UNIVERSIDADE FEDERAL DE GOIÁS / INSTITUTO DE INFORMÁTICA

# Recuperação Híbrida em Retrieval-Augmented Generation (RAG)

## Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos

Thiago Pedroso de Jesus

UNIVERSIDADE FEDERAL DE GOIÁS (UFG)
INSTITUTO DE INFORMÁTICA (INF)

THIAGO PEDROSO DE JESUS

# Recuperação Híbrida em Retrieval-Augmented Generation

Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# TERMO DE CIÊNCIA E DE AUTORIZAÇÃO PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO NO REPOSITÓRIO INSTITUCIONAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio do Repositório Institucional (RI/UFG), regulamentado pela Resolução CEPEC no 1240/2014, sem ressarcimento dos direitos autorais, de acordo com a Lei no 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo dos Trabalhos de Conclusão dos Cursos de Graduação disponibilizado no RI/UFG é de responsabilidade exclusiva dos autores. Ao encaminhar(em) o produto final, o(s) autor(a)(es)(as) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

**1. Identificação do Trabalho de Conclusão de Curso de Graduação (TCCG)**

Nome(s) completo(s) do(a)(s) autor(a)(es)(as): THIAGO PEDROSO DE JESUS

Título do trabalho: Recuperação Híbrida em Retrieval-Augmented Generation

Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos

**2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento** [ X ] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante: a) consulta ao(à)(s) autor(a)(es)(as) e ao(à) orientador(a); b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo do TCCG. O documento não será disponibilizado durante o período de embargo.

**Casos de embargo:**

- Solicitação de registro de patente;

- Submissão de artigo em revista científica;

- Publicação como capítulo de livro.

**Obs.: Este termo deve ser assinado no SEI pelo orientador e pelo autor.**

Documento assinado eletronicamente por **Thiago Pedroso De Jesus**, **Discente**, em 05/02/2026, às 15:34, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Fernando Marques Federson**, **Professor do Magistério Superior**, em 13/03/2026, às 11:45, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5957175** e o código CRC **302F3DD0**.

THIAGO PEDROSO DE JESUS

# Recuperação Híbrida em Retrieval-Augmented Generation

Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Orientador: Prof. Dr. Fernando Marques Federson

Goiânia

2025

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema de Bibliotecas da UFG.

JESUS, THIAGO PEDROSO DE
Recuperação Híbrida em Retrieval-Augmented Generation [manuscrito]: Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos / THIAGO PEDROSO DE JESUS. - 2025.
140 f.: 2025

Orientador: Prof. Dr. Fernando Marques Federson
Trabalho de Conclusão de Curso (Graduação ) - Universidade Federal de Goiás, Instituto de Informática (INF), Inteligência Artificial, Goiânia, 2025.

1. Inteligência Artificial. 2. Retrieval-augmented Generation. 3. Recuperação Híbrida.

I. Federson, Fernando Marques , orient. II. Título.

CDU 004

THIAGO PEDROSO DE JESUS

**Recuperação Híbrida em Retrieval-Augmented Generation**

Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos

Relatório final de Trabalho de Conclusão de Curso, apresentado à Universidade Federal de Goiás, como parte das exigências para a obtenção do título de Bacharel em Inteligência Artificial.

Data da Aprovação: 09 de dezembro de 2025.

Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)

Prof. Dr. Aldo André Díaz Salazar
Coordenador de TCC do BIA (INF-UFG)

Prof. Dr. Anderson da Silva Soares
Coordenador do BIA (INF-UFG)

Daniel Fazzioni
(INF-UFG)

THIAGO PEDROSO DE JESUS

# Recuperação Híbrida em Retrieval-Augmented Generation

Desenvolvimento de Modelo para Atribuição Dinâmica de Pesos


## RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Retrieval-Augmented Generation (RAG)**. Uma ilustração e sua narrativa descrevem os períodos de trabalho. Os Apêndices contêm os Termos de Aceite de Entrega e os resultados obtidos durante cada período de trabalho.


Palavras-chave: Inteligência artificial; Retrieval-augmented generation; Recuperação híbrida.


## ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Retrieval-Augmented Generation (RAG)**. An illustration and its narrative describe the work periods. The Appendices contain the Delivery Acceptance Terms and the results obtained during each work period.


Keywords: Artificial intelligence; Retrieval-augmented generation; Hybrid recovery.

Goiânia

2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

# Minha Jornada


Thiago Pedroso de Jesus

Especialista em: Retrieval-Augmented Generation (RAG)

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

# MINHA JORNADA

**Nome:** **Thiago Pedroso de Jesus**

**Especialidade:** **Retrieval-Augmented Generation (RAG)**

## Objetivo deste documento

Durante o processo da disciplina Residência em IA[1], foram gerados diversos resultados na construção da minha especialização. A cada semana, um conjunto de resultados foi formalizado por um Termo de Aceite de Entrega e avaliado por uma banca, considerando o planejado e o realizado para o período. Este documento tem como objetivo descrever esses resultados obtidos, fazendo referência aos Termos de Aceite de Entrega e seus documentos associados.

## Minha Jornada

O início da minha jornada, nas **Semanas 1 e 2**, foi um momento de convergência dos interesses que desenvolvi ao longo da graduação. Apesar de ter explorado diversas frentes da IA, foi no Processamento de Linguagem Natural que encontrei minha maior motivação, especialmente pela forma como a linguagem pode ser transformada em estruturas vetoriais capazes de representar significados e relações. As aplicações de busca vetorial sempre me chamaram atenção, tanto pela utilidade prática quanto pela profundidade histórica que eu ainda não conhecia. Essa combinação de fatores me levou a definir o *Retrieval-Augmented Generation (RAG)* como tema da minha especialização, decisão detalhada no **Apêndice 1**. A partir disso, me dediquei ao estudo dos fundamentos históricos da Recuperação de Informação e à construção de uma primeira taxonomia da área, processo que resultou em uma linha do tempo e em um mapa mental que serviram como base para toda a minha trajetória. Esses materiais estão registrados no **Apêndice 2**.

[1] Dez Semanas, entre setembro de 2025 e dezembro de 2025.

Durante o período das **Semanas 3 e 4**, meu foco se voltou para o aprofundamento técnico sobre a área de *Retrievers* (Recuperação). Aproveitando um material de revisão que eu já havia produzido na disciplina de PLN do Bacharelado, pude revisitar a evolução dos modelos de representação vetorial e utilizá-la como guia para entender como os algoritmos de recuperação foram se sofisticando ao longo do tempo. Assim, investiguei desde métodos esparsos clássicos, como *TF–IDF* e *BM25* (e suas variações), até as arquiteturas de *embeddings* que sustentam a busca densa, mapeando como cada abordagem influencia a recuperação em diferentes cenários, conforme registrado no **Apêndice 3**. Em seguida, busquei conectar essa base conceitual à prática de engenharia de dados, estudando estratégias de *chunking* e indexação e relacionando-as a diferentes tipos de fontes. Esse estudo mais "aplicado", descrito no **Apêndice 4**, foi fortemente inspirado por casos de uso reais, como a indexação de código do Cursor, e contribuiu para expandir minha visão sobre como segmentar, indexar e recuperar informações de forma eficiente.

As **Semanas 5 e 6** foram cruciais para compreender o "outro lado da moeda" de um sistema RAG: a Geração e a Avaliação. Estudando os desafios da área, deparei-me com o problema do "*Lost in the Middle*", que explicita a dificuldade dos *Large Language Models* (*LLMs)* em lidar com informações posicionadas no meio de contextos muito longos. Esse desafio me levou a investigar com mais profundidade a área de *re-ranking*, entendendo desde os primórdios do *text ranking* até o uso de modelos como *BERT* e variantes (*monoBERT*, *duoBERT*) para reordenação de passagens, estudos que descrevo no **Apêndice 5**. Para fechar esse ciclo teórico, percebi que não bastava saber recuperar e gerar: eu precisava entender como avaliar a qualidade dessas soluções. Assim, descrevo no **Apêndice 6** um estudo completo de métricas de Recuperação de Informação e de PLN, de *benchmarks* e *datasets* amplamente utilizados, de métodos de avaliação mediados por *LLMs* e de *frameworks* práticos como o *RAGAS*, além da revisão de modelos importantes para *ranking*.

Nesse ponto da minha jornada, percebi que era o momento de começar a enxergar o *RAG* de forma mais integrada e aplicada. Portanto, na **Semana 7**, busquei a convergência dos conhecimentos adquiridos por meio do estudo de *frameworks* que unem Recuperação e Geração para compor fluxos de trabalho completos de *RAG*. Motivado por esses materiais,

retomei todos os artigos que havia estudado até então, organizei-os em uma tabela por área e tese principal, e finalizei o mapa mental que vinha construindo desde o início da Residência, usando-o como uma consolidação dos meus estudos na área. Foi neste processo que, ao tabular e revisar os trabalhos, tomei a decisão de me aprofundar na área de Recuperação Híbrida, selecionando o artigo "*Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models*" como ponto de partida para os experimentos. O processo de seleção, a síntese dos *frameworks* estudados e a consolidação do mapa mental final estão apresentados no **Apêndice 7**.

A partir das **Semanas 8 e 9**, a jornada tornou-se, de fato, prática e experimental. Inicialmente, foquei na replicação do artigo escolhido, implementando a recuperação híbrida em duas fases, com destaque para a primeira etapa de busca tri-modal, que combina *retrievers* esparsos, densos e baseados em grafos. Utilizando os mesmos conjuntos de dados do trabalho original, obtive resultados pré–*re-ranking* próximos aos relatados pelos autores, conforme descrevo no **Apêndice 8**. No entanto, ao avançar para a etapa de fusão vetorial e *re-ranking* com *LLMs*, encontrei dificuldades para reproduzir os resultados do artigo, em parte pela ausência de detalhes importantes na implementação descrita pelos autores. Essas limitações me motivaram a expandir o escopo dos experimentos e a incorporar a metodologia do artigo "*DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation*", que propõe um ajuste dinâmico dos pesos de fusão entre modalidades. A partir daí, implementei o mecanismo de atribuição dinâmica de pesos, construí um ambiente modular em código para testar diferentes combinações de *retrievers*, estratégias de normalização e esquemas de fusão e passei a documentar esses experimentos de forma sistemática. Essa migração para a abordagem dinâmica gerou resultados coerentes com os relatados pelo artigo, conforme documentado no **Apêndice 9,** e me deu confiança para avançar para uma contribuição própria.

A **Semana 10** foi especialmente importante para a minha jornada, pois marcou o momento em que pude ir além da replicação de trabalhos e desenvolver uma contribuição própria. Ao perceber que os melhores resultados do método *DAT* dependiam de modelos proprietários (como o *GPT-4o*) para decidir quanto peso cada modalidade de *retriever* deveria receber, formulei a pergunta que guiou esse período: "será possível obter um

desempenho competitivo utilizando um modelo *open-source* ajustado especificamente para essa tarefa?". Para explorar essa hipótese, utilizei o *dataset SQuAD 1.1* para construir um conjunto de dados no qual, para cada consulta, os pesos atribuídos ao *BM25* e ao *retriever* denso eram derivados automaticamente de suas posições nos rankings, permitindo treinar um modelo capaz de aprender essa decisão. Em seguida, realizei o *Fine-tuning Supervisionado* de um modelo *Llama-3.1-8B* e comparei seu desempenho com *baselines* de pesos fixos e com versões que utilizavam modelos proprietários. Dessa forma, observei que o meu modelo treinado não apenas superou os pesos fixos, como também ultrapassou o modelo proprietário em cenários específicos e em subconjuntos mais sensíveis à combinação híbrida. A documentação completa desse processo encontra-se no **Apêndice 10**, com uma contribuição que considero significativa para a área de recuperação híbrida de informação em *RAG*.

Ao final da jornada, aprendi que toda área de conhecimento é, por natureza, infinita e que, à medida que avançamos, novas camadas, conexões e possibilidades surgem. Nesse sentido, o processo da Residência me ensinou a organizar o conhecimento sem me perder e a seguir uma linha de estudo clara e consistente. Hoje, sinto segurança em afirmar que me tornei especialista em *RAG*, mas reconheço que a conquista mais valiosa foi desenvolver a capacidade de me especializar em qualquer outro campo daqui em diante. Por isso, gostaria de expressar meu agradecimento aos professores da Residência em IA, cujas orientações e ensinamentos foram fundamentais em cada etapa da minha jornada, e também aos meus amigos, que me acompanharam ao longo de toda a graduação, me ofereceram suporte nos momentos de maior desafio e contribuíram diretamente para minha formação pessoal e profissional.

# APÊNDICE 1

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 4 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante a primeira Semana, realizei as seguintes atividades:

- Definição do tema da Residência: **Retrieval-Augmented Generation (RAG).**
- Classificação do tema de acordo com os tópicos do Congresso CSCE'25:
  - Information and Knowledge Retrieval and Searching Algorithms (ACC)
  - Natural Language Processing (ICAI)
  - Knowledge Representation (ICAI)

No documento abaixo, descrevo a minha motivação, processo de escolha e classificação do tema:
Residência - Escolha do tema

- Estudo da história da recuperação de informação, com foco em dois trabalhos:
  - Ensaio "As We May Think" de Vannevar Bush;
  - Sistema de Recuperação de Informação SMART, de um grupo liderado por Gerard Salton.

O estudo da história está descrito no documento abaixo:
Residência - História da recuperação de informação

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Com o objetivo de finalizar meu estudo da história da recuperação de informação, eu pretendo:

- Finalizar a leitura de uma das principais publicações de Gerard Salton: A vector space model for automatic indexing;
- Concluir a trajetória histórica do tema até a primeira aparição do termo "Retrieval-Augmented Generation".

Por fim, eu gostaria de criar uma taxonomia da minha área de pesquisa para direcionar estudos futuros:

- Explorar artigos que classificam a área, como por exemplo: [2402.19473] Retrieval-Augmented Generation for AI-Generated Content: A Survey;
- Construir um mapa mental para direcionar meu estudo.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Processo de escolha do tema

## A MINHA MOTIVAÇÃO

Durante minha graduação em Inteligência Artificial, tive a oportunidade de explorar diversas áreas da IA. No entanto, a que mais despertou meu interesse foi o NLP, principalmente com os avanços na pesquisa de LLMs e dos resultados surpreendentes que essa frente vinha alcançando. Meus primeiros estudos em processamento de linguagem natural foram direcionados ao entendimento de como a linguagem poderia ser modelada. Desde o início, me chama muito a atenção o fato de ser possível traduzir algo tão complexo quanto a linguagem natural em representações numéricas, abrindo o caminho para que os computadores pudessem lidar com significados, contextos e relações semânticas.

Mais fascinante ainda foi perceber que, a partir dessas representações vetoriais, torna-se viável construir mecanismos capazes de identificar similaridades semânticas e, com isso, recuperar informações relevantes de grandes volumes de dados. Essa capacidade de buscar, entre milhões de possibilidades, aquilo que melhor responde a uma consulta, não apenas imita o processo humano de associação de ideias, como também amplia a eficiência na exploração e organização do conhecimento.

A partir desse entendimento, ficou claro que os fundamentos de indexação, representação e recuperação da informação são indispensáveis para os avanços das soluções em NLP. E foi justamente nesse ponto que encontrei no RAG (Retrieval-Augmented Generation) uma síntese desses conceitos.

Um exemplo particularmente inspirador para mim está no funcionamento do Cursor, uma plataforma que aplica esses princípios no contexto de desenvolvimento de código. Segue o link para a matéria que mais me despertou interesse nessa área: How Cursor Indexes Codebases Fast - by Engineer's Codex

Fazendo a leitura, percebi que existe uma riqueza conceitual e profundidade teórica que sustenta esse tipo de solução. No caso do Cursor, por exemplo, sua aplicação de RAG exige o domínio de fundamentos que vão desde estruturas de dados

distribuídas, como árvores de Merkle usadas para garantir integridade e rastreabilidade de mudanças, até técnicas avançadas de segmentação de código (chunking) e embeddings treinados para capturar nuances semânticas em ambientes altamente técnicos. Além disso, a busca semântica envolve o uso de índices vetoriais otimizados, métricas de similaridade sofisticadas e estratégias de re-ranking, que demandam conhecimento aprofundado em recuperação da informação. Essa complexidade revela como inovações como o Cursor se apoiam em décadas de pesquisa acadêmica e engenharia de sistemas, exigindo o trabalho de um especialista.

## EXPLORANDO O CONGRESSO CSCE'25

Na disciplina de residência em Inteligência Artificial, tivemos a oportunidade de buscar inspirações nos tópicos do Congresso CSCE'25 (The 2025 World Congress in Computer Science, Computer Engineering, & Applied Computing). Fazendo a leitura, destaco as seguintes áreas que me chamaram mais atenção e, de alguma forma, vão ao encontro com o tema que eu tinha em mente:

### ACC'25 - The 9th International Conference on Applied Cognitive Computing

- Information and Knowledge retrieval and searching algorithms

- Knowledge Representation and Classification Systems

- Big Data Fusion and Information Retrieval

### ICAI'25 - The 27th International Conference on Artificial Intelligence

- Natural language processing

- Knowledge discovery

- Knowledge representation

- Knowledge acquisition

- Knowledge networks and management

- Knowledge acquisition and discovery techniques

- Aspects of knowledge structures

- General Structure-based approaches in information retrieval, web authoring, information extraction, and web content mining

## EXPLORANDO TÓPICOS RELACIONADOS

Em um primeiro momento, é necessário entender melhor os tópicos para entender quais fazem mais sentido para o meu campo de especialização. Para esse objetivo, fiz uma pesquisa rápida para criar uma descrição de cada tópico. Como não estou me aprofundando no momento, fiz a leitura rápida de sites e abstracts de artigos:

### Information and Knowledge Retrieval and Searching Algorithms.

Dedica-se ao estudo de como recuperar informação de forma eficiente e precisa em grandes coleções de dados. O foco está em modelos de ranqueamento e organização de índices que permitam alinhar documentos às necessidades informacionais de usuários ou sistemas. A área envolve tanto o desenvolvimento de algoritmos de busca quanto a análise de critérios de relevância e avaliação de desempenho.

Referências:
Compreendendo os algoritmos de busca de IA | Elastic Blog
Knowledge Retrieval and its ability to deliver more accurate and efficient interactions

### Knowledge Representation and Classification Systems.

Explora as formas de estruturar conhecimento de maneira que possa ser processado por sistemas computacionais. Isso inclui ontologias, taxonomias e grafos, que fornecem bases para organizar conceitos e relações. Em paralelo, sistemas de classificação utilizam essas estruturas para agrupar ou categorizar informações, criando um elo direto entre representação e tomada de decisão automatizada.

Referências:

[1506.04002] Knowledge Representation in Learning Classifier Systems: A Review
Knowledge Representation in AI - GeeksforGeeks

## Big Data Fusion and Information Retrieval.

Trata da integração de múltiplas fontes de dados (geralmente heterogêneas e massivas) em um processo de fusão que permite gerar visões mais consistentes e informativas. Essa combinação é especialmente relevante em contextos de recuperação de informação, onde sinais vindos de diferentes sistemas ou modalidades precisam ser consolidados em uma única resposta coerente.

Referências:
Definição de Big Data: Exemplos e benefícios | Google Cloud
A dynamic big data fusion and knowledge discovery approach for water resources intelligent system based on granular computing - ScienceDirect

## Natural Language Processing (NLP)

Estuda como sistemas computacionais podem compreender e gerar linguagem natural. A área combina aspectos de sintaxe, semântica e pragmática com métodos estatísticos e de aprendizado profundo, e se tornou fundamental para aplicações como tradução automática, análise de texto e interfaces conversacionais.

Referências:
O que é PLN (processamento de linguagem natural)? | IBM

## Knowledge Discovery

Foca no processo de extrair padrões relevantes e conhecimento a partir de grandes volumes de dados. Esse campo articula etapas de preparação, análise e interpretação, envolvendo desde métodos estatísticos até técnicas modernas de mineração de dados e aprendizado de máquina. Tivemos a oportunidade de estudar sobre esse tópico na disciplina “Mineração de dados”

Referências:
KDD and Data Mining

## Knowledge Representation

Concentra-se na forma como o conhecimento é modelado e formalizado para que sistemas inteligentes possam raciocinar sobre ele. Isso inclui o uso de linguagens formais, grafos e ontologias que permitem organizar informações de modo estruturado, equilibrando expressividade e eficiência computacional.

Referências:
Knowledge Representation - an overview | ScienceDirect Topics

## Knowledge Acquisition

Refere-se ao processo de obtenção de conhecimento, seja a partir de especialistas humanos, seja por meio de dados. Essa atividade pode incluir entrevistas, extração de padrões textuais ou aprendizado automatizado, sendo essencial para alimentar e expandir sistemas de representação e gestão de conhecimento.

Referências:
Knowledge Acquisition - an overview | ScienceDirect Topics

## Knowledge Networks and Management

Investiga como organizar e administrar conhecimento em estruturas conectadas, como grafos de entidades e relações. Essa perspectiva não apenas aborda os aspectos técnicos de modelagem e consulta, mas também os de governança, qualidade e integração de informações em escala organizacional.

Referências:
[1911.09899] Knowledge Network and a Knowledge Network Example
(PDF) Knowledge Networks: Linking Knowledge Management to Business Strategy

## Aspects of Knowledge Structures

Analisa como o conhecimento pode ser organizado em diferentes formas estruturais, como hierarquias, redes ou reticulados conceituais. Essa perspectiva permite explorar não apenas a organização em si, mas também como navegar, inferir e derivar novas

relações a partir dela.

Referências:
Knowledge Structure - an overview | ScienceDirect Topics

General Structure-based Approaches in Information Retrieval, Web Authoring, Information Extraction, and Web Content Mining

Foca no uso de estruturas explícitas, como hiperlinks, árvores ou grafos, para enriquecer tarefas de busca, extração e mineração de conteúdo. A estrutura passa a ser uma fonte adicional de significado e relevância, complementando o conteúdo textual puro e permitindo análises mais robustas.

Referências:
(PDF) Web Content Mining: A Review on Concepts, Techniques, and Tools

https://arxiv.org/pdf/cs/0011033

## CONCLUSÃO DO ESTUDO

Fica claro, a partir do estudo, que diversos tópicos estão muito relacionados, tendo diversas intersecções entre os assuntos. Por outro lado, encontrei também alguns tópicos interessantes, mas que não tem tanta relação com o meu tema quanto eu pensei que teria. Como o meu objetivo é estudar mecanismos de busca para LLMs, foi necessário ter uma visão geral da área para conseguir filtrar os tópicos propostos pelo congresso e direcionar o meu estudo. Nesse sentido, fiz uma pesquisa de artigos do tipo “Survey” para estudo:

[2404.19543] RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing
[2402.19473] Retrieval-Augmented Generation for AI-Generated Content: A Survey
[2312.10997] Retrieval-Augmented Generation for Large Language Models: A Survey
[2405.06211] A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models

Para a primeira semana, decidi fazer uma leitura mais calma apenas do artigo “RAG and RAU: A Survey on Retrieval-Augmented Language Model in Natural Language Processing”, que apresenta de forma mais clara uma taxonomia não só da área como um todo, mas também dos elementos que a compõe, como fontes de dados, LMs, aplicações, entre outros.

Em conclusão, como forma de resumir os resultados da pesquisa, acredito que os três tópicos a seguir já sintetizam bem o campo em que eu quero atuar:

- Natural Language Processing (ICAI):
  - Estuda, dentre diversas coisas, como a linguagem natural pode ser modelada para representações vetoriais. É imprescindível que um especialista em RAG seja capaz de compreender modelos de embedding, relações semânticas, etc…
- Information and Knowledge Retrieval and Searching Algorithms (ACC):
  - O coração do problema: algoritmos de busca, ranqueamento, indexação e recuperação, ou seja, a base de qualquer mecanismo de busca acoplado a LLMs.
- Knowledge Representation (ICAI)
  - Como organizar o conhecimento (embeddings, grafos, ontologias, estruturas híbridas) para que ele possa ser recuperado e utilizado de forma eficiente pelos modelos.

# Entendendo o surgimento da área

Como descrito, escolhi me especializar em mecanismos de busca e recuperação de informação para geração com LLMs. Eu poderia ter resumido no termo “Retrieval-Augmented Generation (RAG)”, porém, para esse momento inicial, eu gostaria de explorar melhor a história dos mecanismos de busca que vieram muito antes dos grandes modelos de linguagem.

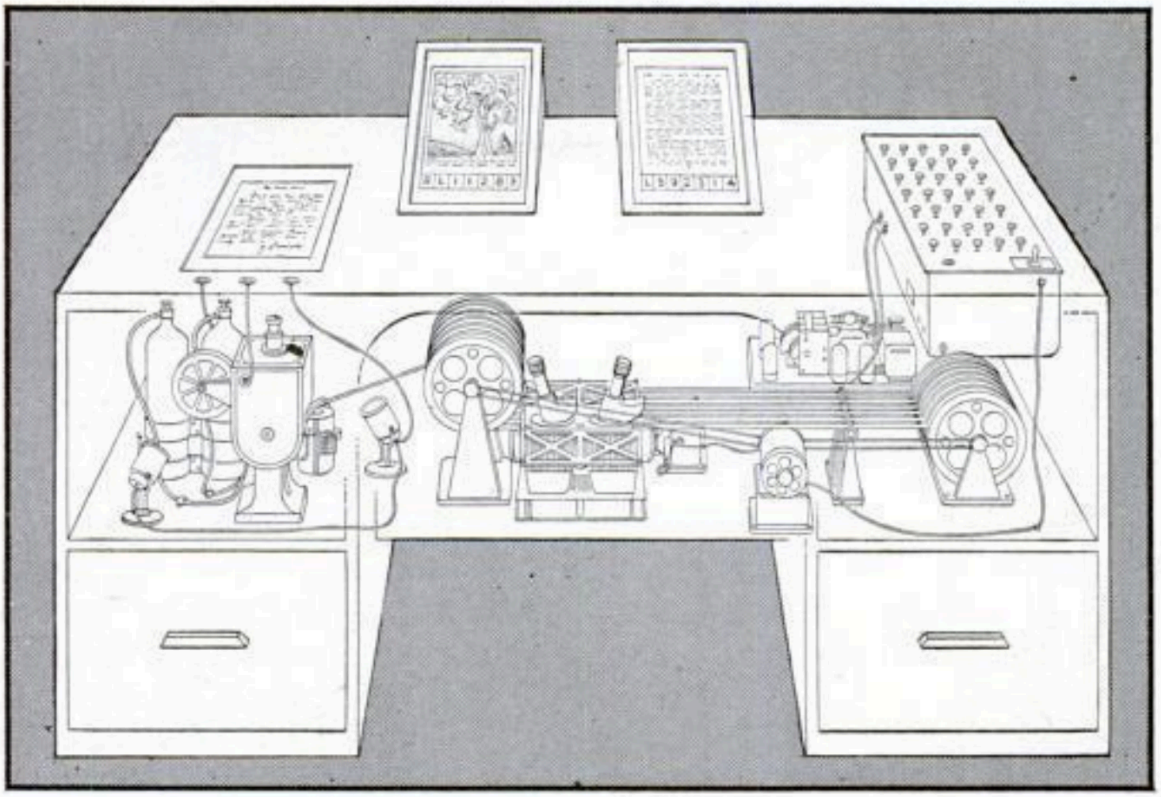
Na primeira metade do século XX, o mundo vivia uma explosão de produção

científica e tecnológica, acelerada em grande parte pelos esforços da Primeira e Segunda Guerra Mundial. Bibliotecas, arquivos e centros de pesquisa acumulavam volumes cada vez maiores de documentos, mas os métodos de organização e acesso permaneceram essencialmente tradicionais: catálogos em fichas, classificações hierárquicas e indexação manual. A recuperação de informação nesse contexto era pensada de forma rígida, baseada em estruturas taxonômicas fixas, nas quais cada documento precisava se enquadrar em uma categoria única e predefinida. Isso significava que a busca era limitada e pouco flexível, refletindo mais a lógica dos sistemas de catalogação do que as necessidades cognitivas do pesquisador.

É nesse cenário que, em 1945, Vannevar Bush publica o ensaio *"As We May Think"*. Durante a minha pesquisa sobre a história da recuperação de informação, foi natural chegar nesse artigo como uma das primeiras ideias para um mecanismo computacional capaz de centralizar e otimizar a consulta por informação.

