UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO
INSTITUTO MULTIDISCIPLINAR

GABRIEL MARINHO DE SOUZA
SYLVINO PREVOT DE OLIVEIRA JUNIOR

Benchmark de RAG para Notícias Esportivas

Prof. Filipe Braida do Carmo, D.Sc.
Orientador

Nova Iguaçu, Dezembro de 2025

Benchmark de RAG para Notícias Esportivas

Gabriel Marinho de Souza

Sylvino Prevot de Oliveira Junior

Projeto Final de Curso submetido ao Departamento de Ciência da Computação do Instituto Multidisciplinar da Universidade Federal Rural do Rio de Janeiro como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Apresentado por:

_____

Gabriel Marinho de Souza

_____

Sylvino Prevot de Oliveira Junior

Aprovado por:

_____

Prof. Filipe Braida do Carmo, D.Sc.

_____

Prof. Leandro Guimarães Marques Alvim, D.Sc.

_____

Prof. Ygor de Mello Canalli, D.Sc.

NOVA IGUAÇU, RJ - BRASIL

Dezembro de 2025

Agradecimentos

Gabriel Marinho de Souza

Agradeço, primeiramente, aos gigantes que me apoiaram nos ombros: meus pais. Ao meu pai, Sergio, por ter me apresentado aos computadores e me ensinado a curiosidade. A minha mãe, Denize, por ter me ensinado a força real da persistência. Vocês me deram suporte para ser quem eu sou e para que eu pudesse ver mais longe. Por tudo isso, serei eternamente grato.

Agradeço também a todos os que trilharam esse caminho ao meu lado, vocês permitiram que esse trajeto fosse mais leve. Em especial, agradeço a Khiara, Sylvino, Marcos e Allan por terem me ensinado e apoiado, tanto nessa caminhada quanto na vida. Obrigado por todas as risadas, conversas, trabalhos, noites de estudos e saídas. Eu não teria conseguido sem vocês.

Agradeço ao professor Filipe Braida pela orientação e ensinamentos ao longo de todo o curso. Suas conversas foram de extrema importância para minha trajetória. Você é um grande exemplo de profissional para mim.

Agradeço a todos os professores do Departamento de Ciência da Computação da UFRRJ pelos ensinamentos e oportunidades.

Agradeço também a todos que fizeram parte da minha caminhada mas não estão mais ao meu lado. Vocês também ajudaram a construir quem sou hoje, e por isso, sou grato.

Por fim, agradeço a mim por, diante de tantas adversidades, dos caminhos ter mantido o certo. Você pode ir além.

Sylvino Prevot de Oliveira Junior

Primeiramente, agradeço a Deus, pois sem Ele nada seria possível. Agradeço pela maior bênção que me concedeu, antes mesmo que eu pudesse respirar: minha mãe, Giseli. Que ela saiba que todo o seu esforço na minha criação valeu a pena. Foi ela quem me ensinou tudo o que sei e, principalmente, me deu coragem para aprender o que ela não poderia ensinar. Obrigado pelo cuidado e proteção, mas, como diz o ditado: um navio no porto é seguro, mas não é para isso que os navios foram construídos.

Agradeço ao meu pai. Tenho orgulho de carregar seu nome e ser lembrado por isso; obrigado pela motivação e ensinamentos. No mesmo sentido, estendo minha gratidão ao meu padrasto, Carlos Alberto, por todo o esforço, respeito e carinho com que me tratou durante todos esses anos.

À minha namorada, Diully Ceni, por ter sido meu refúgio quando mais precisei. Obrigado por me apresentar a vida sob nova perspectiva, por me ajudar a crescer e por me tornar alguém mais forte.

Agradeço àqueles que estiveram ao meu lado, não me permitindo desistir e me ajudando a descobrir quem sou: Allan, Breno, Gabriel Mota, João Pedro Ottoni, Jorge, Mariana e Raiane. Um agradecimento especial ao Gabriel Marinho, que dividiu comigo as angústias e vitórias na produção deste trabalho, reforçando ainda mais a nossa amizade.

Meus sinceros agradecimentos ao professor Filipe Braida e aos demais docentes do Departamento de Ciência da Computação da UFRRJ, pela orientação e pelos ensinamentos fundamentais para minha formação.

Por fim, agradeço a mim mesmo e a tudo que me formou, por ter perseverado através de todas as noites insones e, mesmo nos ambientes mais adversos, ter escolhido o caminho da educação. Encerro com a certeza de que a jornada me transformou:

"Cuidado; pois agora eu já não sinto medo, e por isso sou poderoso."

RESUMO

Benchmark de RAG para Notícias Esportivas

Gabriel Marinho de Souza e Sylvino Prevot de Oliveira Junior

Dezembro/2025

Orientador: Filipe Braida do Carmo, D.Sc.

Os *Large Language Models* (LLM) apresentam limitações intrínsecas, como conhecimento estático e alucinações, que se tornam críticas em domínios factuais e dinâmicos, como o de notícias esportivas. O Retrieval-Augmented Generation (RAG) surge para conectar LLM a bases de conhecimento atualizadas, mas a literatura carece de um benchmark sistemático que compare as arquiteturas RAG neste contexto específico. Este trabalho propõe e executa um benchmark para essa avaliação. Este trabalho contribui com: (1) um conjunto de dados personalizado, gerado com notícias esportivas recentes (setembro de 2025) para evitar contaminação, abrangendo testes factuais, de integração e de rejeição negativa; (2) o desenvolvimento e avaliação de quatro arquiteturas: *Baseline*, *Naive* RAG, *Advanced* RAG e *Graph* RAG; e (3) uma análise aprofundada via framework de avaliação que combina métricas de pipeline (*RAGAS*) e avaliação semântica (*LLM Judge*) para dissecar os trade-offs de cada abordagem. Os resultados demonstram que o *Baseline* (LLM puro) é factualmente obsoleto, obtendo uma qualidade factual média de 1.4413 (em 5), mas com segurança perfeita (5.0). O *Naive* RAG, por sua vez, oferece o maior ganho factual (2.7981), mas exibe falhas de segurança (4.5333). O *Graph* RAG, mais complexo, apresentou desempenho factual fraco (2.0952), sugerindo que a modelagem complexa pode introduzir pontos de falha na recuperação. O *Advanced* RAG, utilizando sumarização e reordenação (re-ranking), obteve desempenho factual comparável ao *Naive* RAG (2.7175), mas foi a única arquitetura RAG a alcançar segurança perfeita (5.0). Conclui-se que otimizações de pipeline, como a reordenação, oferecem o melhor equilíbrio entre desempenho factual, complexidade e confiabilidade para este domínio.

ABSTRACT

Benchmark de RAG para Notícias Esportivas

Gabriel Marinho de Souza and Sylvino Prevot de Oliveira Junior

Dezembro/2025

Advisor: Filipe Braida do Carmo, D.Sc.

Large Language Models (LLM) possess intrinsic limitations, such as static knowledge and hallucinations, which are critical in factual and dynamic domains like sports news. RAG emerges to connect LLM to up-to-date knowledge bases, yet the literature lacks a systematic benchmark comparing RAG architectures in this specific context. This work proposes and executes a benchmark for this evaluation. This study contributes: (1) a custom dataset generated from recent sports news (September 2025) to prevent contamination, covering factual, information integration, and negative rejection tests; (2) the development and evaluation of four architectures: Baseline, Naive RAG, Advanced RAG, and Graph RAG; and (3) an in-depth analysis using an evaluation framework that combines pipeline metrics (RAGAS) and semantic evaluation (LLM Judge) to dissect the trade-offs of each approach. Results show that the Baseline (pure LLM) is factually obsolete, achieving an average factual quality score of 1.4413 (out of 5), despite perfect safety (5.0). The Naive RAG offers the highest factual gain (2.7981) but exhibits safety failures (4.5333). The more complex Graph RAG showed weak factual performance (2.0952), suggesting that modeling complexity can introduce failure points in retrieval. The Advanced RAG, utilizing summarization and re-ranking, achieved factual performance (2.7175) comparable to Naive RAG but was the only RAG architecture to achieve perfect safety (5.0). It is concluded that pipeline optimizations, such as re-ranking, offer the best balance between factual performance, complexity, and reliability for this domain.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

LLM	*Large Language Models*
RAG	Retrieval-Augmented Generation
RGB	Retrieval-Augmented Generation Benchmark
HNSW	Hierarchical Navigable Small Worlds
JSON	JavaScript Object Notation
HTTP	HyperText Transfer Protocol
HTML	HyperText Markup Language
HTML	HyperText Markup Language
URL	Uniform Resource Locator

Sumário

Capítulo 1

Introdução

Os *Large Language Models* (LLMs) revolucionaram o processamento de linguagem natural, demonstrando uma capacidade sem precedentes de compreender e gerar textos com fluidez humana (RADFORD et al., 2019). Esses modelos, pré-treinados em vastos conjuntos de dados, internalizam padrões complexos, gramática e um amplo conhecimento de mundo, o que contribuiu para sua rápida adoção em diversas aplicações. Contudo, essa arquitetura possui limitações intrínsecas, como as alucinações e a fronteira de conhecimento delimitado pela data de treinamento, o que se traduz em baixo desempenho para alguns tipos de aplicação (HUANG et al., 2025).

Essas limitações tornam-se particularmente críticas em domínios de alta volatilidade, como o de notícias esportivas. Nesses cenários, a informação torna-se obsoleta rapidamente, e a dependência do conhecimento paramétrico do modelo leva a respostas desatualizadas ou factualmente incorretas. Como solução para essas deficiências, a arquitetura de RAG emergiu como uma abordagem proeminente (LEWIS et al., 2021). O RAG aprimora os LLMs ao conectá-los a uma base de conhecimento externa e dinâmica, permitindo que o modelo baseie suas respostas em informações verificáveis e recentes no momento da inferência (LEWIS et al., 2021).

Contudo, a literatura atual, embora explore RAG em domínios onde o conhecimento não se torna obsoleto com o tempo, carece de uma análise comparativa sistemática que avalie a eficácia dessas diferentes arquiteturas em domínios de alta

volatilidade, como o jornalismo esportivo. A escolha da arquitetura ideal para esse tipo de domínio é uma questão em aberto, sem um *benchmark* claro que meça o equilíbrio entre a complexidade de desenvolvimento e o ganho prático em factualidade e segurança.

1.1 Objetivo

O objetivo central deste trabalho é propor e executar um *benchmark* para a avaliação sistemática e comparativa de arquiteturas RAG no domínio de notícias esportivas. Diante da inexistência de conjuntos de dados disponíveis para essa finalidade, este estudo foca em dois objetivos principais: (1) a construção de um conjunto de dados personalizado a partir de notícias recentes, garantindo que a avaliação não sofra de contaminação de conhecimento; e (2) o desenvolvimento e avaliação comparativa de três arquiteturas RAG distintas: *Naive*, *Advanced* e *Graph* RAG.

Utilizando uma metodologia de avaliação que combina métricas de *pipeline* com uma avaliação semântica da qualidade da resposta por um *LLM Judge*, este trabalho analisa a relação entre o desempenho factual e a segurança contra alucinações existente nessas arquiteturas. O resultado é uma análise quantitativa que identifica qual abordagem oferece o melhor equilíbrio entre complexidade, factualidade e robustez para o domínio proposto.

1.2 Resumo dos Resultados

A análise dos dados revela que o *Naive* RAG apresentou a maior eficácia nas métricas de qualidade da resposta, atingindo médias de 2.7981 em *Accuracy* e 0.6337 em *Faithfulness*. No entanto, ao examinarmos os componentes de recuperação, o *Advanced* RAG demonstrou superioridade, com médias de 0.9079 para *Context Precision* e 0.7682 para *Context Recall*. Além disso, no critério de segurança via rejeição negativa, esta abordagem e o *Baseline* LLM obtiveram desempenho ideal,

registrando média de 5.0000. Em contraste com os demais, o *Graph* RAG obteve os resultados mais baixos em todas as dimensões avaliadas.

1.3 Organização do Trabalho

Este trabalho está estruturado em cinco capítulos. O Capítulo 1, esta introdução, contextualiza o problema do conhecimento estático dos LLMs e apresenta os objetivos da pesquisa. O Capítulo 2, *Retrieval-Augmented Generation*, estabelece a fundamentação teórica, detalhando a arquitetura dos LLMs, suas limitações e o funcionamento das arquiteturas RAG. O Capítulo 3, *Proposta*, descreve o *benchmark* desenvolvido, detalhando a escolha do domínio, os trabalhos relacionados, a modelagem das arquiteturas RAG selecionadas e o *framework* de avaliação proposto. O Capítulo 4, *Metodologia Experimental e Resultados*, detalha a construção do conjunto de dados, o desenvolvimento das arquiteturas e apresenta a análise quantitativa dos dados coletados no *benchmark*. Finalmente, o Capítulo 5, *Conclusão*, sumariza as descobertas, discute as limitações do estudo e propõe trabalhos futuros.

Capítulo 2

Retrieval-Augmented Generation

Este capítulo estabelece a base teórica necessária para a compreensão da proposta deste trabalho. A exposição inicia-se com uma análise dos LLMs, detalhando seu funcionamento probabilístico e as limitações intrínsecas como alucinações e o conhecimento estático, que motivam esta pesquisa. Em seguida, é apresentada a arquitetura de RAG como uma solução proeminente para essas deficiências, detalhando seus diferentes níveis de complexidade.

2.1 Large Language Models

Estruturalmente, esses modelos se dividem em três arquiteturas principais: *encoder-only*, *decoder-only* e *encoder-decoder* (VASWANI et al., 2023). Modelos *encoder-only*, como o BERT, são otimizados para tarefas de compreensão de texto, como análise de sentimento (DEVLIN et al., 2019). A arquitetura *decoder-only*, popularizada pela série GPT, especializa-se na geração de texto sequencial (RADFORD et al., 2019). Por fim, os modelos *encoder-decoder*, como o T5, são projetados para tarefas de transformação de uma sequência de entrada em outra, sendo ideais para tradução automática (RAFFEL et al., 2023).

A entrada e a saída de um LLM é uma sequência de sub-palavras, chamadas de *tokens*. O processo de inferência desse tipo de modelo é fundamentalmente

probabilístico. Sua tarefa consiste em, a partir de uma sequência de *tokens*, chamada de *prompt*, calcular o *token* mais provável que deveria ser gerado e repetir esse processo para formar uma sequência de *tokens*. Essa abordagem se difere de um processo de raciocínio lógico ou semântico, em vez disso, baseia-se inteiramente nos padrões identificados durante a fase de treinamento em um vasto *corpus* de dados textuais. (VASWANI et al., 2023)

Nos modelos generativos, *i.e.*, decoder-only e encoder-decoder, a geração de texto é um processo autorregressivo. A cada passo, o modelo processa os tokens anteriores e, ao final, sua camada linear de saída calcula um vetor de pontuações brutas, *logits*, para todo o vocabulário (VASWANI et al., 2023). Após a conversão dos *logits* em uma distribuição de probabilidades, aplicam-se estratégias de amostragem para restringir o conjunto de *tokens* candidatos.

