

Automatização de Consultas SQL com Retrieval-Augmented Generation (RAG)

Uma Abordagem para a Conversão de Texto aplicada a
Dados do Campeonato Brasileiro

Evellyn Nicole Machado Rosa



UFG

UNIVERSIDADE
FEDERAL DE GOIÁS

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

EVELLYN NICOLE MACHADO ROSA

**Automatização de Consultas SQL com Retrieval-Augmented
Generation (RAG)**

Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato
Brasileiro

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): EVELLYN NICOLE MACHADO ROSA

Título do trabalho: Automatização de Consultas SQL com Retrieval-Augmented Generation (RAG)

Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato Brasileiro

2. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador) Concorda com a liberação total do documento SIM NÃO¹

[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(a)(s) autor(a)(es)(as) e ao(a) 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 **Evellyn Nicole Machado Rosa, Discente**, em 13/01/2025, às 08:54, 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 15/01/2025, às 16:12, 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 **5089556** e o código CRC **138022ED**.

Referência: Processo nº 23070.001555/2025-15

SEI nº 5089556

EVELLYN NICOLE MACHADO ROSA

**Automatização de Consultas SQL com Retrieval-Augmented
Generation (RAG)**

Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato
Brasileiro

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.

ROSA, EVELLYN NICOLE MACHADO

Automatização de Consultas SQL com Retrieval-Augmented Generation (RAG) [manuscrito] : Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato Brasileiro / EVELLYN NICOLE MACHADO ROSA. - 2025.

99 f.

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. geração aumentada de recuperação. 3. base de dados. I. Federson, Fernando Marques , orient. II. Título.

CDU 004


EVELLYN NICOLE MACHADO ROSA

**Automatização de Consultas SQL com Retrieval-Augmented
Generation (RAG)**

Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato Brasileiro

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: 17 de dezembro de 2024.



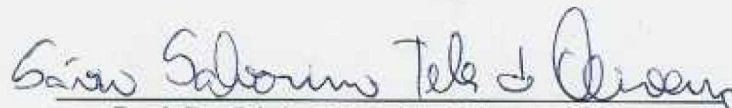
Prof. Dr. Fernando Marques Federson
Orientador (INF-UFG)



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



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



Prof. Dr. Sávio Salvarino Teles de Oliveira
(INF-UFG)

EVELLYN NICOLE MACHADO ROSA

Automatização de Consultas SQL com Retrieval-Augmented Generation (RAG)

Uma Abordagem para a Conversão de Texto aplicada a Dados do Campeonato
Brasileiro

RESUMO

Este Relatório de Conclusão de Curso tem como objetivo reunir os resultados da minha jornada para me tornar um especialista em **Geração Aumentada de Recuperação (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, modelos grandes de linguagem, geração automática de datasets.