Link para o ensaio: Bush - As We May Think (Life Magazine 9-10-1945).pdf

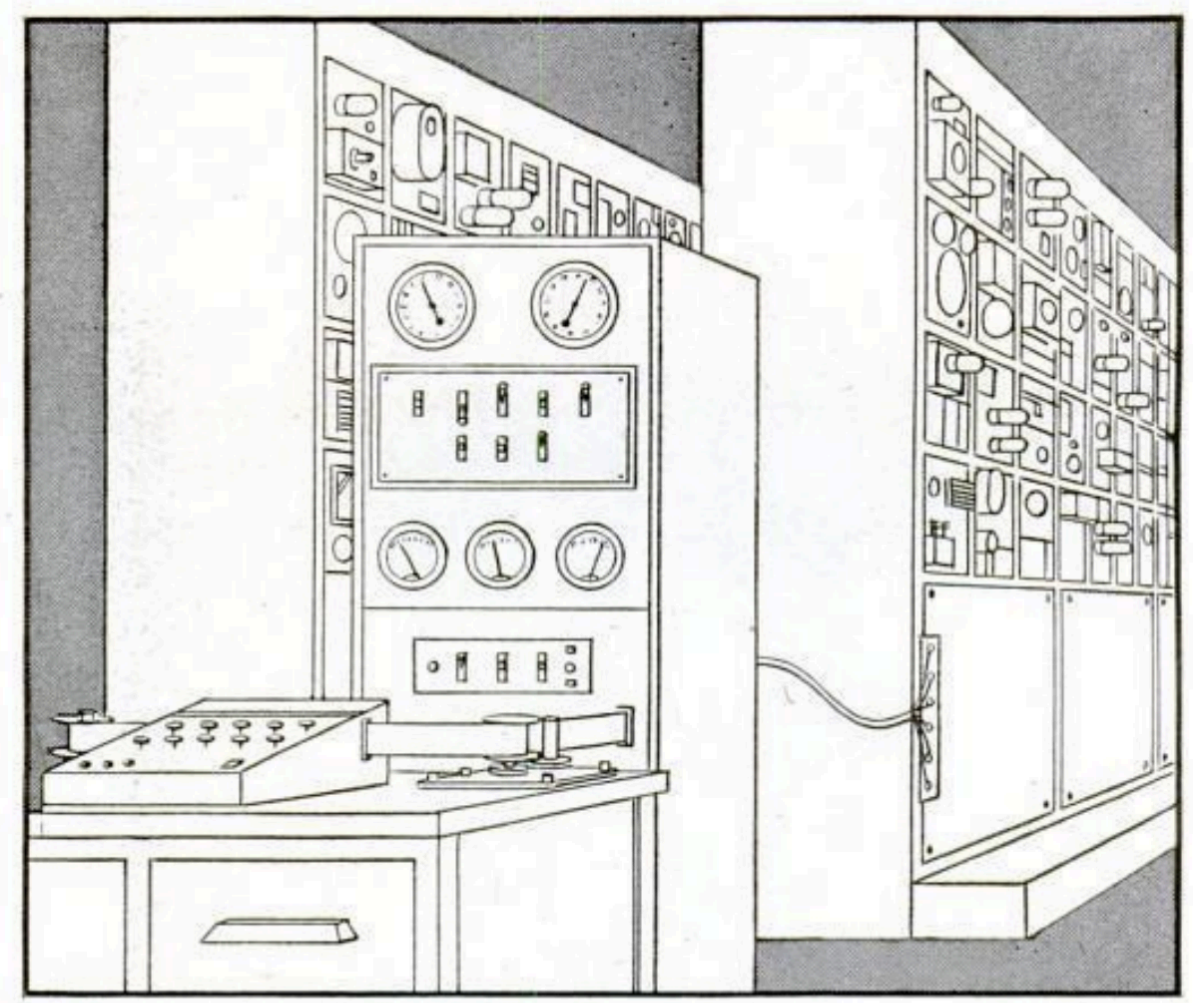
Escrito logo após a Segunda Guerra Mundial, o texto traz um chamado à reorientação do esforço científico: em vez de se voltar para o desenvolvimento de armas, a ciência deveria agora se dedicar a expandir o conhecimento humano e a criar ferramentas que auxiliassem pesquisadores a lidar com a sobrecarga de informação. Para ilustrar essa visão, Bush propôs o memex, um dispositivo teórico baseado em microfilme, que permitiria ao usuário armazenar, consultar e, sobretudo, criar conexões entre documentos de forma dinâmica. Diferentemente dos catálogos hierárquicos tradicionais, o memex operava pela lógica da associação de ideias, imitando o funcionamento da memória humana. Essa proposta introduziu o conceito de "trilhas associativas", nas quais o usuário poderia vincular diferentes documentos em cadeias de significado, formando percursos de navegação personalizados.



**MEMEX** in the form of a desk would instantly bring files and material on any subject to the operator's fingertips. Slanting translucent viewing screens magnify supermicrofilm filed by code numbers. At left is a mechanism which automatically photographs longhand notes, pictures and letters, then files them in the desk for future reference.

A importância do artigo para a história da Recuperação da Informação é dupla. Primeiro, ele enunciou com clareza o problema central que se tornaria a base da área: como organizar e tornar acessível um volume crescente de informação de maneira eficiente e útil. Segundo, ao propor uma forma de navegação por associações em vez de classificações fixas, Bush antecipou ideias que décadas depois se materializariam em conceitos fundamentais da RI moderna, como a busca por relevância, a navegação por hipertexto e até mesmo os modelos de recuperação baseados em redes de conexões. Sua visão inspirou diretamente pioneiros como Douglas Engelbart, Ted Nelson e Gerard Salton, que levariam adiante a tarefa de transformar a recuperação de informação em uma disciplina científica apoiada em computadores digitais.

Evidentemente fiz a leitura do ensaio com foco no estudo de recuperação de informação, porém acho válido mencionar também que ele trouxe a ideia de uma "thinking machine". Trata-se de uma máquina capaz de raciocinar sobre premissas e chegar a conclusões lógicas, funcionando quase como uma extensão do intelecto humano. Já era uma visão de automação do raciocínio humano, mostrando também o quão visionário foi o autor.



**THINKING MACHINES** would solve not only the most difficult mathematical problems but even problems of logical thought. Mathematical problems would be fed by punched tape to the electronic device in the racks at rear. Results, accomplished in a fraction of the time it takes man, would be recorded on dials at top and bottom of control board.

Em resumo, Bush identificou já em 1945 um dos desafios centrais da RI: há

muito mais informação disponível do que a mente humana pode processar.

Durante minha pesquisa, percebi que Gerard Salton foi um dos nomes que se inspiraram no trabalho de Bush. Em paralelo, enquanto eu pesquisava por referência na literatura de recuperação da informação, percebi que o nome de Salton também era frequente. Isso me despertou a atenção para esse autor. Coloquei o nome do Salton no Google e acabei descobrindo, segundo diversas fontes, o “pai da recuperação de informação".

Gerard Salton foi o arquiteto de um dos primeiros e mais duradouros programas de pesquisa em Recuperação da Informação. Em Harvard e, sobretudo, em Cornell, ele estruturou o SMART (System for the Mechanical Analysis and Retrieval of Text) como um “laboratório vivo” para testar ideias de indexação, ranqueamento e avaliação em coleções reais.

Referência: Who is Gerard Salton? - Babbar.tech Blog

No SMART, Salton impulsionou a passagem da busca booleana para a recuperação ranqueada por similaridade, representando documentos e consultas como vetores e medindo proximidade (por exemplo, por cosseno). Essa linha culmina no artigo clássico “A Vector Space Model for Automatic Indexing”, que formaliza a abordagem vetorial e reporta seus ganhos em testes controlados e cuja implementação prática foi incubada no ecossistema do SMART. (A própria historiografia da área registra o SMART como o primeiro grande uso do modelo vetorial em IR.)

# APÊNDICE 2

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 11 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

**THIAGO PEDROSO DE JESUS**

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Na primeira Semana, defini meu tema como sendo Retrieval-Augmented Generation (RAG), e iniciei o estudo histórico da recuperação de informação, com Bush e Salton.

Durante a segunda Semana, meu foco foi consolidar o estudo histórico até o surgimento do termo "RAG" e iniciar a estruturação de uma taxonomia/mapeamento da área, para direcionar estudos futuros.

- Leitura do artigo "A Vector Space Model for Automatic Indexing", escrito por Gerard Salton em 1975. A partir desse estudo, encontrei fundamentos sobre:
  - Representação vetorial de documentos
  - Estudo e experimentos em ponderação de termos
  - Similaridade de vetores
  - Avaliação de sistemas de indexação

  Anotações produzidas sobre o artigo: Residência - A VSM for Automatic Indexing

- Desenvolvimento de uma linha do tempo que descreve a evolução da Recuperação de Informação até o surgimento do termo RAG, em 2020. Meu processo de desenvolvimento:
  - Utilizando o Google Search e o Google Acadêmico, pesquisei pelo termo "História da recuperação de informação" para encontrar trabalhos que já descreviam a história da área;
  - Selecionei trabalhos que mais faziam sentido para o meu propósito, de modo especial 2 artigos que detalham a história e deixam boas referências;
  - Anotei os principais marcos históricos encontrados e produzi uma linha do tempo.

  Descrição: Residência - Timeline histórica RAG

- Estruturação inicial de uma taxonomia do RAG
  - Utilizando o Google Acadêmico, fiz a busca por surveys recentes sobre o tema;
  - Selecionei 4 dos resultados e fiz uma leitura parcial com foco em entender como a área é classificada e subdividida
  - Criei um mapa mental para organizar e direcionar meus estudos

  Mapa mental: Link de acesso

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Com o mapa mental, ficou claro que RAG pode ser inicialmente dividido no estudo de recuperação, geração e integração.

Portanto, com o objetivo de aprofundar meu estudo nos mecanismos de recuperação (retrievers), eu pretendo:

- Estudar os principais tipos de retrievers (lexicais, densos, híbridos e multimodais), destacando suas características, vantagens e limitações;
- Relacionar cada tipo de retriever com os marcos históricos já estudados (ex.: TF-IDF, BM25, DPR);
- Produzir uma tabela comparativa que organize os diferentes métodos e suas aplicações em RAG.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Linha do Tempo histórica

## CONTEXTUALIZAÇÃO

Para maior compreensão histórica da área, me dediquei ao desenvolvimento de uma timeline de marcos históricos importantes para o entendimento de como os mecanismos de representação e recuperação de informação se desenvolveram até o surgimento do termo “RAG”.

Descrição de acontecimentos relevantes:

1. **1945 - Ensaio “As We May Think” de Vannevar Bush:** Em um contexto de fim das Segunda Guerra Mundial, Bush reconheceu que a produção científica crescia além da capacidade humana de acompanhá-la, destacando a necessidade de novos métodos (eventualmente computacionais) de indexar e acessar informações relevantes dentre grandes volumes de documentos. Segue o estudo detalhado que eu realizei sobre esse artigo, conforma consta no termo de entrega: Residência - História da recuperação de informação

2. **1950 - Paper “Information Retrieval Viewed As Temporal Signalling” de Calvin Mooers.** A primeira menção do termo “Information Retrieval” foi feita nesse artigo de Mooers. Ele foi também autor do Lei de Mooers (1959): “An information retrieval system will tend not to be used whenever it is more painful and troublesome for a customer to have information than for him not to have it.”

3. **1972 – Ponderação TF-IDF, criado por Karen Spärck Jones:** No início dos anos 1970, pesquisadores aperfeiçoaram as técnicas de indexação ponderada. Em 1972, Karen Spärck Jones introduziu o conceito de Inverse Document Frequency (IDF), quantificando a especificidade de um termo em uma coleção

4. **1975 – Modelo de espaço vetorial por Gerard Salton:** Durante a década de 1960, Gerard Salton liderou a primeira grande equipe de pesquisa em IR (em Harvard e depois Cornell), desenvolvendo em 1975 o modelo de espaço vetorial. Eu faço uma descrição do paper responsável por esse modelo no documento Residência - A Vector Space Model for Automatic Indexing. Dessa forma, o trabalho de Salton lançou as bases matemáticas para mecanismos de busca, e seu sistema SMART demonstrou empiricamente a eficácia do modelo vetorial, influenciando fortemente o design de motores de busca nas décadas seguintes.

5. **1976 - Framework de Recuperação Probabilística de Stephen Robertson e Karen Spärck Jones:** Eles desenvolveram o framework de recuperação probabilística que posteriormente levou ao algoritmo BM25. Este trabalho estabeleceu fundamentos teóricos rigorosos para scoring de relevância baseado em princípios probabilísticos. Segue o link para o trabalho deles: (PDF) Relevance weighting of search terms.

6. **1988 - Indexação Semântica Latente (LSI), de Scott Deerwester, Susan Dumais, George Furnas e colegas.** Foi uma técnica pioneira projetada para superar os problemas de sinonímia (diferentes palavras com o mesmo significado) e polissemia (a mesma palavra com múltiplos significados) que limitavam os sistemas de RI baseados em palavras-chave. Pioneirismo na recuperação semântica através da projeção de texto para um espaço latente de baixa dimensão. Link para o paper: (PDF) Indexing by Latent Semantic Analysis (1990) | Scott Deerwester | 13330 Citations

7. **1992 - Primeira TREC (Text Retrieval Conference):** Em 1992, o Departamento de Defesa dos EUA e o NIST organizaram a primeira TREC, uma conferência competitiva para avaliação de técnicas de recuperação em larga escala. Pela primeira vez, pesquisadores puderam comparar sistemas usando dados e métricas idênticas.

8. **1994 - Algoritmo de Relevância Okapi BM25 por Stephen Robertson:** Houve um avanço prático significativo derivado do modelo probabilístico de Robertson: o algoritmo Okapi BM25. Publicado por Robertson, o BM25 tornou-se um dos mais eficazes métodos de ranqueamento de documentos, combinando TF, IDF e uma normalização pelo tamanho do documento. O mais legal de se observar é que ele foi proposto no TREC 3! Segue o link para o artigo: Okapi at TREC

9. **1998 - Google e o PageRank (Revolução na Busca Web):** Em setembro de 1998, Larry Page e Sergey Brin fundaram a Google Inc., introduzindo o mecanismo de busca Google que revolucionaria a recuperação de informação na Web. O elemento-chave foi o algoritmo PageRank, descrito por eles em 1998, que analisa a estrutura de hyperlinks entre páginas para atribuir um nível de importância a cada página. Link para o artigo: The PageRank Citation Ranking: Bringing Order to the Web. - Stanford InfoLab Publication Server

10. **2003 - Modelos de Linguagem Neurais:** Nomes como o de Yoshua Bengio fundamentaram teoricamente o conceito de aprender representações de palavras (embeddings) distribuídas e densas.

11. **2011 - IBM Watson vence "Jeopardy!":** Em fevereiro de 2011, o supercomputador IBM Watson derrotou os campeões humanos no programa de perguntas Jeopardy!, marcando um momento histórico para a IA. O Watson foi desenvolvido entre 2007 e 2011 sob o projeto DeepQA da IBM. Sua arquitetura era essencialmente um pipeline que integrava recuperação de informações e geração de respostas: diante de uma pergunta em linguagem natural, o Watson fazia buscas em milhões de documentos (enciclopédias, dicionários, notícias), extraía trechos possivelmente relevantes e então aplicava modelos estatísticos e regras para gerar e avaliar possíveis respostas.

12. **2013 - Word2Vec por Tomáš Mikolov e equipe do Google:** Foi um avanço no processamento de linguagem natural que mudou a forma de representar texto. Uma técnica para treinar redes neurais simples que produziam vetores contínuos para cada palavra (embeddings). Link para o artigo: Efficient Estimation of Word Representations in Vector Space

13. **2015 - Google RankBrain:** Ele foi o primeiro sistema de deep learning do Google a ser aplicado na Pesquisa em 2015 e continua a ser um dos principais sistemas de IA que alimentam a Pesquisa hoje, melhorando a relevância e a satisfação do utilizador.

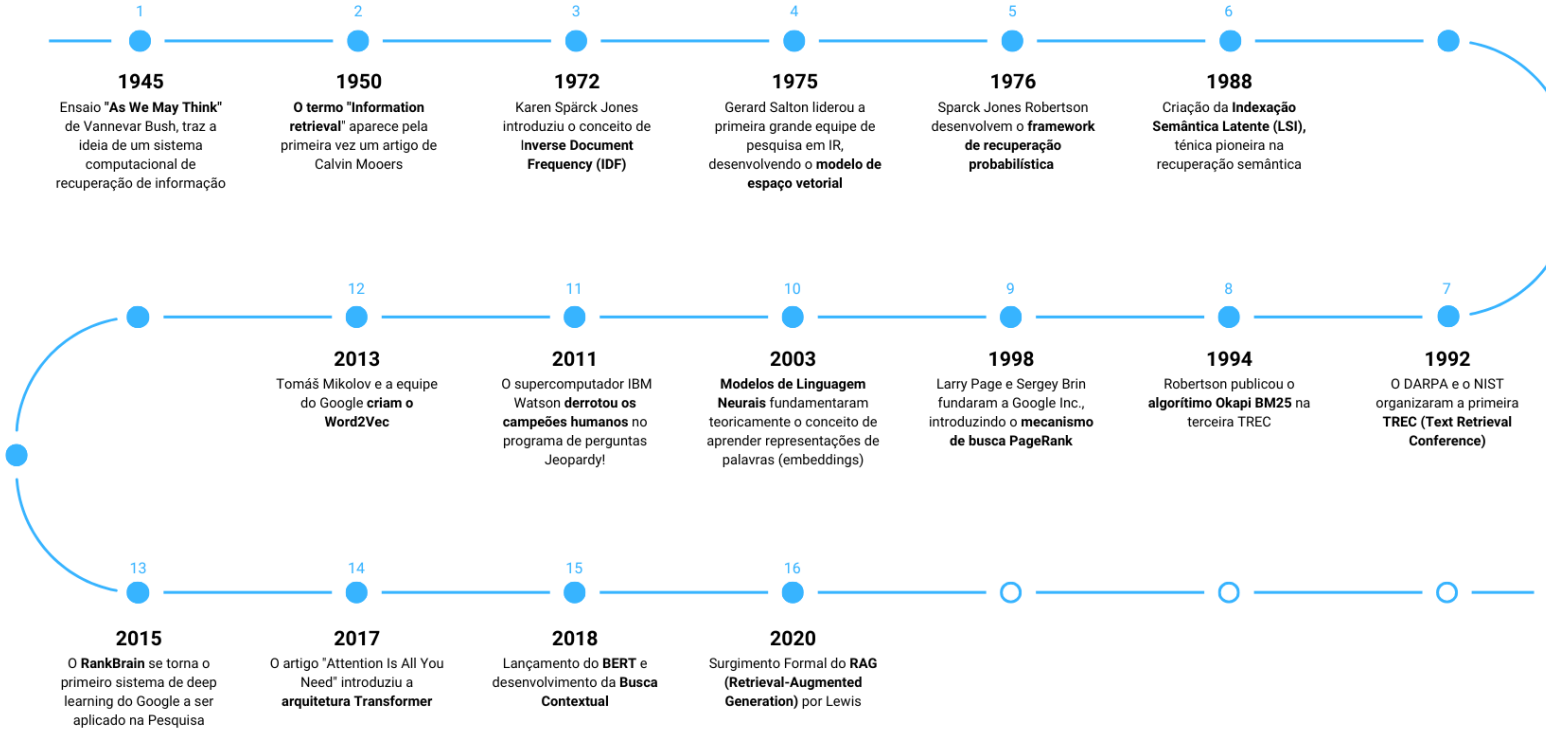
14. **2017 - Arquitetura Transformer de Ashish Vaswani e uma equipe de pesquisadores do Google:** O artigo "Attention Is All You Need" introduziu a arquitetura Transformer, que mudou fundamentalmente o campo do PLN. A inovação central foi abandonar completamente as arquiteturas recorrentes (RNNs) e convolucionais, que processam os dados sequencialmente, em favor de um mecanismo baseado unicamente em self-attention. Link para o paper: [1706.03762] Attention Is All You Need

15. **2018 – BERT e a Busca Contextual:** A Google lançou o BERT (Bidirectional Encoder Representations from Transformers), modelo de linguagem baseado em Transformer treinado de forma bi-direcional, que trouxe compreensão contextual de alto nível para palavras em texto. Com BERT, o motor de busca passou a captar nuances de linguagem, melhorando significativamente os resultados para buscas em linguagem natural. Link para o paper do BERT: [1810.04805] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

16. **2020 – Surgimento Formal do RAG (Retrieval-Augmented Generation):** O termo foi introduzido em 2020 por Patrick Lewis e colegas, no artigo seminal “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. Esse

trabalho apresentou um framework unificado onde um modelo primeiro realiza busca externa e, em seguida, um modelo gerativo (seq2seq) produz a resposta usando as evidências recuperadas. Na implementação original, o sistema utilizou o DPR (Dense Passage Retrieval). Link para o artigo: [2005.11401] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

## LINHA DO TEMPO DO RAG

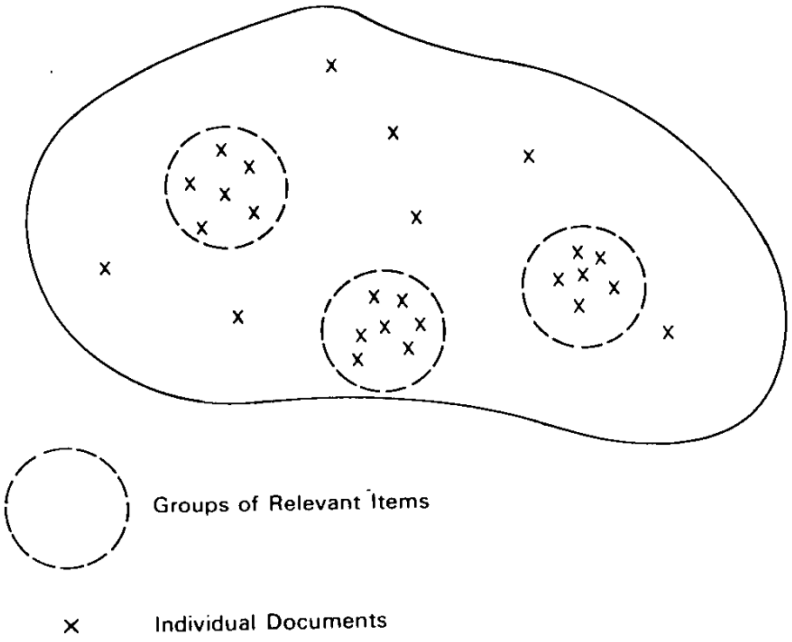



# A Vector Space Model for Automatic Indexing

## CONTEXTUALIZAÇÃO

O artigo "A vector space model for automatic indexing" parte da ideia de representar cada documento como um vetor em um espaço de termos, onde a similaridade é medida pelo ângulo ou produto interno entre vetores normalizados. Nesse cenário, um document space ideal seria aquele em que documentos relacionados entre si (do ponto de vista de relevância) ficam próximos, enquanto documentos não relacionados ficam bem distantes. Como não é possível conhecer de antemão todos os julgamentos de relevância dos usuários, os autores propõem uma aproximação: em vez de tentar "arrumar" o espaço perfeito, busca-se espalhar ao máximo os documentos, reduzindo a densidade média do espaço vetorial. Em um espaço mais esparso, aumenta a chance de um documento relevante ser recuperado sem "arrastar" muitos vizinhos irrelevantes junto.

Fig. 2. Ideal document space.




## CORRELAÇÃO ENTRE DENSIDADE E DESEMPENHO

Para ligar a intuição geométrica ao desempenho real do sistema de busca, os

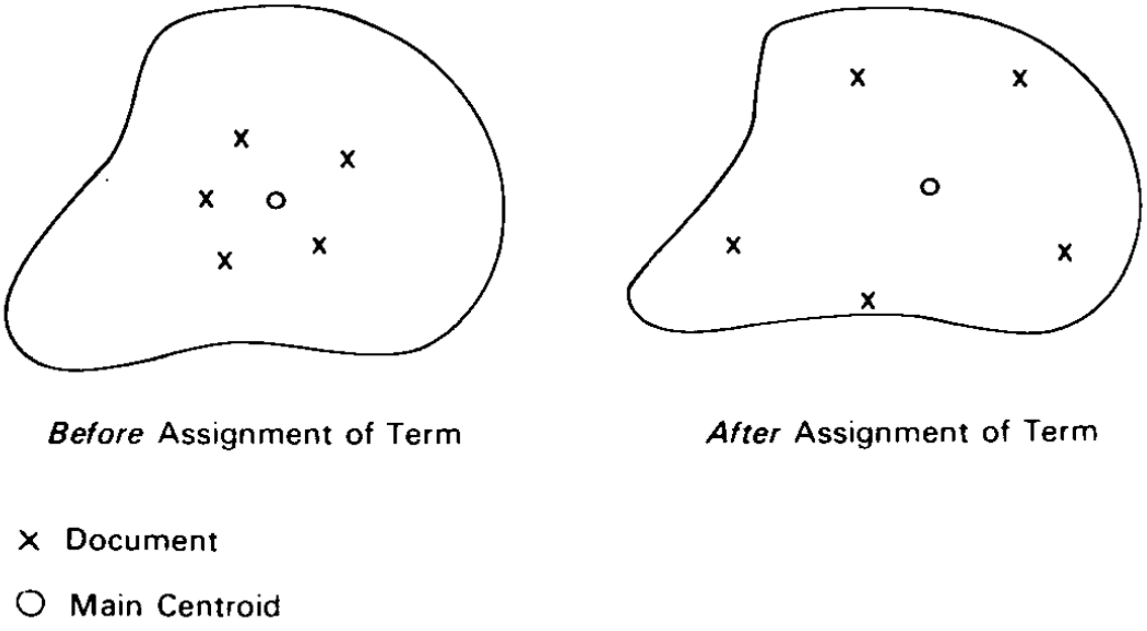
autores organizam os documentos em clusters e medem duas quantidades: (x) a similaridade média entre cada documento e o centróide do seu cluster (coesão interna) e (y) a similaridade média entre os centróides dos clusters (separação entre grupos). A razão y/x funciona como um índice global de densidade: quanto menor, mais bem separados e organizados estão os documentos. Testando diferentes esquemas de ponderação de termos (TF simples, TF–IDF e uma variante que enfatiza termos muito frequentes), o artigo mostra que reduzir a densidade (y/x menor) está diretamente associado a melhores curvas de recall–precisão, enquanto aumentar a densidade piora o desempenho.

## O MODELO DE VALOR DE DISCRIMINAÇÃO

**Pergunta:** O que seria um termo bom ou um termo ruim?

Um bom termo é aquele que, ao ser usado na representação dos documentos, aumenta a separação entre eles no espaço vetorial. Já um mau termo é aquele que aproxima artificialmente documentos diferentes, comprimindo o espaço.

Fig. 5. Operation of good discriminating term.




Para medir esse efeito, os autores propõem o Discrimination Value (DV). Ele compara a densidade do espaço com e sem determinado termo. Se a remoção do termo torna o espaço mais compacto, é sinal de que o termo era importante para espalhar os documentos (DV positivo). Se, ao contrário, a remoção torna o espaço mais disperso, o termo era prejudicial (DV negativo).

# APÊNDICE 3

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 18 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as duas primeiras Semanas, escolhi como tema de estudo a Retrieval-Augmented Generation (RAG) e tive a oportunidade de compreender melhor a história da área de Recuperação da Informação, produzindo uma linha do tempo com os marcos mais relevantes até o surgimento do termo "RAG", além de um mapa mental que oferece uma visão ampla do tema.

Dentro do meu planejamento, eu havia destacado que o RAG pode ser dividido em três frentes principais: Recuperação, Geração e Integração. Para a terceira Semana, defini como prioridade o aprofundamento na frente de Recuperação (Retriever), com o objetivo de compreender os fundamentos e relacioná-los aos marcos históricos estudados.

Nesse sentido, desenvolvi a Semana com os seguintes estudos:

- Retrievers esparsos:
  - TF-IDF (Relacionando ao surgimento em 1972 por K Sparck Jones)
  - BM25 (Relacionando ao surgimento em 1976 por SE Robertson e K Sparck Jones)
  - BM25F, BM25L, BM25+
- Retrievers densos:
  Para compreender esse eixo, investiguei a evolução dos embeddings e suas arquiteturas, que servem como base para os retrievers densos. Os principais marcos estudados foram:
  - Word2Vec (Relacionando ao surgimento em 2013 pela Google)
  - Glove
  - FastText
  - Elmo

Em preparação para a prova final da disciplina de Processamento de Linguagem Natural, eu produzi, junto com alguns colegas, um material de revisão com todos os conteúdos da disciplina. Na ocasião, eu fui responsável, coincidentemente, pelos tópicos "Embeddings" e "RAG". Portanto, o meu estudo das arquiteturas durante essa Semana foi baseado na leitura do meu próprio documento de revisão, dos materiais ofertados na disciplina e das fontes que eu já havia destacado como relevantes para o tema.

Deixo minhas anotações e estudos no documento: Residência - Retrievers

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Dando continuidade ao estudo da frente de Recuperação (Retrieval) do RAG, é necessário compreender como a área evoluiu não apenas em relação aos modelos de busca, mas também em relação às estratégias de preparação e organização da informação.

Nesse sentido, o meu planejamento para a próxima Semana envolve:

- Investigar diferentes estratégias de chunking, como segmentação fixa, segmentação semântica e sobreposição de janelas.
- Estudar métodos de indexação clássicos (como o inverted index) e analisar sua evolução ao longo do tempo, relacionando-os à história da Recuperação de Informação.
- Mapear os principais tipos de fontes de dados utilizadas em sistemas RAG (estruturadas, não estruturadas e semi-estruturadas), destacando desafios de curadoria, atualização e confiabilidade.

Para materializar o meu estudo, eu pretendo:

- Produzir um documento com anotações, referências e guias sobre cada técnica estudada.
- Expandir o mapa mental sobre RAG, desenvolvido na Semana 2, incorporando os conceitos de Retrieval e situando-os no contexto mais amplo do tema.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Estudo de Retrievers

## DEFINIÇÃO

De acordo com o artigo *"Retrieval-Augmented Generation for AI-Generated Content: A Survey"*, o "Retriever" consiste em identificar e obter informações relevantes diante de uma necessidade de informação. Especificamente, vamos considerar recursos de informação que podem ser conceitualizados como um armazenamento de chave-valor, onde cada chave corresponde a um valor (chaves e valores podem ser idênticos). Dada uma consulta, o objetivo é buscar as k chaves mais semelhantes usando uma função de similaridade e, em seguida, obter os valores associados. Com base em diferentes funções de similaridade, os métodos de recuperação existentes podem ser categorizados em recuperação esparsa, recuperação densa e outros. Nas formas amplamente utilizadas de recuperação esparsa e densa, todo o processo pode ser dividido em duas fases distintas: (i) cada objeto é primeiro codificado em uma representação ; e então (ii) um índice é construído para organizar a fonte de dados a fim de permitir uma busca eficiente.