As técnicas mais comuns de restrição do conjunto de tokens incluem *Top-K sampling*, que considera apenas os K *tokens* mais prováveis, e *Top-P*, *i.e. Nucleus sampling*, que seleciona o menor conjunto de *tokens* cuja probabilidade acumulada ultrapassa um limiar P. Um *token* é então aleatoriamente selecionado desse subconjunto filtrado e anexado à sequência. A aleatoriedade é controlada por parâmetros como a temperatura, que ajusta a distribuição dos *logits* antes da amostragem. (HOLTZMAN et al., 2020)

Uma limitação fundamental dos LLMs, decorrente de sua natureza conexionista, é que seu processo de geração de *tokens* não se baseia em dedução lógica, mas sim em reconhecimento de padrões estatísticos. Essa característica é a causa principal de sua falha mais notória, as alucinações, fenômeno onde o modelo gera conteúdo plausível, porém falso. Este problema é classificado em duas categorias principais: *Factuality Hallucination*, que é a discrepância entre o conteúdo gerado e fatos verificáveis do mundo real, e *Faithfulness Hallucination*, que ocorre quando a resposta contradiz o contexto fornecido pelo usuário ou as próprias instruções. Essas falhas também ocorrem porque o conhecimento do modelo, derivado de seu treinamento, pode ser falho, enviesado ou simplesmente desatualizado, levando à incorporação de informações incorretas com aparente confiança. (HUANG et al., 2025)

Adicionalmente, o conhecimento de um LLM é estático, limitado aos dados de seu treinamento, o que o impede de acessar informações novas ou de domínios específicos. Para superar essa deficiência, duas estratégias principais podem ser empregadas: o fine-tuning, que envolve o retreinamento do modelo para internalizar novos dados, e o Retrieval-Augmented Generation (RAG), que consiste na utilização de conhecimento externo no momento da inferência.

O fine-tuning especializa o modelo através do retreinamento em um corpus de domínio específico (ANISUZZAMAN et al., 2025). Contudo, tal processo se torna mais computacionalmente custoso que o RAG para domínios dinâmicos, que demandam atualizações constantes de conhecimento (GAO et al., 2024). Considerando que o domínio de notícias esportivas, foco deste trabalho, é caracterizado por um fluxo contínuo de novas informações, o retreinamento frequente do modelo se mostra inviável. Diante dessa limitação, a abordagem de RAG se mostra mais adequada para os objetivos propostos.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) é uma abordagem projetada para mitigar os problemas de alucinações e conhecimento estático nos LLMs. Proposto originalmente por Lewis et al. (2021), o RAG foi apresentado como uma abordagem de fine-tuning de propósito geral, projetada para combinar a memória paramétrica com uma memória não-paramétrica.

A arquitetura combinava um recuperador pré-treinado com um gerador *seq2seq* pré-treinado. A memória não-paramétrica consistia em um índice vetorial da Wikipedia. Crucialmente, o sistema era treinado de forma *end-to-end*, permitindo que o recuperador aprendesse a buscar documentos que fossem úteis especificamente para o gerador produzir a resposta correta. (LEWIS et al., 2021)

Neste trabalho, duas formulações para essa arquitetura híbrida foram descritas. Formalmente, o modelo recebe uma sequência de entrada x e utiliza documentos recuperados z, tratados como variável latente, para gerar a sequência de saída y,

composta por N tokens. (LEWIS et al., 2021)

A primeira formulação, *RAG-Sequence*, recupera os documentos uma única vez e utiliza o mesmo conjunto de informações latentes para gerar toda a sequência de resposta. A probabilidade da sequência final $p(y|x)$ para este modelo é definida pela Equação 2.1, onde $p_\eta(z|x)$ é a probabilidade do recuperador e p_θ a do gerador. (LEWIS et al., 2021)

A segunda, *RAG-Token*, segue o mesmo fluxo, mas permite que o modelo busque documentos diferentes a cada *token* gerado (y_i), possibilitando a síntese de informações de múltiplas fontes, conforme a Equação 2.2. (LEWIS et al., 2021)

$$p_{\text{RAG-Sequence}}(y|x) \approx \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x) \prod_{i=1}^{N} p_\theta(y_i|x, z, y_{1:i-1}) \tag{2.1}$$

$$p_{\text{RAG-Token}}(y|x) \approx \prod_{i=1}^{N} \sum_{z \in \text{top-k}(p(\cdot|x))} p_\eta(z|x) p_\theta(y_i|x, z, y_{1:i-1}) \tag{2.2}$$

Embora o RAG tenha sido originalmente proposto como um método de fine-tuning (LEWIS et al., 2021), com o surgimento de modelos com capacidade de realizar aprendizado por contexto, *i.e.*, *In-Context Learning*, o RAG se tornou um *pipeline* de inferência. Tanto o recuperador quanto o gerador são usados sem retreinamento, e o conhecimento é fornecido dinamicamente ao LLM através do *prompt*.

Ao fundamentar a geração de texto em fontes de dados externas e verificáveis, esta abordagem de RAG aumenta a factualidade das respostas e permite que o modelo utilize conhecimento além de seu treinamento original. Conforme representado na Figura 2.1, o processo se divide em três etapas: indexação, recuperação e geração. (GAO et al., 2024)

A fase de indexação é o processo offline de preparação da base de conhecimento. Inicialmente, os documentos brutos são carregados e segmentados em trechos menores e gerenciáveis, conhecidos como *chunks*. Essa segmentação é crucial para garantir que as unidades de informação recuperadas sejam concisas e semanticamente autocontidas.

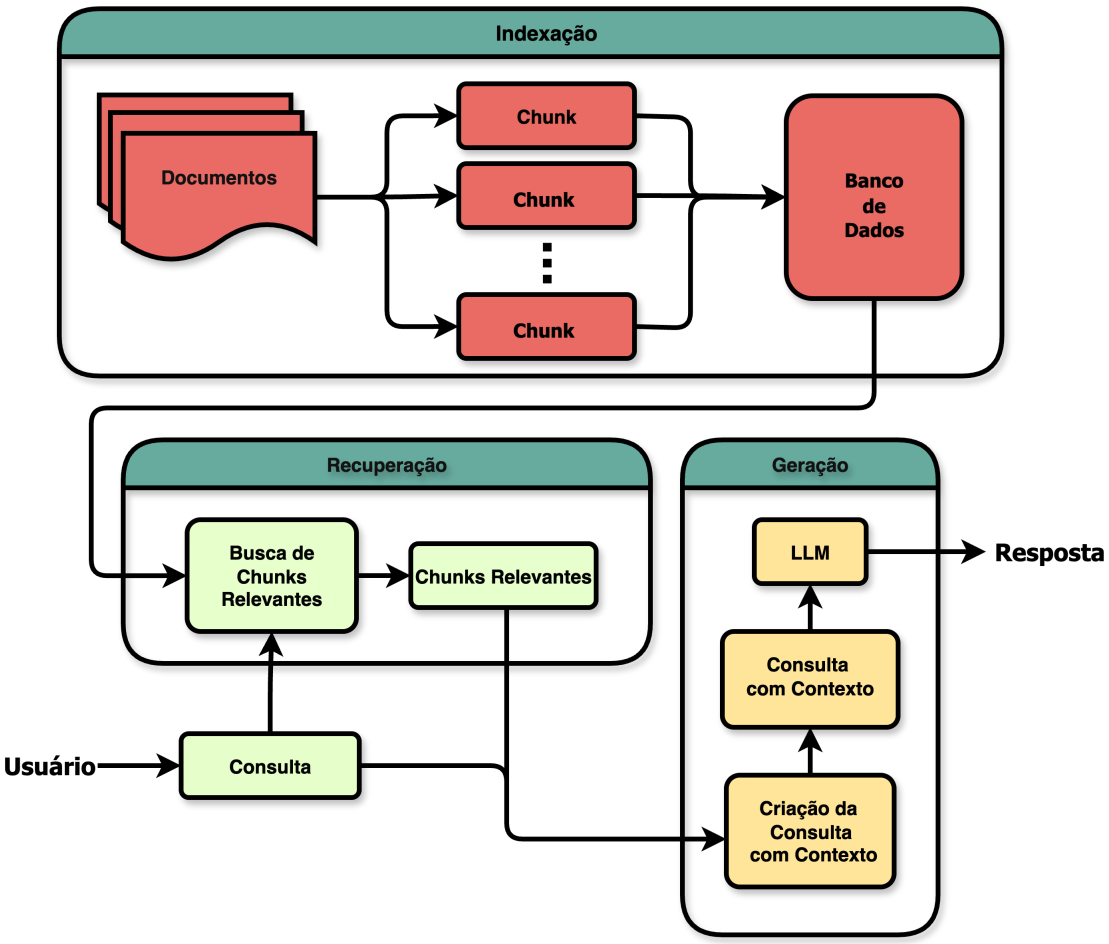

Figura 2.1: Representação esquemática do fluxo de funcionamento de um sistema RAG, destacando as etapas fundamentais de Indexação, Recuperação e Geração. Fonte: Adaptado de Gao et al. (2024).

Cada *chunk* é então processado por um modelo de *embedding*, que o converte em um vetor numérico de alta dimensão. Esses vetores são armazenados em um índice, comumente uma base de dados vetorial, que é otimizada para realizar buscas por similaridade em alta velocidade, permitindo a localização eficiente de informações relevantes posteriormente. (GAO et al., 2024)

A etapa de recuperação ocorre em tempo de inferência e é acionada pela consulta do usuário. A pergunta é primeiramente transformada em um vetor utilizando o mesmo modelo de *embedding* da fase de indexação, garantindo que a comparação seja feita no mesmo espaço vetorial. Este vetor de consulta é então utilizado para buscar no índice os *chunks* mais relevantes. A busca tipicamente se baseia em algoritmos de

similaridade de cosseno ou outras métricas de distância, que identificam os vetores de texto cujos conteúdos são semanticamente mais próximos da pergunta. O resultado desta fase é um conjunto de k trechos de texto recuperados, que constituem o contexto a ser fornecido ao LLM. (GAO et al., 2024)

Finalmente, na fase de geração, o sistema combina as informações recuperadas com a pergunta original do usuário. Os *chunks* de texto relevantes são formatados e inseridos em um *prompt*, que instrui o LLM a formular sua resposta com base no contexto fornecido. Este processo, conhecido como *grounding* ou ancoragem, orienta o modelo a atuar como um sintetizador de informação, em vez de depender exclusivamente de seu conhecimento parametrizado. O resultado é uma resposta que não apenas tende a ser mais precisa e factual, mas que também pode citar as fontes utilizadas, aumentando a transparência e a confiabilidade do sistema. (GAO et al., 2024)

No trabalho de Gao et al. (2024) é proposta uma taxonomia que classifica as arquiteturas RAG em três categorias: *Naive* RAG, *Advanced* RAG e *Modular* RAG. A distinção entre elas se concentra em duas fases principais do processo: a de indexação e a de recuperação. A fase de indexação envolve o pré-processamento e armazenamento de trechos de documentos, *i.e.*, *chunks*, na base de conhecimento. Subsequentemente, na fase de recuperação, os *chunks* mais relevantes para a consulta do usuário são selecionados para servirem de contexto na geração da resposta pelo modelo de linguagem. A seguir são delimitadas as diferenças entre as categorias.

O *Naive* RAG consiste na arquitetura mais simples e representa o paradigma fundamental de *retrieve-then-read* (MA et al., 2023). Esse tipo abrange qualquer arquitetura que siga um fluxo sequencial direto, sem otimizações. O processo inicia-se com a indexação, onde os documentos são segmentados em *chunks* de texto, vetorizados através de um modelo de *embedding* e armazenados em um banco de dados vetorial. Na etapa de recuperação, a pergunta do usuário é igualmente vetorizada e utilizada para realizar uma busca por similaridade, recuperando os *chunks* mais relevantes do banco de dados. Por fim, esses *chunks* são inseridos de maneira organizada ao *prompt* original para fornecer o contexto necessário ao LLM,

que então gera a resposta final.

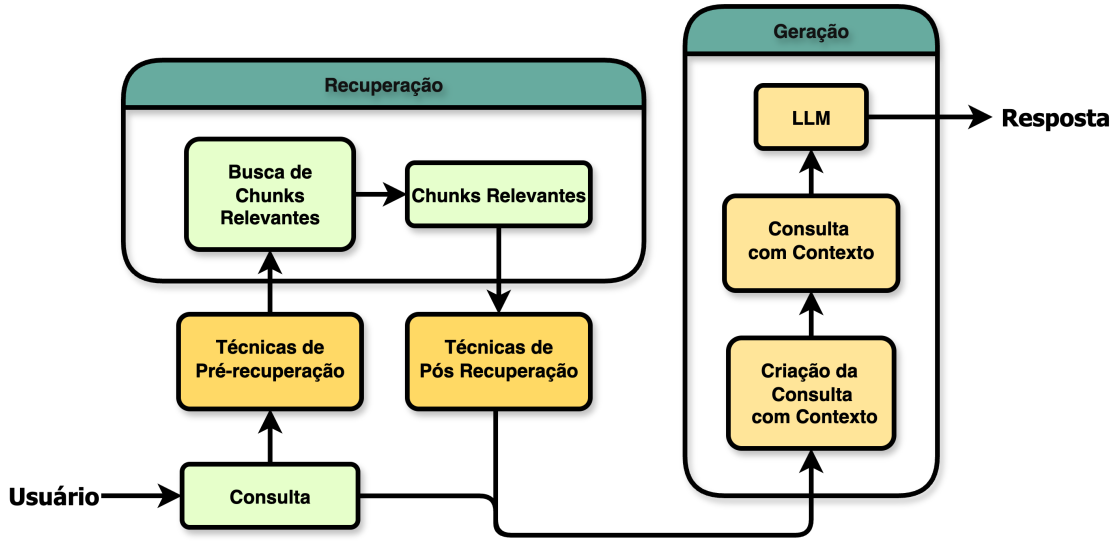

Figura 2.2: Fluxograma do paradigma *Advanced* RAG, evidenciando a adição de técnicas de otimização nas etapas de pré-recuperação e pós-recuperação. Fonte: Adaptado de Gao et al. (2024).

O *Advanced* RAG engloba técnicas que adicionam etapas de processamento *pre-retrieval* e *post-retrieval*, *i.e.*, antes e após da busca, respectivamente, para aprimorar a qualidade do contexto fornecido ao LLM, como na Figura 2.2. As estratégias de *pre-retrieval* focam em otimizar tanto a consulta do usuário quanto os dados indexados, incluindo técnicas como *query rewriting* (MA et al., 2023), que consiste na reescrita de perguntas para maior clareza ou otimização da própria indexação com metadados. Já as estratégias de *post-retrieval* atuam sobre os *chunks* já recuperados, aplicando métodos como *re-ranking* (NOGUEIRA; CHO, 2019), que reordena os *chunks* recuperados, para priorizar a informação mais relevante ou a compressão do contexto para eliminar ruídos e adequá-lo à janela de contexto do LLM.