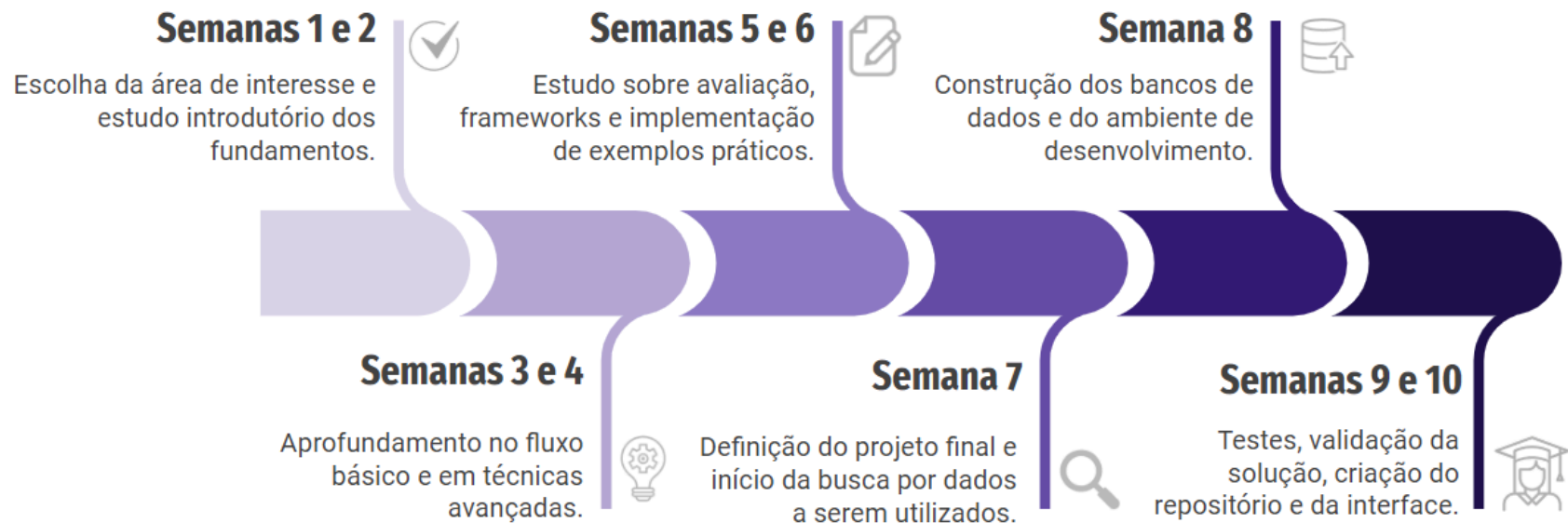
ABSTRACT

This Course Completion Report aims to bring together the results of my journey to become an expert in **Recovery 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, large language models, automatic dataset generation.

Goiânia
2025

Minha Jornada



Evellyn Nicole Machado Rosa

Especialista em: Geração Aumentada de Recuperação (RAG)

MINHA JORNADA

Nome: Evellyn Nicole Machado Rosa

Especialidade: Geração Aumentada de Recuperação (RAG)

Objetivo deste documento

Durante o processo da disciplina Residência em IA¹, 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

Minha jornada começou com a decisão de explorar as áreas de interesse dentro da conferência Computer Science, Computer Engineering, and Applied Computing (CSCE'24). Para organizar minhas ideias e avaliar as possibilidades, criei um mapa mental que me ajudou a visualizar as opções disponíveis. Esse processo me levou a escolher Natural Language Processing (NLP) como o tema central da minha residência. No entanto, reconheci que a área de NLP é extremamente ampla e que, para me tornar especialista, seria necessário delimitar um foco específico. Após refletir sobre os avanços e desafios do campo, decidi direcionar meus estudos para a área de Retrieval-Augmented Generation (RAG), um tema promissor e altamente relevante no contexto atual. Na **Semana 1**, como parte do primeiro Stage, que envolvia história, fundamentos e categorias do tema escolhido, procurei construir uma base sólida. Encontrei um survey abrangente que cobria tópicos essenciais sobre RAG, o que me proporcionou um ponto de partida valioso. A leitura detalhada desse material me ajudou a compreender as principais problemáticas, conceitos, técnicas e métricas de avaliação da área. Além disso, tive uma conversa muito

¹ Dez semanas, entre setembro de 2024 e dezembro de 2024.

enriquecedora com o professor Sávio Salvarino Teles de Oliveira, que trouxe insights valiosos sobre a relevância do tema na prática. Ele destacou que as empresas estão ansiosas por ferramentas conversacionais capazes de interagir com os próprios dados, reforçando ainda mais a importância e a aplicabilidade do Retrieval-Augmented Generation. Essa perspectiva me motivou a explorar como as técnicas de RAG podem ser direcionadas para atender a essa demanda crescente. Na **Semana 2**, meu objetivo foi compreender o fluxo básico do Retrieval-Augmented Generation (RAG) e identificar as partes mais importantes que o compõem. Para alcançar esse objetivo, organizei meus estudos em três fases principais: Indexação, Recuperação e Geração. Para essa etapa, utilizei a documentação do LangChain, que apresenta uma estrutura clara e bem organizada sobre cada uma dessas fases, além de listar as técnicas associadas a cada etapa de forma sistemática. Complementando meus estudos, explorei diferentes blogs que abordam essas partes separadamente, oferecendo uma riqueza de detalhes e exemplos práticos que contribuíram significativamente para meu aprendizado. As referências lidas, assim como algumas observações que considero importantes, podem ser obtidas em detalhes no material disponibilizado no **Apêndice 1**.