## SPARSE RETRIEVAL (ESPARSO)

Em resumo, são métodos tradicionais de recuperação textual baseados em correspondência exata ou estatística de termos (palavras). Representam consulta e documentos como vetores esparsos no espaço de palavras ou n-gramas: cada dimensão corresponde a um termo do vocabulário. Técnicas consagradas incluem o cálculo de TF-IDF (Term Frequency–Inverse Document Frequency) e modelos probabilísticos como Query Likelihood (modelo de linguagem de consulta) e o famoso BM25. Esses modelos atribuem pesos aos termos – por exemplo, TF-IDF aumenta o peso de palavras frequentes na consulta, mas penaliza termos muito comuns na coleção. O BM25, proposto por Stephen Robertson e colegas nos anos 1990, combina heurísticas de frequência de termo no documento, frequência inversa nos documentos (IDF) e comprimento do documento, sendo até hoje um forte baseline em buscas na web. A principal vantagem dos métodos sparse é a interpretabilidade e

eficiência: eles utilizam um índice invertido, que mapeia cada termo a uma lista de documentos onde ele ocorre. Assim, ao receber a consulta, o sistema verifica quais documentos contêm os termos consultados e calcula um escore (e.g., BM25) para ordená-los. Esse paradigma foi a base de motores de busca clássicos (ex.: Google no início, Elasticsearch, etc.), escalando para coleções de bilhões de documentos.

Uma boa referência que eu encontrei para comparar os métodos de busca esparsa é: Sparse Embedding: Bag of Words, TF-IDF, BM25

## TF-IDF

### História

Em seu artigo original de 1972, Spärck Jones fez modestas afirmações teóricas para o IDF, apresentando-o principalmente como uma heurística poderosa com forte apelo intuitivo e uma conexão com a Lei de Zipf, que descreve a distribuição de frequência de palavras na linguagem natural. A pesquisa de Jones tinha por objetivo entender e avaliar como atribuir pesos para termos. Durante décadas após sua publicação, pesquisadores buscaram estabelecer uma base teórica mais formal para a notável eficácia do IDF. Essa busca por uma justificativa baseada em princípios levou ao desenvolvimento de modelos probabilísticos de recuperação, mais notavelmente por Stephen Robertson, que, em colaboração com Spärck Jones, mais tarde demonstraria que o IDF pode ser derivado como uma aproximação dentro de uma estrutura probabilística.

### Definição

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

**TF-IDF**

Term $x$ within document $y$

$tf_{x,y}$ = frequency of $x$ in $y$
$df_x$ = number of documents containing $x$
$N$ = total number of documents

TF (Term Frequency):
Quanto mais vezes uma palavra aparece em um documento, maior deve ser o peso dela naquele documento.
Exemplo: Se "inteligência" aparece 20 vezes em um artigo, provavelmente é um tema central.

IDF (Inverse Document Frequency):
Palavras que aparecem em todos os documentos (como "de", "o", "e") não ajudam a diferenciar os textos. Por isso, elas recebem peso baixo.
Exemplo: A palavra "inteligência" pode ser mais rara em toda a coleção. Logo, ganha peso maior.

Multiplicação TF × IDF:
O peso final é alto quando:

- O termo aparece muitas vezes em um documento específico (alto TF).
- Mas aparece em poucos documentos no total (alto IDF).

Apesar de sua simplicidade, o TF-IDF estabeleceu a base para modelos de espaço vetorial, nos quais documentos são representados como vetores esparsos e de alta dimensão de pesos de termos. No entanto, ele opera em um modelo de "bag of words", o que significa que trata documentos como uma coleção desordenada de palavras, ignorando completamente a sintaxe, a ordem das palavras e o contexto semântico. Esta é sua principal limitação: o TF-IDF não reconhece sinônimos (por exemplo, "carro" vs. "automóvel") nem compreende paráfrases. Uma consulta por "algoritmos de IA" não corresponderia a um documento que discute "redes neurais" se os termos exatos da consulta não estivessem presentes. Esse problema de incompatibilidade lexical se tornaria a motivação central para o desenvolvimento dos recuperadores semânticos densos que se seguiram.

## BM25

### História

Com base no trabalho fundamental de Spärck Jones, o campo da recuperação de informação passou por uma evolução significativa nas décadas de 1970 e 1980, passando de heurísticas empiricamente bem-sucedidas para modelos mais formais e teoricamente fundamentados. Essa mudança foi liderada por Stephen E. Robertson,

cujo trabalho culminou em uma das funções de classificação mais bem-sucedidas e duradouras da história do campo.

O desenvolvimento do “Probabilistic Relevance Framework” (PRF) por Stephen E. Robertson, Karen Spärck Jones e outros foi um marco que reformulou a tarefa de recuperação de informações. O objetivo central da PRF é estimar a probabilidade de um documento ser relevante para uma determinada consulta do usuário, um afastamento dos esquemas de ponderação de termos mais diretos, como o TF-IDF. Paper: (PDF) The Probabilistic Relevance Framework: BM25 and Beyond

Esta estrutura baseia-se no “Probability Ranking Principle” (PRP), uma hipótese formal articulada por Robertson em 1977. O PRP afirma que, se a resposta de um sistema de recuperação a uma consulta for uma classificação de documentos em ordem decrescente de sua probabilidade de relevância, a eficácia geral do sistema será a melhor possível para esses dados. Este princípio forneceu um alvo teórico claro para a otimização dos sistemas de recuperação. Paper: (PDF) The Probability Ranking Principle in IR

A instanciação mais famosa e bem-sucedida do PRF é a função de classificação “Okapi BM25” (abreviação de Best Matching 25). Ela foi desenvolvida como parte do sistema de recuperação de informações Okapi na City University de Londres nas décadas de 1980 e 1990 e foi descrita no artigo "Okapi no TREC-3" de Robertson e seus colegas em 1994.

## Definição

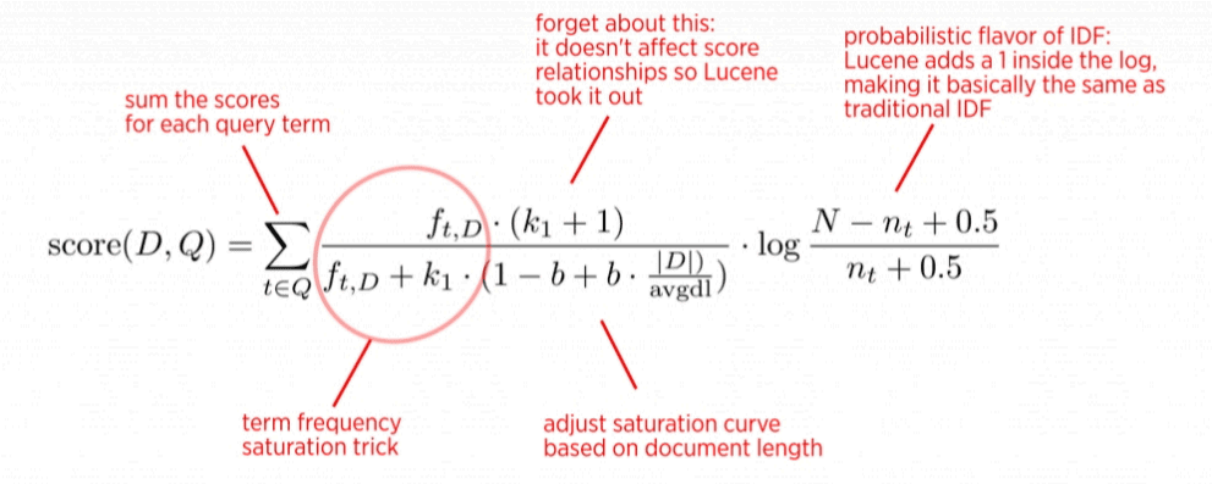
O BM25 é um modelo de recuperação de informações que aprimora as limitações do TF-IDF. Semelhante ao TF-IDF, ele se baseia na frequência de palavras e documentos, mas utiliza uma fórmula mais sofisticada para avaliar a similaridade entre documentos e consultas. Em particular, ele fornece resultados de recuperação realistas, ajustando o efeito de saturação na frequência de palavras e no comprimento do documento.

$$\text{score}(d_i, q) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{f(t, D) \cdot (k_1 + 1)}{f(t, d_i) + k_1 \cdot \left(1 - b + b \cdot \frac{|d_i|}{\text{avgdl}}\right)}$$

Onde:

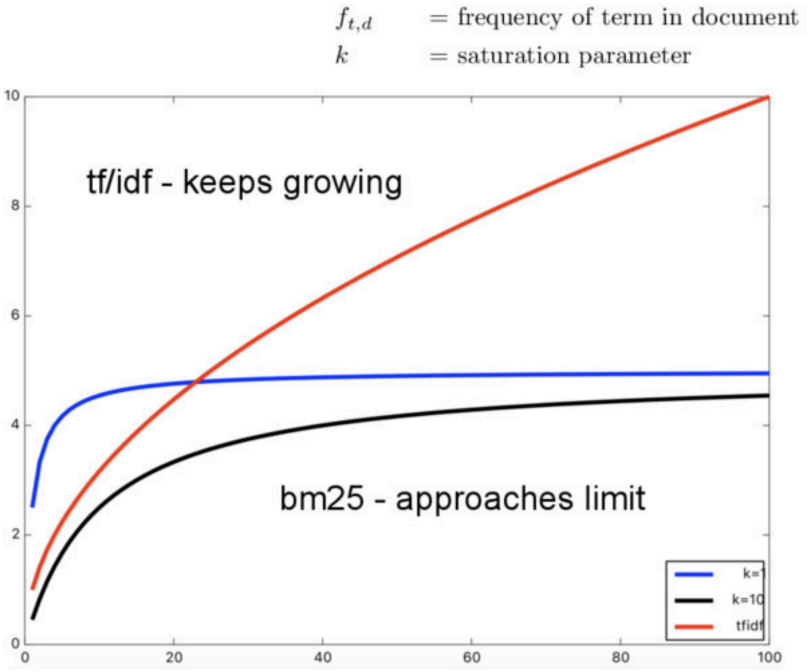
- $t$ é o termo de busca.
- $f(t, d_i)$ é a frequência do termo $t$ no documento $d_i$.
- $|d_i|$ é o tamanho do documento $d_i$ em palavras.
- $\text{avgdl}$ é a média do tamanho dos documentos na coleção.
- $k_1$ Parâmetro que controla a sensibilidade à frequência do termo (normalmente 1.2).
- $b$ Parâmetro que controla a normalização do comprimento (normalmente 0.75).
- $\text{IDF}(t)$ é a "inverse document frequency" do termo de busca $t$ na coleção. Calculado como:
  $\text{IDF}(t) = \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5}\right)$, onde N é o número total de documentos na coleção, 0.5 é uma "smoothing constant", e n(t) é o número total de documentos contendo o termo t.

A fórmula pode também ser lida e compreendida da seguinte forma:

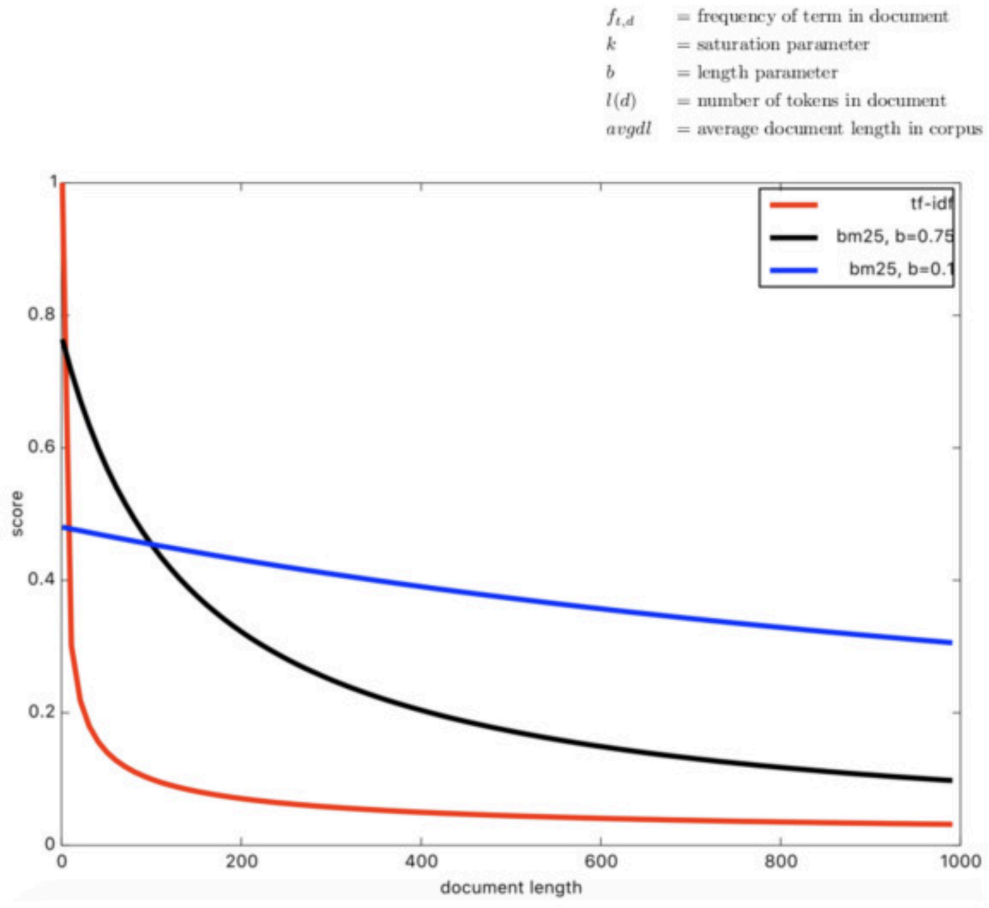



Fonte: [BM25Similarity: An Effective Relevance Model for Information Retrieval | by Everton Gomede, PhD | The Modern Scientist | Medium]

Como observado no TF-IDF, quanto mais vezes um termo aparece em um documento, maior o peso. Além disso, documentos mais longos tendem a ter mais ocorrências de termos só porque têm mais palavras, o que pode gerar viés a favor deles.



O TF-IDF assume que, quanto mais vezes uma palavra aparece em um documento, mais relevante ela é. O problema é que esse crescimento é ilimitado: se uma palavra se repete 50, 100 ou 200 vezes, o TF-IDF continua aumentando o peso de forma linear, mesmo que, do ponto de vista semântico, depois de certo ponto repetir a palavra não acrescente nova informação. O BM25 corrige isso introduzindo uma função de saturação. Em vez de crescer indefinidamente, o ganho da frequência tende a um limite. Isso significa que as primeiras ocorrências de uma palavra são muito valiosas, mas cada repetição adicional vai contribuindo cada vez menos, até praticamente não fazer diferença. Assim, o modelo se aproxima mais do comportamento humano, em que notar uma palavra uma vez ou algumas vezes já é suficiente para associá-la fortemente ao tema de um texto. No BM25, é possível controlar esse comportamento usando o parâmetro K, conforme indica a figura.




Outro ajuste importante que o BM25 faz é no comprimento dos documentos. O TF-IDF penaliza documentos longos de maneira excessiva, porque, ao normalizar pela

quantidade total de termos, acaba reduzindo demais o peso de palavras em textos extensos. Isso cria um viés: textos maiores podem parecer menos relevantes, mesmo que tenham muito conteúdo útil. O BM25 resolve isso de forma mais equilibrada, comparando o tamanho do documento com a média de comprimento da coleção. Assim, documentos que são maiores do que a média sofrem uma penalização controlada, mas não injusta. Isso permite que textos longos ainda sejam considerados relevantes se tiverem conteúdo consistente. Esse comportamento é controlado pelo parâmetro “b”

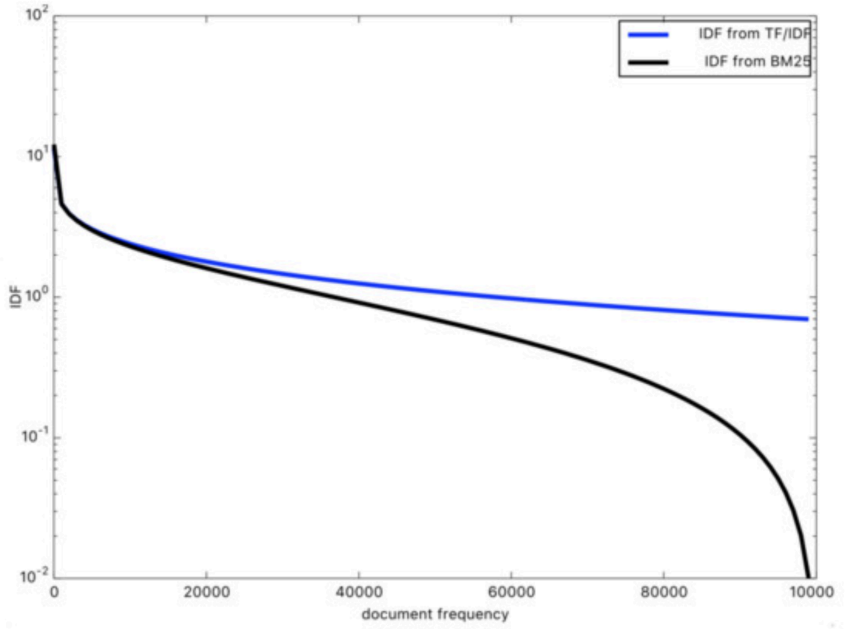
Intuição do b:

- Se b = 0 - não há normalização pelo comprimento. Ou seja, documentos longos não sofrem nenhuma penalização.

- Se b = 1 - a normalização pelo comprimento é máxima. Documentos maiores são penalizados proporcionalmente à sua extensão em relação à média.

- Se 0 < b < 1 (valores típicos ficam em torno de 0,75) - temos um meio-termo: documentos muito grandes são penalizados, mas de forma mais suave do que no TF-IDF.

$N$ = number of documents
$df_t$ = number of docs that contain the term



Por fim, o BM25 também faz um refinamento na forma como calcula o IDF, o fator que mede a raridade de uma palavra na coleção. No TF-IDF, o IDF é sempre positivo, o que significa que até termos extremamente comuns, que aparecem em quase todos os documentos, acabam recebendo algum peso. O BM25 suaviza esse efeito, chegando até a reduzir o peso a praticamente zero para palavras extremamente frequentes. Isso evita que termos que são quase “stopwords” continuem influenciando o ranking, tornando a busca mais seletiva e precisa.

### Conclusão

O estudo do BM25 mostra como ele se consolidou como um dos métodos mais eficazes de recuperação de informação, servindo como referência por décadas. Sua força está no equilíbrio entre frequência de termos, raridade no corpus e ajustes de comprimento de documentos, o que o torna mais realista e robusto que o TF-IDF. Além disso, extensões como o BM25F ampliaram sua aplicabilidade em contextos mais complexos, permitindo dar pesos diferenciados a campos importantes como título e resumo.

A evolução do TF-IDF para o BM25 ilustra um movimento maior na inteligência artificial: transformar heurísticas intuitivas em modelos probabilísticos e matemáticos bem fundamentados. Essa trajetória reforça a relevância teórica e prática do BM25, mostrando como ideias simples, quando formalizadas, podem se tornar pilares duradouros.

Por fim, o BM25 permanece essencial porque captura algo único: a precisão lexical. Em tarefas que exigem correspondência exata de termos raros, ele supera modelos semânticos densos. Por isso, nas arquiteturas modernas de RAG, sua utilização em conjunto com modelos vetoriais mostra que precisão lexical e semântica não são alternativas excludentes, mas sim sinais complementares que, quando combinados, produzem sistemas de recuperação mais completos e eficazes.

## Dense Retrieval (Denso)

## Introdução

Uma grande mudança de paradigma na recuperação de informações foi impulsionada não por novas teorias estatísticas, mas por uma revolução fundamental na forma como as máquinas representam a linguagem. O advento de modelos de aprendizado profundo pré-treinados para Processamento de Linguagem Natural (NLP) permitiu a transição de representações lexicais esparsas para representações semânticas densas. Essa revolução possibilitou que os sistemas de recuperação associassem documentos a consultas com base em seu significado, em vez de apenas pela sobreposição de palavras-chave. Um sistema poderia finalmente entender que uma consulta sobre "algoritmos de IA" é altamente relevante para um documento que discute "redes neurais", mesmo que as palavras sejam diferentes.

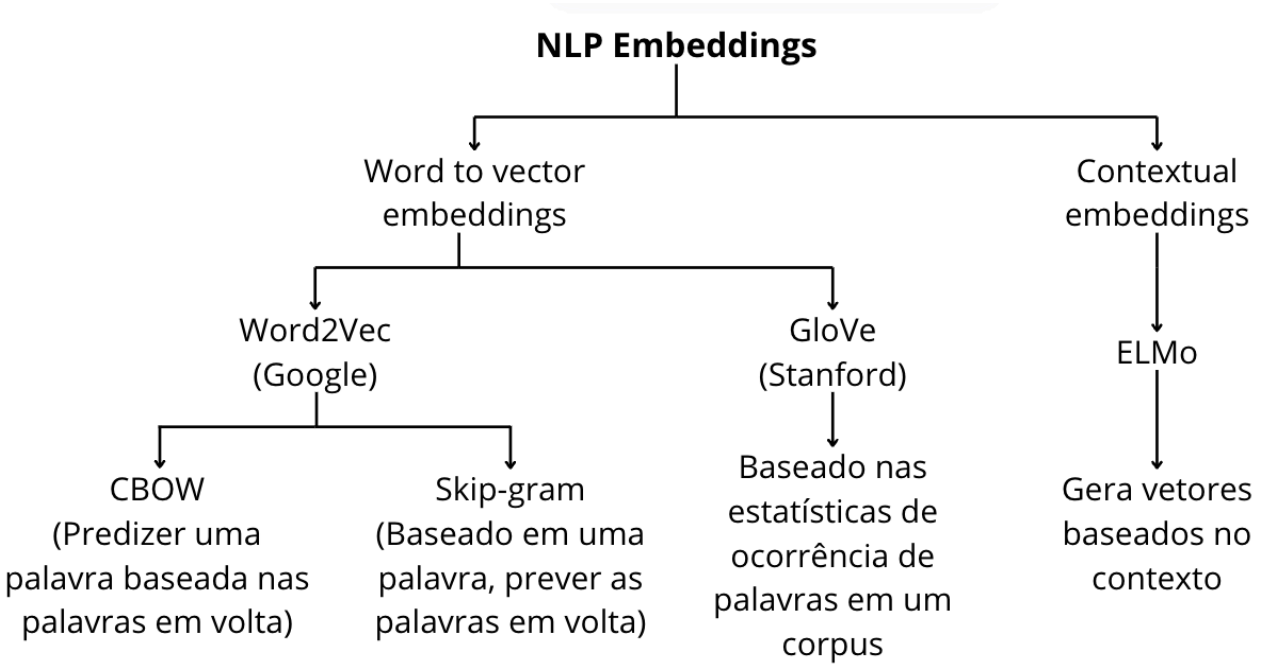
Nos métodos densos, tanto a consulta quanto os documentos são convertidos em vetores densos de dimensionalidade fixa usando modelos de linguagem pré-treinados ou outras redes neurais. Em vez de matching exato de termos, a similaridade é calculada por métricas como cosseno do ângulo ou produto interno entre vetores. Essa abordagem ganhou tração a partir de avanços em embeddings: primeiro word embeddings estáticos (e.g. Word2Vec em 2013, GloVe) que representavam palavras em vetores semânticos, e depois embeddings contextuais com modelos transformers como o BERT (Devlin et al., 2018). Com BERT e similares, tornou-se viável codificar consultas e passagens de texto em vetores comparáveis – técnica conhecida como Dense Passage Retrieval (DPR) desde o trabalho pioneiro de Karpukhin et al. (2020).

## Embeddings

Para a revisão da disciplina de “Processamento de Linguagem Natural” do Bacharelado em IA, eu criei, junto com mais 4 colegas, um material contendo todos os conteúdos ministrados na disciplina. Aproveitando esse estudo, eu recuperei as anotações sobre o tema "embeddings” que eu havia feito e complementei com mais algumas informações.

O material pode ser encontrado no link a seguir: Residência - Retrievers

Dentro dele, está um estudo da evolução histórica dos embeddings:



# APÊNDICE 4

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 24 de set. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

**THIAGO PEDROSO DE JESUS**

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as três primeiras Semanas, escolhi Retrieval-Augmented Generation (RAG) como tema da minha Residência, estudei a história da Recuperação de Informação (com linha do tempo até o surgimento do termo "RAG"), produzi um mapa mental da área e aprofundei a frente de Retrievers, focando inicialmente nos modelos de recuperação.

Para a quarta Semana, defini como prioridade continuar estudando Retrievers com o estudo de Chunking (segmentação de documentos), bem como uma revisão das possíveis fontes de dados e suas implicações para indexação e busca.

Dessa forma, realizei as seguintes atividades durante a Semana:

- Estudo das principais técnicas de Chunking
  - Busquei por guias técnicos que descrevem e classificam as principais práticas, como por exemplo o blog "Finding the Best Chunking Strategy for Accurate AI Responses | NVIDIA Technical Blog", escrito por Steve Han, um especialista em RAG pela NVIDIA;
  - A partir dos guias, mapeei técnicas simples (como segmentação fixa e segmentação recursiva) e técnicas mais avançadas (como segmentação semântica e segmentação baseada em LLMs);
  - Relacionei os estudos com os três tipo de fontes de dados possíveis (estruturados, não estruturados e semiestruturados);
  - Anotações: Residência - Retrievers .

- Na primeira Semana eu havia pontuado que uma das aplicações que me motivaram a estudar RAG foi o Cursor. Portanto, entendi que esse era um bom momento para entender como eles fazem chunking e indexação na prática.
  - Estudo do blog "How Cursor Indexes Codebases Fast - by Engineer's Codex", que explica, usando trechos documentação do Cursor, como ele consegue eficiência na indexação de código;
  - Anotações: Residência - Indexação Cursor

- Com o objetivo de consolidar o estudo na frente de Retrievers, expandi o meu mapa mental

produzido inicialmente para incorporar os novos conceitos.
    - Link do mapa mental: Link para o mapa

## Planejamento: [descrever o que pretende fazer para realizar a próxima ENTREGA]

Após duas Semanas de estudo na frente de Recuperação (Retrieval) do RAG, o próximo é avançar para o estudo de Geração.

Portanto, meu objetivo é:

- Estudar a função dos LLMs como geradores em pipelines RAG, analisando como eles utilizam o contexto recuperado para formular as respostas;
- Mapear os principais desafios associados à fase de Geração, como alucinações, dificuldade em sintetizar informações conflitantes e a perda de contexto (o problema do *"lost in the middle"*).

Além disso, como estudo em paralelo, pretendo fazer a leitura do artigo "[2508.21038] On the Theoretical Limitations of Embedding-Based Retrieval" que foi proposto para discussão no grupo de estudos em NLP.

Como resultado do meu estudo, pretendo:

- Produzir um documento com anotações e guias sobre o papel do Gerador, as abordagens de personalização e os desafios mapeados.
- Expandir o mapa mental sobre RAG, detalhando o pilar de "Geração" e conectando-o aos conceitos já estudados.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Estudo de Chunking

## DEFINIÇÃO

Em termos simples, chunking é o processo de dividir documentos grandes em pedaços menores e mais gerenciáveis, chamados de “chunks”. A forma como você divide seus documentos afeta a capacidade do seu sistema de encontrar informações relevantes e fornecer respostas precisas. Quando um sistema RAG tem um desempenho ruim, o problema geralmente não é o recuperador, mas sim os chunks. Mesmo um sistema de recuperação perfeito falha se pesquisar dados mal preparados.

O desafio fundamental do chunking reside em encontrar um equilíbrio:

- Chunks muito grandes tendem a misturar múltiplas ideias, criando um embedding "ruidoso" que não representa claramente um único tópico, dificultando a recuperação. Além disso, grandes entradas de contexto podem diluir a atenção do LLM, prejudicando o desempenho.
- Chunks muito pequenos podem falhar em fornecer contexto suficiente. Se um chunk não fizer sentido quando lido sozinho, será difícil para o LLM gerar uma resposta útil.

O Ponto Ideal de Chunking é alcançado ao preservar o "fluxo de pensamento" do autor, criando chunks que são pequenos o suficiente para uma recuperação precisa, mas completos o suficiente para dar ao LLM contexto total.

## FONTES UTILIZADAS


- Chunking Strategies to Improve Your RAG Performance | Weaviate
- Late Chunking: Balancing Precision and Cost in Long Context Retrieval | Weaviate
- Late Chunking in Long-Context Embedding Models
- Finding the Best Chunking Strategy for Accurate AI Responses | NVIDIA Technical Blog
- 11 Chunking Strategies for RAG — Simplified & Visualized | by Mastering LLM

(Large Language Model)
- Chunking Strategies for LLM Applications | Pinecone
- [2410.13070] Is Semantic Chunking Worth the Computational Cost?

## TIPOS DE CHUNKING

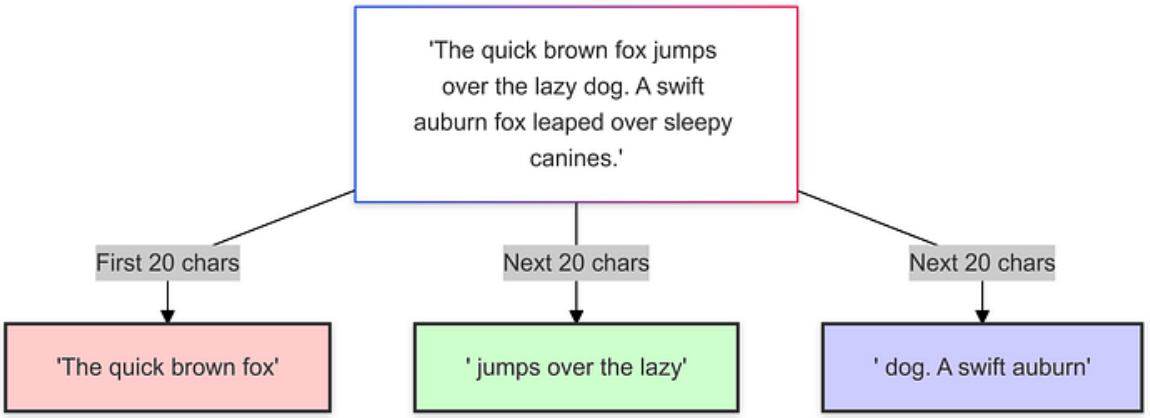
Cada método oferece vantagens únicas e é adequado para casos de uso específicos. Vamos entender cada um desses métodos de fragmentação em detalhes, comparar diferentes estratégias de fragmentação, como escolher a estratégia de fragmentação correta e entender as melhores práticas para implementar a fragmentação no RAG.