O *Modular* RAG é o mais poderoso e flexível dos três, pois substitui o *pipeline* linear por uma arquitetura adaptativa composta por múltiplos módulos especializados e intercambiáveis (GAO et al., 2024). Essa abordagem permite a adição de novas capacidades e a execução de fluxos não lineares e mais dinâmicos. Em vez de uma sequência fixa, o *Modular* RAG pode incorporar componentes como o *query router*, que consiste em um roteador de perguntas que decide qual fonte de dados

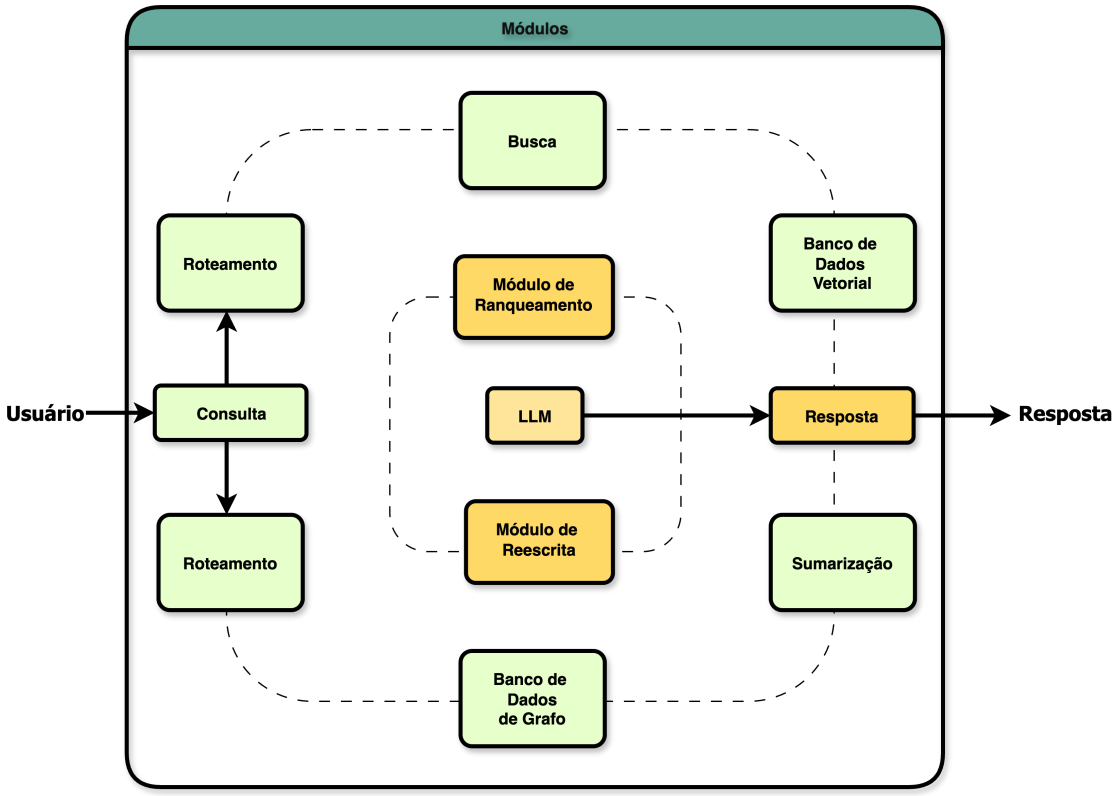

Figura 2.3: Arquitetura do paradigma *Modular* RAG, ilustrando a flexibilidade de composição entre módulos independentes como roteamento, reescrita e ranqueamento. Fonte: Adaptado de Gao et al. (2024).

consultar ou se a busca é necessária, e até mesmo módulos que executam outros tipos de RAG de forma iterativa. Analisando a arquitetura da Figura 2.3, por exemplo, uma única consulta poderia ser inicialmente avaliada por um roteador, que a direciona para o módulo de busca. Os resultados poderiam então ser enviados a um módulo de reescrita para maior clareza, antes de serem finalmente organizados por um ranqueador que otimiza a relevância para a geração final. Técnicas como o *Self-RAG* (ASAI et al., 2023), onde o próprio LLM aprende a decidir quando e o que recuperar, são exemplos dessa arquitetura, que permite um raciocínio mais sofisticado e adaptado à complexidade de cada consulta.

Capítulo 3

Proposta

Este capítulo apresenta uma proposta de *benchmark* para a avaliação sistemática de diferentes arquiteturas de RAG. A motivação reside na vulnerabilidade dos LLMs em domínios em que o conhecimento é de natureza volátil, como o de notícias esportivas, onde seu conhecimento estático rapidamente se torna obsoleto. Para enfrentar este desafio, o experimento deste trabalho consiste em utilizar notícias recentes do portal Globo Esporte[1] como uma base de conhecimento externa e dinâmica, avaliando qual arquitetura RAG se desempenha melhor na tarefa de consumir essas informações para responder a perguntas factuais.

O objetivo é conduzir uma análise comparativa profunda, fundamentada nos frameworks Retrieval-Augmented Generation Benchmark (RGB) (CHEN et al., 2023) e TRACe (FRIEL; BELYI; SANYAL, 2025). Para isso, o trabalho aplica uma metodologia robusta que combina métricas de *pipeline* e avaliação semântica, visando analisar o *trade-off* entre a complexidade da arquitetura e seu impacto prático no desempenho factual e na segurança.

As seções subsequentes detalham os pilares desta proposta. A argumentação inicia com a justificativa para a escolha do domínio de notícias esportivas e a consequente necessidade de um conjunto de dados personalizado, seguida por uma análise dos trabalhos relacionados para contextualizar as lacunas existentes. O cerne da proposta

[1] <https://ge.globo.com/>

é então apresentado através da metodologia de construção do conjunto de dados, da concepção das arquiteturas, *Naive* RAG, *Advanced* RAG e *Graph* RAG, e da definição do *framework* de avaliação que será utilizado para analisar os resultados.

3.1 Domínio de Notícias Esportivas

O domínio de notícias esportivas foi selecionado como caso de estudo por encapsular de forma exemplar os desafios centrais para os LLMs. A informação neste campo é extremamente volátil, com novas notícias sobre resultados, transferências e análises de partidas sendo publicados diariamente, tornando o conhecimento obsoleto em questão de horas. Parte do conhecimento sobre esportes é inerentemente estruturado, composto por entidades bem definidas, como jogadores, times, competições e os relacionamentos explícitos que os conectam. Nas notícias, entretanto, essa estrutura é apresentada em formato não estruturado, o que impõe o desafio de extrair e compreender estas conexões densas e interligadas, exigindo um sistema que supere a fronteira de conhecimento estático dos LLMs.

Mesmo com essas características, a aplicação de RAG no jornalismo esportivo se mostra uma área de pesquisa com aparente lacuna na literatura. Investigações existentes tendem a focar em domínios de conhecimento mais estáveis, como fazem Amugongo et al. (2025) na área da saúde, ou em tarefas de pergunta e resposta de escopo aberto, como no trabalho de Friel, Belyi e Sanyal (2025). A natureza relacional dos dados esportivos levanta a questão fundamental de como diferentes arquiteturas RAG se comportam diante desses desafios. A literatura carece de *benchmarks* que comparem sistematicamente as vantagens e desvantagens dessas abordagens em um cenário tão dinâmico e interconectado.

As arquiteturas de RAG, portanto, surgem como uma solução aparentemente adequada para este problema, dada a sua capacidade de externalizar a fonte de conhecimento, tornando-a independente dos parâmetros do LLM e permitindo a sua atualização contínua sem excessivo custo computacional (SOUDANI; KANOULAS; HASIBI, 2024). Ao fundamentar a geração em fontes externas e recentes, o RAG

pode, em tese, fornecer respostas factuais sobre eventos que ocorreram após o corte de treinamento do modelo. Antes de propor uma nova metodologia de avaliação, é imperativo analisar o estado da arte para contextualizar a contribuição deste trabalho e fundamentar a lacuna previamente identificada. A seção a seguir, portanto, examina os principais *benchmarks* e arquiteturas RAG existentes na literatura.

3.2 Trabalhos Relacionados

A validação da eficácia de sistemas RAG tem sido objeto de crescente interesse acadêmico, resultando na proposição de diversos *benchmarks* robustos. A revisão da literatura revelou um foco em duas frentes principais: a criação de metodologias de avaliação e o desenvolvimento de arquiteturas RAG mais sofisticadas. No entanto, a aplicação e teste dessas inovações em domínios de conhecimento de alta volatilidade, como o de notícias, permanecem como uma lacuna significativa.

A fundação para a avaliação moderna de RAG foi estabelecida por trabalhos como o de Chen et al. (2023), que propôs o RGB. O foco de avaliação do RGB se concentra na robustez a ruído, rejeição negativa e integração de informação. Entretanto, ele emprega uma correspondência exata de texto para medir a acurácia, uma abordagem rígida que pode penalizar respostas semanticamente corretas, mas textualmente diferentes. Este trabalho se baseia nas capacidades cognitivas definidas pelo RGB, mas avança ao propor um método de avaliação mais flexível.

Paralelamente ao desenvolvimento de *benchmarks*, a pesquisa tem explorado arquiteturas RAG mais autônomas e complexas. Um exemplo notável é o *SELF-RAG*, proposto por Asai et al. (2023), um *framework* que treina o LLM para decidir por si só quando a recuperação é necessária e para criticar suas próprias gerações através de tokens de reflexão. A proposta do *SELF-RAG* evidencia a fronteira da pesquisa, focada em sistemas com maior capacidade de autocrítica.

O *framework* TRACe, proposto por Friel, Belyi e Sanyal (2025), oferece uma metodologia centrada no pipeline, com o objetivo de medir o desempenho do *retriever*, i.e, recuperador, e do gerador. As métricas *Context Utilization* e *Adherence* medem,

respectivamente, a eficiência e a factualidade do gerador, enquanto *Context Relevance* avalia a qualidade do recuperador. Em conjunto, essas métricas permitem um diagnóstico granular sobre a saúde mecânica do processo, identificando se as falhas se originam em uma recuperação deficiente ou em uma geração defeituosa.

A importância de validar a confiabilidade de RAG em domínios especializados é reforçada pela revisão sistemática de Amugongo et al. (2025), na qual se investiga a aplicação de RAG no setor da saúde. O estudo consolida os desafios e as oportunidades do uso de RAG em um campo onde a factualidade é crítica, alinhando-se diretamente à motivação deste trabalho. Embora focado na área de saúde, o artigo destaca a necessidade universal de *benchmarks* robustos para validar a aplicação de RAG em cenários onde a precisão da informação é primordial.

Em suma, a análise dos trabalhos relacionados revela que, embora existam avaliações de propósito geral e arquiteturas avançadas, estas concentram-se predominantemente em domínios de conhecimento estático. A literatura atual carece de estudos empíricos dedicados a avaliar e comparar o desempenho de arquiteturas RAG no contexto de domínios de alta volatilidade, como o de notícias. Esta lacuna impede a compreensão de como diferentes estratégias de recuperação se comportam quando a factualidade da informação é sensível à temporalidade. Para viabilizar essa análise, o primeiro passo é a construção de um novo conjunto de dados de avaliação que simule tais desafios.

3.3 Conjunto de Dados de Avaliação

A validade experimental de sistemas RAG depende estritamente da garantia de que o modelo não resolva as questões por simples memorização. Uma barreira significativa para essa validação é a contaminação de dados: como os LLMs são pré-treinados com vastos *corpora* coletados da internet, é frequente que os próprios conjuntos de dados de avaliação públicos já tenham sido ingeridos pelo modelo durante seu treinamento. Nesses casos, o modelo pode acertar a resposta recorrendo à sua memória paramétrica, mascarando a ineficiência do sistema de recuperação

(CHEN et al., 2023). Para mitigar esse risco e isolar a capacidade de fundamentação do sistema, torna-se mandatório a curadoria de um conjunto de dados inédito. No contexto de notícias esportivas, isso é alcançado selecionando eventos recentes, ocorridos comprovadamente após a data de corte do conhecimento dos modelos avaliados (FRIEL; BELYI; SANYAL, 2025).

Com base nesta premissa, foi necessário construir um conjunto de dados para criar um ambiente de avaliação controlado. Esse conjunto é estruturado como uma coleção de artigos de notícias, coletadas em 30 de setembro de 2025, onde cada artigo funciona como uma fonte de conhecimento autocontida. Associado a cada um, é gerado um conjunto de pares de pergunta e resposta de referência. A lógica de geração define que cada pergunta simples seja formulada sinteticamente a partir do texto do artigo correspondente, ou, no caso das perguntas multi-contexto, de múltiplos artigos. Respectivamente, a resposta de referência é produzida utilizando exclusivamente o conteúdo do mesmo artigo ou dos múltiplos artigos usados para geração. Este princípio de ancoragem factual é crucial, pois assegura que a avaliação seja feita contra uma verdade fundamental estritamente contida na base de conhecimento e cria um cenário ideal para testar a precisão da recuperação e a fidelidade da geração a partir de uma fonte bem definida (GAO et al., 2024).

Com este conjunto de dados estabelecido como a base para o *benchmark*, o próximo passo é detalhar as arquiteturas que serão submetidas à avaliação, cada uma representando uma abordagem distinta para a modelagem da base de conhecimento e para a recuperação da informação.

3.4 Arquiteturas de RAG Selecionadas

Para conduzir esta avaliação, o desempenho de um LLM ao responder as perguntas sem acesso à base de conhecimento estabelece o desempenho mínimo esperado, o *Baseline*. A partir daí, três arquiteturas RAG foram desenvolvidas para avaliar o ganho incremental de cada abordagem. A primeira, *retrieve-then-read*, serve como a arquitetura fundamental. A segunda, *Advanced* RAG, utilizando sumarização e

reordenação, introduz otimizações no *pipeline* através de etapas de sumarização e reordenação. Por fim, a terceira, *Graph* RAG, representa uma mudança paradigmática ao modelar o conhecimento em uma estrutura de grafo híbrida.

A abordagem *retrieve-then-read* representa a instância mais fundamental do paradigma RAG, o *Naive* RAG e representa a arquitetura mais simples. Nessa abordagem, a base de conhecimento é concebida como uma coleção não estruturada de *chunks*, extraídos diretamente do conjunto de dados. A lógica de recuperação é um processo linear: a pergunta do usuário é transformada em uma representação vetorial e utilizada para realizar uma busca por similaridade. Os *n chunks* mais semanticamente similares são recuperados e concatenados para formar o contexto que será fornecido ao LLM. (GAO et al., 2024)

O modelo de *Advanced* RAG investiga o impacto de otimizações no *pipeline* de recuperação, introduzindo etapas de pré e pós-processamento. Na fase *pre-retrieval*, este trabalho modela o uso da sumarização para criar representações mais densas dos documentos antes da busca, visando aprimorar a identificação inicial das fontes mais relevantes. Subsequentemente, na fase *post-retrieval*, é utilizado um modelo *reranker*. Este modelo reavalia os *chunks* inicialmente recuperados com um critério de relevância mais sofisticado, com o objetivo de refinar a seleção e priorizar as informações mais pertinentes no contexto final. (GAO et al., 2024)

A arquitetura *Graph* RAG representa uma mudança paradigmática na modelagem da base de conhecimento, substituindo a representação linear de texto por uma estrutura de grafo de conhecimento híbrida. Nesta abordagem, cada notícia extraída é instanciada no grafo como um nó *Documento* central. O conteúdo textual é então decomposto em nós *Chunk*, menores e vetorizados, onde cada um é explicitamente ligado ao seu nó *Documento* de origem por meio de um relacionamento. Coexistindo com essa camada textual, há uma camada de conhecimento estruturado: nós *Entidade* e os relacionamentos explícitos que os conectam, ambos extraídos do texto completo por um LLM. Essa estrutura híbrida é interconectada por *Relações*, que ligam os *Chunks* às *Entidades* que eles citam. O processo de recuperação explora essa topologia, combinando a busca vetorial inicial nos *Chunks* com uma expansão contextual através

do grafo para coletar um subgrafo de conhecimento rico, que é então sintetizado em linguagem natural para servir de contexto ao LLM. (GAO et al., 2024)

3.5 Metodologia de Avaliação

A complexidade inerente aos sistemas RAG, que combinam recuperação de informação e geração de linguagem, torna a avaliação uma tarefa multifacetada. Segundo Friel, Belyi e Sanyal (2025), não há um conjunto de métricas universalmente aceito, exigindo uma abordagem que meça tanto a qualidade da resposta final quanto a eficácia das etapas intermediárias. Esses desafios são enfrentados na literatura por diferentes focos.