Nas **Semanas 3 e 4**, meu foco esteve na exploração de técnicas avançadas de Retrieval-Augmented Generation (RAG) e na compreensão do GraphRAG como uma evolução desse conceito. No Stage 3, estudei melhorias já consolidadas no LangChain, como Multi-Query, Step Back, RAG Fusion e HyDE, que aprimoram a tradução de consultas, além de estratégias avançadas de indexação, como Multi-representation Indexing, Specialized Embeddings e Indexação Hierárquica. Essas técnicas proporcionam maior precisão e flexibilidade no fluxo básico de RAG. No Stage 4, aprofundi-me no GraphRAG, guiada por um paper da Microsoft, que introduz grafos de conhecimento para estruturar entidades e relações em um corpus. Analisei suas vantagens, como a pesquisa global e local e a criação de comunidades densamente conectadas, bem como seus desafios, como o custo elevado. Para consolidar os aprendizados, produzi anotações detalhadas e criei um desenho que representa visualmente o fluxo de RAG e GraphRAG, evidenciando suas diferenças e complementaridades. Essas semanas foram fundamentais para compreender a

evolução e as possibilidades de aplicação dessas técnicas. Isso tudo pode ser conferido mais profundamente no **Apêndice 2**.

Nas **Semanas 5 e 6**, avancei no estudo de métricas de avaliação, frameworks e na implementação prática de RAG. Na **Semana 5**, explorei métricas para sistemas RAG com base no artigo Evaluation of Retrieval-Augmented Generation: A Survey. O texto explora os desafios na definição de métricas eficazes para sistemas RAG, dada a complexidade de seus componentes interativos. Uma das contribuições centrais do artigo é o Auepora, um Processo Unificado de Avaliação que estrutura a análise em torno de três questões principais: “O que avaliar?” (definição do alvo), “Como avaliar?” (dataset de referência) e “Como medir?” (métricas). As métricas apresentadas foram divididas em duas categorias principais: Recuperação, que mede a pertinência dos documentos recuperados em relação à consulta, e Geração, que avalia a qualidade das respostas com base nas informações recuperadas. Após o entendimento de todos os processos que envolvem o RAG, desenvolvi um código básico indexando o PPC do Bacharelado em Inteligência Artificial para aplicar os conceitos aprendidos, registrando os detalhes em anotações e código correspondentes. Na **Semana 6**, foquei em aprimorar o fluxo básico de RAG criado anteriormente, incorporando técnicas avançadas como Chunk Semântico, Multi Query Retriever, Pesquisa Híbrida e Reranker. Também implementei uma estrutura de grafos para explorar os relacionamentos nos documentos analisados. Fiz isso para entender como técnicas avançadas podem melhorar o fluxo mais básico. Com isso, desenvolvi um código no Google Colab que pode ser acessado no **Apêndice 3**.

Após uma jornada intensa explorando história, fundamentos, categorias e frameworks de Retrieval-Augmented Generation (RAG), percebi que era hora de definir uma trilha clara para as semanas finais do meu projeto. Refleti sobre as áreas que mais me interessam e uma delas se destacou: o futebol. No entanto, me perguntei: "Como posso aplicar RAG no contexto do futebol?" Foi então que me lembrei da conversa que tive com o Professor Sávio, onde discutimos o crescente desejo por ferramentas conversacionais capazes de fornecer respostas rápidas e precisas. Isso despertou em mim a vontade de criar algo semelhante: uma ferramenta que, com apenas uma pergunta, pudesse entregar

informações relevantes de forma ágil. Na **Semana 7**, decidi que meu projeto final seria o desenvolvimento de um chatbot conversacional utilizando RAG, focado em dados do Campeonato Brasileiro de Futebol (Série A). Para começar, busquei uma fonte confiável de dados sobre o Brasileirão. Entrei em contato com a Transfermarkt, uma plataforma de análise de futebol, e após preencher os documentos necessários, obtive acesso a alguns dados. Contudo, percebi que esses dados eram limitados, abrangendo apenas informações sobre jogadores, enquanto eu precisava de dados mais abrangentes, como partidas, tabelas, artilharia e rodadas futuras. Diante dessa limitação, procurei alternativas e encontrei a API Futebol, que se mostrou ideal para atender às necessidades do projeto. A API forneceu dados ricos e variados, incluindo informações sobre partidas, artilheiros, próximas rodadas e a tabela do campeonato, permitindo uma visão completa do Brasileirão. Todo o processo descrito pode ser encontrado com mais detalhes no **Apêndice 4**.