### Técnicas Simples / Estruturadas

#### Segmentação fixa

Nesta estratégia, o texto é dividido em partes de comprimento pré-definido (por número de tokens, caracteres ou palavras). É um método simples e uniforme, fácil de implementar, produzindo trechos de tamanho consistente. Por exemplo, pode-se definir chunks de 500 palavras ou 1000 caracteres.

A dica é escolher um tamanho de chunk apropriado (chunk size): deve ser pequeno o suficiente para facilitar a busca precisa, mas grande o bastante para conter uma ideia completa. Muitas implementações usam ~200 a 300 tokens como base, ajustando conforme o caso. Também é comum combinar segmentação fixa com janelas sobrepostas (chunk overlap) para mitigar a perda de contexto em fronteiras arbitrárias




Vantagens:

- Simplicidade e a facilidade de indexação

Desvantagens:

- Trechos fixos podem cortar frases ou ideias ao meio, levando à perda de contexto.
- Informações cruciais podem ficar divididas em dois chunks diferentes, prejudicando a recuperação relevante.

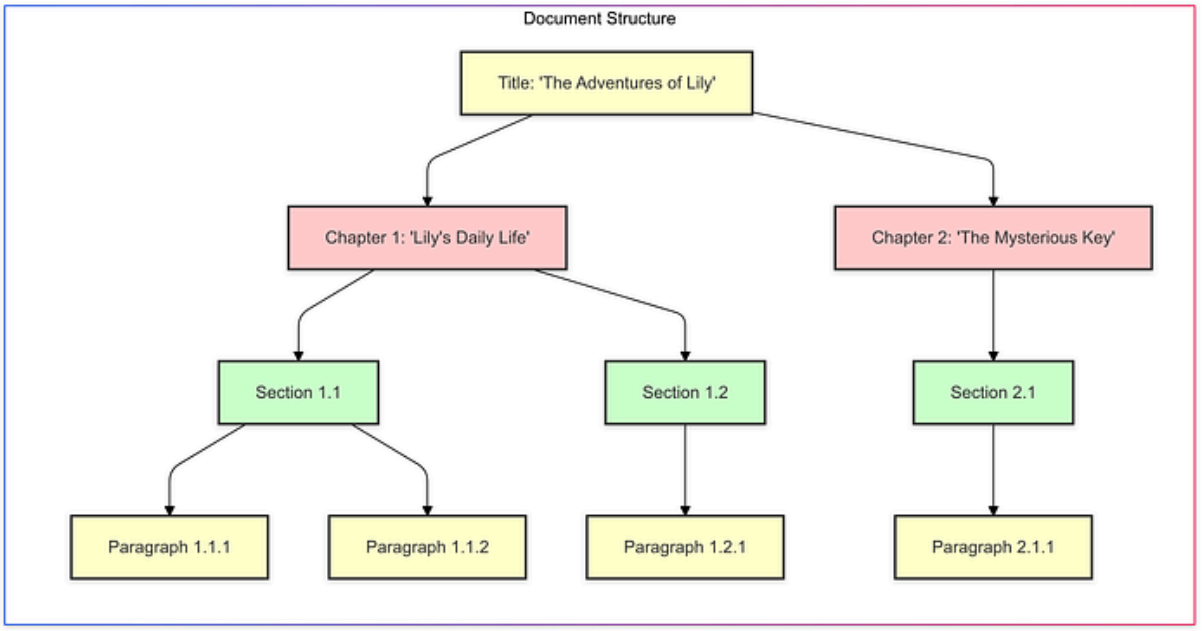
## Segmentação recursiva

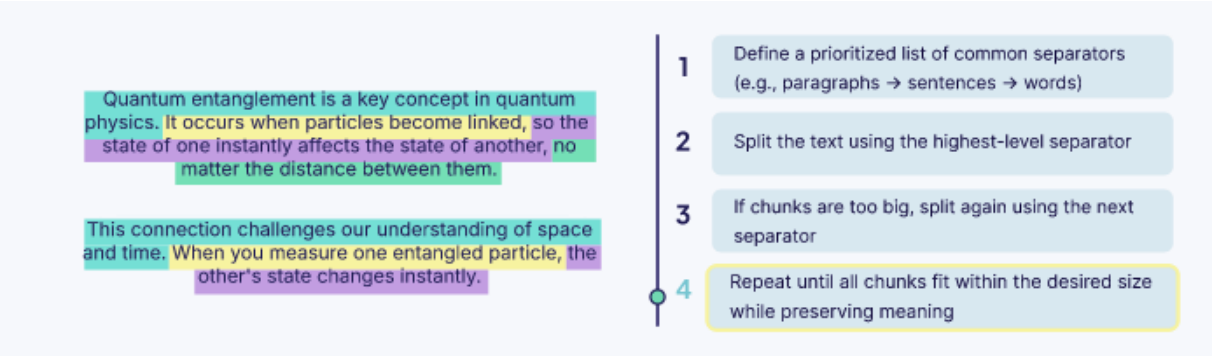
A Segmentação Recursiva funciona quebrando o texto progressivamente em chunks menores usando uma ordem hierárquica de delimitadores ou separadores, como títulos, subtítulos, parágrafos e frases

O mecanismo tipicamente utiliza uma lista priorizada de separadores. O algoritmo tenta primeiro dividir o texto pelo separador de mais alta prioridade (por exemplo, quebras de linha duplas para parágrafos).

Se algum chunk resultante for considerado muito grande (ou seja, exceder o tamanho máximo de chunk), o algoritmo aplica recursivamente o próximo separador na lista (por exemplo, quebras de linha simples para frases) a esse chunk específico.

Dica de Implementação: Para mitigar a potencial perda de contexto e a complexidade, sugere-se usar a estrutura do documento (como tags HTML) para identificar os níveis hierárquicos e armazenar metadados sobre a posição de cada chunk na hierarquia para facilitar a contextualização durante a recuperação.




Vantagens:

- Preservação do Contexto Hierárquico: O método mantém as relações estruturais do documento.
- Adaptação Estrutural: Ele se adapta à organização natural do texto, mantendo as unidades estruturalmente relacionadas juntas e assegurando que os chunks

retenham a estrutura do seu formato original.

- Evita Cortes Abruptos: A estratégia evita os cortes abruptos que ocorrem no meio de frases ou ideias, o que é um desafio comum na fragmentação de tamanho fixo (Fixed-Length Chunking).
- Escalabilidade: É eficaz para trabalhar com textos muito grandes.
- Equilíbrio: É considerada uma escolha padrão sólida porque respeita a organização natural do texto, sendo um "ótimo meio-termo" entre a divisão sempre por um caractere específico e o uso de um divisor mais semântico, ao mesmo tempo que garante, quando possível, tamanhos de chunk fixos.

Desvantagens:

- Implementação Complexa: Requer lidar com múltiplos níveis de estrutura do texto, o que aumenta a complexidade de implementação. A complexidade geral desta estratégia é classificada como Média.
- Potencial Perda de Contexto: Mesmo utilizando delimitadores hierárquicos, os chunks menores resultantes ainda podem perder contexto se o gerenciamento não for adequado.

## Fragmentação Baseada na Estrutura do Documento

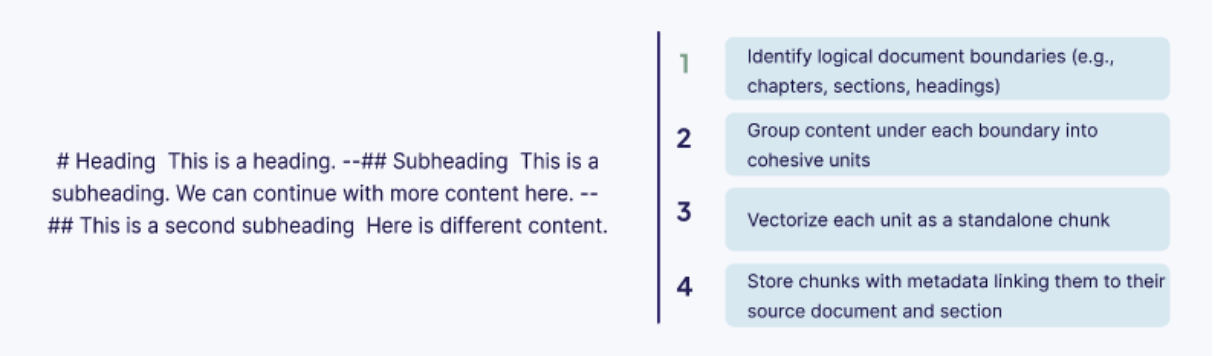
Em vez de depender de contagem de caracteres ou separadores genéricos, este método usa os elementos específicos do formato do documento para parsing e criação de unidades lógicas de informação. Alguns exemplos:

• Markdown: A fragmentação ocorre por cabeçalhos (como # ou ##) para delinear seções ou subseções.

• HTML: O texto é dividido por tags como <p> (parágrafos) ou  para preservar os blocos de conteúdo lógicos da página web.

• Código de Programação: A divisão pode ocorrer por unidades lógicas de código, como funções ou classes (e.g., def em Python).

• PDF e LaTeX: Após o pré-processamento, a divisão é feita por cabeçalhos, parágrafos, tabelas ou outros elementos estruturais.




Vantagens:

- Coerência Lógica: Os chunks permanecem alinhados com a organização lógica do documento, o que, por sua vez, geralmente se correlaciona com o significado semântico.
- Preservação da Estrutura: O método preserva a estrutura original do conteúdo durante a criação do chunk, resultando em chunks mais semântica e contextualmente relevantes.
- Divisão Apropriada: A divisão por classes ou funções em código de programação, por exemplo, garante unidades lógicas completas.

Desvantagens:

- Requer Pré-processamento (para certos formatos): Documentos como PDFs exigem uma etapa de pré-processamento (como o OCR ou a conversão para um formato estruturado como Markdown) para obter um texto limpo e estruturado antes que a fragmentação possa ser aplicada.
- Ineficácia em Documentos Não Estruturados: Esta estratégia é melhor para documentos altamente estruturados, onde o formato pode facilmente definir separações lógicas.
- Limitações de Desempenho (Fragmentação por Seção vs. Página): Em avaliações comparativas, a fragmentação por nível de seção (Section-level

chunking) foi superada pela fragmentação por nível de página (Page-level chunking) na média de vários datasets. O Page-level chunking alcançou a maior precisão média geral e o desempenho mais consistente nos testes. (Fonte: Relatório da NVIDIA)

## Técnicas Avançadas / Contextuais

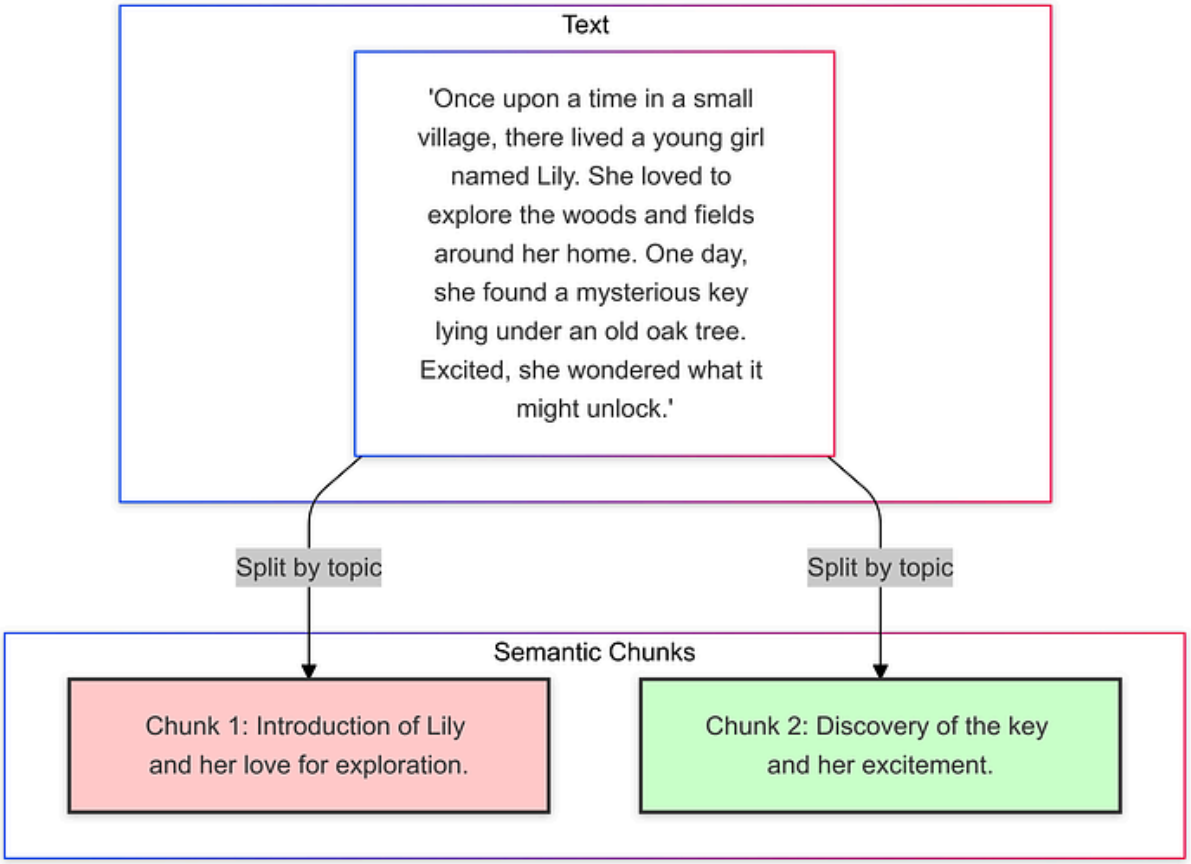
### Semantic Chunking

O Semantic Chunking, também conhecido como Context-Aware Chunking, é uma estratégia avançada de chunking que tem como objetivo superar as limitações das técnicas baseadas em regras fixas, como o fixed-size chunking, dividindo o texto com base no significado inerente

O Semantic Chunking utiliza embeddings ou modelos de aprendizado de máquina (machine learning) para segmentar o texto com base na similaridade semântica.

O processo geralmente envolve as seguintes etapas:

1. Segmentação de Sentenças: O texto é primeiro quebrado em sentenças individuais.
2. Geração de Embeddings: Cada sentença ou grupo de sentenças (incluindo o contexto circundante) é convertido em um vetor de embedding.
3. Análise de Similaridade: O sistema compara a distância semântica (dissimilaridade) entre esses embeddings de sentenças ou grupos de sentenças.
4. Formação de Chunks: Pontos de quebra (chunk boundaries) são definidos onde a similaridade semântica diminui drasticamente, indicando uma mudança de tema ou tópico.




Dessa forma, o Semantic Chunking adapta-se à estrutura e ao conteúdo inerentes do texto, resultando em chunks altamente coerentes.

Vantagens:

- Consultas Complexas: É ideal para consultas que exigem uma compreensão profunda, como aquelas em manuais técnicos ou artigos acadêmicos.
- Preservação do Contexto: É uma das estratégias recomendadas quando o objetivo principal é preservar o contexto.
- Textos Não Estruturados e Densos: É recomendado para textos densos e não estruturados (como trabalhos acadêmicos, documentos legais ou longas histórias) onde se deseja preservar o contexto semântico completo de uma ideia. É útil quando os limites semânticos não se alinham perfeitamente com a estrutura do documento, como parágrafos ou cabeçalhos

Desvantagens:

- Exige complexidade: Requer modelos avançados de NLP e mais recursos computacionais.
- Demora mais: A análise semântica pode ser demorada.

O artigo [2410.13070] Is Semantic Chunking Worth the Computational Cost? teve por objetivo principal determinar se os benefícios de desempenho obtidos pelo chunking semântico justificam os custos computacionais adicionais em comparação com a abordagem mais simples de fixed-size chunking. A conclusão é que não vale a pena e isso ocorre pelos seguintes motivos:

1. Os benefícios do fatiamento semântico em diversos cenários são inconsistentes e, muitas vezes, insuficientes para justificar o custo computacional adicionado.
2. Em datasets que refletem melhor documentos do mundo real (não sintéticos), o Fixed-size Chunker frequentemente apresentou desempenho superior.
3. Nas tarefas de recuperação de evidências e de geração de respostas, as diferenças de desempenho entre as estratégias de fatiamento foram mínimas, não indicando uma clara vantagem para o fatiamento semântico.
4. O impacto da estratégia de fatiamento em si é frequentemente ofuscado por outros fatores, como a qualidade dos modelos de embedding.

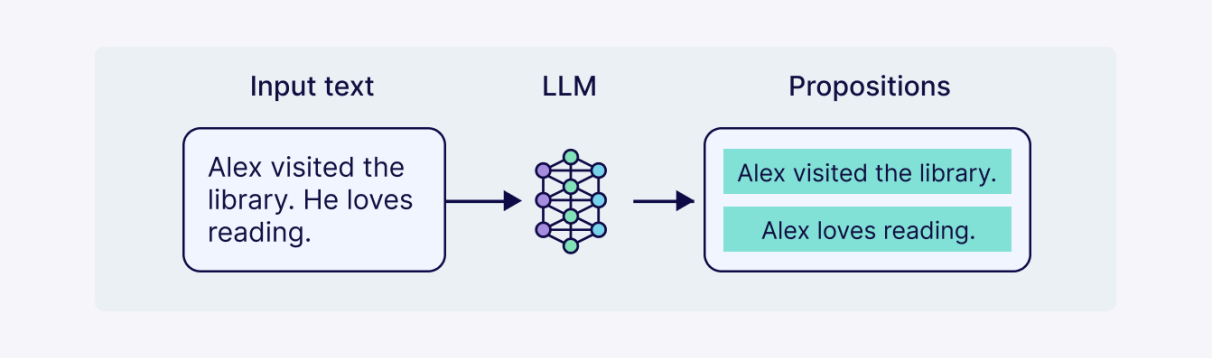
## LLM-Based Chunking

Emprega um LLM para analisar o texto e sugerir limites de chunk. Este processo pode ser realizado de várias maneiras:

1. Identificação de Proposições: O LLM pode quebrar o texto em declarações lógicas claras.
2. Resumo e Destaque: Pode resumir seções ou destacar pontos-chave para garantir que a informação mais relevante seja capturada nos chunks

resultantes.

3. Segmentação Inteligente: A técnica aproveita a capacidade de compreensão do LLM para criar chunks significativos.




Vantagens:

- Segmentação Inteligente e Adaptativa: A estratégia alavanca a compreensão do LLM para criar chunks significativos. Ela é adaptável e pode lidar eficazmente com conteúdo diverso e não estruturado.
- Preservação Semântica Precisa: Os chunks preservam o significado semântico de forma mais precisa do que os métodos tradicionais.
- Enriquecimento do Contexto: O LLM pode gerar chunks que resumem ou destacam ideias-chave.

Desvantagens:

- Intensidade Computacional: É o método mais intensivo em termos de computação, exigindo recursos significativos para processar o documento inteiro.
- Custo Elevado: Devido aos gastos computacionais, o custo pode não ser prático para aplicações em larga escala.
- Lentidão no Processamento: É o método mais lento em comparação com outras técnicas de chunking.

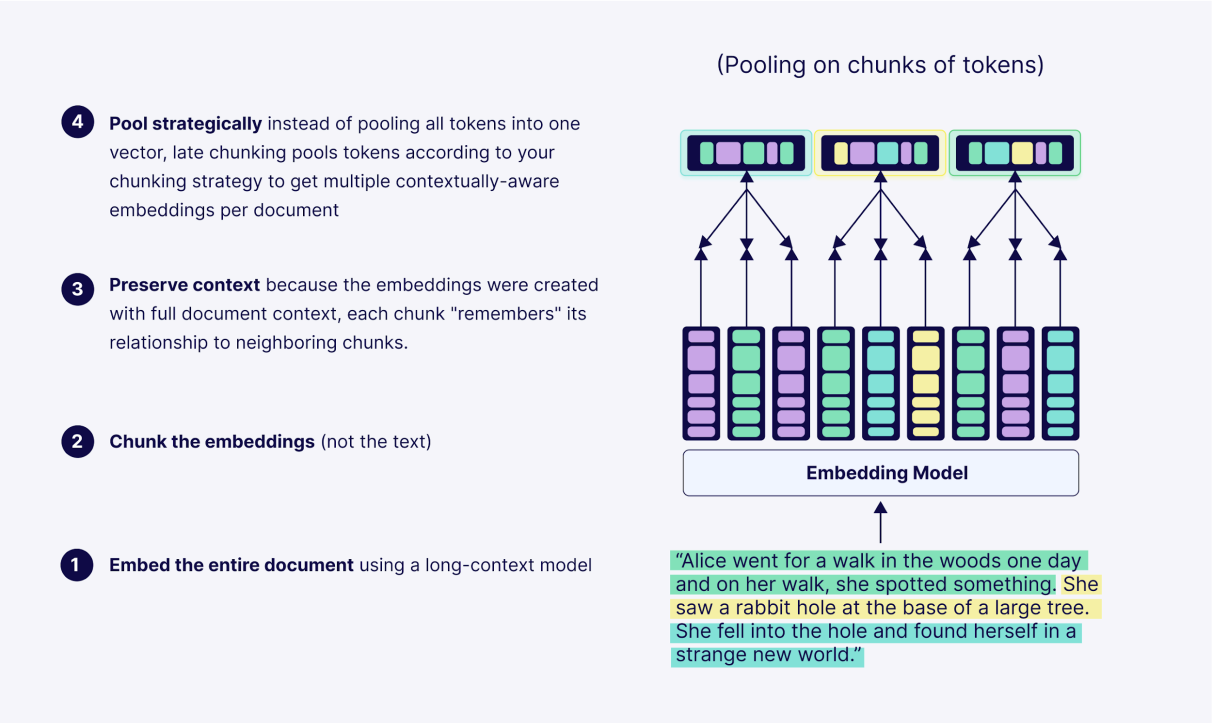
# Late Chunking

O Late Chunking (Late Chunking: Balancing Precision and Cost in Long Context Retrieval | Weaviate) é uma técnica avançada de segmentação que busca resolver um problema comum em outras estratégias de chunking: a perda de contexto.

Em métodos tradicionais, a divisão do documento é feita primeiro, isolando cada chunk antes da criação do embedding. No Late Chunking, o processo é invertido para garantir que cada segmento retenha o conhecimento do contexto completo do documento.

Como o Late Chunking Funciona:

1. Embedding do Documento Inteiro: Em vez de dividir primeiro, o sistema começa alimentando o documento inteiro em um modelo de embedding de contexto longo.
2. Criação de Embeddings de Token: Este processo inicial cria embeddings detalhados em nível de token que compreendem a imagem completa e o contexto integral do documento.
3. Divisão em Chunks: A divisão do documento em chunks (segmentos menores) ocorre somente depois que os embeddings de token com o contexto total são gerados.
4. Geração de Embeddings do Chunk: Para criar o embedding final de cada chunk, o sistema simplesmente calcula a média dos embeddings de token relevantes que caem dentro daquele chunk. Como os embeddings de token foram criados considerando todo o contexto do documento, o chunk resultante retém o contexto sobre o documento inteiro.




Vantagens:

- Mitigação da Perda de Contexto: Resolve o problema comum de context loss (perda de contexto) que ocorre quando os chunks são isolados antes da incorporação.
- Retenção do Contexto Global: Cada chunk retém o contexto sobre o documento inteiro, pois seu embedding é criado pela média dos embeddings de token gerados com o documento completo
- Ideal para Documentos Referenciais: Ajuda a capturar conexões entre diferentes partes do documento que métodos regulares perderiam (ex: documentos técnicos, papers de pesquisa ou textos legais com referências cruzadas)

Desvantagens:

- Latência no Primeiro Acesso: Se o Late Chunking for considerado uma forma de

Post-chunking, ele pode introduzir latência no primeiro acesso a um documento, embora os resultados possam ser armazenados em cache para acessos subsequentes.

- Requisitos de Infraestrutura: Requer decisões e suporte de infraestrutura adicionais.

# APÊNDICE 5

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 1 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

**THIAGO PEDROSO DE JESUS**

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as quatro primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, estudei a história da Recuperação de Informação, produzi um mapa mental da área e aprofundei a frente de Retrievers, explorando tanto modelos de recuperação quanto estratégias de chunking e indexação.

Para a quinta Semana, defini como prioridade iniciar os estudos da frente de **"Geração"**, a fim de compreender os principais desafios que surgem na utilização de longos contextos em LLMs nas pipelines de RAG.

As atividades desenvolvidas foram:

- Participação em grupo de estudos de NLP
  - Estive presente na apresentação e discussão do paper "[2508.21038] On the Theoretical Limitations of Embedding-Based Retrieval" no grupo de estudos em NLP. Embora seja um artigo avançado para meu momento de estudo, foi interessante acompanhar discussões da área e identificar pontos que pretendo aprofundar futuramente.

- Início dos estudos da frente de Geração
  - Para começar meus estudos, eu me fiz a leitura do tópico "Geração" do artigo "[2312.10997] Retrieval-Augmented Generation for Large Language Models: A Survey";
  - Anotações: Residência - Geradores;
  - Nessa leitura, ficou evidente que o grande problema quando geramos conteúdo com LLM a partir de documentos recuperados, é "Lost in the middle".

- Problema do "Lost in the Middle"
  - Leitura do artigo "[2307.03172] Lost in the Middle: How Language Models Use Long Contexts", que aborda a dificuldade dos LLMs em lidar com informações localizadas na parte central do contexto;
  - Anotações: Residência - Lost in the Middle;
  - O artigo aponta uma área de estudo para resolver esse problema: Ajustando o conteúdo recuperado (técnicas de ranking).

- Entendimento da área de “Ranking”
  - Fiz buscas no Google Scholar com o objetivo de encontrar um material que pudesse contextualizar melhor essa área (Os surveys sobre RAG estavam muito superficiais);
  - Nesse sentido, encontrei o survey “[2010.06467] Pretrained Transformers for Text Ranking: BERT and Beyond”, que possui 204 páginas;
  - Fiz a leitura apenas da introdução (20 páginas), que já apresenta:
    - Problemas do text ranking
    - Primórdios do text ranking
    - Desafios do Exact Match
    - Ascensão do Learning to Rank
    - Advento do Deep Learning
    - A Chegada do BERT
  - Anotações: Residência - História e fundamentos text ranking ;
  - Entendi que eu deveria entender melhor como os modelos BERT revolucionaram os ranqueadores e como essa área têm evoluído para os dias atuais.

- Modelos clássicos de ranking com BERT
  - Leitura do artigo “[1901.04085] Passage Re-ranking with BERT", que parece ter sido a primeira aparição de BERT para tarefas de ranking;
  - Leitura do artigo “[1910.14424] Multi-Stage Document Ranking with BERT”, que introduz monoBERT e duoBERT como arquiteturas fundamentais.
  - Anotações: Residência - monoBERT e duoBERT

- Rankers recentes aplicados ao RAG
  - Estudo do blog *“Enhancing RAG Pipelines in Haystack: Introducing DiversityRanker and LostInTheMiddleRanker | Towards Data Science”*, produzido por Vladimir Blagojevic, que mostra como um framework atual implementa métodos de ranking para otimizar a utilização da janela de contexto em pipelines de RAG.
  - Anotações: Residência - DiversityRanker e LostInTheMiddleRanker

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Levando em consideração uma sugestão feita pela banca na última apresentação, defini como prioridade para a próxima entrega aprofundar o entendimento sobre avaliação em RAG, mapeando como cada componente pode ser medido e comparado.

Nesse sentido, pretendo:

- Mapear métricas relevantes para cada frente do RAG, destacando quais datasets costumam ser utilizados em benchmarks da área;

- Construir um guia organizado de métricas e datasets que possa servir como referência para estudos futuros.

Além disso, a partir das leituras recentes e das discussões do grupo de estudos, percebi a relevância de duas derivações do BERT para a área de RAG: ModernBERT e ColBERT. Portanto, acredito que é um bom momento para:

- Realizar a leitura dos papers:
  - ModernBERT: [2412.13663] Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference
  - ColBERT: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT
- Evidenciar como esses modelos são avaliados e relacioná-los ao guia de métricas desenvolvido.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Geração com LLMs

## VISÃO GERAL

Em sistemas de geração aumentada por recuperação (RAG), não basta simplesmente recuperar muitos documentos e jogar tudo dentro do prompt do modelo de linguagem. Há um consenso crescente de que é preciso controlar melhor tanto o conteúdo recuperado quanto o próprio comportamento do LLM, para evitar ruído, vieses de contexto longo e perda de foco nas informações realmente relevantes.

Do lado do conteúdo, aparecem duas linhas principais: curadoria e compressão de contexto. A curadoria inclui re-ranking dos trechos recuperados para colocar primeiro o que é mais relevante (e potencialmente descartar o resto), seja com regras baseadas em métricas (relevância, diversidade, etc.) ou com modelos especializados de re-ranqueamento e até LLMs gerais atuando como "juízes" de relevância. Já a compressão/seleção de contexto combate a ideia equivocada de "quanto mais texto, melhor": usar modelos menores para remover tokens pouco informativos, treinar condensadores de informação ou pedir ao próprio LLM que avalie e descarte documentos pouco úteis são estratégias para reduzir o contexto sem perder o essencial, o que ajuda a evitar ruído, reduzir custo e mitigar problemas como o *"lost in the middle"*.