O *framework* TRACe oferece uma metodologia centrada no pipeline, com o objetivo de dissecar e medir a eficácia do recuperador (*retriever*) e do gerador. Métricas como *Context Relevance* avaliam a qualidade do recuperador, enquanto *Context Utilization* e *Adherence* medem, respectivamente, a eficiência e a factualidade do gerador. Em conjunto, essas métricas permitem um diagnóstico granular sobre a saúde do processo, identificando se as falhas se originam em uma recuperação deficiente ou em uma geração defeituosa. (FRIEL; BELYI; SANYAL, 2025)

Em contraste, o *framework* RGB, proposto por Chen et al. (2023), foca na avaliação das capacidades cognitivas do LLM ao ser apresentado com um contexto aumentado. Esta metodologia testa habilidades de raciocínio em quatro diferentes focos. O *Noise Robustness* consiste na capacidade de ignorar informação distrativa. A *Negative Rejection* é a habilidade de se recusar a responder quando a informação é insuficiente. A *Information Integration* é a capacidade de sintetizar uma resposta a partir de múltiplos documentos. Por fim, a *Counterfactual Robustness* avalia a capacidade do modelo de identificar e não ser enganado por informações incorretas, *i.e.*, contrafactuais presentes no contexto recuperado. Embora o RGB seja abrangente, este trabalho foca especificamente nos eixos de *Negative Rejection*, *Information Integration* e *Noise Robustness* pois são os mais alinhados aos desafios do domínio proposto. A *Counterfactual Robustness* não é do escopo desta avaliação, uma

vez que o domínio de notícias esportivas não apresenta cenários onde informações deliberadamente incorretas sejam o problema central.

Com base na análise desses *frameworks* de avaliação de RAG, este trabalho propõe uma avaliação híbrida que sintetiza e aprimora as abordagens existentes, justificando-se por duas diferenciações estratégicas. Em contraste com a métrica de acurácia do RGB, que se baseia em uma correspondência exata de texto, nossa proposta emprega um *LLM Judge*, baseado no trabalho de Gu et al. (2025), para uma avaliação semântica. Esta abordagem, embora computacionalmente mais custosa, é mais robusta, pois é capaz de reconhecer respostas corretas que são parafraseadas ou estruturadas de forma diferente da resposta de referência. Além disso, enquanto o *framework* TRACe oferece um diagnóstico quantitativo essencial para a saúde do pipeline, ele não capta a qualidade geral da resposta. O *LLM Judge* complementa essa análise julgando simultaneamente critérios complexos como completude e a integração de informação, o que oferece uma visão multifacetada da performance que se aproxima do julgamento humano.

A partir dessa abordagem de avaliação híbrida, este trabalho adota um *framework* de análise multifacetado. A metodologia foca na análise granular de cada métrica de forma independente, permitindo uma compreensão mais profunda dos *trade-offs* de cada arquitetura.

A avaliação do desempenho de cada sistema utiliza a média aritmética das pontuações de cada resposta dada pelo modelo para aferir o desempenho geral e o desvio padrão para medir a consistência da arquitetura, apresentando os resultados no formato padrão (média $\pm$ desvio padrão). Um desvio baixo é interpretado como um comportamento estável e previsível, enquanto um desvio alto indica que o desempenho variou significativamente entre as perguntas. Além disso, a análise é dividida de acordo com as duas responsabilidades distintas de um sistema RAG, que não podem ser avaliadas da mesma maneira: o desempenho factual e a segurança contra alucinação.

Para medir o desempenho factual, aplicado às perguntas simples e de multi-contexto, foi utilizado o conjunto completo de métricas de *pipeline*, *Context Precision*

e *Context Recall*, e de geração, *Accuracy* e *Faithfulness*. As métricas *Faithfulness* e *Context Relevance* seguem as definições formais do trabalho de Es et al. (2025), focado em avaliação sem referência, definido nas Equações 3.1 e 3.2. Já o *Context Recall* avalia a capacidade do recuperador em encontrar a informação contida no gabarito (*ground-truth*) e é definido na Equação 3.4. A *Accuracy* é definida pela Equação 3.3.

O *Faithfulness* (F) avalia se a resposta é fundamentada no contexto, evitando alucinações. Dado um conjunto de afirmações S extraídas da resposta gerada, calcula-se a razão daquelas (V) que podem ser inferidas a partir do contexto recuperado $c(q)$:

$$F = \frac{|V|}{|S|}, \quad \text{onde } V = \{s \in S \mid c(q) \models s\}. \tag{3.1}$$

A *Context Relevance*, utilizada como métrica de precisão, penaliza a inclusão de informações redundantes no contexto. Ela é calculada pela razão entre o número de sentenças relevantes extraídas (S_{ext}) e o total de sentenças no contexto ($c(q)$):

$$\text{Context Relevance} = \frac{|S_{ext}|}{|\text{Total de Sentenças em } c(q)|}. \tag{3.2}$$

Para as métricas que dependem de referência, a *Accuracy* é definida por uma função de avaliação f_{judge}, executada por um *LLM Judge* que atribui uma nota discreta à qualidade da resposta y em relação ao gabarito y_{gt}:

$$\text{Accuracy}(y, y_{gt}) = f_{\text{judge}}(q, y, y_{gt}) \in \{1, \ldots, 5\}. \tag{3.3}$$

O *Context Recall* verifica a proporção de sentenças da resposta de referência $S(y_{gt})$ que podem ser atribuídas ao contexto recuperado C:

$$\text{Context Recall} = \frac{|\{s \in S(y_{gt}) \mid C \models s\}|}{|S(y_{gt})|}. \tag{3.4}$$

Para medir a segurança contra alucinação, aplicada aos testes de rejeição negativa, são desconsideradas as métricas de saúde de *pipeline*, *i.e*, *Faithfulness*, *Precision* e

Recall. Nesses casos, um sistema ideal deve identificar o contexto como irrelevante e se recusar a responder, o que faria com que essas métricas pontuassem próximo de zero. Portanto, a avaliação de segurança é definida exclusivamente pela média da pontuação de *Accuracy*. Isso é possível pelo fato de que o LLM gerador é instruído a fornecer uma resposta de recusa padrão, e o *LLM Judge* foi instruído a atribuir nota 5 para essa recusa correta.

Esta abordagem analítica, ao decompor a avaliação nos eixos de qualidade factual, saúde do *pipeline* e segurança, permite uma análise dos trade-offs inerentes a cada arquitetura. A metodologia busca expor de forma transparente como um sistema pode, por exemplo, otimizar o *Context Recall* em detrimento do *Context Precision*, ou como uma arquitetura com alta acurácia factual em perguntas diretas pode falhar no pilar de segurança. A análise permite, assim, uma comparação direta mas de como cada sistema equilibra essas diferentes demandas.

Capítulo 4

Metodologia Experimental e Resultados

Este capítulo detalha a execução do *benchmark* proposto, apresentando a metodologia de construção do conjunto de dados, o *framework* de avaliação e o desenvolvimento das arquiteturas. O objetivo central é conduzir uma avaliação sistemática de arquiteturas de RAG em um domínio de natureza altamente dinâmica, utilizando o *corpus* de notícias esportivas desenvolvido para este fim.[1]

A exposição inicia-se com o detalhamento do *pipeline* de construção do conjunto de dados, incluindo as fases de coleta de dados, geração sintética de perguntas e a análise de sua composição. Em seguida, o capítulo descreve o desenvolvimento e a arquitetura de cada sistema avaliado: o *Baseline*, o *Naive* RAG, o *Advanced* RAG e o *Graph* RAG. Sequencialmente, é formalizada a metodologia de avaliação, explicando as métricas quantitativas de *pipeline* e a abordagem qualitativa do *LLM Judge*.

Finalmente, o capítulo apresenta e analisa os resultados detalhados da avaliação. Os dados coletados para os cenários de teste são consolidados, permitindo uma comparação quantitativa do desempenho factual, perguntas simples e multi-contexto, e da capacidade de rejeição negativa de cada abordagem.

[1] O código-fonte completo de todas as arquiteturas e o *pipeline* de dados está disponível em: <https://github.com/sylvininholol/RAGBenchmark>.

4.1 Conjunto de Dados

Um pré-requisito fundamental para a avaliação de sistemas RAG é a utilização de um conjunto de dados que não sofra da contaminação de conhecimento, *i.e.*, cujas informações não estejam presentes nos dados de treinamento dos LLMs avaliados. Conforme destacado por Chen et al. (2023), o uso de informações recentes é uma estratégia eficaz para mitigar esse viés, garantindo que o desempenho do modelo reflita sua capacidade de raciocinar sobre o contexto fornecido, em vez de recorrer ao seu conhecimento paramétrico. Para este fim, foi desenvolvido um *pipeline* de dados automatizado para construir um conjunto de dados focado no domínio de notícias esportivas.

4.1.1 Construção do Conjunto de Dados

A fonte de dados selecionada foi o portal Globo Esporte[2], um veículo de alta relevância e com um fluxo contínuo de publicações. Devido à ausência de uma API pública, a extração de dados foi realizada por meio de *web crawling*, em um processo multifásico realizado no dia 30 de setembro de 2025.

Inicialmente, foi desenvolvida uma rotina em Python utilizando a biblioteca Selenium para automatizar a navegação na página principal, que depende da interação do usuário para carregar seu conteúdo dinamicamente. Esta rotina simulou o comportamento de um usuário, executando rolagens de página sucessivas até o final do *feed*. Quando a simples rolagem não resultava em novo conteúdo, o elemento "Veja mais" era localizado e acionado, permitindo o carregamento de notícias mais antigas. Este ciclo de rolagem e clique foi repetido até que o elemento "Veja mais" não estivesse mais disponível ou o limite pré-definido de 25 interações fosse atingido, assegurando a captura de um volume substancial de notícias.

Uma vez que a página principal estava integralmente carregada, seu código-fonte HyperText Markup Language (HTML) foi extraído e analisado para identificar e coletar o Uniform Resource Locator (URL) de cada notícia individual. Em uma

[2]https://ge.globo.com/

segunda etapa, para obter o conteúdo completo de cada artigo, a rotina iterou sobre a lista de URLs coletadas. Requisições HyperText Transfer Protocol (HTTP) foram executadas para cada URL, utilizando um cabeçalho User-Agent para simular um navegador padrão. Esta abordagem, que não depende da renderização de JavaScript, mostrou-se mais eficiente para a coleta em massa dos textos dos artigos.

Finalmente, realizou-se a análise sintática da estrutura DOM de cada página para isolar o conteúdo textual relevante. Utilizando seletores baseados na marcação semântica do HTML, identificando, por exemplo, *tags* de parágrafo contidas nos elementos principais do artigo, foi possível segregar o texto da notícia de elementos ruidosos, como menus de navegação, publicidade e rodapés. Os dados estruturados resultantes, que incluem título, URL e texto limpo, foram armazenados em uma tabela em um banco de dados relacional, formando o *corpus* base utilizado neste trabalho.

O processo resultou na coleta de 210 notícias distintas. Este volume amostral foi dimensionado para garantir a diversidade temática necessária à validação, mantendo-se dentro da viabilidade financeira para a execução massiva das avaliações via LLMs proprietários. Os registros foram armazenados em uma tabela de um banco de dados PostgreSQL, formando o *corpus* base para a geração dos dados de avaliação. Com o *corpus* de notícias estabelecido, a etapa seguinte consistiu na geração de um conjunto de avaliação diversificado, projetado para testar diferentes capacidades dos sistemas RAG. O processo foi dividido em três fases de geração de perguntas e respostas utilizando o modelo de linguagem *GPT-3.5-turbo*.

A primeira fase focou na criação de perguntas simples cuja resposta está contida integralmente em um único documento. Para cada uma das 210 notícias do *corpus*, o modelo foi instruído a gerar três perguntas relevantes que pudessem ser respondidas apenas pelo conteúdo do texto-fonte. Subsequentemente, para cada pergunta, uma resposta de referência foi produzida utilizando exclusivamente o conteúdo do mesmo artigo. Este subconjunto de dados visa criar um ambiente controlado para avaliar o desempenho fundamental de cada arquitetura, que são a precisão da recuperação e a fidelidade da geração.

A segunda fase foi projetada para criar cenários de teste mais complexos, avaliando a capacidade de integração de informações. O processo primeiramente, usando um LLM, classifica as notícias por entidades, nesse caso, times de futebol, e, em seguida, cria 30 pares de documentos distintos. Para cada par, o modelo foi instruído com um *prompt* específico para gerar uma única pergunta que só pudesse ser respondida forçando a síntese e a combinação de informações de ambos os textos.

Formalmente, baseando-se no conceito de *Information Integration* proposto por Chen et al. (2023), define-se o cenário de necessidade conjunta. Seja um par de documentos $D = \{d_i, d_j\}$ que compartilham um assunto em comum, a pergunta q e a resposta de referência y são geradas de tal forma que a resposta seja dedutível apenas pela união dos contextos, satisfazendo a condição definida pela Equação 4.1:

$$(d_i \cup d_j) \models y \quad \wedge \quad d_i \not\models y \quad \wedge \quad d_j \not\models y. \tag{4.1}$$

Dessa forma, garante-se que o sistema RAG deve obrigatoriamente recuperar e sintetizar informações de ambas as fontes para atingir a resposta correta, validando a capacidade de integração.

Finalmente, uma terceira fase foi executada para compor o subconjunto de testes de rejeição negativa. O objetivo desta etapa foi gerar perguntas que, embora factíveis dentro do domínio esportivo, não pudessem ser respondidas pelo *corpus* de notícias. Para isso, o modelo foi instruído a gerar 15 perguntas plausíveis cujas respostas não estivessem contidas nos tópicos fornecidos, extraídas da base de conhecimento, por exemplo, sobre um evento fictício, uma estatística inventada ou um jogador não mencionado. Esta abordagem visa avaliar a capacidade do sistema RAG de se abster de responder quando a informação necessária não está disponível no contexto recuperado.

Como consolidação das etapas de coleta e geração sintética, o artefato final desenvolvido para este trabalho constitui um conjunto de dados heterogêneo, estruturado para avaliar diferentes dimensões de competência do sistema. A composição definitiva do conjunto de dados apresenta-se da seguinte forma:

1. **Corpus de Conhecimento:** 210 artigos de notícias esportivas na íntegra, constituindo a base documental para a recuperação;

2. **Perguntas de Contexto Único:** 630 pares de pergunta-resposta fundamentados em documentos específicos, destinados à avaliação de recuperação direta e precisão local;

3. **Perguntas Multicontexto:** 30 pares de pergunta-resposta que exigem a agregação de informações de diferentes documentos, destinados à avaliação da capacidade de síntese e recuperação complexa;

4. **Amostras Negativas:** 15 perguntas adversariais sem resposta na base documental, projetadas para testar os limites de segurança e a rejeição de alucinações.

A infraestrutura de dados foi projetada para suportar buscas de similaridade semântica de alto desempenho. Foi utilizado o PostgreSQL com a extensão pgvector[3], que habilita o armazenamento e a consulta de dados vetoriais. O processo de vetorização foi conduzido utilizando o modelo *text-embedding-3-small* da OpenAI[4]. Foram gerados *embeddings* para todas as perguntas e respostas de referência, que foram armazenados em tabelas dedicadas. Para otimizar as operações de busca, foram criados índices *Hierarchical Navigable Small Worlds (HNSW)* sobre as colunas de vetores, o que reduz drasticamente a latência das consultas. Feita a construção e geração do conjunto de dados, a subseção seguinte analisa a composição quantitativa e qualitativa dos dados resultantes, que servem como base para o *benchmark* das arquiteturas RAG.

4.1.2 Análise do Conjunto de Dados

A Figura 4.2 apresenta uma visão quantitativa do conjunto de dados final. O *corpus* é composto por 210 notícias distintas. A partir delas, foi gerado o conjunto de avaliação, totalizando 675 perguntas e 675 respostas de referência. Este conjunto

[3]<https://github.com/pgvector/pgvector>
[4]<https://openai.com/>

é dividido em três categorias, conforme a metodologia de construção: 630 perguntas simples, 30 perguntas de múltiplo contexto e 15 perguntas de rejeição.

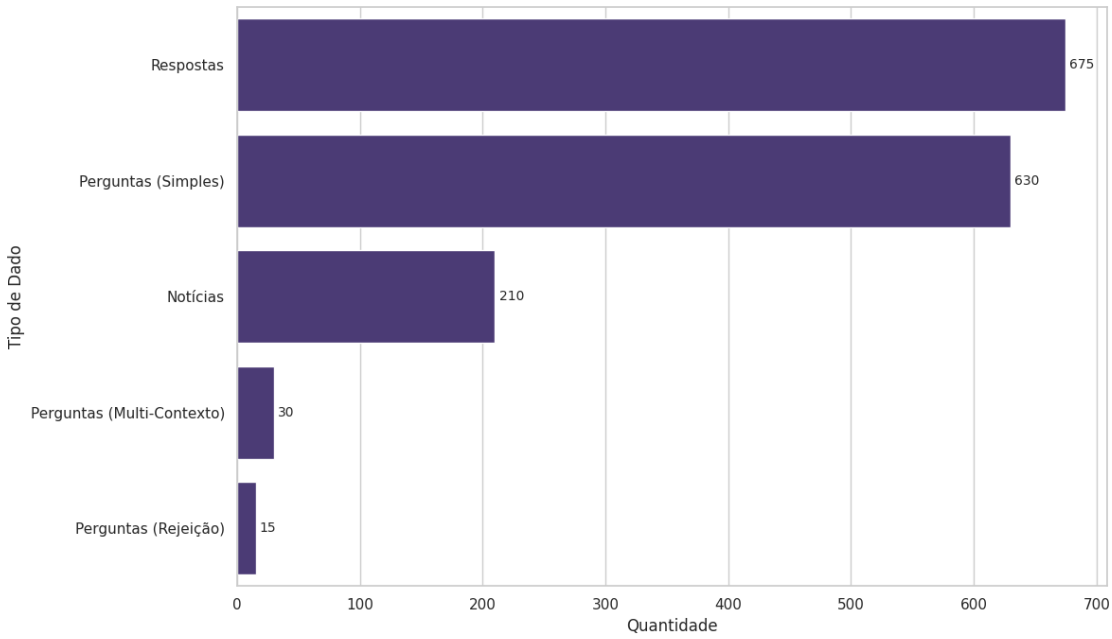

Figura 4.1: Quantidade total de registros no conjunto de dados gerado a partir da captura de notícias de um portal de esportes

Figura 4.2: Quantidade total de registros no conjunto de dados.

A Figura 4.3 demonstra que a maioria das 210 notícias possui entre 1.000 e 3.000 caracteres. Essa dimensão é suficientemente longa para conter informações factuais ricas, mas também justifica a necessidade de segmentação (*chunking*) implementada nas arquiteturas RAG.

As Figuras 4.4 e 4.5 mostram que as perguntas são, em sua maioria, curtas e diretas, com tamanho entre 50 e 150 caracteres, simulando consultas de usuários reais, enquanto as respostas de referência são concisas e factuais, com tamanho entre 100 e 300 caracteres.

A Figura 4.6 apresenta a distribuição das modalidades esportivas identificadas no conjunto de dados. A classificação de cada notícia foi realizada de forma automatizada, utilizando o modelo de linguagem *GPT-3.5-turbo* como classificador. O modelo foi instruído a analisar o título de cada artigo e identificar o esporte específico, escolhendo entre uma lista predefinida de modalidades, como futebol, basquete, tênis

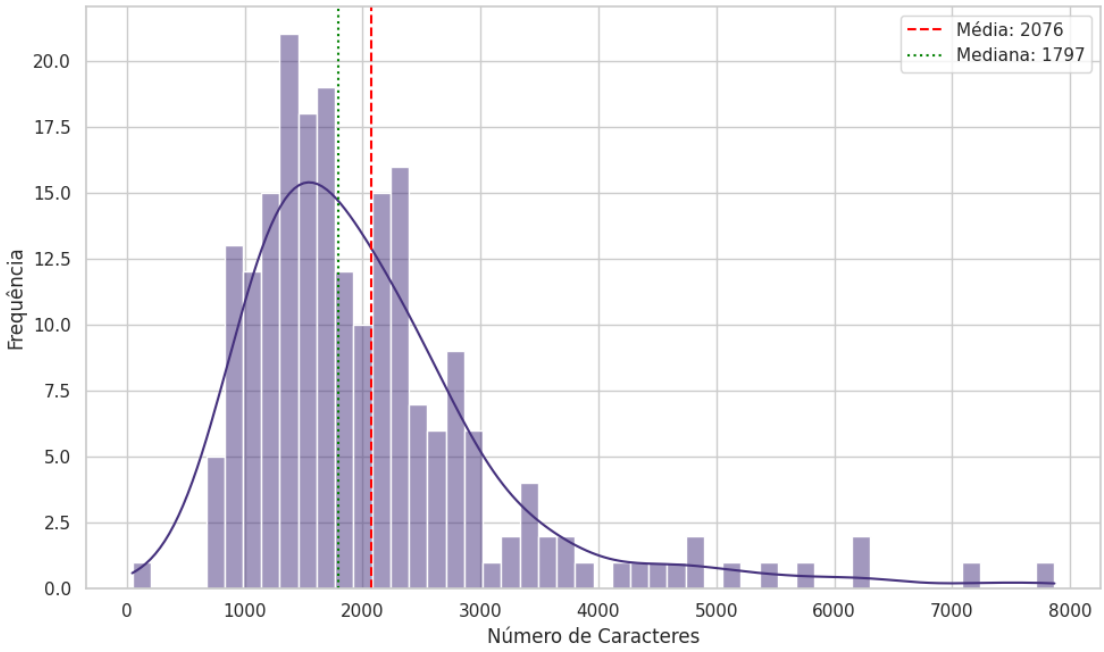

Figura 4.3: Histograma da distribuição do número de caracteres por notícia coletada no *corpus* base

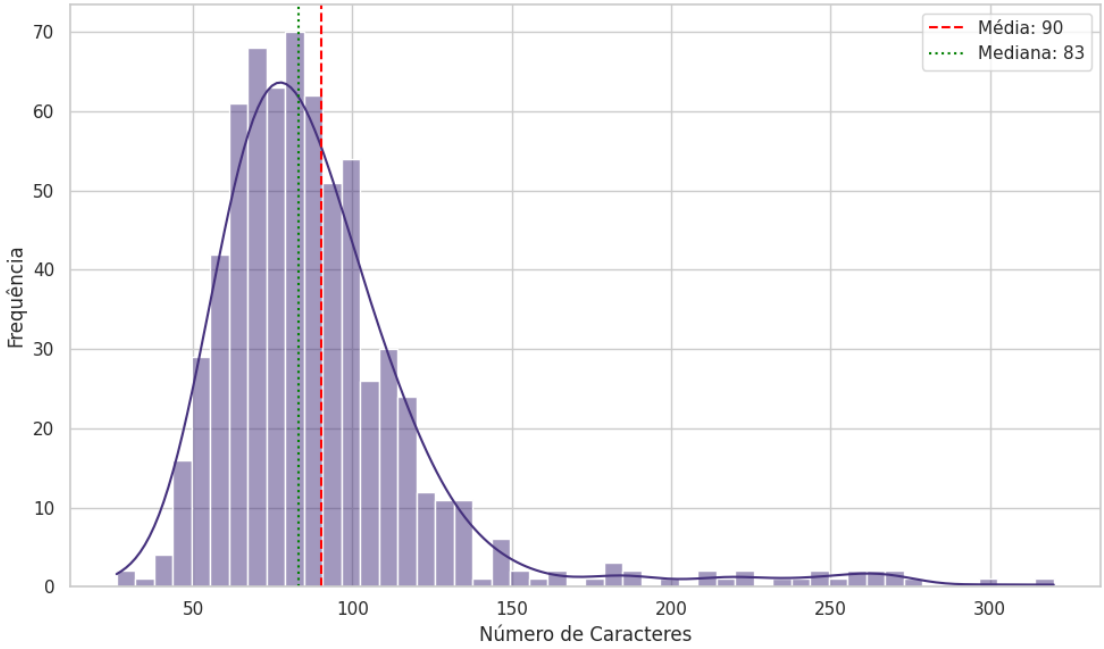

Figura 4.4: Histograma da distribuição do número de caracteres das perguntas geradas sinteticamente

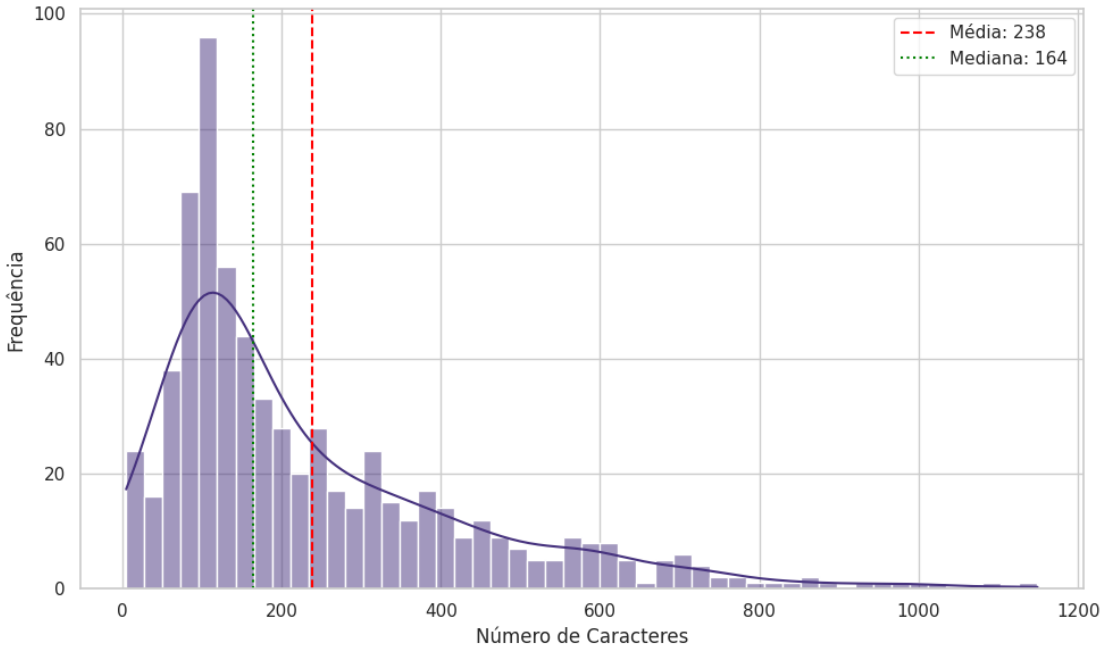

Figura 4.5: Histograma da distribuição do número de caracteres das respostas de referência (*ground-truth*)

e automobilismo. Embora mais de vinte esportes distintos tenham sido detectados pelo classificador, o gráfico mostra apenas o Top 6 esportes mais mencionados, seguido da categoria "Outros", para tornar a visualização mais clara e informativa. Essa categoria agrupa as demais modalidades com menos de quatro ocorrências, como Boxe, Futebol Americano, Vôlei, MMA, Judô, Surfe e Natação, entre outras.

Essa consolidação permite reduzir o ruído visual e destacar a forte predominância do futebol, que representa a maioria absoluta das notícias coletadas. Essa concentração é reforçada pela quantidade de notícias que mencionam times, vista na Figura 4.7, que destaca a proeminência dos clubes de futebol do Rio de Janeiro, com Flamengo e Botafogo liderando as menções.

Finalmente, a nuvem de palavras gerada a partir dos títulos das notícias, Figura 4.8, serve como uma síntese visual dos tópicos. Termos como "Fluminense", "Vasco", "Botafogo" e nomes de jogadores e competições relevantes, como "Gerson" e "brasileirão", são centrais, confirmando que o conjunto de dados em sua maioria é composto por notícias de futebol.

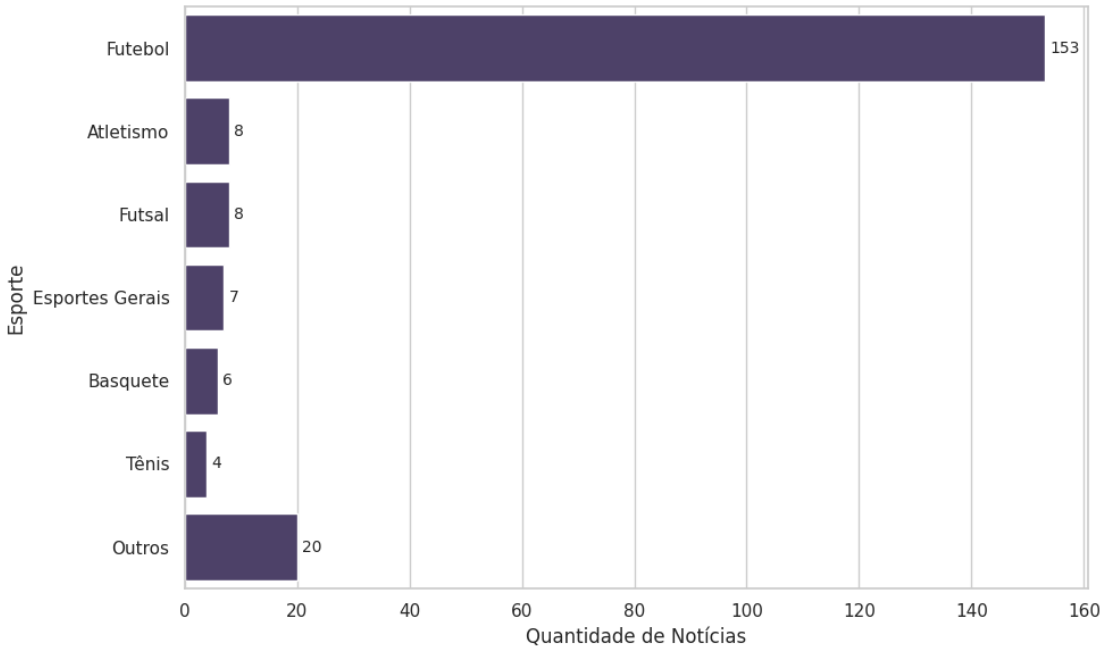

Figura 4.6: Frequência absoluta de notícias classificadas por modalidade esportiva identificada no conjunto de dados.