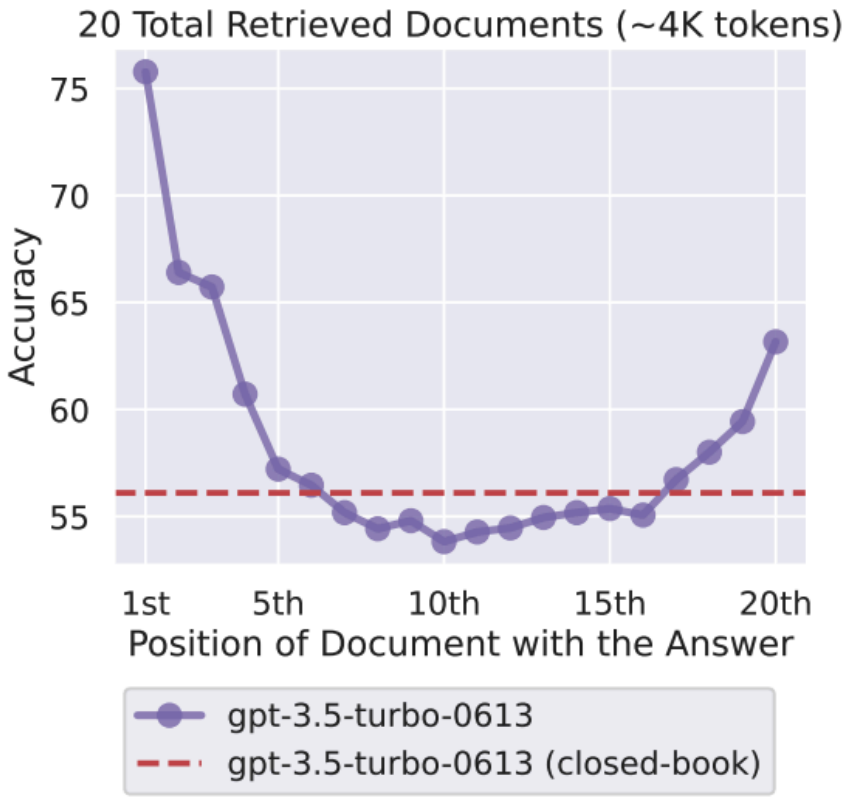
Do lado do modelo, o ajuste fino (fine-tuning) é usado para adaptar o LLM ao domínio, ao formato dos dados e ao estilo de saída desejado, bem como para alinhar melhor o modelo às necessidades do recuperador e às preferências humanas. Isso inclui treinar o modelo para lidar melhor com dados estruturados, usar aprendizado contrastivo para integrar melhor consulta e documentos, aplicar técnicas de aprendizado por reforço com feedback humano ou de outros modelos, e recorrer à destilação de modelos mais poderosos quando não se tem acesso direto a eles. Em muitos casos, o ganho vem justamente de alinhar, em conjunto, o LLM e o recuperador, de modo que o que é recuperado, ranqueado e apresentado no prompt seja exatamente o tipo de informação que o modelo sabe explorar melhor.

Mais detalhes podem ser encontrados no documento de apoio:

Residência - Geradores

## O PROBLEMA "LOST IN THE MIDDLE"

O artigo "*Lost in the Middle: How Language Models Use Long Contexts*" investiga um problema central para sistemas de geração aumentada por recuperação (RAG): apesar de modelos de linguagem aceitarem contextos longos, eles não conseguem usar essas informações de forma robusta, especialmente quando o trecho relevante está no meio do contexto. Os autores estudam esse fenômeno em dois cenários que lembram diretamente pipelines de RAG: (i) resposta a perguntas com múltiplos documentos, em que o modelo recebe uma pergunta e vários trechos de texto (como em buscadores generativos) e precisa encontrar o único documento que contém a resposta; e (ii) uma tarefa sintética de recuperação de pares chave-valor em JSON, onde basta "ler" o contexto e devolver o valor correto. Em ambos os casos, eles controlam cuidadosamente o comprimento do contexto e a posição da informação relevante (início, meio ou fim) e mostram uma curva em formato de U: o desempenho é mais alto quando a resposta está no começo ou no fim do prompt (viés de primazia e recência) e cai significativamente quando a informação fica "perdida" no meio.




Isso é especialmente crítico para RAG, porque muitas aplicações (busca generativa, QA sobre documentação, suporte ao usuário, etc.) funcionam como o cenário de múltiplos documentos: um retriever devolve k trechos e o modelo gera a resposta condicionado neles. O paper mostra que, mesmo em modelos de contexto estendido (como GPT-3.5 16K ou Claude 100K), não basta "caber" mais texto na janela: quando o documento relevante é colocado no meio do contexto, modelos como GPT-3.5 chegam a ir pior do que no modo closed-book (sem documento nenhum), o que significa que mais contexto pode atrapalhar em vez de ajudar. Em um estudo de caso com QA em domínio aberto, os autores ainda mostram que, à medida que se aumenta o número de documentos recuperados (por exemplo, de 20 para 50 trechos da Wikipedia), o recall do retriever continua subindo, mas a acurácia do modelo praticamente satura, indicando que o leitor (o LLM) não consegue aproveitar bem os documentos adicionais. Para RAG, isso reforça que simplesmente "colocar tudo no prompt" não é uma solução mágica; é preciso pensar em reranking, truncamento inteligente e organização do contexto para aproximar o conteúdo relevante do início da janela.

## O DESENVOLVIMENTO DO TEXT RANKING

Para entender melhor sobre o funcionamento do text ranking, recorri a um artigo chamado "[2010.06467] Pretrained Transformers for Text Ranking: BERT and Beyond", que detalha a sua história e fundamentos.

O texto apresenta o text ranking como peça central dos sistemas de acesso à informação: dado uma consulta, o sistema precisa ordenar textos (páginas, parágrafos, tweets, respostas em fóruns etc.) por relevância. Embora o exemplo mais óbvio seja a busca na web, essa mesma ideia aparece em várias tarefas: responder perguntas, encontrar perguntas semelhantes em comunidades online, filtrar fluxos contínuos de textos, recomendar conteúdos, selecionar trechos para módulos downstream (extração de informação, sumarização, tradução, verificação de fatos) e até montar conjuntos de treino via distant supervision e data augmentation. Em todos esses cenários, há sempre uma "query" (explícita ou implícita) e um conjunto de textos candidatos que precisa ser ranqueado.

Historicamente, o text ranking começou com indexação manual feita por bibliotecários, passou para representações automáticas baseadas em contagem de palavras e chegou a modelos clássicos como o vector space model e funções de pontuação baseadas em exact match, como BM25. Logo ficou claro o problema de vocabulary mismatch: usuário e autor do documento usam palavras diferentes para falar da mesma coisa. Isso motivou técnicas como expansão de consulta, expansão de documento e modelos estatísticos mais sofisticados, além do surgimento do learning to rank, que combina dezenas ou centenas de features (estatísticas de termos, sinais de clique, estrutura de página, grafos de links etc.) em modelos supervisionados. Depois, com o deep learning pré-transformers, surgiram modelos neurais de ranking que aprendiam representações contínuas de queries e documentos (modelos de representação) ou modelavam explicitamente as interações termo a termo (modelos de interação), abrindo caminho para semantic matching, mas ainda exigindo muitos dados rotulados e, em vários cenários públicos, sem superar de forma consistente as abordagens baseadas apenas em palavras-chave.

A virada acontece com modelos pré-treinados baseados em transformers, que combinam self-supervised pretraining em grandes corpus com fine-tuning leve para tarefas de ranking. Eles passam a ser usados principalmente de duas formas: como

rerankers em arquiteturas multiestágio (primeiro uma busca tradicional gera um conjunto candidato, depois o transformer refina e reordena) e como base de métodos de recuperação densa, em que queries e documentos são mapeados diretamente para vetores densos em um espaço semântico compartilhado. Isso traz ganhos grandes e replicáveis de efetividade, torna o uso de modelos neurais viável fora da indústria e impulsiona toda uma família de variantes e extensões. Ao mesmo tempo, permanecem desafios importantes: lidar com documentos longos (para além de frases isoladas), equilibrar eficácia e eficiência (latência, custo, tamanho de modelo e índice) e entender melhor como esses modelos realmente realizam matching de relevância. O texto conclui que, embora muitas técnicas já sejam maduras, ainda há um campo aberto de pesquisa em arquiteturas, treinamento e uso de transformers para text ranking.

## ESTUDO DE CASO: MONOBERT E DUOBERT

Um trabalho interessante nessa área é o "*Multi-Stage Document Ranking with BERT*". A partir da leitura, é possível ter um estudo de caso de como colocar BERT para ranquear documentos sem matar o sistema de busca. O problema inicial é simples: BERT melhora muito a qualidade dos resultados, mas é pesado demais para ser rodado em milhões de documentos a cada query. Então os autores montam um pipeline em etapas: primeiro, um buscador tradicional (BM25) faz o serviço "bruto" de recuperar muitos candidatos rapidamente; depois entra um BERT mais simples, o monoBERT, para reordenar esses candidatos individualmente; por fim, um duoBERT compara documentos em pares para fazer um ajuste fino na ordem final. Cada etapa pega menos documentos e usa um modelo mais "caro", o que permite equilibrar qualidade x tempo de resposta.

Nos experimentos com dois cenários reais de busca (MS MARCO e TREC CAR), esse arranjo mostra bem a lógica prática: BM25 já funciona razoavelmente, mas o monoBERT dá um salto grande na qualidade das respostas; o duoBERT ainda melhora um pouco mais, com um custo extra de tempo que pode ser controlado ajustando quantos documentos entram em cada fase. As análises mostram também casos concretos em que o modelo neural corrige erros óbvios da busca por palavra-chave (como rankings inflados só porque o termo da query se repete muito) e consegue captar melhor sinônimos e nuances de significado. No fim, o artigo mostra na prática como dá para usar modelos pesados como o BERT em sistemas de busca reais se você

os encaixa em uma arquitetura em cascata, em vez de tentar fazer tudo com um único modelo gigante.

# APÊNDICE 6

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 8 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as cinco primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, estudei a história da Recuperação de Informação, aprofundei a frente de Retrievers (modelos, chunking e indexação) e fiz os estudos da frente de Geração, explorando os desafios de longos contextos e o papel dos Rankers.

Para a sexta Semana, minha prioridade foi compreender como avaliar sistemas RAG, isto é, quais **métricas e benchmarks** são utilizados tanto em pesquisas acadêmicas quanto em aplicações práticas.

Dessa forma, realizei as seguintes atividades:

- Estudo de métricas e benchmarks
  - Com o objetivo de direcionar meus estudos sobre métricas, busquei no Google Scholar por materiais que já apresentassem uma revisão abrangente da área.
  - Encontrei e li o artigo **"**Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey **"** (Abril de 2025).
  - Foi suficiente para abranger tudo o que eu procurava sobre avaliação, explorando assuntos como:
    - Métricas de Recuperação de Informação (IR),
    - Métricas de Processamento de Linguagem Natural (NLP),
    - Métodos de avaliação via LLM,
    - Métodos de avaliação de segurança e eficiência,
    - Principais benchmarks e conjuntos de dados
  - Anotações: Residência - Métricas e benchmarks.

- Estudo sobre o framework RAGAS
  - Dentro do próprio survey, encontrei menção ao RAGAS, um framework automatizado para avaliação de pipelines RAG, que eu já havia visto em alguns projetos práticos.
  - Por isso, realizei a leitura completa do paper **"**[2309.15217] Ragas: Automated Evaluation of Retrieval Augmented Generation**".**
  - Anotações: Residência - RAGAS.

- Leitura complementar de modelos de retrieval e ranking
  - Para complementar os estudos da Semana, realizei também a leitura de dois artigos que apresentam modelos muito usados em pipelines de RAG.
  - ModernBERT: [2412.13663] Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference.
  - Anotações: Residência - ModernBERT.
  - ColBERT: ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT.
  - Anotações: Residência - ColBERT.

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Na segunda Semana, eu havia dividido o estudo de RAG em três partes: **Recuperação**, **Geração** e **Integração**. Portanto, com o objetivo de fechar o ciclo de estudos, pretendo:

- Aprofundar a frente de **Integração**, entendendo como essas etapas se conectam em pipelines completas.
- Finalizar o meu mapa mental que eu venho desenvolvendo desde a segunda Semana.

Além disso, ao longo da minha trajetória anotei artigos e trabalhos que eu tinha interesse em estudar, mas que ainda não era o momento de explorar com profundidade. Pensando nisso, estou me planejando para:

- Retomar essas leituras e identificar possíveis temas de interesse.
- Escolher uma área que me chame atenção e planejar os primeiros experimentos práticos.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

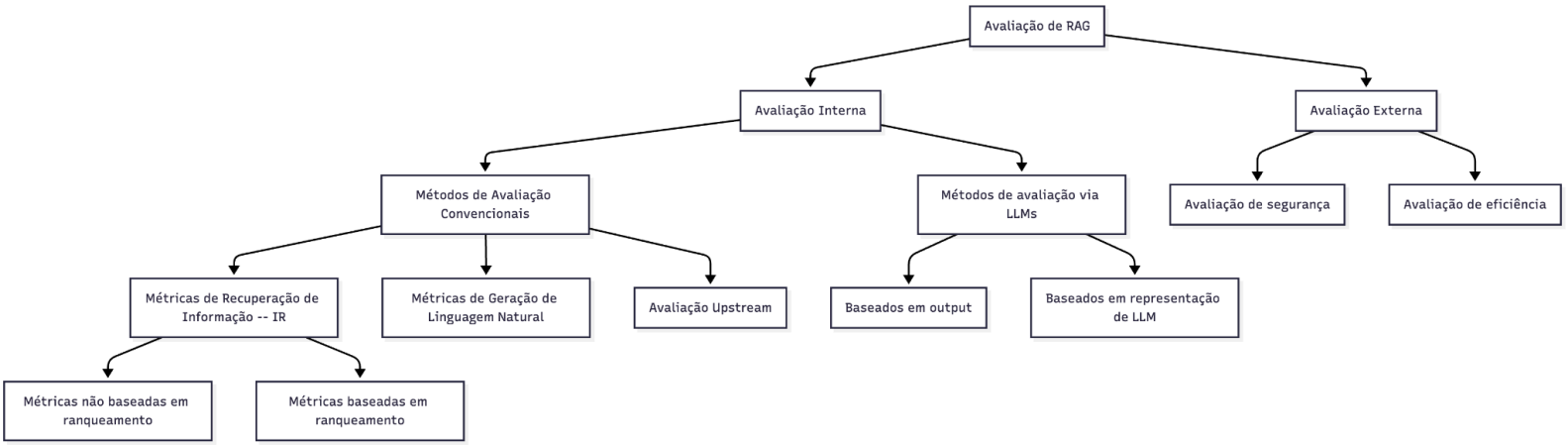
## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Métodos de avaliação

## VISÃO GERAL

O material Residência - Métricas e benchmarks, elaborado a partir do survey "Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey", organiza a avaliação de sistemas RAG em torno de duas grandes perspectivas: interna (focada em componentes) e externa (focada no sistema como um todo e em seu uso real).




## FORMAS DE AVALIAÇÃO

Na avaliação interna, o foco recai sobre dois problemas centrais: (i) a recuperação de documentos relevantes (retrieval), e (ii) a geração de respostas alinhadas à consulta e às evidências (generation). Para o módulo de recuperação, os alvos principais são: relevância (documentos ↔ consulta), abrangência/cobertura (quão bem o conjunto recuperado cobre a informação relevante) e corretude no ranqueamento de candidatos. Entram aqui as métricas clássicas de IR: accuracy/Hit@K, precision@K, recall@K, F1, além das métricas baseadas em ranqueamento, como MRR, MAP e NDCG, que levam em conta a posição dos

documentos relevantes. Para o módulo de geração, os alvos são: relevância (resposta e consulta), fidelidade (resposta e documentos recuperados) e corretude em relação à resposta de referência. Nessa parte, usam-se métricas de NLG como Exact Match, ROUGE, BLEU, METEOR, BERTScore, bem como medidas de similaridade semântica, cobertura de conteúdo e, em alguns casos, perplexidade para analisar a qualidade linguística. O estudo também enfatiza a avaliação upstream, envolvendo chunking e embeddings: estratégias de segmentação do corpus podem ser avaliadas tanto por métricas intrínsecas (cobertura de palavras-chave, número de tokens até cobrir o contexto necessário) quanto por impacto em métricas downstream, enquanto modelos de embedding são comparados em benchmarks como MTEB/MMTEB, que mostram que nenhum modelo único é ótimo em todas as tarefas.

Com a popularização dos LLMs, ganham importância os métodos de avaliação via LLM, divididos em abordagens baseadas em output e baseadas em representação. Na primeira, o próprio LLM atua como avaliador (“LLM-as-a-judge”), recebendo consulta, contexto e resposta e atribuindo julgamentos de relevância, completude, fidelidade ou risco, como em frameworks do tipo RAGAS e avaliações semelhantes (“a resposta está suportada pelo contexto?”, “a resposta cobre todos os pontos?”). Também surgem métricas que decompõem o texto em fatos atômicos, extraem key points do contexto ou estimam a “confiança interna” do modelo na resposta. Na segunda linha, métodos baseados em representação analisam embeddings e estados internos do LLM para medir o alinhamento entre resposta, documentos e conhecimento do modelo, seja para verificar consistência com as evidências, seja para estudar fenômenos como filtragem de ruído, conhecimento de cauda longa ou estrutura de clusters no espaço latente.

Na avaliação externa, o RAG é visto como um sistema completo, e a ênfase recai sobre segurança (safety) e eficiência. Em segurança, o estudo destaca: (i) robustez a contextos enganosos ou irrelevantes, medindo o quanto respostas se degradam diante de documentos incorretos; (ii) factualidade, monitorando hallucinations, acurácia factual e correção das citações em relação às fontes; (iii) ataques adversariais, como envenenamento da base de conhecimento, manipulação do ranqueamento ou uso de documentos-gatilho, avaliados por taxas de sucesso de ataque e falhas de recuperação; (iv) privacidade, com métricas de vazamento de PII e sucesso de ataques de extração; (v) fairness e vieses, verificando disparidades entre grupos e presença de estereótipos; e (vi) transparência e rastreabilidade, analisando qualidade das explicações, facilidade

de seguir do output às fontes e precisão das citações. Em eficiência, as principais métricas são de latência (Time to First Token e tempo total de resposta) e de custo, incluindo uso de tokens, recursos de infraestrutura, armazenamento e esforço operacional. São propostas ainda razões de custo-benefício, como cost-effectiveness e retrieval precision ROI, além de esquemas que permitem ajustar explicitamente o trade-off entre custo e acurácia conforme o caso de uso.

Por último, o estudo reúne uma visão geral de benchmarks, datasets e frameworks de avaliação específicos para RAG. Os conjuntos de dados vão desde benchmarks clássicos de QA até corpora dinâmicos (por exemplo, notícias recentes) e domínios especializados (direito, saúde, finanças), além de coleções sintéticas com premissas falsas para testar robustez e tratamento de desinformação. Os frameworks combinam métricas tradicionais e métricas baseadas em LLM, alguns com foco em avaliação geral de qualidade e outros especializados em segurança ou em domínios específicos. Em conjunto, essas dimensões (métricas internas, avaliação via LLM, segurança, eficiência e benchmarks) formam a base conceitual necessária para projetar, justificar e comparar a avaliação de sistemas RAG em trabalhos acadêmicos e aplicações práticas.

## ESTUDO DE CASO: FRAMEWORK RAGAS

A premissa de partida é que LLMs sozinhos não resolvem bem dois problemas: não sabem sobre fatos posteriores ao treinamento e têm dificuldade com conhecimento raro. O paradigma RAG ataca isso recuperando trechos de uma base textual e passando esse contexto para o modelo gerar a resposta. Porém, avaliar RAG na prática é complicado: nem sempre há respostas de referência, muitas aplicações não são simplesmente QA extrativo e, em cenários reais, o time de produto quer iterar rápido sem depender de anotações humanas. É exatamente aí que entra o RAGAS, proposto em "[2309.15217] Ragas: Automated Evaluation of Retrieval Augmented Generation", como um framework de avaliação automatizada, sem necessidade de ground truth e pensado para se encaixar em pipelines construídos com ferramentas como LangChain e LlamaIndex.

O RAGAS define três dimensões centrais de qualidade em RAG: fidelidade, relevância da resposta e relevância do contexto. A fidelidade pergunta se o que a resposta afirma realmente pode ser inferido do contexto recuperado; para isso, o LLM primeiro decompõe a resposta em pequenas afirmações e, depois, verifica para cada

uma se ela é suportada pelo contexto. A relevância da resposta foca em “quão bem isso responde à pergunta”, sem olhar para factualidade: o modelo gera perguntas que teriam aquela resposta e compara semanticamente essas perguntas com a pergunta original, usando embeddings para estimar alinhamento. Já a relevância do contexto mede o quão enxuto e focado é o material recuperado: dado “pergunta + contexto”, o LLM extrai apenas as sentenças que realmente ajudam a responder, e a métrica penaliza trechos longos cheios de ruído. Todas essas avaliações são feitas por prompts sobre um LLM (no artigo, gpt-3.5-turbo-16k), o que permite usar o framework mesmo quando não se tem acesso às probabilidades internas do modelo principal.

Para validar o framework, os autores constroem o WikiEval, um conjunto de dados próprio com trios pergunta, contexto e resposta baseados em páginas da Wikipédia recentes, e o anotam manualmente nas três dimensões de qualidade. Eles então comparam o julgamento humano com as pontuações do RAGAS e com duas baselines simples: pedir ao ChatGPT para dar uma nota (0–10) ou escolher diretamente entre duas respostas/contextos. Os resultados mostram que as métricas do RAGAS se alinham melhor com os humanos do que esses baselines, especialmente em fidelidade e relevância da resposta; relevância de contexto é a dimensão mais difícil tanto para o modelo quanto para as pessoas. No fim, o trabalho posiciona o RAGAS como uma ferramenta prática para avaliar e iterar pipelines RAG no mundo real, quando não se dispõe de labels nem de respostas de referência, mas ainda assim se deseja medir se o sistema está fundamentando bem suas respostas, respondendo o que foi perguntado e usando contexto de forma eficiente.

# APÊNDICE 7

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 15 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as seis primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, estudei a história da Recuperação de Informação, aprofundei as frentes de Recuperação e Geração, e compreendi os principais métodos de avaliação de sistemas RAG.

Para a sétima Semana, defini como prioridade o estudo da frente de Integração, buscando entender como as etapas de Recuperação e Geração se conectam dentro das pipelines de RAG e quais frameworks são mais utilizados na prática.

As atividades desenvolvidas foram:

- Leitura geral da frente de "Integração"
  - Realizei a leitura da seção dedicada à "Integração" no artigo " Retrieval-Augmented Generation for Large Language Models: A Survey", para compreender como os autores abordam essa área e quais as suas referências.
  - Anotações: Residência - Integração .

- Estudo comparativo de frameworks e métodos de RAG
  - Busquei por materiais que analisassem frameworks sob uma perspectiva prática e comparativa. Li o artigo "Maximizing RAG efficiency: A comparative analysis of RAG methods", que apresenta diferentes estratégias de integração e compara em eficiência e custo computacional.
  - Anotações: Residência - Maximizing RAG efficiency .

- Consolidação do mapa mental de RAG
  - Finalizei o mapa mental que eu vinha construindo desde a segunda Semana, alocando todos os principais pontos que eu estudei durante todo o meu processo até o momento.
  - É possível acessar em: Link mapa mental.

Por fim, tendo finalizado os estudos, busquei me aprofundar em alguma área dentro de RAG e começar os

experimentos práticos. Com esse objetivo, realizei:

- Organização e análise de artigos de interesse
  - Reuni todos os artigos que me despertaram interesse durante a Residência em uma tabela que os classifica por área e destaca sua tese principal.
  - Tabela: Residência - Artigos de Interesse.

- Escolha de uma área de estudo
  - Passei por cada artigo de interesse fazendo uma breve leitura.
  - Selecionei o trabalho "Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models" como um exemplo da área que eu gostaria de trabalhar.

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Com a consolidação do estudo teórico, o objetivo da próxima Semana será iniciar a transição para a parte prática da Residência.

Nesse sentido, pretendo:

- Finalizar a leitura do artigo "Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models";
- Identificar os recursos necessários para replicar seus experimentos (modelos, datasets e métricas);
- Iniciar os primeiros testes práticos com pipelines híbridas de recuperação e re-ranking.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

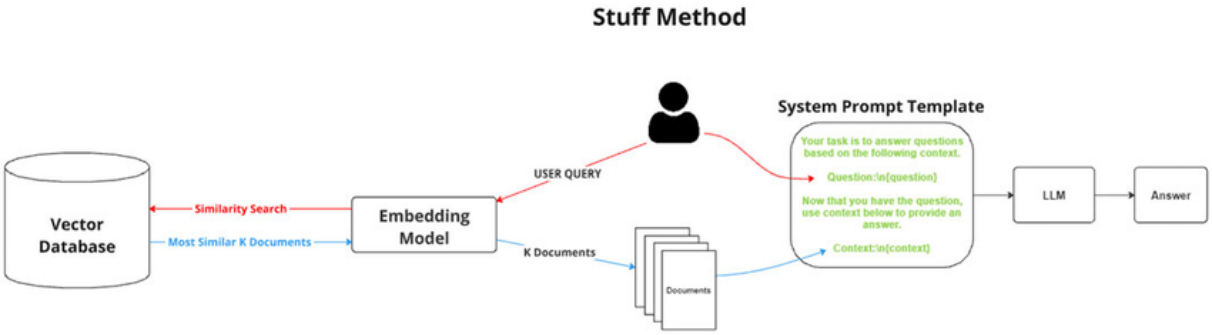
# Arquiteturas de RAG

## ARTIGO DE REFERÊNCIA

A partir da leitura do artigo "*Maximizing RAG efficiency: A comparative analysis of RAG methods*", é possível compreender de forma aprofundada os desafios e estratégias envolvidas na otimização de sistemas de RAG. O estudo parte da necessidade de equilibrar qualidade de resposta, custo computacional e eficiência operacional em pipelines que combinam LLMs, mecanismos de recuperação vetorial e diferentes formas de integração entre ambos.

## ARQUITETURAS AVALIADAS

### Método Stuff




**Conceito**

É o método mais direto e básico de retrieval-augmented generation. Após recuperar os K documentos mais relevantes a partir do repositório vetorial, todos são "empacotados" (stuffed) juntos e enviados como contexto em uma única chamada ao LLM.

**Fluxo**

1. O usuário envia uma pergunta.
2. O sistema busca os K documentos mais similares.

3. Todos os textos são concatenados em um único prompt.
4. O LLM responde considerando todo o contexto.

**Vantagens**

- Simplicidade e velocidade: apenas uma chamada ao LLM.
- Fácil implementação (ótimo para protótipos e chatbots rápidos).
- Menor sobrecarga lógica: nenhuma etapa adicional de filtragem ou fusão.

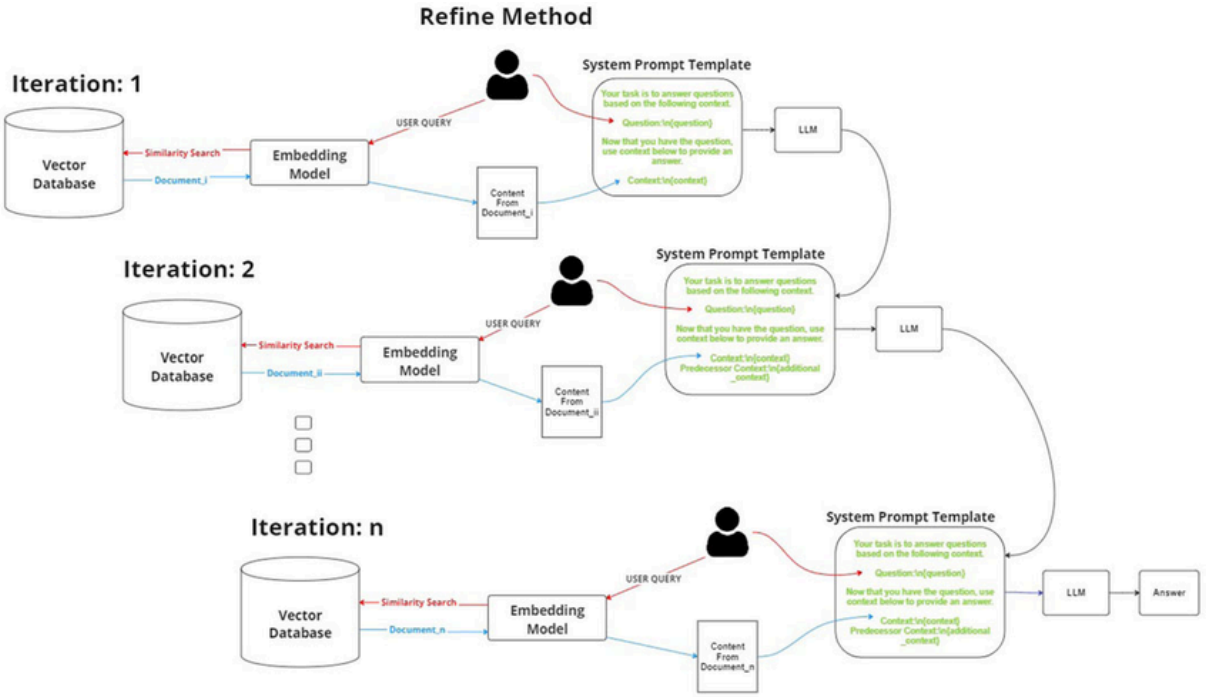
**Limitações**

- Alto custo de tokens, se o contexto for grande.
- Pode gerar ruído: partes irrelevantes confundem o LLM.
- Limitado pelo tamanho máximo de contexto do modelo.