A análise revela um conjunto de dados com características alinhadas aos objetivos deste estudo. A alta especialização temática e a natureza factual e recente das notícias, visíveis na análise de tópicos, criam um cenário de teste focado em informações de nicho, que provavelmente não integram o conhecimento paramétrico do LLM sem RAG. Esta condição é metodologicamente fundamental para mitigar o risco de contaminação de conhecimento, forçando os sistemas RAG a dependerem do contexto recuperado. Dessa forma, o conjunto de dados viabiliza uma avaliação focada na capacidade dos sistemas de recuperar e raciocinar sobre informações fornecidas, em vez de apenas memorizar fatos pré-treinados.

4.2 Desenvolvimento das Arquiteturas RAG

Para a avaliação comparativa proposta neste trabalho, foram desenvolvidas quatro arquiteturas distintas. A primeira, *Baseline*, serve como controle experimental ao utilizar o LLM sem qualquer recuperação de contexto. As outras três são as arquiteturas *Naive* RAG, *Advanced* RAG e *Graph* RAG.

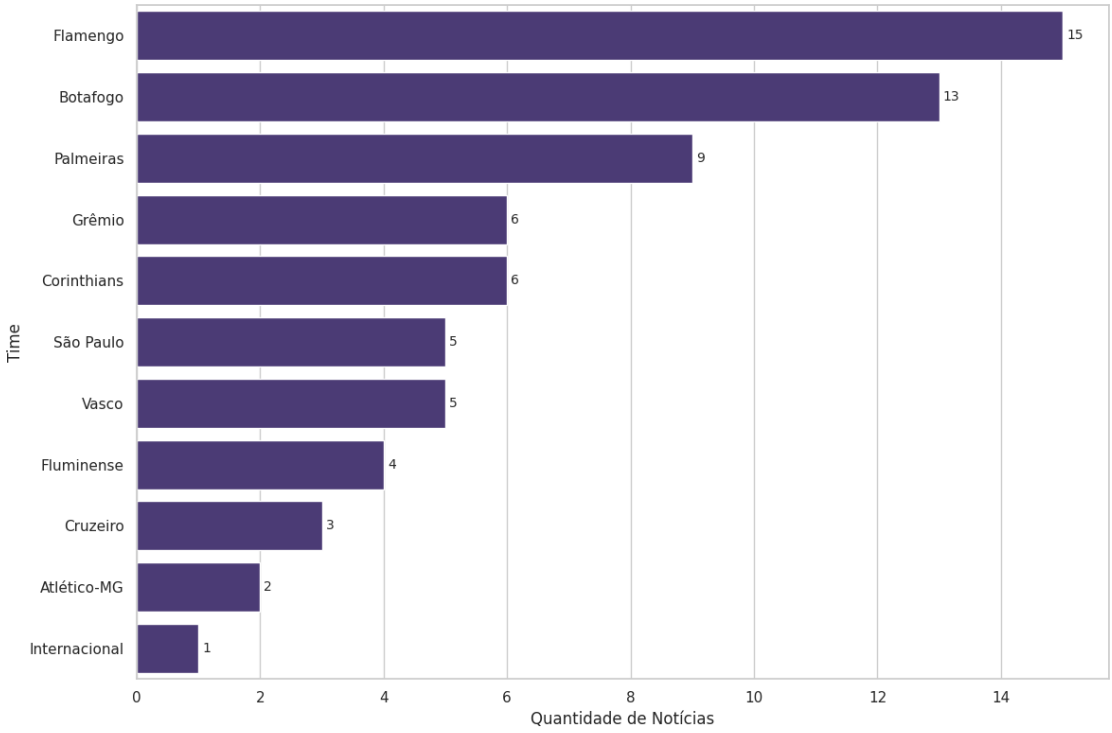

Figura 4.7: Frequência de menções aos principais clubes de futebol nos textos das notícias do *corpus*.

Para garantir uma comparação justa e controlada, as arquiteturas foram padronizadas em seu componente de geração textual. Todas as quatro arquiteturas utilizam o modelo *GPT-3.5-turbo* da OpenAI para a geração final de respostas, a escolha deste modelo fundamentou-se em sua relação custo-desempenho favorável, adequando-se às limitações orçamentárias do projeto sem comprometer a capacidade de raciocínio necessária para a tarefa. Adicionalmente, as três arquiteturas RAG utilizam o modelo *text-embedding-3-small* para a criação dos *embeddings* vetoriais, assegurando consistência na capacidade de compreensão semântica durante a recuperação.

A estrutura de cada arquitetura foi concebida de forma modular, isolando o componente de geração de resposta, constante em todas as abordagens, do componente de recuperação de contexto, que varia entre as arquiteturas RAG ou é omitido no *Baseline*. Esta modularidade permite que a estratégia de recuperação, o componente central que diferencia cada abordagem, seja substituída, modificada ou completamente removida, permitindo uma avaliação isolada do impacto da recuperação de

Figura 4.8: Nuvem de palavras gerada a partir da frequência de termos presentes nos títulos das notícias coletadas.

contexto, mantendo a geração textual constante.

Adicionalmente, o ambiente de execução e as dependências de cada arquitetura, incluindo seus respectivos bancos de dados, são gerenciados por meio do *Docker*, garantindo a reprodutibilidade dos experimentos e o isolamento dos ambientes. A seguir, são definidas as especificações e os detalhes de implementação de cada uma das quatro arquiteturas avaliadas neste trabalho, começando pela linha de base experimental.

4.2.1 Baseline: LLM sem Recuperação de Contexto

Para estabelecer um controle experimental e quantificar o ganho de desempenho introduzido pelas arquiteturas de RAG, foi testado um LLM sem recuperação de contexto para servir como linha de base. Esta abordagem representa o ponto zero de comparação, omitindo intencionalmente todo o processo de recuperação de informação e avaliando o desempenho do LLM em sua forma pura, utilizando apenas

seu conhecimento paramétrico.

Está arquitetura utiliza o mesmo modelo de geração das demais, assegurando que a capacidade de geração textual seja uma constante em todos os experimentos. O *pipeline* de execução, no entanto, é fundamentalmente distinto, pois a consulta do usuário não é pré-processada por um recuperador e nenhum contexto é recuperado de uma base de dados externa.

Em vez disso, a consulta é enviada diretamente ao LLM. A interação é gerenciada por um *prompt* que instrui o modelo a atuar como um assistente especializado em noticias de esporte, mas com uma restrição explícita: responder à pergunta baseando-se apenas em seu conhecimento prévio, *i.e.*, seu pré treinamento, e não inventar informações caso não saiba a resposta.

Esta configuração tem por objetivo isolar o conhecimento estático do modelo. As respostas geradas por esta arquitetura servem como o ponto de referência fundamental contra o qual a fidelidade, a relevância e a precisão das respostas aumentadas das arquiteturas *Naive*, *Advanced* e *Graph* RAG são comparadas, permitindo uma medição clara do impacto da recuperação de contexto.

4.2.2 Naive RAG: retrieve-then-read

A arquitetura *Naive* RAG representa a primeira e mais fundamental arquitetura RAG avaliada neste estudo. Esta arquitetura adere estritamente ao paradigma *retrieve-then-read*, no qual a resposta é gerada com base em um contexto dinamicamente recuperado de uma base de conhecimento vetorial. Como as demais, ela possui três fases principais: a indexação, a recuperação e a geração.

O processo de ingestão e indexação de dados constitui a primeira etapa da construção da base de conhecimento. Cada um dos 210 documentos do *corpus* de notícias é submetido a um processo de segmentação textual, *i.e.*, *chunking*. Para esta tarefa os textos divididos em fragmentos de 500 caracteres, com uma sobreposição estratégica de 50 caracteres entre eles. Essa técnica de sobreposição é fundamental para preservar a coesão semântica entre fragmentos adjacentes, mitigando o risco de

que informações contextuais relevantes, que se encontram nas fronteiras dos *chunks*, sejam perdidas.

Após a segmentação, cada *chunk* de texto é transformado em uma representação vetorial de alta dimensionalidade, um *embedding*, por meio do modelo de vetorização padrão. Os *chunks* e seus respectivos *embeddings* são então armazenados em um banco de dados.

Para garantir que a recuperação de informações seja computacionalmente viável e rápida, mesmo com um grande volume de vetores, foi utilizada uma otimização crucial na camada do banco de dados. Um índice do tipo HNSW foi criado sobre a coluna de *embeddings*. O HNSW é um algoritmo de busca aproximada de vizinhos mais próximos que organiza os vetores em uma estrutura de grafo multinível, permitindo uma redução drástica na complexidade computacional das buscas de similaridade em comparação com uma busca exaustiva, sendo essencial para aplicações em tempo real.

O mecanismo de recuperação é ativado quando uma nova consulta é submetida. Primeiramente, a consulta é vetorizada com o mesmo modelo de vetorização para garantir a consistência no espaço vetorial. Em seguida, uma busca de similaridade é executada no banco de dados para encontrar os k vetores de *chunks* mais próximos do vetor da consulta. A métrica de proximidade utilizada foi a distância Euclidiana, que identifica os fragmentos de texto semanticamente mais relevantes. Na etapa final, os k *chunks* recuperados são concatenados para formar um contexto coeso, que é inserido em um *prompt* estruturado e submetido ao LLM de geração. O *prompt* instrui o modelo a formular sua resposta baseando-se estritamente no contexto fornecido, buscando garantir a fidelidade da resposta à base de conhecimento.

4.2.3 Advanced RAG: Sumarização e Reordenação

O *Advanced* RAG foi projetado para aprimorar a qualidade do contexto recuperado, visando superar as limitações da busca vetorial simples, como a recuperação de informações relevantes, porém pouco específicas, um problema abordado por Liu et al. (2023). Esta arquitetura introduz um *pipeline* de recuperação multifásico que

implementa etapas de pré e pós-processamento: uma camada de sumarização para a busca inicial e um mecanismo de reordenação com um *Cross-Encoder* para refinar a seleção final dos *chunks*, atacando diretamente a precisão do contexto fornecido ao LLM.

A fase de indexação expande significativamente o processo da arquitetura *Naive* RAG. Embora a segmentação e a vetorização dos *chunks* de cada documento sejam mantidas, a sumarização é adicionada como etapa de pré-processamento. Para cada artigo do *corpus*, o LLM de geração é utilizado para gerar um resumo denso e semanticamente rico. Este resumo é então vetorizado com o modelo de vetorização padrão e armazenado em uma tabela dedicada no banco de dados.

A criação desses resumos vetorizados estabelece um pilar estratégico para a recuperação. Esta abordagem cria uma camada de abstração, um índice semântico de alto nível para os documentos. A ideia é que a busca usando resumos, que condensam a informação mais saliente de um texto, seja mais eficaz para identificar os documentos verdadeiramente relevantes para uma consulta. Isso contrasta com a busca direta em fragmentos de texto, que podem ser semanticamente próximos a uma consulta por conterem termos similares, mas que podem carecer do contexto geral para responder à pergunta de forma completa.

O processo de recuperação reflete essa estratégia em um *pipeline* de três estágios. Primeiramente, na fase de pré-recuperação, a busca por similaridade é realizada sobre os *embeddings* dos resumos e não dos *chunks*. Esta busca inicial atua como um filtro de alta eficiência, identificando um subconjunto dos documentos mais promissores. Em seguida, um conjunto inicial e mais amplo de *chunks* candidatos é recuperado exclusivamente dos documentos selecionados na etapa anterior. Essa estratégia de duas fases otimiza a busca, focando os recursos computacionais apenas nos documentos com maior probabilidade de conter a resposta.

A otimização mais significativa ocorre na etapa de pós-recuperação com a reordenação. Esta etapa atua sobre um conjunto ampliado de *chunks* candidatos, recuperados na fase anterior, antes da seleção final. Esses *chunks* candidatos são reordenados por um modelo *Cross-Encoder*, o *ms-marco-MiniLM-L-6-v2*. Diferen-

temente dos modelos *bi-encoder*, como o modelo de vetorização utilizado no *Naive* RAG, que calculam os vetores da consulta e do documento de forma independente, medindo apenas a similaridade de distância, o *Cross-Encoder* processa ambos simultaneamente. Ele recebe como entrada o par (*consulta*, *chunk*) e produz uma pontuação de relevância muito mais precisa, pois avalia a interação semântica direta entre os dois textos. Os *chunks* são então reordenados com base nessa pontuação, e os k mais relevantes são selecionados para compor o contexto final que será enviado ao LLM. Esta reordenação por *Cross-Encoder* é o que garante uma maior densidade de informação pertinente, uma vez que a precisão contextual é superior à simples medição de distância vetorial, minimizando a inclusão de ruído e maximizando a relevância factual do contexto.

4.2.4 Graph RAG: Modelagem com Grafo de Conhecimento

A arquitetura *Graph* RAG representa uma mudança paradigmática na modelagem da base de conhecimento, tratando a informação não como uma coleção de fragmentos de texto isolados, mas sim como uma rede de entidades e relacionamentos interconectados. A premissa desta abordagem é que o contexto estrutural e as conexões explícitas entre os conceitos, frequentemente perdidas na linearidade do texto, são cruciais para um raciocínio mais profundo. O objetivo é recuperar não apenas texto relevante, mas um subgrafo de conhecimento que represente as relações em torno da consulta do usuário.

A construção do grafo implementa um *pipeline* de ingestão de dados para um banco de dados de grafos *Neo4j*. O processo é aplicado individualmente a cada uma das 210 notícias que compõem o *corpus* de dados. Para cada um desses documentos, o LLM de geração é utilizado para realizar a extração de conhecimento. Guiado por um *prompt* com um esquema rigoroso de nós, e.g., *Pessoa*, *Organização*, e relacionamentos, e.g., *JOGA_EM*, *VENCEU*, definido na tabela 4.1, o LLM analisa o texto e retorna as informações extraídas em formato JavaScript Object Notation (JSON), servindo como um padrão para a construção do grafo.