**Quando usar**

- O conjunto de documentos é pequeno (poucos parágrafos).
- A precisão não precisa ser muito alta.
- A latência deve ser mínima.

# Método Refine




## Conceito

O método iterativo e cumulativo. O LLM processa um documento de cada vez, refinando a resposta à medida que novos contextos são introduzidos.

## Fluxo

1. O LLM gera uma resposta inicial com o primeiro documento.
2. Depois, para cada novo documento recuperado:
    - O modelo recebe a resposta anterior + novo documento.
    - Ele “refina” a resposta anterior, ajustando-a com base nas novas informações.
3. O processo continua até o último documento.

## Vantagens

- Mantém consistência e foco: cada passo revisa o anterior.
- Reduz chance de contradições entre fontes.
- Usa o contexto de forma mais “inteligente” do que Stuff.

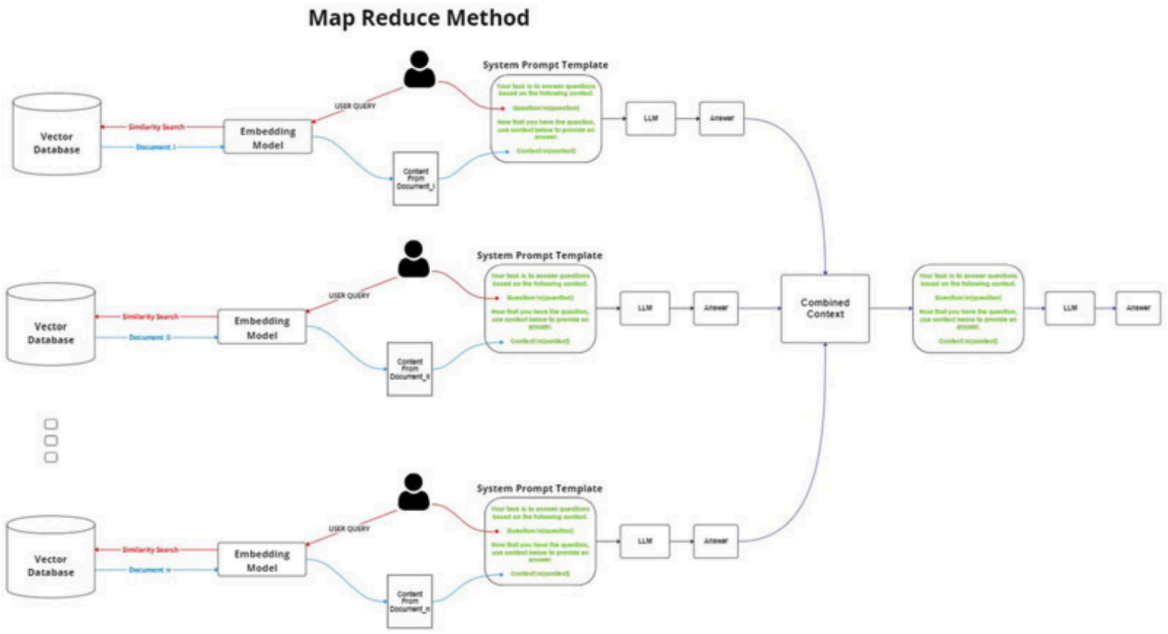
**Limitações**

- Múltiplas chamadas ao LLM ⇒ custo e tempo maiores.
- Pode “esquecer” detalhes iniciais ao longo das iterações.
- Propagação de erros: se o modelo interpretar algo errado cedo, o erro pode persistir.

**Quando usar**

- Quando a qualidade e coerência global são prioritárias.
- Casos de síntese incremental (ex: relatórios longos, resumos de várias fontes).

# Método Map-Reduce




## Conceito

Inspirado no paradigma de computação paralela “Map-Reduce” (Google, 2004). Divide o problema em duas fases:

- Map: cada documento é processado separadamente pelo LLM, gerando respostas parciais.
- Reduce: essas respostas são combinadas em uma resposta final (por concatenação, resumo ou votação).

## Fluxo

1. Para cada documento recuperado: o LLM responde à pergunta usando apenas aquele documento.
2. Depois, uma etapa de redução sintetiza as respostas locais em uma resposta única.

## Vantagens

- Permite processamento paralelo → mais eficiente em escala.
- Cada resposta parcial é “limpa” (sem interferência de outros contextos).
- Fusão final pode ser calibrada para diferentes metas (ex: sumarização, votação, consenso).

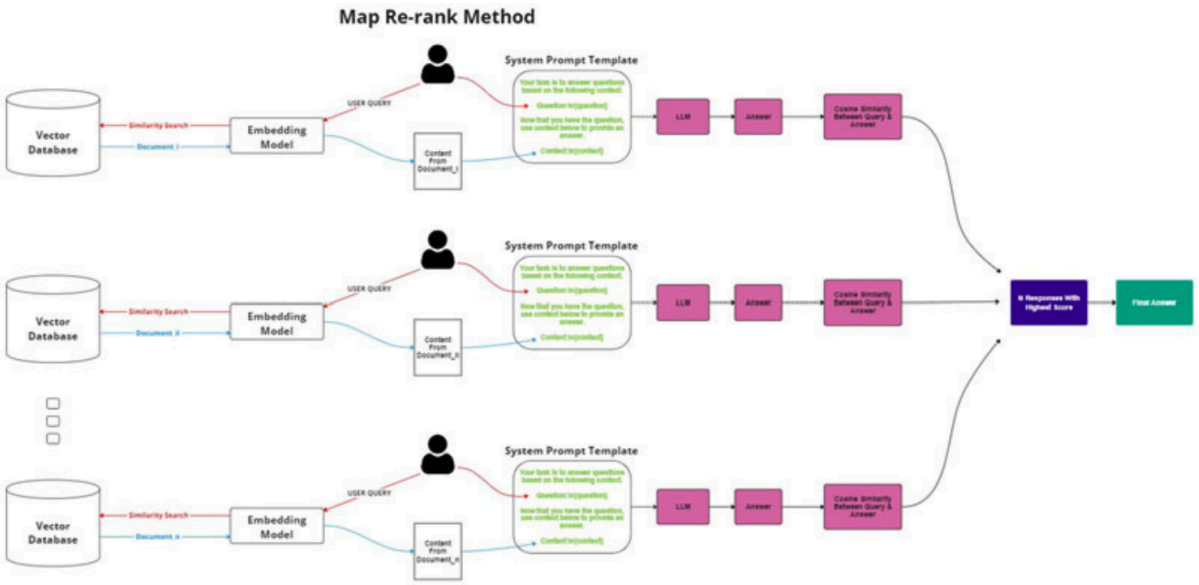
**Limitações**

- Combinar respostas parciais exige uma segunda chamada ao LLM.
- Pode perder nuances cruzadas entre documentos (não há contexto global).

**Quando usar**

- Bases documentais grandes e heterogêneas.
- Situações onde é fácil sintetizar resultados (ex: FAQs, revisões).

## Método Map-Re-Rank




Conceito

Extensão do Map-Reduce, mas com uma etapa adicional de ranking e seleção antes da fusão final. Após o “map”, cada resposta parcial recebe um score de relevância (via modelo de similaridade ou LLM), e apenas as mais relevantes são usadas na etapa “reduce”.

**Fluxo**

1. Map: o LLM gera uma resposta por documento.
2. Re-rank: as respostas são classificadas (por similaridade com a pergunta, confiança ou embeddings).
3. Reduce: as top-N respostas são combinadas em uma final.

**Vantagens**

- Reduz ruído e redundância.
- Mantém a precisão sem precisar processar todos os documentos.
- Pode equilibrar custo × qualidade (descartando contextos irrelevantes).

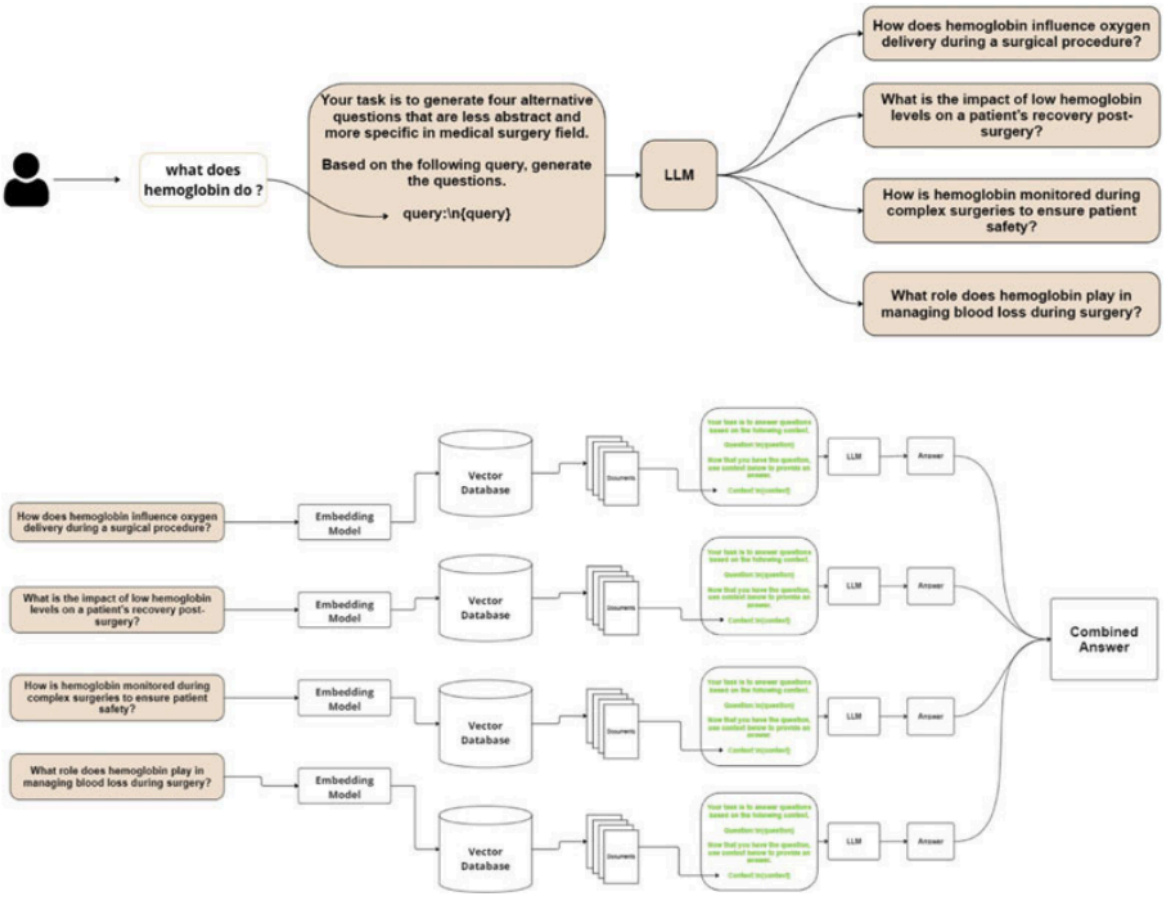
**Limitações**

- Exige uma camada de scoring adicional.
- Pode eliminar documentos úteis se o ranker não for preciso.
- Tempo de execução um pouco maior que o Map-Reduce puro.

**Quando usar**

- Quando o repositório é grande e redundante.
- Para aplicações sensíveis à qualidade (ex: pesquisa científica, jurídico).

# Método Query Step-Down




## Conceito

Abordagem voltada para consultas ambíguas ou complexas. O sistema gera versões refinadas da pergunta original (por decomposição, simplificação ou reformulação) e faz retrieval para cada uma.

## Fluxo

1. O LLM ou outro módulo gera subconsultas (ex: decompor “Explique o impacto econômico e ambiental da IA” –> duas perguntas).

2. Cada subconsulta faz retrieval + resposta.

3. As respostas são combinadas na saída final.

### Vantagens

- Melhora a cobertura em consultas ambíguas.
- Garante que diferentes aspectos da pergunta sejam abordados.
- Evita “viés de recuperação única” (pegar apenas um tipo de documento).

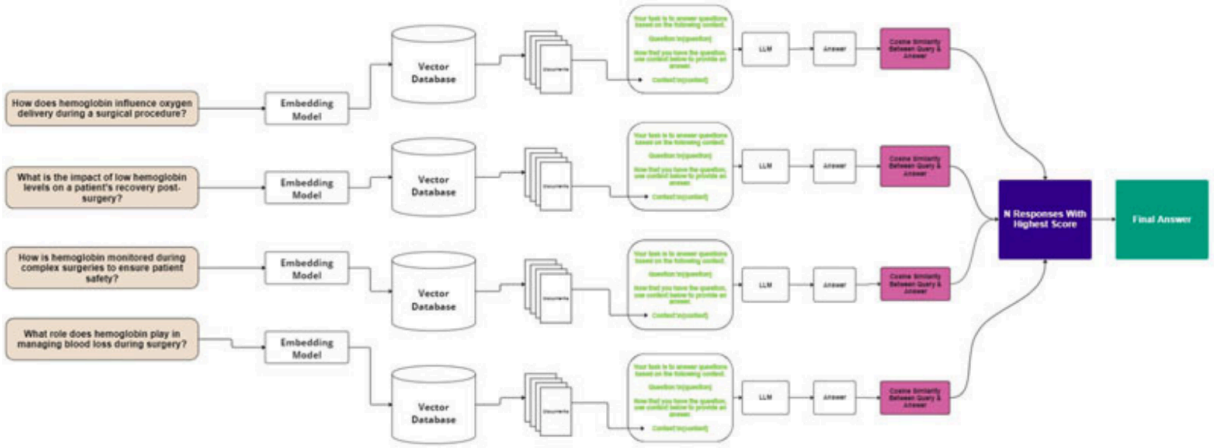
### Limitações

- Multiplica o custo (retrieval e geração para cada subconsulta).
- Requer boa decomposição automática da pergunta (senão gera redundância).
- Integração das respostas pode ser inconsistente.

### Quando usar

- Consultas complexas e multifacetadas (ex: pesquisa científica, relatórios técnicos).
- Ambientes com documentos de escopo diverso.

# Método Reciprocal




## Conceito

Um dos métodos mais sofisticados: combina ideias de re-ranking, autoavaliação e reciprocidade de consulta-resposta. Funciona como um ciclo de refinamento mútuo entre a consulta e as respostas geradas, escolhendo a melhor resposta final.

## Fluxo

1. São geradas múltiplas variantes da consulta (como no Step-Down).
2. Para cada uma, o LLM gera uma resposta (pode usar Stuff ou Map-Reduce).
3. As respostas são avaliadas entre si (re-rank) com base na similaridade à consulta original ou em métricas de consistência semântica.
4. A resposta mais relevante ou uma fusão das melhores é devolvida como final.

## Vantagens

- Altíssima precisão e consistência (as respostas se validam mutuamente).
- Tolerante a consultas vagas ou mal formuladas.
- Excelente desempenho em benchmarks (97% de similaridade média no estudo).

**Limitações**

- Muito caro computacionalmente (várias chamadas e comparações).
- Implementação mais complexa.
- Latência elevada: inadequado para uso em tempo real.

**Quando usar**

- Domínios críticos de precisão: jurídico, financeiro, médico.
- Avaliações offline, onde custo/tempo não são problema.

## RESULTADOS

Os resultados quantificam claramente os trade-offs entre esses integradores. O Reciprocal RAG atinge os melhores índices de similaridade em quase todos os cenários, chegando a cerca de 97% no dataset financeiro 10-Q, seguido de perto por Query Step-Down e Map-Re-Rank (faixa de 85–90%). O método Stuff, apesar de ingênuo, mantém um desempenho razoável (≈86%), o que mostra que ele é competitivo em tarefas menos complexas. Em contrapartida, o custo de tokens e o tempo de execução crescem de forma acentuada nos métodos iterativos e multiestágio: Refine e, sobretudo, Reciprocal RAG são os mais caros e lentos, enquanto Stuff é claramente o mais rápido e com menor custo, especialmente quando combinado com compressão de contexto. Map-Reduce e Map-Re-Rank ficam em um meio-termo interessante, oferecendo boa qualidade com custo moderado, principalmente quando se deseja aproveitar paralelismo.

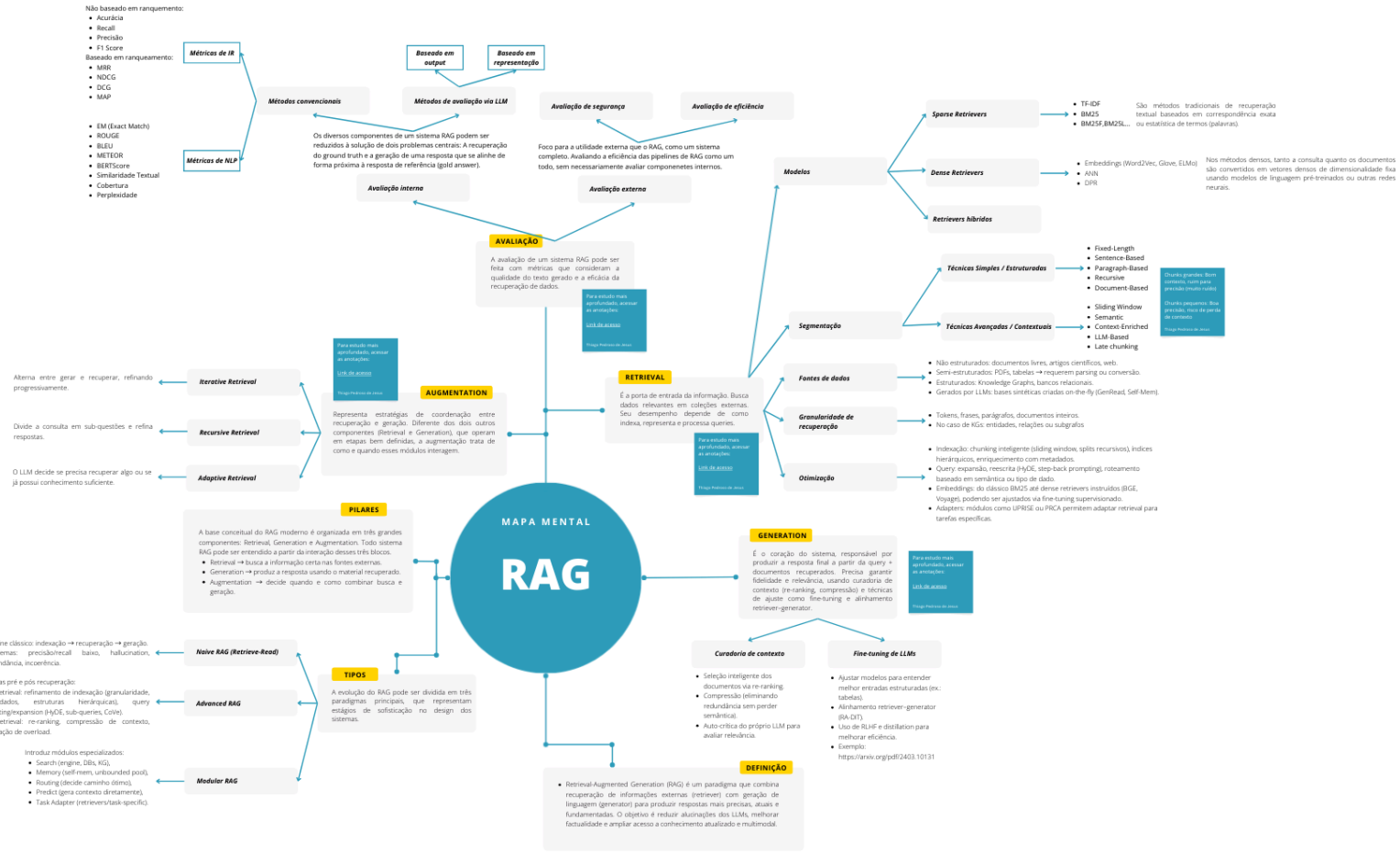
Um ponto transversal do artigo é o uso de compressão contextual (filtros de similaridade/redundância) como mecanismo de eficiência. Ao aplicar filtros com thresholds de similaridade entre 0,5 e 0,9, os autores observam uma redução média de cerca de 12,5% no uso de tokens e 4,4% no tempo de execução, com perda de similaridade inferior a 1% na maioria dos casos. Métodos mais “pesados”, como Reciprocal RAG e Map-Re-Rank, foram os que mais se beneficiaram: ao limpar o

contexto antes de chamar o LLM, mantêm praticamente a mesma qualidade com um custo significativamente menor. Isso reforça a ideia de que, em pipelines RAG, não basta escolher um bom LLM,  é crucial controlar o quanto de contexto chega até ele e com que nível de ruído.

Por fim, o estudo também mostra que a eficiência do RAG depende da infraestrutura de recuperação. Comparando ChromaDB, FAISS e Pinecone, o ChromaDB aparece como a opção com melhor equilíbrio entre velocidade e uso de CPU/memória nos cenários testados; o FAISS tem bom desempenho, mas consome mais memória, enquanto o Pinecone apresenta maior latência. Em conjunto, os achados sugerem que a escolha do integrador (Stuff, Refine, Map-Reduce, etc.), do esquema de compressão e do vectorstore deve levar em conta o domínio e a criticidade da aplicação: em contextos sensíveis (jurídico, financeiro, médico), métodos como Reciprocal RAG e Query Step-Down fazem sentido, mesmo com maior custo; já em chatbots simples, FAQs ou aplicações com forte restrição de latência e orçamento, combinações mais leves (Stuff + boa recuperação + compressão de contexto) tendem a ser a solução mais realista.

# Mapa mental da área

Durante toda a minha jornada na Residência em IA, conduzi os meus estudos a partir de um mapa mental que serviu como um guia no conteúdo e me deu suporte para não me perder no conteúdo. Com o passar das Semanas, eu fui preenchendo esse mapa e, ao fim da sétima Semana, consolidei a versão final, que pode ser encontrada no link: Link mapa mental




# Listagem dos artigos de interesse

Com o objetivo de decidir um tema para desenvolvimento de estudos práticos, reuni todos os artigos que me despertaram interesse durante a Residência em uma tabela que os classifica por área e destaca sua tese principal. Dessa forma, fica fácil ter uma visão geral de tudo o que foi estudado e qual caminho mais me desperta o interesse. Segue a tabela final:

| | | | | |
|---|---|---|---|---|
| RAG-FUSION: A NEW TAKE ON RETRIEVAL-AUGMENTED GENERATION | https://arxiv.org/abs/2402.03367 | Retrieval, Augmentation | Retrieval | O artigo avalia o RAG-Fusion, que gera múltiplas consultas a partir da pergunta do usuário e depois aplica Reciprocal Rank Fusion (RRF) para fundir e reranquear os documentos recuperados, antes de entregar o contexto ao LLM. A ideia é contextualizar a intenção do usuário por diferentes ângulos e melhorar a cobertura e a relevância do material passado ao gerador; os resultados qualitativos mostram respostas mais completas, mas com risco de desvio quando as consultas geradas ficam pouco alinhadas ao pedido original. |
| RaFe: Ranking Feedback Improves Query Rewriting for RAG | https://aclanthology.org/2024.findings-emnlp.49/ | Retrieval | Retrieval | O trabalho apresenta o RaFe, um framework para treinar modelos de reescrita de consultas sem anotações, usando um reranker como fonte de feedback de ranking (offline com DPO/KTO e online com PPO). A tese é que o sinal do reranker se alinha diretamente ao objetivo da reescrita (recuperar documentos mais relevantes), evitando custos de rótulos e recompensas ad-hoc, e melhorando a recuperação em cenários de QA em domínio aberto. |
| Precise Zero-Shot Dense Retrieval without Relevance Labels | https://aclanthology.org/2023.acl-long.99.pdf | Retrieval | Retrieval | Os autores introduzem o HyDE (Hypothetical Document Embeddings): dado um query, um LLM gera um “documento hipotético” que captura o padrão semântico do que seria uma resposta; esse texto é então codificado e usado para recuperar vizinhos reais por similaridade vetorial. Assim, o método dispensa rótulos de relevância, supera retrievers não supervisionados e aproxima o desempenho de retrievers ajustados, inclusive em múltiplas tarefas e idiomas. |

| | | | | |
|---|---|---|---|---|
| SELF-RAG: LEARNING TO RETRIEVE, GENERATE, AND CRITIQUE THROUGH SELF-REFLECTION | https://arxiv.org/abs/2310.11511 | Retrieval, Augmentation, Generation | Generation | LLMs erram por depender apenas de conhecimento paramétrico. O Self-RAG treina um único modelo a decidir se precisa recuperar passagens, gerar e criticar tanto os documentos recuperados quanto a própria resposta, via tokens de reflexão que tornam o comportamento controlável na inferência. Mostra ganhos de factualidade e citações em QA aberto, verificação de fatos e geração longa, superando baselines RAG e até ChatGPT em certos cenários. |
| DMQR-RAG: DIVERSE MULTI-QUERY REWRITING FOR RETRIEVAL-AUGMENTED GENERATION | https://arxiv.org/abs/2411.13154 | Retrieval, Augmentation | Retrieval | Perguntas de usuário trazem ruído/ambiguidade; reescrever a consulta melhora a recuperação. O DMQR-RAG propõe múltiplas reescritas com níveis diferentes de informação e uma estratégia adaptativa para escolher poucas reescritas eficazes, aumentando diversidade de documentos e a qualidade da resposta final — com validações em cenários acadêmicos e industriais. |
| RAFT: Adapting Language Model to Domain Specific RAG | https://arxiv.org/abs/2403.10131 | Augmentation, Generation | Augmentation | Em domínios específicos, combinar RAG com SFT ainda é aberto. O RAFT (Retrieval-Augmented Fine-Tuning) é uma receita de treinamento “open-book”: dado o enunciado e documentos (com distratores), o modelo é treinado a ignorar o que não ajuda e citar literalmente o trecho relevante do documento certo. Isso melhora a precisão em QA in-domain e a robustez a recuperações imperfeitas, sem precisar de rótulos de relevância |
| Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions | https://arxiv.org/abs/2212.10509 | Retrieval, Augmentation | Retrieval | Em perguntas multi-hop, “pegar tudo de uma vez” (retrieve-once) não basta: o que recuperar depende do que já foi deduzido. O IRCoT alterna passos de raciocínio em CoT com novas buscas guiadas por cada passo, intercalando reason → retrieve → reason…. Isso melhora recall de passagens (até +11–21 pontos) e F1 de QA (até +15), reduzindo |

| | | | | |
|---|---|---|---|---|
| | | | | alucinações, em HotpotQA, 2WikiMultihopQA, MuSiQue e IIRC — inclusive com modelos menores (Flan-T5) sem treino adicional. Em termos de pipeline, é uma forma de retrieval iterativo guiado por raciocínio e montagem incremental do contexto antes da geração final. |
| Active Retrieval Augmented Generation | https://arxiv.org/abs/2305.06983 | Retrieval, Augmentation | Retrieval | Em textos longos, recuperar só no início falha porque novas partes do texto exigem evidências futuras. O FLARE faz RAG ativo: prevê a próxima sentença, usa essa previsão como consulta para recuperar trechos adicionais e regenera a sentença quando há baixa confiança em tokens. Assim, decide quando e o que recuperar ao longo da geração, obtendo ganhos consistentes em tarefas de geração de conhecimento de longo formato. Em suma, é retrieval adaptativo e contínuo, estreitando o laço entre enriquecimento de contexto e geração. |
| Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models | https://arxiv.org/abs/2506.00049 | Retrieval, Augmentation | Retrieval | Em retrieval híbrido para RAG (misturando denso semântico + esparso lexical + grafos), os autores mostram que embeddings menores combinados com reranqueamento por LLM podem superar modelos maiores (e mais caros). O foco é custo-efetividade: a fusão tri-modal seguida de LLM re-ranking entrega melhores candidatos para o contexto do RAG, diminuindo dependência de embeddings “SOTA” pesados. Em termos de pipeline, isso eleva a qualidade do conjunto top-k antes do prompt final, com benefícios práticos de latência/custo. |
| LongLLMLingua: Accelerating and Enhancing LLMs in Long Context Scenarios via Prompt Compression | https://arxiv.org/abs/2310.06839 | Augmentation | Augmentation | Em cenários de contexto longo, LLMs sofrem com custo, queda de desempenho e viés posicional. O LongLLMLingua comprime prompts preservando “informação-chave”, reduzindo drasticamente tokens/latência e melhorando a acurácia em QA de longo contexto (ex.: ganhos no |

| | | | | |
|---|---|---|---|---|
| | | | | NaturalQuestions com ~4× menos tokens). É uma técnica de seleção/condensação do material passado ao gerador no pipeline RAG. |
| PRCA: Fitting Black-Box Large Language Models for Retrieval Question Answering via Pluggable Reward-Driven Contextual Adapter | https://arxiv.org/abs/2310.18347 | Augmentation, Generation | Augmentation | Quando o gerador é um LLM caixa-preta (p.ex., via API), é caro/ inviável tunar. O PRCA insere um adaptador contextual treinável entre o retriever e o LLM e o otimiza por RL para maximizar uma recompensa de QA. Ele refina os conteúdos recuperados antes do LLM e melhora métricas de ReQA em múltiplos datasets, sem tocar no gerador. |
| Structure-Aware Language Model Pretraining Improves Dense Retrieval on Structured Data | https://arxiv.org/abs/2305.19912 | Retrieval | Retrieval | Recuperar dados estruturados (código, produtos) exige representar estrutura. O SANTA pré-treina LMs com (1) Structured Data Alignment (alinha dados estruturados ↔ textos) e (2) Masked Entity Prediction (máscara orientada a entidades), aprendendo um espaço único de embeddings para consulta e item. Resulta em SOTA para code/product search e bons resultados zero-shot. |
| Retrieval-Augmented Generative Question Answering for Event Argument Extraction | https://arxiv.org/pdf/2211.07067 | Retrieval, Augmentation, Generation | Augmentation | Para extração de argumentos de evento, o trabalho propõe R-GQA: recuperar demonstrações similares e aumentar o prompt para então gerar respostas (argumentos) com LMs, superando métodos extrativos e gerativos anteriores em regimes fully-supervised, transferência e few-shot. É um uso clássico de RAG orientado a IE (information extraction). |
| Dual-Feedback Knowledge Retrieval for Task-Oriented Dialogue Systems | https://arxiv.org/abs/2310.14528 | Retrieval, Augmentation | Retrieval | Em diálogo orientado a tarefas com bases grandes, o paper separa retriever e generator e propõe treinar o retriever sem rótulos, usando feedback duplo do gerador (positivo/negativo como pseudo-rótulos). Isso melhora a seleção de conhecimento e a qualidade da resposta nos benchmarks de TOD. |

| | | | | |
|---|---|---|---|---|
| BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation | https://arxiv.org/abs/2402.03216 | Retrieval | Retrieval | O BGE M3 entrega embeddings multilíngues, multi-função (busca semântica, reranking, etc.) e multi-granularidade (frase, parágrafo, documento) via self-knowledge distillation, visando um único backbone forte e versátil para pipelines RAG em vários idiomas e tarefas. |

# APÊNDICE 8

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 23 de out. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as sete primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, estudei a história da Recuperação de Informação, aprofundei as frentes de Recuperação, Geração e Integração, compreendi os principais métodos de avaliação de sistemas RAG e, por fim, escolhi me aprofundar mais no tema de **Recuperação Híbrida**.