O armazenamento de dados implementado no *Neo4j* é de natureza híbrida, arma-

Rótulos de Nós	Tipos de Relacionamento
Pessoa	TRABALHA_EM
Organizacao	JOGA_EM
Localizacao	TREINA
Evento	CONVOCADO_PARA
Competicao	TRANSFERIDO_PARA
Posicao	EMPRESTADO_POR
Conceito	PARTICIPOU_DE
	COMPETIU_EM
	VENCEU
	PERDEU_PARA
	CAMPEAO_DE
	VICE_CAMPEAO_DE
	MARCOU_GOL_EM
	DEU_ASSISTENCIA_PARA
	SOFREU_FALTA_DE
	LOCALIZADO_EM
	SEDE_DE
	PARTE_DE
	E_COMPANHEIRO_DE_EQUIPE_DE
	E_RIVAL_DE
	E_IDOLO_DE
	JOGA_NA_POSICAO

Tabela 4.1: Esquema do Grafo de Conhecimento

zenando tanto a informação estruturada extraída quanto o texto não estruturado original. O texto de cada documento é segmentado em *chunks* vetorizados, armazenados como nós *:Chunk*. A estrutura é enriquecida por múltiplas camadas de relações que conectam os dados de diferentes formas. Existem relações estruturais, como *:PARTE_DE*, que conectam um *chunk* ao seu documento; relações textuais, como *:PROXIMO_CHUNK*, que preservam a sequência lógica do conteúdo; e relações de conexão, como *:MENCIONA*, que vinculam um *chunk* às entidades que ele contém. As próprias entidades são interligadas por diversas relações de domínio, como *:JOGA_EM*, que representam os fatos extraídos do texto. Um índice vetorial é criado nos nós *:Chunk* para habilitar buscas de similaridade dentro do próprio ambiente do grafo.

O mecanismo de recuperação explora a estrutura conectada do grafo. A recuperação inicia com uma busca vetorial nos nós *:Chunk* para identificar os pontos de entrada mais relevantes. A partir desses *chunks*, uma consulta em linguagem *Cypher*

executa uma expansão de vizinhança. Esta consulta navega pelo grafo para coletar mais informações, incluindo o texto dos *chunks* vizinhos, as entidades mencionadas e os fatos, que são relacionamentos de primeiro grau, associados a essas entidades, construindo um subgrafo de conhecimento altamente contextualizado.

O contexto estruturado resultante, que combina texto e fatos do grafo, não é diretamente fornecido ao LLM final. Ele passa por uma etapa crucial de síntese. O LLM de geração é então utilizado para traduzir essa informação híbrida em um parágrafo coeso em linguagem natural. Este parágrafo sintetizado, que encapsula a riqueza semântica do grafo, serve como o contexto final e aprimorado para a etapa de geração da resposta, que então prossegue de maneira análoga às outras arquiteturas, garantindo a comparabilidade dos resultados.

Com a construção das quatro arquiteturas, o sistema de *benchmark* pode ser executado. A seção a seguir detalha o processo sistemático de execução dos experimentos, descrevendo como cada tipo de pergunta foi avaliado e seus resultados.

4.3 Resultados e Análise

A avaliação das quatro arquiteturas gerou um grande conjunto de dados, permitindo uma análise granular do desempenho de cada sistema. Os resultados são apresentados separadamente para os cenários de teste factuais e de segurança. Os valores representam a média e o desvio padrão da pontuação de cada resposta de cada métrica. O desvio padrão é crucial para entender a consistência; um valor alto indica que o desempenho da arquitetura foi volátil, enquanto um valor baixo sugere um comportamento mais previsível. A seguir são descritos os resultados do desempenho factual e da rejeição negativa.

4.3.1 Desempenho Factual (Perguntas Simples e Multi-Contexto)

Com essa configuração composta por 660 perguntas, 630 simples e 30 de integração, avalia a capacidade central das arquiteturas em responder a consultas factuais. A Tabela 4.3.1 compara o desempenho dos sistemas nesta tarefa.

Arquitetura	Accuracy $[1, 5]$	Faithfulness $[0, 1]$	Context Precision $[0, 1]$	Context Recall $[0, 1]$
Baseline LLM	1.4413 ± 0.7125	N/A	N/A	N/A
Naive RAG	2.7981 ± 1.6681	0.6337 ± 0.4477	0.8683 ± 0.3385	0.7502 ± 0.3919
Advanced RAG	2.7175 ± 1.6688	0.6146 ± 0.4577	0.9079 ± 0.2893	0.7682 ± 0.3748
Graph RAG	2.0952 ± 1.3286	0.3684 ± 0.4497	0.5921 ± 0.4918	0.3041 ± 0.4076

Tabela 4.2: Resultados para Perguntas Factuais
Nota: Valores apresentados no formato Média ± Desvio Padrão. N/A indica não aplicável.

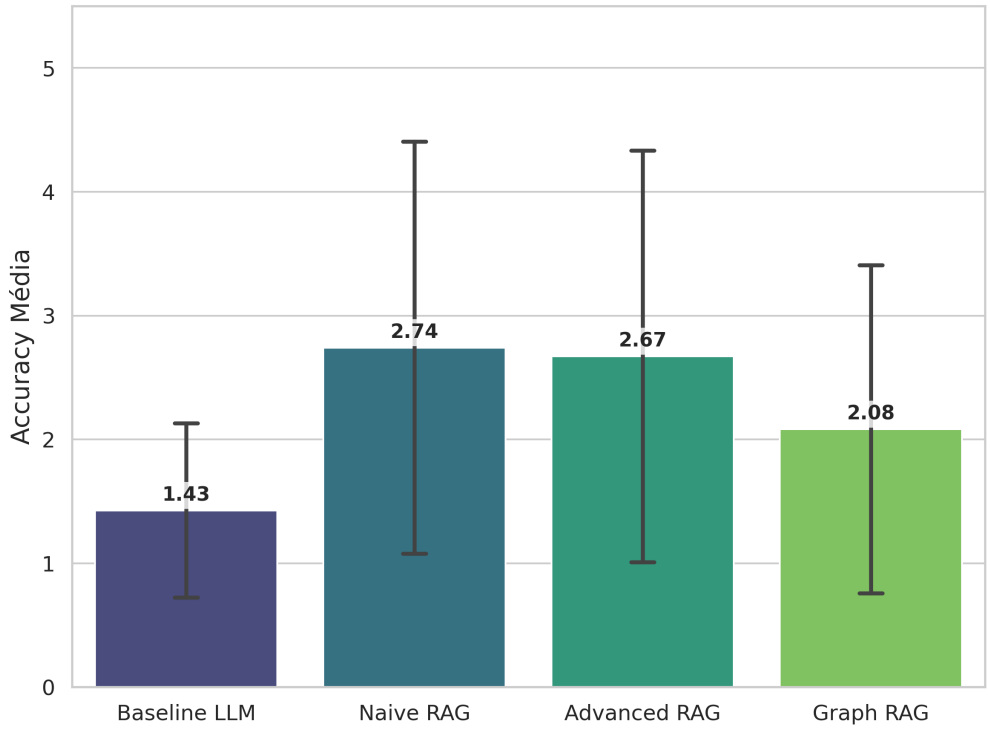

Figura 4.9: Média e desvio padrão da pontuação de cada resposta de *Accuracy* (1-5) atribuída pelo *LLM Judge* para cada arquitetura no cenário factual.

O *Baseline* LLM apresentou uma pontuação de qualidade de 1.4413, extremamente baixa, como esperado, fato visualizado na Figura 4.9. O desvio padrão de 0.7125 também é baixo, indicando que o modelo foi consistentemente incapaz de responder corretamente às perguntas sobre notícias recentes. Isso é corroborado pela Figura 4.10, que mostra sua distribuição de notas concentrada no valor 1, tendo 439 ocorrências. Isso valida a premissa do *benchmark* de que o conhecimento paramétrico do LLM é obsoleto para este domínio.

O *Naive* RAG foi o vencedor em *Accuracy* (2.7981), como ilustra a Figura 4.9, demonstrando que a simples implementação do *retrieve-then-read* produz o ganho mais substancial sobre o *Baseline*. Suas métricas de *pipeline*, visíveis nas Figuras 4.11

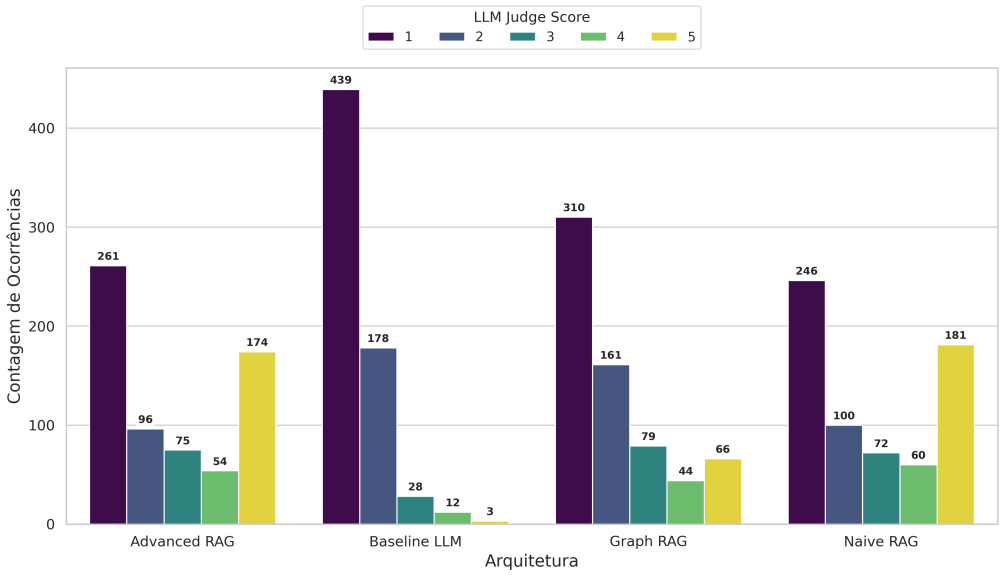

Figura 4.10: Contagem de ocorrências de cada nota de Accuracy (1 a 5) atribuída pelo LLM Judge para cada arquitetura no cenário factual. Este gráfico evidencia a distribuição real das pontuações.

e 4.12, são fortes: o *Context Recall* (0.7502) mostra que a busca vetorial simples foi eficaz em *encontrar* o contexto necessário, e o *Faithfulness* (0.6337) indica que o LLM utilizou esse contexto na maior parte do tempo. O alto desvio padrão na qualidade (1.6681), o maior de todos e notável nas barras de erro da Figura 4.9, sugere um desempenho polarizado. Essa polarização é ilustrada perfeitamente na Figura 4.10, que mostra uma distribuição acentuadamente bimodal, com a maioria das respostas concentrada nas notas 1 e 5, com 246 e 181 ocorrências, respectivamente.

O *Advanced* RAG, conforme visto na Figura 4.9, ficou marginalmente atrás do *Naive* RAG em qualidade de resposta (2.7175). Sua principal força reside na Precisão do Contexto (0.9079), a mais alta de todas, como destacam tanto a Figura 4.11 quanto a Figura 4.12, e com o menor desvio padrão (0.2893). Isso indica que o *Cross-Encoder* é um filtro de ruído excepcional. Contudo, o *Faithfulness* (0.6146) foi ligeiramente inferior ao do *Naive* RAG, como visto na Figura 4.11. Isso sugere que o *reranker*, ao ser muito agressivo para filtrar ruído, pode ter ocasionalmente descartado *chunks* que continham a resposta, levando o LLM a alucinar ou a gerar respostas incompletas. Isso também pode indicar que a sumarização pode ter causado perda de informação suficiente para evitar que os *chunks* fossem recuperados.

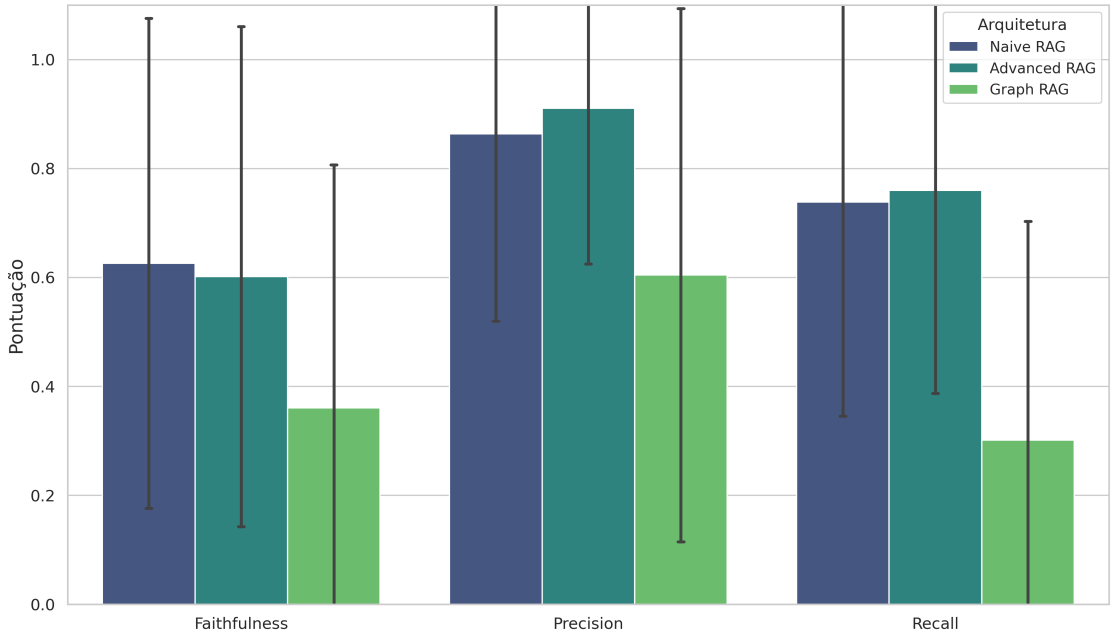

Figura 4.11: Comparativo da média e desvio padrão das métricas de saúde do *pipeline* (*Faithfulness*, *Precision* e *Recall*) entre as arquiteturas de RAG.

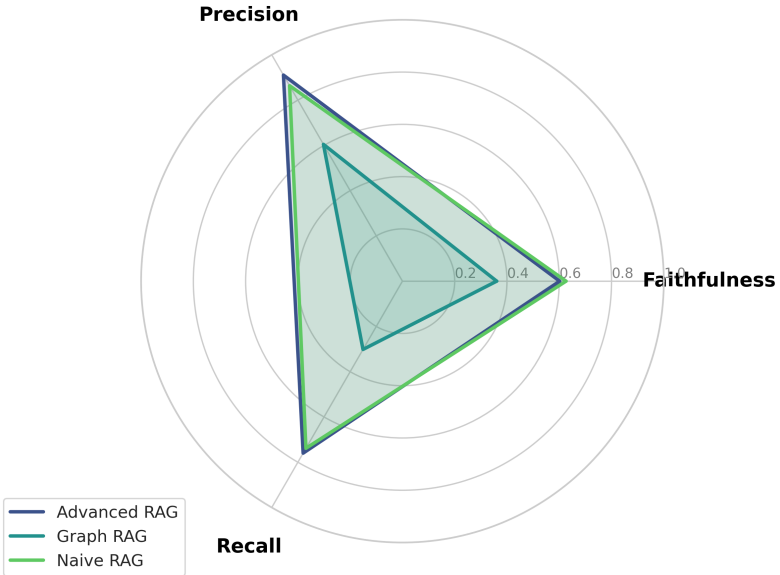

Figura 4.12: Gráfico de radar comparando o perfil de desempenho das arquiteturas nas três métricas de *pipeline*: *Faithfulness*, *Precision* e *Recall*.