Para a oitava Semana, defini como prioridade iniciar os experimentos práticos a partir da leitura e replicação do artigo "Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models", com o objetivo de compreender sua arquitetura e testar os resultados apresentados.

As atividades desenvolvidas foram:

- Finalização da leitura do artigo "Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models"
  - O artigo define a recuperação em duas fases principais:
    - Fase 1: Recuperação híbrida tri-modal (esparsa, densa e grafo);
    - Fase 2: Re-ranking com LLMs, utilizando pesos dinâmicos para balancear as modalidades.
  - Anotações: Residência - Rethinking Hybrid Retrieval

- Replicação da primeira fase (retrieval híbrido):
  - Desenvolvimento e teste das três modalidades de retriever de forma isolada;
  - Download e preparação dos mesmos conjuntos de dados utilizados no artigo para avaliação (FiQA, SciFact e NFCorpus);
  - Execução de experimentos de busca híbrida, obtendo resultados próximos aos reportados no estudo original (comparação com os experimentos sem re-ranking do artigo).
  - Repositório: https://github.com/Thiago-Pedroso/hybrid-retrieval

- Leitura complementar sobre busca híbrida dinâmica:
  - Dentro da seção de trabalhos relacionados do artigo principal, foi mencionado o trabalho "DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation", que

despertou meu interesse por tratar da otimização dinâmica dos pesos entre modalidades de busca.
- Anotações: Residência - Dynamic Alpha Tuning

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

O objetivo para a próxima Semana é finalizar a replicação dos experimentos propostos no artigo, avançando para a segunda fase do método. Nesse sentido, pretendo:

- Realizar novos experimentos variando parâmetros, buscando resultados mais próximos dos reportados no artigo original;
- Replicar a Fase 2, construindo o mecanismo de *re-ranking* com LLM e avaliando seu impacto nos resultados de recuperação;
- Terminar de replicar os benchmarks descritos no paper.

Além disso, durante os testes, surgiu o interesse em estender o estudo para o contexto brasileiro, aplicando a mesma metodologia em dados em português. Portanto, também pretendo:

- Identificar e selecionar um dataset em português adequado para experimentos de retrieval;
- Executar os experimentos seguindo a metodologia do artigo;
- Propor ajustes e melhorias na abordagem para o português, como o fine-tuning do modelo de embeddings utilizado.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Artigo “Rethinking Hybrid Retrieval”

## INTRODUÇÃO E CONTEXTUALIZAÇÃO

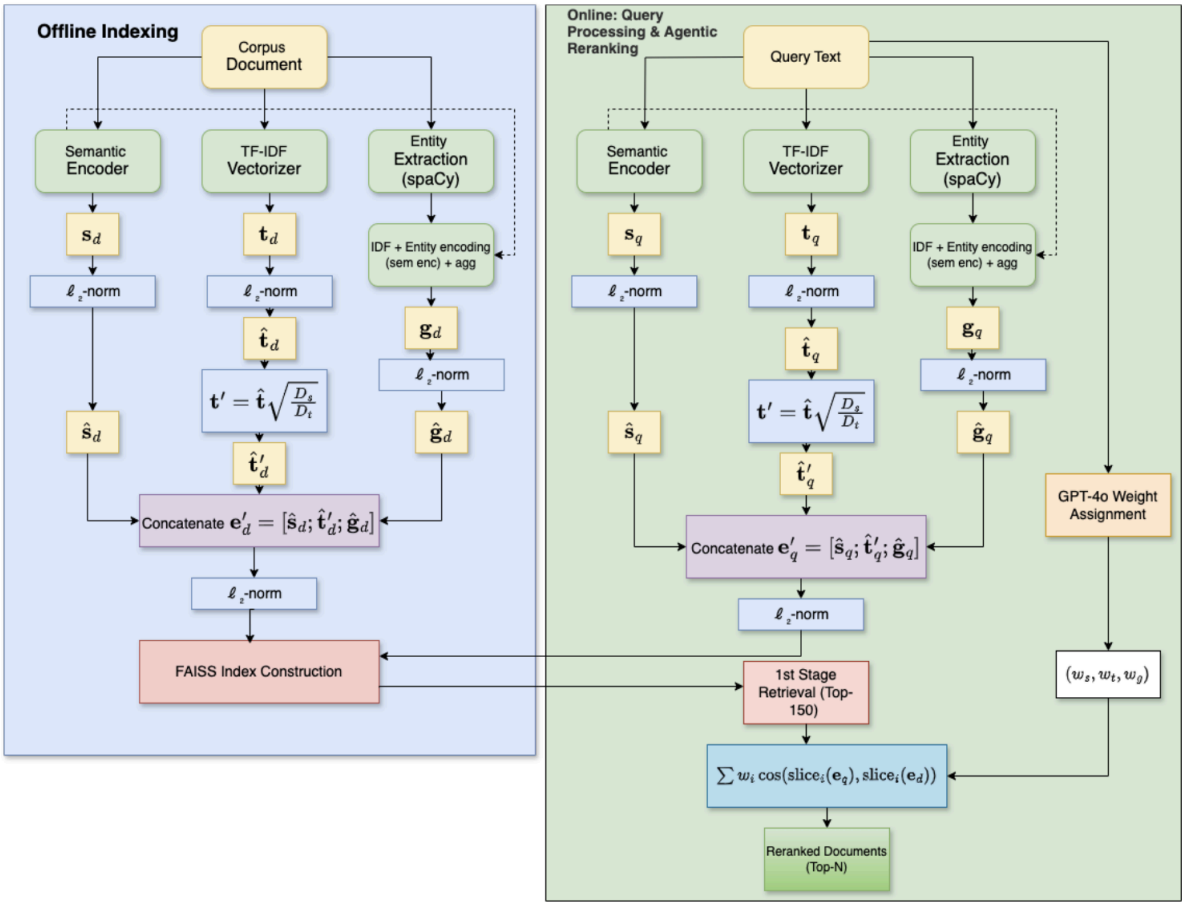
Referência: “Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models”

- A área de IR (Information Retrieval) evoluiu com LLMs, mas ainda é difícil combinar bem múltiplas “modalidades” de busca: semântica (embeddings densos), lexical (TF-IDF/BM25) e conhecimento estruturado de grafos.
- Em cenários reais (ex.: e-commerce, busca em bases de conhecimento), as consultas costumam ter múltiplos atributos e categorias; nesses casos, a importância relativa de cada modalidade muda conforme o contexto da consulta.
- Métodos tradicionais usam combinações estáticas desses sinais e, por isso, não se adaptam quando o contexto da consulta muda, perdendo relevância em situações dinâmicas.
- Integrar grafos (knowledge graphs) à busca textual é especialmente desafiador: há diferenças estruturais e, se mal integrados, perde-se o ganho potencial de entender relações entre entidades.
- O trabalho propõe ponderação dinâmica baseada em LLM: o sistema ajusta, para cada consulta, os pesos dos três recuperadores (semântico, lexical e grafo).
- Para isso, apresenta um índice tri-híbrido que une os três tipos de embeddings numa representação única e permite fusão adaptativa por consulta.
- A promessa é melhorar a precisão e a relevância da recuperação, superando abordagens estáticas, com baixa latência e aplicável em tempo real.

## PERGUNTAS DE PESQUISA

- Como a ponderação dinâmica com LLM melhora a relevância e o desempenho em sistemas multimodais de recuperação?
- Quais os benefícios de um índice tri-híbrido (semântico, lexical, grafo) para recuperação em tempo real?
- Como a ponderação dinâmica se compara à estática em consultas complexas e multiatributo?
- Como aplicar essa ponderação dinâmica de forma eficiente sem comprometer o desempenho?

# SOLUÇÃO PROPOSTA




A solução proposta pelo artigo envolve a comparação de modelos de modelos de embedding em recuperação híbrida trimodal para sistemas RAG. O trabalho investiga a fusão de recuperações semânticas densas, lexicais esparsas e baseadas em grafos. O fluxo de trabalho completo é composto por duas etapas:

- Indexação offline: documentos → embeddings tri-modais → normalização, escalonamento e concatenação em um vetor híbrido armazenado no índice para buscas rápidas.
- Processamento online da consulta: gera embeddings tri-modais da query → recupera candidatos no índice → aplica reranqueamento guiado por LLM, que ajusta dinamicamente os pesos (semântico/lexical/grafo) conforme o contexto da consulta, priorizando os resultados mais relevantes.

## RESULTADOS

MiniLM-v6 + GPT-4o (re-ranking) superou BGE-Large + GPT-4o em todos os datasets (SciFact, FIQA, NFCorpus), apesar de o MiniLM ser bem menor. Ganhos em nDCG@10: SciFact +8,3%, FIQA +23,1%, NFCorpus +7,7%. O Recall ficou parecido entre os modelos, mas o MRR favoreceu o MiniLM-v6 (≈ +10,9% a +24,3%).

O ganho é mais forte em k baixos (top-1/3), que é o que mais importa para RAG. Em k=1, o MiniLM-v6 vence por +14,6% (SciFact), +36,5% (FIQA) e +22,9% (NFCorpus).

Antes do re-ranking por LLM, o BGE-Large ia melhor no "primeiro passe": nDCG@10 ≈ 0,6608 (BGE) vs 0,6505 (MiniLM). Depois do re-ranking (GPT-4o), o quadro inverte: 0,6170 (BGE) vs 0,6681 (MiniLM). Os autores chamam isso de "FAISS Hybrid Paradox": embeddings maiores ajudam a recuperar candidatos inicialmente, mas não se alinham tão bem ao critério de relevância do LLM; o MiniLM-v6 é o oposto (ganha muito após o re-rank).

Principais achados de acordo com os autores:

- MiniLM-v6 + GPT-4o > BGE-Large + GPT-4o em todos os datasets;
- Reranqueamento agentivo amplia a diferença, sobretudo em k baixos;
- FIQA tem o maior salto, sugerindo boa adaptação a domínios financeiros;
- Eficiência: MiniLM-v6 tem ~93% menos parâmetros e embeddings menores (≈ 384d vs 1024d no BGE-Large), mantendo ou superando a qualidade após re-rank.

# Artigo "Dynamic Alpha Tuning"

## INTRODUÇÃO E CONTEXTUALIZAÇÃO

Referência: "DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation"

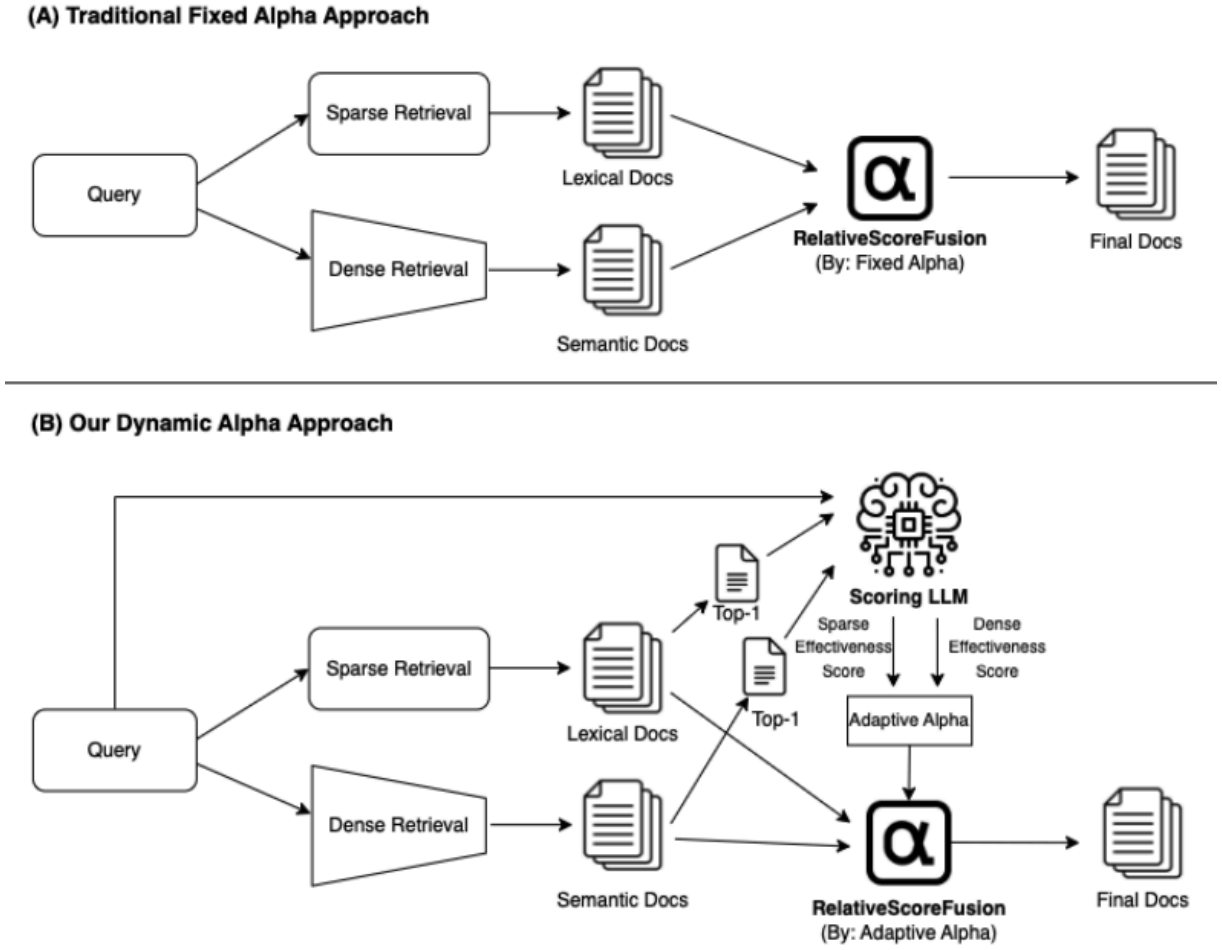
O artigo conduz uma abordagem de recuperação híbrida que combina métodos esparsos (como o BM25, bom para palavras-chave) e densos (bons para semântica). Embora essa combinação funcione bem, o desafio é como equilibrar (ponderar) a contribuição de cada método. A abordagem atual usa um peso fixo (chamado $\alpha$) definido em testes offline, o que é inadequado porque o equilíbrio ideal varia para cada consulta de usuário e base de conhecimento.

Os autores argumentam que as tentativas de ajustar esse peso com base em "tipos" de consulta ainda são limitadas e que usar um valor médio fixo pode, na verdade, prejudicar o desempenho da maioria das consultas individuais, que

poderiam se beneficiar mais de um método ou de outro (extremos).

Para resolver isso, o artigo apresenta o DAT, um novo framework que ajusta adaptativamente o peso (α) para cada consulta específica. A ideia principal é avaliar a eficácia relativa dos métodos esparsos e densos comparando apenas o primeiro resultado (top-1) de cada um usando um mecanismo de pontuação baseado em LLM. Isso permite ao DAT determinar dinamicamente qual método é mais forte para aquela consulta específica e calcular o peso ideal, tudo com um custo computacional mínimo.

As principais contribuições do trabalho são: (1) O framework adaptativo (DAT) que elimina a necessidade de pesos estáticos; (2) O mecanismo leve de pontuação baseado em LLM; (3) A demonstração de que avaliar apenas o top-1 resultado é suficiente; e (4) A comprovação experimental de que o DAT supera significativamente as abordagens de peso fixo, especialmente em consultas difíceis, e oferece resultados mais consistentes.




## METODOLOGIA DAT (DYNAMIC ALPHA TUNING)

O artigo detalha o funcionamento do DAT, o framework proposto para superar a limitação do peso estático (alpha fixo) na recuperação híbrida.

A intuição central é que diferentes consultas favorecem diferentes estratégias: algumas exigem correspondência precisa de palavras-chave (favorecendo o BM25), enquanto outras dependem de alinhamento semântico (favorecendo a recuperação densa).

O DAT utiliza as capacidades de raciocínio dos LLMs para estimar o coeficiente de ponderação ideal, alpha(q), para cada consulta (q) em tempo de execução.

1. O processo funciona da seguinte maneira:
2. Dada uma consulta q, o sistema recupera apenas o resultado top-1 (o primeiro

resultado) do método denso e o resultado top-1 do BM25.
3. Esses dois documentos são tratados como "indicadores representativos" da eficácia de cada método para aquela consulta específica.
4. Essa estratégia de amostragem focada (apenas o top-1 de cada) é projetada para minimizar o custo computacional, ao mesmo tempo que fornece informação suficiente para a decisão de ponderação.

Um componente chave do DAT é o uso de LLMs como avaliadores da qualidade da recuperação.

O DAT define uma função de pontuação S(q, d) = f_LLM(q, d), onde o LLM avalia a relevância de um documento (d) para a consulta (q) e atribui uma pontuação discreta na escala de {0, 1, 2, 3, 4, 5}.

A pontuação é definida da seguinte forma:

- **5 pontos (Direct hit):** O documento responde diretamente à pergunta.
- **3–4 pontos (Good wrong result):** O documento está conceitualmente próximo da resposta correta, indicando alta probabilidade de que as respostas certas estejam "próximas" dele na recuperação.
- **1–2 pontos (Bad wrong result):** O documento é vagamente relacionado, mas enganoso, com baixa probabilidade de respostas corretas estarem próximas.
- **0 pontos (Completely off-track):** O resultado é totalmente não relacionado à consulta.

O LLM avalia independentemente os dois documentos top-1 (o denso e o do BM25), gerando duas pontuações: S_v(q) (para o resultado denso) e S_b(q) (para o resultado do BM25). Essa avaliação dupla permite capturar a eficácia relativa de cada método de recuperação para aquela consulta específica.

o DAT calcula o coeficiente de ponderação dinâmico alpha(q) para a consulta. Isso é feito através de uma formulação baseada em regras (Equação 6) para garantir um comportamento robusto:

$$\alpha(q) = \begin{cases} 0.5, & \text{if } S_v(q) = 0 \text{ and } S_b(q) = 0, \\ 1.0, & \text{if } S_v(q) = 5 \text{ and } S_b(q) \neq 5, \\ 0.0, & \text{if } S_b(q) = 5 \text{ and } S_v(q) \neq 5, \\ \frac{S_v(q)}{S_v(q)+S_b(q)} & \text{otherwise.} \end{cases}$$

- **Se ambos os métodos falharem** (pontuação 0 para ambos), o peso é dividido

igualmente: alpha(q) = 0.5.

- **Se um método obtiver um "acerto perfeito"** (pontuação 5) e o outro não, esse método recebe 100% do peso.
  - Se apenas o denso acertar perfeitamente: alpha(q) = 1.0.
  - Se apenas o BM25 acertar perfeitamente: alpha(q) = 0.0.
- **Em todos os outros casos** (quando ambos retornam resultados parcialmente relevantes), o peso é calculado proporcionalmente às suas pontuações

Para garantir estabilidade, o valor final de alpha(q) é arredondado para uma casa decimal.

Uma vez que o alpha(q) dinâmico é determinado, o framework o utiliza para calcular a pontuação de ranking híbrido final R(q, d) para todos os documentos recuperados (não apenas o top-1), conforme a Equação:

$$R(q,d) = \alpha(q) \cdot \tilde{S}_{\text{dense}}(q,d) + (1 - \alpha(q)) \cdot \tilde{S}_{\text{BM25}}(q,d)$$

Finalmente, os documentos são classificados com base nesta pontuação híbrida final R(q, d), e os K melhores resultados (top-K) são selecionados e enviados ao componente de geração do sistema RAG. O texto conclui afirmando que essa abordagem de adaptação específica para cada consulta permite ao DAT superar as limitações dos métodos de peso fixo, melhorando a relevância geral dos resultados

# RESULTADOS

| Method | SQuAD | DRCD |
|---|---|---|
| BM25 Only ($\alpha = 0.0$) | 0.7981 | 0.8110 |
| Dense Only ($\alpha = 1.0$) | 0.7789 | 0.6156 |
| Fixed Hybrid ($\alpha = 0.6$) | 0.8975 | 0.8623 |
| DAT (DeepSeek-R1-Distill-Qwen-14B) | 0.9163 | 0.8897 |
| DAT (GPT-4o-mini) | 0.9194 | **0.9013** |
| DAT (GPT-4o) | **0.9234** | 0.9010 |

Table 2: Alpha Selection Accuracy on Complete Datasets $\mathcal{Q}_{\text{eval}}$

| Method | SQuAD | | DRCD | |
|---|---|---|---|---|
| | Precision@1 | MRR@20 | Precision@1 | MRR@20 |
| BM25 Only ($\alpha = 0.0$) | 0.7594 | 0.8223 | 0.7630 | 0.8134 |
| Dense Only ($\alpha = 1.0$) | 0.7396 | 0.8119 | 0.5743 | 0.6708 |
| Fixed Hybrid ($\alpha = 0.6$) | 0.8461 | 0.8997 | 0.8113 | 0.8619 |
| DAT (DeepSeek-R1-Distill-Qwen-14B) | 0.8663 | 0.9079 | 0.8347 | 0.8711 |
| DAT (GPT-4o-mini) | 0.8676 | 0.9093 | 0.8417 | 0.8796 |
| DAT (GPT-4o) | **0.8740** | **0.9130** | **0.8440** | **0.8807** |

Table 3: Retrieval Performance on Complete Datasets $\mathcal{Q}_{\text{eval}}$

# APÊNDICE 9

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 6 de nov. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

**THIAGO PEDROSO DE JESUS**

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as oito primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, percorri seu desenvolvimento histórico, aprofundei as frentes de Recuperação, Geração, Integração e Avaliação, e realizei estudos práticos sobre recuperação híbrida. Na última entrega, iniciei a replicação do artigo "[2506.00049] Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models" (maio/2025),

Na última entrega:

- Consegui reproduzir a primeira fase do artigo, montando a recuperação híbrida com as modalidades esparsa, densa e grafo.
- Fiz uma avaliação pré-rank que teve resultados próximos aos relatados no trabalho original.

Para a nona Semana, o objetivo foi avançar para a fase de fusão vetorial e re-ranking, aprofundando os mecanismos que compõem o núcleo da proposta do artigo.

Nesse sentido, as atividades desenvolvidas foram:

- Replicação e análise da fase de fusão e re-ranking
  - Conduzi experimentos para reproduzir o processo de fusão tri-modal e re-ranking com LLMs descrito no artigo;
  - Durante a replicação, notei resultados que divergiam parcialmente das premissas originais, sugerindo lacunas na implementação relatada;
  - Essa constatação levou à necessidade de investigar e comparar diferentes técnicas de fusão, normalização e ponderação de vetores, ampliando o escopo dos experimentos para além do paper original.

- Estudos e experimentos complementares sobre retrievers híbridos
  - Realizei testes individuais com os retrievers esparso, denso e grafo, avaliando seu desempenho isolado antes da fusão;
  - Essas análises me permitiram explorar questões fundamentais, como:
    - Qual a melhor forma de realizar a fusão dos vetores?

        - Quanto cada modalidade impacta na busca final?
        - Em que medida o retriever híbrido melhora em relação aos individuais?
    - Aprofundei o estudo desses aspectos a partir de leituras complementares.
        - Anotações: Residência - Estudo complementar retrievers híbridos

- Aplicação do método DAT (Dynamic Alpha Tuning)
    - Diante das limitações conceituais e metodológicas observadas no primeiro artigo, mantive sua estrutura experimental, mas incorporei princípios do trabalho "[2503.23013] DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation" (março/2025);
    - O paper apresenta duas propostas principais que eu repliquei:
        - A construção de um dataset que foca em casos onde diferentes modalidades de recuperadores trazem resultados diferentes;
        - Implementei o mecanismo de tuning dinâmico dos pesos de fusão (α);
    - A aplicação do DAT resultou em ganhos consistentes de desempenho, superando tanto os retrievers individuais quanto o método de fusão do artigo Rethinking Hybrid Retrieval;
    - Experimentos: Residência - Documentação experimentos

- Desenvolvimento de uma estrutura experimental modular
    - Para facilitar a execução e comparação dos experimentos, estruturei um ambiente modular em código, que permite configurar múltiplas modalidades de retriever, estratégias de fusão e normalização;
    - Esse ambiente evoluiu para um framework experimental para recuperação híbrida, acompanhado de laboratórios práticos em notebooks Python, que auxiliam na depuração, visualização e compreensão do processo de recuperação híbrida;
    - Link do repositório: github.com/Thiago-Pedroso/hybrid-retrieval

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

Com o objetivo de consolidar os meus experimentos práticos, pretendo:

- Finalizar a documentação do código, detalhando a estrutura desenvolvida para execução e análise dos experimentos;
- Finalizar a documentação dos laboratórios práticos, assegurando que cada módulo esteja descrito e reproduzível;
- Consolidar um benchmark unificado, reunindo todos os métodos de recuperação híbrida estudados até o momento e padronizando sua avaliação em um mesmo conjunto de métricas e dados;
- Propor e avaliar um método próprio de recuperação híbrida, aplicando-o no benchmark construído para analisar seu desempenho em relação às abordagens anteriores.

**Observação: [caso precise fazer alguma observação, de qualquer "natureza"]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

Instituto de Informática (INF) - UFG
Alameda Palmeiras, Quadra D, Câmpus Samambaia
Fone: (62) 3521-1181 / Fax: (62) 3521-1182

# Estudo de estratégias para recuperação híbrida

## VISÃO GERAL

A recuperação híbrida combina buscadores densos (vetoriais) e esparsos (lexicais) para explorar, ao mesmo tempo, correspondência semântica e exata. Em sistemas de RAG, essa integração melhora recall, precisão e robustez fora do domínio, especialmente quando o conteúdo é variado ou a consulta é ambígua. As estratégias atuais se organizam em torno de como fundir scores/resultados e de como reordenar (re-rankear) os documentos candidatos, muitas vezes em pipelines multiestágio.

## ESTRATÉGIAS

### Fusão de pontuações

Na forma mais simples, cada recuperador (denso e lexical) gera um score e eles são combinados via soma ponderada com um peso $\alpha$ que indica quanta confiança se dá a cada método. Com poucos exemplos de validação já é possível calibrar esse peso para um domínio específico, e essa abordagem costuma superar técnicas clássicas como RRF e CombSUM/CombMNZ em cenários modernos. A limitação é ser global e estática: o mesmo $\alpha$ vale para todas as consultas, mesmo quando algumas se beneficiariam mais de semântica e outras de matching exato.

### Fusão adaptativa (peso dinâmico)

Estratégias recentes ajustam a contribuição de cada método por consulta. No Dynamic Alpha Tuning (DAT), um LLM inspeciona os resultados iniciais de BM25 e do retriever denso, “julga” qual parece melhor para aquela query e, a partir disso, define um $\alpha$ específico para combinar os rankings. Abordagens baseadas em entropia de scores seguem lógica semelhante: se a distribuição de pontuações de um método é muito incerta (alta entropia, sem um claro vencedor), seu peso é reduzido em favor do outro. Outras variantes usam heurísticas simples (comprimento da query, termos raros, linguagem mais natural) ou até um modelo leve supervisionado para prever o peso ideal. Uma extensão direta é a fusão tri-modal, que adiciona um terceiro sinal (por exemplo, grafos de entidades) e deixa que um LLM ou modelo auxiliar pondere dinamicamente lexical, denso e grafo, especialmente útil em consultas complexas.

### Fusão de resultados e indexação unificada

Na prática, o padrão ainda é a fusão tardia: executar uma busca em cada

índice, unir os top-K de cada um e reordenar tudo com alguma regra de pontuação global. Isso garante alta cobertura, mas exige manter dois pipelines e lidar com a mesclagem dos resultados. Como alternativa, surgem índices híbridos unificados, onde cada documento é representado por uma combinação de dimensões densas e esparsas e consultado em um único passo. Para isso funcionar, é preciso alinhar escalas e distribuições desses dois tipos de vetor e, muitas vezes, usar buscas aproximadas em duas fases (primeiro só a parte densa, depois refino incluindo a componente esparsa). Esses métodos trazem ganhos de eficiência em larga escala, mas são mais complexos de projetar e ainda estão em maturação.

### Pipelines multiestágio

Em cenários de grande volume, é comum usar um pipeline em dois grandes estágios. No primeiro estágio, o foco é recall: BM25 e o retriever denso recuperam, em paralelo ou em sequência, um conjunto amplo de candidatos, que são unidos para reduzir o risco de perder documentos relevantes. No segundo estágio, um modelo mais caro (normalmente neural) faz o re-ranking desses candidatos, ordenando-os de forma muito mais precisa. Aqui tanto sinais lexicais quanto semânticos já estão embutidos no texto, permitindo ao modelo capturar relações de relevância que não aparecem apenas nas pontuações originais.