O *Graph* RAG apresentou o desempenho factual mais fraco entre as arquiteturas RAG (2.0952), como mostra a Figura 4.9. As métricas de *pipeline*, visualizadas nas Figuras 4.11 e 4.12, revelam a causa da falha: o *Context Recall* (0.3041) e o *Faithfulness* (0.3684) são drasticamente baixos. O perfil de desempenho do *Graph* RAG na Figura 4.12 é visivelmente o menor. Uma hípotese para isso é uma falha no *retriever*, pois o sistema não consegue encontrar a informação correta e, como detalhado na Seção 4.2, a complexa etapa de síntese provavelmente distorce ou omite fatos cruciais, resultando em um contexto pobre.

4.3.2 Rejeição Negativa (Segurança)

Este cenário, com 15 perguntas, avalia a capacidade do sistema de se abster de responder quando nenhuma informação relevante é encontrada. Para este teste, as métricas de *pipeline*, como *Faithfulness*, *Context Precision* e *Context Recall*, são desconsideradas, pois um sistema ideal não deve encontrar contexto relevante. O foco é exclusivamente na pontuação do *LLM Judge*, onde uma média de 5.0 significa uma recusa perfeita.

Arquitetura	**Accuracy (LLM Judge)** $[1, 5]$
Baseline LLM	5.0000 ± 0.0000
Naive RAG	4.5333 ± 1.0601
Advanced RAG	5.0000 ± 0.0000
Graph RAG	4.4667 ± 1.4075

Tabela 4.3: Resultados para Testes de Rejeição Negativa (N=15)
Nota: Valores apresentados no formato Média ± Desvio Padrão.

Os resultados da Tabela 4.3.2 podem passar do limite máximo de cinco pois os resultados são variádos o suficiente para ter um desvio padrão alto e média está próxima do máximo, porém não há pontuação acima de cinco. O *Baseline* LLM e o *Advanced* RAG alcançaram um desempenho de segurança perfeito (5.0000 ± 0.0000). O *Baseline* foi perfeito porque, na ausência de contexto, seguiu sua instrução de *prompt* para não inventar respostas. O *Advanced* RAG foi perfeito porque seu *pipeline* de sumarização e reordenação foi sofisticado o suficiente para identificar o contexto recuperado como irrelevante e descartá-lo, fazendo com que o LLM executasse a

mesma regra de recusa do *Baseline*.

Em contraste, o *Naive* RAG e o *Graph* RAG mostraram vulnerabilidades significativas (4.5333 e 4.4667). Seus mecanismos de recuperação, mais simples ou mais complexos, não conseguiram reconhecer *chunks* que eram semanticamente similares às perguntas, mas factualmente irrelevantes. Isso fez com que o LLM tentasse formular respostas baseadas em ruído, resultando em alucinações e, consequentemente, em pontuações de segurança mais baixas e inconsistentes, como indicam os altos desvios padrão de 1.0601 e 1.4075.

4.3.3 Análise dos Resultados

Em síntese, os resultados demonstram que, embora a introdução de qualquer forma de RAG forneça um ganho substancial sobre o LLM puro, a complexidade adicional nem sempre se traduz em melhor desempenho factual. O *Advanced* RAG provou ser a arquitetura mais robusta e equilibrada. Embora tenha sido marginalmente inferior ao *Naive* RAG em desempenho factual, foi a única arquitetura de recuperação a alcançar a perfeição (5.0000 ± 0.0000) em segurança e rejeição de ruído.

Esta troca é ilustrada visualmente na Figura 4.13, que mostra o Desempenho Factual Médio (eixo-x) contra a Segurança Média (eixo-y). Nela, o *Advanced* RAG (Factual=2.7175, Segurança=5.0000) posiciona-se claramente como a arquitetura mais próxima da área ideal, de alto desempenho e alta segurança. Em contraste, o *Naive* RAG (Factual=2.7981, Segurança=4.5333) sacrifica a segurança para obter um ganho factual marginal, enquanto o *Baseline* LLM (Factual=1.4413, Segurança=5.0000) é perfeitamente seguro, mas factualmente inútil. O *Graph* RAG (Factual=2.0952, Segurança=4.4667) apresenta baixo desempenho em ambos os eixos.

O *Naive* RAG se destacou como o melhor em desempenho factual, mas sua alta inconsistência, com um desvio padrão de 1.6681, e sua vulnerabilidade a ruído em testes de rejeição (4.5333) o tornam menos confiável. O *Graph* RAG, por sua vez, indicou que a complexidade de sua implementação introduziu pontos de falha, provavelmente na etapa de síntese de contexto, que degradaram severamente o desempenho factual (2.0952), exigindo uma otimização mais refinada para superar

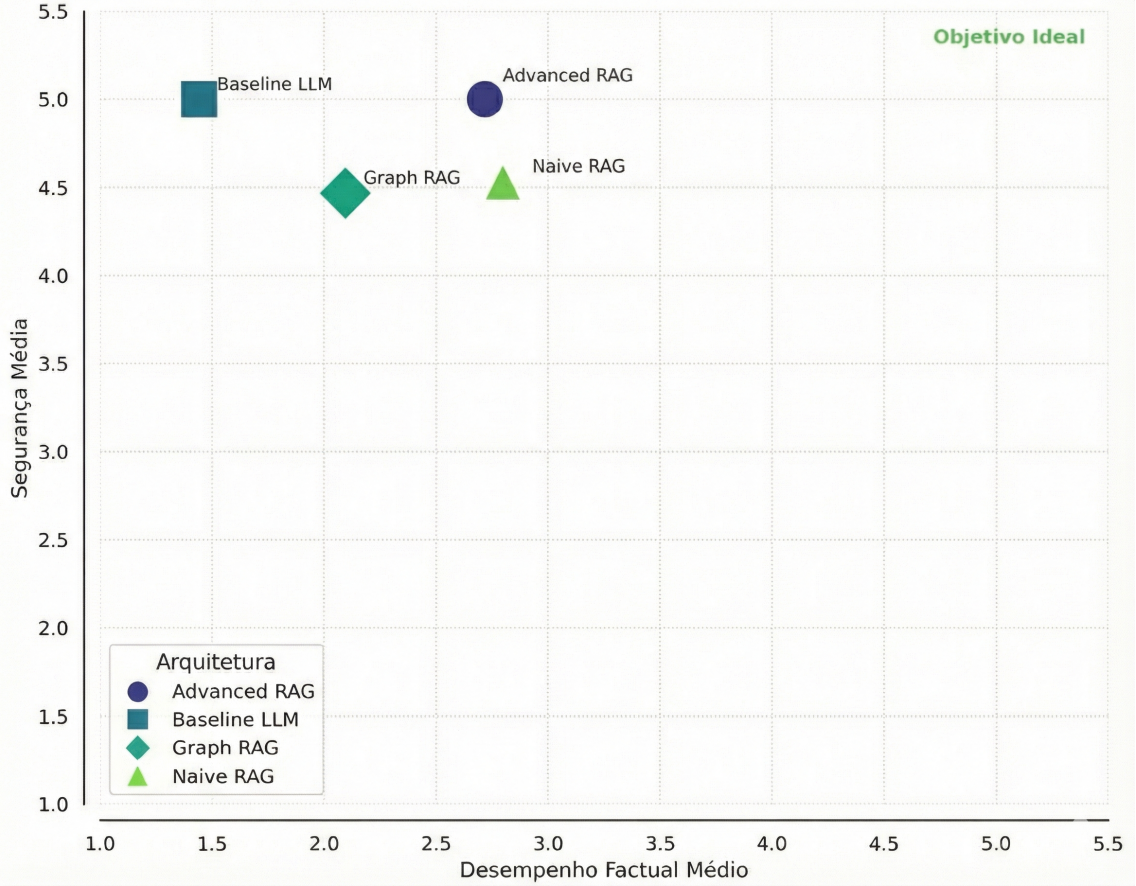

Figura 4.13: Análise de *trade-off* entre o Desempenho Factual vs. Segurança das arquiteturas.

as abordagens mais simples.

Capítulo 5

Conclusão

Este capítulo sumariza as principais descobertas do trabalho, discute as implicações dos resultados obtidos no *benchmark* proposto e apresenta as considerações finais. Adicionalmente, aborda as limitações do estudo e sugere direções para pesquisas futuras, com base nos desafios identificados durante a avaliação das arquiteturas de RAG no domínio de notícias esportivas.

5.1 Considerações finais

Este trabalho se propôs a avaliar sistematicamente o desempenho de arquiteturas RAG em domínios de alta volatilidade, utilizando notícias esportivas como caso de estudo. Os resultados sugerem que o conhecimento estático dos LLMs é insuficiente para este domínio, conforme evidenciado pelo baixo desempenho factual da arquitetura *Baseline*, com *Accuracy* média de 1.4413.

O desenvolvimento e o teste das arquiteturas RAG revelaram que a introdução do paradigma *retrieve-then-read*, *Naive* RAG, foi suficiente para um salto substancial no desempenho factual, alcançando uma *Accuracy* média de 2.7981, em relação ao *Baseline*, que obteve média de 1.4413. Este resultado evidencia a eficácia e a necessidade do RAG para o domínio proposto.

Os resultados demonstraram que a complexidade da arquitetura não possui uma

relação linear com o desempenho, evidenciando os riscos associados à implementação de sistemas complexos. A arquitetura *Graph* RAG apresentou o desempenho factual mais fraco, com uma *Accuracy* média de 2.0952. As métricas críticas de *Context Recall* (0.3041) e *Faithfulness* (0.3684) indicam que houveram falhas na implementação do *pipeline* de extração e consulta ao grafo. Esses erros de implementação são sintomáticos da alta complexidade técnica da abordagem: a necessidade de definir esquemas ontológicos rígidos e consultas estruturadas cria múltiplos pontos de falha que, se não perfeitamente executados, degradam severamente a recuperação.

Em contraste, a arquitetura *Advanced* RAG demonstrou ser a abordagem mais equilibrada e tecnicamente viável. Embora seu desempenho factual, com *Accuracy* média de 2.7175, tenha sido marginalmente inferior ao do *Naive* RAG, com *Accuracy* média de 2.7981, sua vantagem principal residiu na robustez da implementação. Esta foi a única arquitetura RAG a alcançar uma pontuação perfeita nos testes de rejeição negativa, provando que a complexidade moderada de seus componentes permite uma implementação mais segura e menos sujeita a erros operacionais do que o modelo em grafo.

Este trabalho conclui que, para este caso de uso, a viabilidade técnica favorece otimizações de menor complexidade de código, como a sumarização e a reordenação com *Cross-Encoder*. A abordagem *Graph* RAG mostrou-se menos viável pois sua alta complexidade de implementação aumenta a probabilidade de erros na codificação do sistema, tornando-a uma escolha arriscada em comparação a alternativas mais robustas.

5.2 Limitações e trabalhos futuros

Apesar dos resultados, este trabalho possui limitações que abrem caminhos para pesquisas futuras. A principal limitação reside no escopo e na estrutura do conjunto de dados. A coleta se concentrou em um único portal de notícias e, como revelado pela análise, foi dominada pelo futebol. Adicionalmente, o conjunto de avaliação possui uma distribuição desbalanceada entre os tipos de pergunta. Investigações

futuras devem expandir o *corpus* para incluir múltiplas fontes e uma variedade maior de esportes, bem como balancear a quantidade de perguntas em cada categoria de teste. Seria fundamental também conduzir testes longitudinais, utilizando notícias de datas diferentes, para simular ativamente a temporalidade do domínio e avaliar como cada arquitetura lida com a obsolescência de dados antigos.

Adicionalmente, o *benchmark* focou na qualidade da resposta, mas não avaliou sistematicamente o custo computacional. O custo associado à utilização das LLMs proprietárias, usadas como gerador e juiz, representou uma limitação prática. Este trabalho não pôde expandir o volume de testes ou o tamanho do conjunto de dados, pois o custo operacional se tornaria elevado. Um trabalho futuro relevante seria incluir essa análise de custo-benefício, avaliando o tempo de inferência e o custo de ingestão de dados de cada arquitetura, fatores críticos para domínios que exigem atualizações constantes.

Além disso, para *Graph* RAG, que teve um desempenho abaixo do esperado, investigações futuras poderiam explorar a geração de relacionamentos do tipo customizado entre nós em tempo de execução. Isso poderia aumentar a riqueza da informação, embora com um potencial aumento de complexidade. Por fim, sugere-se a avaliação do impacto de diferentes modelos de *embedding*, *Cross-Encoders* e LLMs, tanto geradores quanto juízes, no desempenho geral do *pipeline*, além da utilização de avaliadores humanos para validar a acurácia do LLM Judge.

Referências

AMUGONGO, L. M. et al. Retrieval augmented generation for large language models in healthcare: A systematic review. *PLOS Digital Health*, Public Library of Science, v. 4, n. 6, p. 1–33, 06 2025. Disponível em: <https://doi.org/10.1371/journal.pdig.0000877>.

ANISUZZAMAN, D. et al. Fine-tuning large language models for specialized use cases. *Mayo Clinic Proceedings: Digital Health*, v. 3, n. 1, p. 100184, 2025. ISSN 2949-7612. Disponível em: <https://www.sciencedirect.com/science/article/pii/S2949761224001147>.

ASAI, A. et al. *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. 2023. Disponível em: <https://arxiv.org/abs/2310.11511>.

CHEN, J. et al. *Benchmarking Large Language Models in Retrieval-Augmented Generation*. 2023. Disponível em: <https://arxiv.org/abs/2309.01431>.

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

ES, S. et al. *Ragas: Automated Evaluation of Retrieval Augmented Generation*. 2025. Disponível em: <https://arxiv.org/abs/2309.15217>.

FRIEL, R.; BELYI, M.; SANYAL, A. *RAGBench: Explainable Benchmark for Retrieval-Augmented Generation Systems*. 2025. Disponível em: <https://arxiv.org/abs/2407.11005>.

GAO, Y. et al. *Retrieval-Augmented Generation for Large Language Models: A Survey*. 2024. Disponível em: <https://arxiv.org/abs/2312.10997>.

GU, J. et al. *A Survey on LLM-as-a-Judge*. 2025. Disponível em: <https://arxiv.org/abs/2411.15594>.

HOLTZMAN, A. et al. *The Curious Case of Neural Text Degeneration*. 2020. Disponível em: <https://arxiv.org/abs/1904.09751>.

HUANG, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, Association for Computing Machinery (ACM), v. 43, n. 2, p. 1–55, jan. 2025. ISSN 1558-2868. Disponível em: <http://dx.doi.org/10.1145/3703155>.

LEWIS, P. et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. Disponível em: <https://arxiv.org/abs/2005.11401>.

LIU, N. F. et al. *Lost in the Middle: How Language Models Use Long Contexts*. 2023. Disponível em: <https://arxiv.org/abs/2307.03172>.

MA, X. et al. *Query Rewriting for Retrieval-Augmented Large Language Models*. 2023. Disponível em: <https://arxiv.org/abs/2305.14283>.

NOGUEIRA, R.; CHO, K. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

RADFORD, A. et al. Language models are unsupervised multitask learners. 2019.

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

SOUDANI, H.; KANOULAS, E.; HASIBI, F. Fine tuning vs. retrieval augmented generation for less popular knowledge. *arXiv preprint arXiv:2403.01432*, 2024.

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