### Estratégias de re-ranking

O re-rank pode ser feito com LLMs gerais, via prompting para que o modelo atribua notas de relevância ou indique a melhor entre várias passagens, ou com re-ranqueadores neurais dedicados, como MonoBERT/MonoT5, ajustados especificamente para ranking em datasets como MS MARCO. Há ainda arquiteturas em cascata, nas quais um modelo leve filtra a maior parte dos candidatos e apenas um subconjunto reduzido é enviado para um modelo pesado (cross-encoder ou LLM). Resultados recentes mostram que, em setups híbridos, o ganho vem tanto da combinação denso+lexical na fase de recall quanto da “compatibilidade” entre o retriever e o re-ranqueador: às vezes embeddings menores, mas alinhados ao critério de relevância do LLM, funcionam melhor do que modelos de embedding muito grandes usados isoladamente.

## CONCLUSÃO

No conjunto, as estratégias de recuperação híbrida evoluem de fusões lineares simples para esquemas dinâmicos, multimodais e guiados por LLMs, sempre navegando os trade-offs entre cobertura, precisão, custo e latência. Em sistemas RAG, isso se traduz em contextos mais relevantes, menos ruído e melhor aproveitamento da janela de contexto do modelo gerador.

# Experimentos realizados

## Relatório de Benchmark

- Escopo: comparação entre BM25 puro, baselines híbridos com $\alpha$ fixo e DAT Hybrid (LLM-judge) em squad_small.
- Métricas: nDCG, MRR, MAP, Recall, Precision para $k \in \{1, 3, 5, 10, 20\}$.
- Latência: tempo aproximado por retriever.

## Principais resultados

- Melhor qualidade média (nDCG): dat_hybrid_gpt4o (0.868), seguido por dat_hybrid_gpt4o_mini (0.864).
- Melhor custo-benefício (nDCG/latência @k=10): bm25_only (0.001392) pela latência extremamente baixa (~0.57s), embora com qualidade bem inferior às híbridas.
- baseline_alpha_0.6 (0.856 avg nDCG) ficou próximo dos DATs, porém com latência substancialmente menor (~248s vs ~561–594s).
- baseline_alpha_1.0 (0.786) e baseline_alpha_0.0 (0.754) confirmam o ganho de combinar sinais.
- Por k (nDCG): dat_hybrid_gpt4o lidera em todos os ks (1,3,5,10,20).

## RESUMO DOS NÚMEROS (MÉDIA)

- DAT (gpt-4o): nDCG 0.868 | latência 593.6s
- DAT (gpt-4o-mini): nDCG 0.864 | latência 561.0s
- Baseline $\alpha$=0.6: nDCG 0.856 | latência 248.3s
- Baseline $\alpha$=1.0: nDCG 0.786 | latência 241.6s
- Baseline $\alpha$=0.0: nDCG 0.754 | latência 270.9s
- BM25 only: nDCG 0.753 | latência 0.57s

# APÊNDICE 10

# Termo de Aceite de Entrega

## Objetivo deste documento

Este documento faz parte do Processo da disciplina Residência em IA e tem como objetivo formalizar o aceite da entrega considerando o planejado e o realizado para o período.

## Data da Reunião ("Gate") de aprovação: 13 de nov. de 2025

## Participantes da Entrega [matriculados em Residência em IA]:

| |
|---|
| **THIAGO PEDROSO DE JESUS** |

## Entrega: [descrever a ENTREGA - requisitos e produtos gerados: links para textos, códigos, vídeos etc.]

Durante as nove primeiras Semanas, escolhi **Retrieval-Augmented Generation (RAG)** como tema da minha Residência, percorri seu desenvolvimento histórico, aprofundei as frentes de Recuperação, Geração, Integração e Avaliação, e desenvolvi uma estrutura em código para conduzir experimentos especificamente na área de **recuperação híbrida**, com foco na replicação dos trabalhos Rethinking Hybrid Retrieval: When Small Embeddings and LLM Re-ranking Beat Bigger Models (maio/25) e DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation (março/25).

Na última entrega, concluí a implementação do método DAT e alcancei resultados consistentes com os apresentados no artigo original. Com essa etapa validada e com um desempenho que considerei satisfatório, abri espaço para explorar possíveis expansões do estudo,  o que motivou os experimentos e desenvolvimentos realizados nesta Semana.

Portanto, as atividades desenvolvidas foram:

- Problema e proposta de melhoria
  - Durante a décima Semana, direcionei meus estudos para investigar se modelos abertos podem substituir modelos proprietários nas estratégias de recuperação híbrida dinâmica.O método DAT utiliza modelos da OpenAI (gpt-4o-mini e gpt-4o) como julgadores, e eu considero que isso pode ser inviável em cenários reais por custo e dependência do provedor.
  - Com isso, surgiu a pergunta norteadora da Semana: "É possível atingir um bom desempenho usando um modelo open-source ajustado especificamente para essa tarefa?"

- Coleta e preparação dos dados
  - Como fonte de dados de treino e avaliação, utilizei o dataset SQuAD 1.1 (The Stanford Question Answering Dataset).
  - Eu já havia trabalhado com esse dataset nas Semanas anteriores e já tinha familiaridade com a sua estrutura.

- Exploração e modelagem dos dados

- A partir dos rankings gerados pelos retrievers (BM25 e Denso), calculei automaticamente o peso de cada um convertendo sua posição no ranking em uma nota de relevância. Assim, para cada query, obtive um par de pesos que serviu como rótulo no dataset e permitiu treinar o modelo a arbitrar entre as duas modalidades.
- O dataset foi balanceado para incluir cenários em que as modalidades de retrievers concordam, discordam ou erram, garantindo diversidade e evitando vieses.
- Descrição do processo: Residência - Modelagem do Dataset

- Treinamento do modelo
  - Realizei o treinamento do modelo Llama-3.1-8B por meio de fine-tuning supervisionado, utilizando uma GPU A100-80GB na plataforma Lightning AI (https://lightning.ai/).
  - Organizei os hiperparâmetros e detalhes do treinamento em documento.
  - Descrição do treinamento:: Residência - Processo de Fine-Tuning

- Resultados dos experimentos
  - Conduzi experimentos comparando BM25, Dense, híbridos com pesos fixos e a estratégia DAT utilizando tanto o Llama-3.1-8B base quanto o modelo ajustado (Llama-FT).
  - Usei nDCG como métrica principal.
  - Dataset de testes
    - Llama-base: desempenho ligeiramente inferior a pesos fixos (0.779 vs 0.781).
    - Llama-FT: alcançou 0.804, superando o uso de pesos fixos (+2,9%) e também o GPT-4o (0.797).
  - Subset hybrid-sensitive (1500 queries) - Dataset mais “difícil” para atribuir pesos.
    - O Llama-base (0.569) foi superior ao baseline fixo (0.523), mas ainda fraco.
    - Llama-FT atingiu 0.703, superando o GPT-4o (0.660) em +6,5% no Top-1.
    - O Llama-FT também praticamente empatou com o GPT-4o no ranking geral (nDCG@20 de 0.832 vs 0.835).
  - Relatório completo dos experimentos: Residência - Relatório de Experimentos

- Documentação do repositório e laboratórios
  - Finalizei a documentação completa do repositório, que hoje funciona como uma estrutura flexível para testes de recuperação híbrida.
  - https://github.com/Thiago-Pedroso/hybrid-retrieval

## Planejamento: **[descrever o que pretende fazer para realizar a próxima ENTREGA]**

**Observação: [caso precise fazer alguma observação, de qualquer “natureza”]**

## ACEITE DA ENTREGA:

**CEDRIC LUIZ DE CARVALHO:** Go!

# Modelagem do dataset

Foi necessário fazer a criação de um dataset de treinamento com o objetivo de realizar o fine-tuning supervisionado (SFT) de um LLM, como o Llama. O objetivo é que este LLM atue como um "juiz" de relevância, substituindo o GPT-4o no framework DAT (Dynamic Alpha Tuning).

O LLM Judge será treinado para prever dois scores de relevância (0-5), um para o documento top-1 do retriever BM25 e outro para o documento top-1 do retriever Dense.

## INTUIÇÃO DA MODELAGEM

O principal desafio é ensinar um LLM local a replicar a capacidade de julgamento de um modelo proprietário (como o GPT-4o) para a tarefa de reranking híbrido. O framework DAT necessita de um componente que, para qualquer query, decida dinamicamente a importância relativa (o alfa) do retriever esparso (BM25) versus o denso.

Para treinar um LLM a fazer isso, precisei criar um dataset que contenha "exemplos de julgamento" com respostas corretas (labels). A modelagem do meu dataset foi guiada pelas seguintes intuições:

### O Problema: Decisão Dinâmica (Alfa)

O objetivo final não é apenas classificar um documento como "relevante" ou "irrelevante". O objetivo é calcular um alfa dinâmico:

alfa = score_dense / (score_dense + score_bm25)

Se o LLM for treinado para prever score_dense e score_bm25 de forma acurada, posso calcular esse alfa no momento da inferência. Portanto, a tarefa de SFT do LLM é:

- Input: Query, Contexto BM25 Top-1, Contexto Dense Top-1
- Output (Previsão): score_bm25 (int 0-5), score_dense (int 0-5)

## O Label: De "Ground Truth" para Scores (0-5)

Como eu não tinha um "juiz humano" ou o GPT-4o para gerar os scores, precisei de um proxy de relevância. Usei o dataset SQuAD, que fornece um "ground truth" (documento gold) para cada query.

A intuição central é: a relevância de um retriever é proporcional à sua capacidade de ranquear o documento gold o mais alto possível.

Decidi observar o Top-5 de cada retriever. Um resultado que coloca o documento gold em 1º lugar é excelente (Score 5), enquanto um que o coloca em 5º é fraco (Score 1). Se o documento gold nem aparece no Top-5, é irrelevante (Score 0).

Isso me deu o seguinte mapeamento (rank → score):

- Rank 1 → Score 5 (Excelente)
- Rank 2 → Score 4 (Bom)
- Rank 3 → Score 3 (Médio)
- Rank 4 → Score 2 (Fraco)
- Rank 5 → Score 1 (Muito Fraco)
- Não encontrado no Top-5 → Score 0 (Irrelevante)

Este mapeamento rank_to_score se tornou o meu label (a "resposta correta") para score_bm25 e score_dense.

## O Foco: Treinando nos Casos "Difíceis" (Hybrid-Sensitive)

O dataset SQuAD completo possui ~98k queries. No entanto, em muitos casos:

1. Ambos os retrievers acertam (ex: ambos dão Score 5).
2. Ambos os retrievers erram (ex: ambos dão Score 0).

Embora úteis, esses casos "fáceis" não ensinam o LLM a escolher entre os dois. O verdadeiro aprendizado ocorre nos casos "difíceis" ou "hybrid-sensitive": aqueles onde os retrievers divergem.

Por isso, a minha estratégia de modelagem foca em criar um subset de treino que:

1. Filtra por Discordância: Inclui primariamente queries onde score_bm25 != score_dense. É aqui que o julgamento do alfa é crucial.
2. Balanceia os Cenários: O subset é estratificado por buckets de alpha_label (o alfa calculado a partir dos scores reais). Isso garante que o LLM veja uma distribuição equilibrada de exemplos:
    - alpha = 0.0 (Apenas BM25 está correto)

- alpha = 1.0 (Apenas Dense está correto)
- 0 < alpha < 1 (Ambos têm algum nível de relevância, mas diferente)

Essa abordagem foca o esforço computacional do SFT nos exemplos que mais importam para a tarefa de decisão dinâmica.

# Processo de Fine-Tuning do LLM Judge

## OBJETIVO DO FINE-TUNING

O objetivo principal foi treinar um modelo de linguagem (LLM) de código aberto para uma tarefa de "juiz" muito específica: prever dois scores de relevância (0-5), um para o resultado top-1 do BM25 e outro para o resultado top-1 do Dense.

O modelo treinado (o "adapter") seria então usado para calcular o alfa dinâmico (alfa = score_dense / (score_dense + score_bm25)), permitindo que um LLM local e eficiente substituísse o GPT-4o no framework DAT (Dynamic Alpha Tuning).

## DECISÕES DE MODELAGEM E SETUP

Para executar esta tarefa de forma eficiente em uma única GPU (NVIDIA A100-80GB), tomei as seguintes decisões:

### Escolha do Modelo Base

**Decisão:** Utilizar o meta-llama/Llama-3.1-8B-Instruct.

**Raciocínio:**

1. **Tamanho:** 8B parâmetros é um excelente equilíbrio entre capacidade (performance) e eficiência (custo de inferência/treino).
2. **Versão "Instruct":** Modelos "Instruct" já são pré-treinados para seguir instruções. Como minha tarefa de SFT é essencialmente instruir o modelo a seguir um conjunto de regras (o critério de scoring 0-5), começar com um modelo Instruct acelera o aprendizado e melhora a aderência ao formato de saída desejado.

## Eficiência de Treinamento (QLoRA)

**Decisão:** Adotar a técnica de QLoRA (Quantized Low-Rank Adaptation).

**Raciocínio:** Treinar um modelo de 8B parâmetros completamente exigiria múltiplos GPUs. Para viabilizar o treino em um único hardware, o QLoRA foi a escolha ideal, combinando duas técnicas:

1. **Quantização (4-bit):** Carreguei o modelo base Llama-3.1-8B em 4-bit. Isso reduziu drasticamente a memória de GPU necessária para carregar o modelo.
2. **Adapters LoRA (PEFT):** Em vez de treinar todos os 8 bilhões de parâmetros, congelei o modelo base e treinei apenas "adapters" (camadas LoRA) de baixa dimensionalidade (r=16, lora_alpha=32) em módulos chave de atenção (q_proj, k_proj, v_proj, o_proj). Isso reduziu a memória necessária para o treinamento.

## O Processo de Fine-Tuning Supervisionado (SFT)

O SFT funciona ensinando o modelo a mapear um input (prompt) a um output desejado.

O processo de SFT foi configurado para replicar a metodologia do paper original.

- Prompt (Input): Utilizei o template de prompt exato descrito no paper. Este prompt instrui o modelo sobre sua tarefa, define os critérios de scoring de 0 a 5 e especifica o formato de saída. O prompt recebia a pergunta, o dense retrieval Top1 Result e o BM25 retrieval Top1 Result.
- Dataset (Input): Utilizei o subset "hybrid-sensitive" balanceado (train.parquet, 30.000 exemplos) criado na etapa anterior. A decisão de usar este subset foi para focar o treino nos casos "difíceis", onde score_bm25 != score_dense, pois é nesses casos que o julgamento do LLM é mais valioso.
- Completion (Output): O output que o modelo deveria aprender a gerar era simplesmente a string com os scores "ground truth" (derivados do rank), no formato "D B" (exemplo: 5 1), conforme especificado no prompt.

O SFTTrainer (da biblioteca TRL - Transformer Reinforcement Learning) foi configurado para alimentar o modelo com o prompt e treiná-lo para gerar o completion esperado.

# Experimentos finais

## RESUMO

Este relatório detalha uma série de experimentos de hybrid retrieval realizados no dataset SQUAD. O objetivo principal foi avaliar a eficácia do método DAT (Dynamic Alpha Tuning) em comparação com baselines de retrieval tradicionais (BM25, Dense) e híbridos de alpha fixo.

Os resultados confirmam que a abordagem híbrida (BM25 + Dense) é superior às suas contrapartes puras. O método DAT, que ajusta dinamicamente o peso (alfa) entre os retrievers BM25 e Dense por query, demonstrou ser a estratégia mais eficaz, especialmente em cenários complexos ("hybrid sensitive").

O principal achado desta investigação é que um modelo de linguagem aberto (Llama-3.1-8B), após um processo de fine-tuning específico para a tarefa de "juiz" do DAT, apresenta um desempenho não apenas competitivo, mas superior ao GPT-4o no Top-1 (nDCG@1) em um subconjunto de validação robusto. Este resultado sugere que é viável substituir modelos proprietários caros por soluções open-source customizadas, obtendo melhor desempenho em métricas-chave, com maior privacidade e menor custo operacional.

## INTRODUÇÃO E CONTEXTO

O Hybrid Retrieval, que combina a força léxica do BM25 (esparso) com a compreensão semântica de modelos dense (vetoriais), tornou-se um padrão. O método DAT (Dynamic Alpha Tuning) propõe uma melhoria sobre a fusão híbrida de alpha fixo, utilizando um LLM como "juiz" para decidir o peso ideal entre os retrievers para cada query específica.

Meus experimentos buscam:

1. Estabelecer o desempenho de baselines (BM25, Dense e Híbrido Fixo) no SQUAD.
2. Replicar os resultados do paper do DAT usando LLMs proprietários (GPT-4o, GPT-4o-mini) sob uma única base de dados.
3. Investigar a viabilidade de substituir o LLM juiz por um modelo open-source (Llama-3.1-8B).
4. Melhorar o modelo open-source através de fine-tuning e avaliar seu impacto.

5. Analisar o desempenho em um subconjunto de teste rápido ("hybrid sensitive" pequeno).
6. Validar os resultados em um subconjunto "hybrid sensitive" maior e mais robusto (1500 queries).

As métricas-chave avaliadas são nDCG (relevância e ranking), MRR (ranking do primeiro documento correto) e Recall (cobertura total de documentos relevantes), em diferentes valores de k.

## EXPERIMENTO 1: ANÁLISE DOS BASELINES (DATASET ORIGINAL)

Primeiro, estabeleço o desempenho dos métodos de retrieval padrão no dataset SQUAD original.

- BM25 (alpha=0.0): Método léxico puro.
- Dense (alpha=1.0): Método vetorial/semântico puro.
- Híbrido Fixo (alpha=0.6): Misto com peso fixo (60% Dense, 40% BM25), apontado como ótimo na literatura.

**Tabela 1: Comparativo de Desempenho dos Baselines (Dataset SQUAD Completo)**

| Retriever | k | nDCG | MRR | Recall | Precision |
|---|---|---|---|---|---|
| **BM25 (α=0.0)** | 1 | 0.666 | 0.666 | 0.666 | 0.666 |
| | 5 | 0.761 | 0.737 | 0.831 | 0.166 |
| | 20 | 0.802 | 0.752 | 0.971 | 0.049 |
| **Dense (α=1.0)** | 1 | 0.685 | 0.685 | 0.685 | 0.685 |

| | | | | | |
|---|---|---|---|---|---|
| | 5 | 0.805 | 0.772 | 0.904 | 0.181 |
| | 20 | 0.831 | 0.782 | 0.990 | 0.049 |
| **Híbrido Fixo (α=0.6)** | 1 | **0.781** | **0.781** | **0.781** | **0.781** |
| | 5 | **0.874** | **0.848** | **0.950** | **0.190** |
| | 20 | **0.887** | **0.853** | **0.994** | **0.050** |

### Análise - Baselines

O Híbrido Fixo (alpha=0.6) supera tanto o BM25 puro quanto o Dense puro em todas as métricas e em todos os valores de k. Isso confirma a hipótese de que a combinação dos sinais léxicos e semânticos é superior a qualquer um deles isoladamente. O ganho mais expressivo está no Top-1 (nDCG@1 e MRR@1), onde o modelo híbrido atinge 0.781, um salto de quase 10 pontos percentuais sobre o Dense (0.685).

## EXPERIMENTO 2: ANÁLISE DO DAT (DATASET ORIGINAL)

Nesta fase, comparo o melhor baseline (Híbrido Fixo) com a metodologia DAT no dataset SQUAD completo, utilizando diferentes LLMs como juiz.

- GPT-4o-mini e GPT-4o: Réplica dos experimentos do paper com modelos proprietários.
- Llama-3.1-8B: Teste com modelo open-source base.
- Llama-3.1-8B (Finetuned): Teste com meu modelo open-source customizado.

**Tabela 2: Comparativo de Desempenho - DAT vs. Baseline (Dataset SQUAD Completo)**

| Retriever | k | nDCG | MRR | Recall | t_retrieve_sec |
|---|---|---|---|---|---|
| Híbrido Fixo (α=0.6) | 1 | 0.781 | 0.781 | 0.781 | 248.3 s |
| (Baseline) | 20 | 0.887 | 0.853 | 0.994 | |
| **DAT (GPT-4o-mini)** | 1 | 0.787 | 0.787 | 0.787 | 561.0 s |
| | 20 | 0.894 | 0.861 | 0.996 | |
| **DAT (GPT-4o)** | 1 | 0.797 | 0.797 | 0.797 | 593.6 s |
| | 20 | 0.895 | 0.865 | 0.990 | |
| **DAT (Llama-3.1-8B)** | 1 | 0.779 | 0.779 | 0.779 | 947.9 s |
| | 20 | 0.882 | 0.848 | 0.992 | |
| **DAT (Llama-FT)** | 1 | **0.804** | **0.804** | **0.804** | 959.1 s |
| | 20 | 0.886 | 0.858 | 0.979 | |

## Análise - DAT (Dataset Completo)

1. **DAT vs. Baseline Fixo:** Os modelos DAT (GPT-4o e GPT-4o-mini) superam o baseline de alpha fixo, validando a eficácia do ajuste dinâmico. O GPT-4o atinge o melhor desempenho geral (nDCG@20 de 0.895).
2. **Desempenho do Llama-Base:** O Llama-3.1-8B base (sem fine-tuning) teve um desempenho ligeiramente inferior ao baseline fixo. Isso indica que o modelo, "out-of-the-box", não possui a capacidade de discernimento necessária para atuar como um juiz eficaz nesta tarefa.
3. **Impacto do Fine-Tuning:** O resultado mais legal é o do Llama-3.1-8B Finetuned (Llama-FT). Ele não apenas supera o Llama-base, mas ultrapassa o GPT-4o no nDCG@1 e MRR@1 (0.804 vs 0.797). Isso é crucial para aplicações onde a precisão do primeiro resultado é o mais importante.
4. **Trade-off de Recall:** O Llama-FT apresenta um Recall@20 ligeiramente menor (0.979) em comparação com os outros (~0.99). Isso sugere que o fine-tuning especializou o modelo em ranquear o documento mais relevante no topo (maior precisão no Top-1), possivelmente em detrimento de trazer todos os documentos relevantes para o Top-20.
5. **Custo Computacional:** O método DAT introduz uma latência significativa. Os modelos Llama (rodando via Ollama, localmente) são os mais lentos (~950s), enquanto os modelos GPT (API externa) são mais rápidos (~580s), mas ainda mais lentos que o baseline (248s). A lentidão do Llama se deve provavelmente à inferência não otimizada (sem batching) e às limitações do hardware utilizado.

## EXPERIMENTO 3: ANÁLISE DO SUBSET “HYBRID SENSITIVE” (PEQUENO - 479 QUERIES)

Desconfiei que o dataset SQUAD completo pode ser "fácil", não exigindo muito do juiz DAT. Criei um primeiro subconjunto "Hybrid Sensitive" (479 queries, 585 documentos) para um teste rápido, contendo apenas queries onde o documento Top-1 do BM25 é diferente do Top-1 do modelo Dense. Este é um teste de estresse que força o juiz DAT a fazer uma escolha significativa.

**Tabela 3: Comparativo de Desempenho - Subset "Hybrid Sensitive" (Pequeno)**

| Retriever | k | nDCG | MRR | Recall | t_retrieve_sec |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| **Baseline (α=0.6)** | 1 | 0.662 | 0.662 | 0.662 | 133.2 s |
| | 5 | 0.796 | 0.758 | 0.911 | |
| | 20 | 0.821 | 0.767 | 0.989 | |
| **DAT (Llama-3.1-8B)** | 1 | 0.651 | 0.651 | 0.651 | 541.0 s |
| | 5 | 0.782 | 0.745 | 0.892 | |
| | 20 | 0.811 | 0.756 | 0.989 | |
| **DAT (Llama-FT)** | 1 | **0.691** | **0.691** | **0.691** | 550.5 s |
| | 5 | 0.786 | 0.761 | 0.859 | |
| | 20 | 0.818 | 0.773 | 0.963 | |
| **DAT (GPT-4o)** | 1 | **0.691** | **0.691** | **0.691** | 319.7 s |
| | 5 | **0.815** | **0.781** | **0.914** | |

| | | | | | |
|---|---|---|---|---|---|
| | 20 | **0.836** | **0.789** | **0.985** | |

### Análise - Subset "Hybrid Sensitive" (Pequeno)

6. **Confirmação da Dificuldade:** O desempenho de todos os modelos caiu significativamente neste subconjunto (ex: nDCG@1 do DAT caiu de ~0.80 para ~0.69), confirmando que ele é mais desafiador.
7. **DAT vs. Baseline Fixo:** O DAT (Llama-FT e GPT-4o) superou o baseline de alpha fixo (0.662) no nDCG@1, atingindo 0.691, mostrando o valor do ajuste dinâmico.
8. **Falha do Llama-Base:** O Llama-base (sem tuning) teve um nDCG@1 de 0.651, ficando abaixo do baseline de alpha fixo.
9. **Llama-FT vs. GPT-4o:** Neste teste rápido, o Llama-FT iguala perfeitamente o desempenho do GPT-4o no k=1, demonstrando que o fine-tuning foi bem-sucedido.
10. **Vantagem do GPT-4o (k > 1):** Para k > 1, o GPT-4o manteve uma ligeira vantagem no ranking geral (ex: nDCG@20 de 0.836 vs 0.818 do Llama-FT).

## EXPERIMENTO 4: VALIDAÇÃO NO SUBSET “HYBRID SENSITIVE” (GRANDE - 1500 QUERIES)

Para validar os resultados promissores do "Experimento 3" em um cenário mais robusto, construí um subset "Hybrid Sensitive" maior, com 1500 queries e 19027 documentos. Os resultados deste teste em maior escala são apresentados abaixo.

**Tabela 4: Comparativo de Desempenho - Subset "Hybrid Sensitive" (Grande)**

| Retriever | k | nDCG | MRR | Recall | t_retrieve_sec |
|---|---|---|---|---|---|
| **Baseline (α=0.6)** | 1 | 0.523 | 0.523 | 0.523 | 1592.1 s |
| | 5 | 0.757 | 0.694 | 0.945 | |

| | | | | | |
|---|---|---|---|---|---|
| | 20 | 0.776 | 0.702 | 1.000 | |
| **DAT (Llama-3.1-8B)** | 1 | 0.569 | 0.569 | 0.569 | 3083.5 s |
| | 5 | 0.749 | 0.699 | 0.897 | |
| | 20 | 0.778 | 0.710 | 0.994 | |
| **DAT (GPT-4o)** | 1 | 0.660 | 0.660 | 0.660 | 2021.9 s |
| | 5 | 0.822 | 0.778 | 0.952 | |
| | 20 | **0.835** | **0.783** | 0.993 | |
| **DAT (Llama-FT)** | 1 | **0.703** | **0.703** | **0.703** | 3249.4 s |
| | 5 | **0.813** | **0.783** | 0.901 | |
| | 20 | 0.832 | **0.789** | 0.969 | |

## Análise - Subset "Hybrid Sensitive" (Grande)

Este experimento em maior escala não apenas validou, mas fortaleceu as conclusões anteriores.

1. **DAT vs. Baseline Fixo (Aprofundado):** A superioridade do método DAT tornou-se muito mais evidente. O baseline de alpha fixo (α=0.6) teve um colapso no desempenho (nDCG@1 de 0.523), mostrando-se inadequado para cenários complexos. Todos os modelos DAT (incluindo o Llama-base) o superaram no Top-1.
2. **Impacto do Fine-Tuning (Aprofundado):** O Llama-base (nDCG@1 de 0.569) novamente se mostrou fraco, reforçando que o fine-tuning é o componente essencial. O Llama-FT (nDCG@1 de 0.703) teve um desempenho massivamente superior ao Llama-base.
3. **Llama-FT vs. GPT-4o (Resultado Principal):** Este é o achado mais significativo. Ao contrário do empate no subset pequeno, neste teste mais robusto, o Llama-FT superou o GPT-4o no nDCG@1 (0.703 vs 0.660). Isso sugere que o modelo open-source "finetunado" é, de fato, um juiz mais preciso para o ranking Top-1 em queries difíceis.
4. **Vantagem (k > 1):** O Llama-FT também se mostrou extremamente competitivo no ranking geral, praticamente empatando com o GPT-4o no nDCG@20 (0.832 vs 0.835) e superando-o ligeiramente no MRR@20 (0.789 vs 0.783).

## Conclusões Gerais

1. **Híbrido é Superior:** O Hybrid Retrieval (com alpha fixo ou dinâmico) supera consistentemente os métodos puros (BM25 ou Dense) no dataset SQUAD completo.
2. **DAT é Essencial para Casos Difíceis:** O Dynamic Alpha Tuning (DAT) supera drasticamente o baseline de alpha fixo em cenários "hybrid sensitive", provando que o ajuste dinâmico por query é crucial para a robustez.
3. **O "Juiz" Importa e "Out-of-the-Box" Falha:** Um LLM juiz "out-of-the-box" (como o Llama-3.1-8B base) não possui a capacidade de discernimento necessária e tem um desempenho fraco, às vezes pior que um baseline fixo.
4. **Viabilidade do Open-Source Finetuned:** O fine-tuning de um modelo open-source (Llama-3.1-8B) provou ser uma estratégia de grande sucesso. Meu modelo customizado (Llama-FT) alcançou um desempenho competitivo, superando o GPT-4o em métricas Top-1 (nDCG@1) no subconjunto "Hybrid Sensitive" mais robusto. Isso fortalece a evidência de que o modelo customizado é um substituto viável e, para esta tarefa específica, superior.
5. **Implicações Práticas:** É viável construir um sistema de hybrid retrieval de ponta, privado e de baixo custo (inferência), substituindo LLMs proprietários por um modelo open-source menor e especializado, obtendo ganhos de performance em algumas métricas sem sacrificar o desempenho geral do ranking.