Com os dados em mãos, na **Semana 8**, minha abordagem foi construir o meu banco de dados SQLite a partir do que foi coletado e integrar RAG com SQL, permitindo que o LLM formulasse queries precisas e relevantes para consultar o banco de dados. O fluxo foi estruturado da seguinte forma: as informações sobre o Brasileirão foram indexadas em um banco de dados vetorial no Qdrant, possibilitando a recuperação semântica dos dados a partir de perguntas feitas pelo usuário. A partir dessas informações recuperadas, o modelo recebeu instruções no prompt para gerar queries SQL com base no esquema do banco de dados, documentação das tabelas e exemplos de queries corretas. Assim, as perguntas feitas ao chatbot resultam em respostas informadas e personalizadas, alimentadas por dados confiáveis e organizados. Com a indexação no Qdrant concluída, iniciei a construção da parte de geração do modelo, focando na eficiência da integração entre RAG, SQL e o modelo de linguagem. Com isso, já comecei a testar a geração de sql para ver quais são os acertos e as dificuldades do modelo e como a recuperação ajudou e pode ajudar. Notei que o modelo estava tendo uma dificuldade para gerar queries que envolvem datas, número de vitórias, derrotas e empates consecutivos e queries que envolvem mais de uma pergunta. Assim, precisa traçar estratégias para superar esses desafios. No **Apêndice 5** é detalhado todo esse desenvolvimento.

Na **Semana 9**, após identificar as dificuldades encontradas nos testes anteriores, meu foco foi explorar estratégias para melhorar a performance do modelo. Com esse objetivo, expandi o banco de dados vetorial adicionando mais exemplos, enriquecendo a base de conhecimento do sistema. Além disso, desenvolvi a solução fora do ambiente Vanna AI, utilizando o Qdrant e o LangChain como ferramentas principais. Paralelamente, ampliei o dataset de testes, o que permitiu avaliar o sistema em cenários mais diversificados, resultando em respostas satisfatórias. Logo, finalizei essa etapa implementando uma parte conversacional que proporciona uma interação fluida com o usuário e subi os códigos no meu repositório do GitHub. Na **Semana 10**, dediquei-me à criação de uma interface que viabilizasse a interação do usuário com o chatbot, considerando que a proposta do projeto é oferecer uma experiência conversacional intuitiva e eficiente. Para complementar, desenhei o fluxo final do projeto, consolidando as etapas e funcionalidades desenvolvidas ao longo da residência. Todo o processo e os detalhes dessa fase estão registrados no **Apêndice 6**. Observação: o fim do campeonato brasileiro foi no dia 08 de dezembro. Dessa maneira, irei atualizar o banco de dados com tudo o que aconteceu no ano de 2024.

Ao final desta jornada, gostaria de registrar o quanto esse processo foi transformador. Ele me mostrou que, com organização, comprometimento e uma vontade genuína de evoluir a cada semana, é possível se desenvolver em áreas de interesse, mesmo nos momentos em que nos sentimos incapazes. A Residência não apenas ampliou meus conhecimentos técnicos, mas também reforçou a importância de adotar e intercalar entre a mentalidade do Faraó e do Tales para alcançar o tão desejado pote de ouro. Saio desse curto período, porém intenso, mais confiante, determinada e com a certeza de que, com dedicação e amor pelo processo, tudo é possível.

APÊNDICE 1

Termo de Aceite de Entrega 1

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 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Como eu ainda estava no processo de escolha dentro de uma área de NLP, incluí alguns tópicos da semana 0 no meu termo de aceite.

Semana 0:

- Criação de um [mapa mental](#) para explorar áreas de interesse dentro do CSCE.
- Decisão de dedicar a residência ao tema Retrieval-Augmented Generation (RAG).

Semana 1:

Como parte do primeiro stage envolvia **história, fundamentos e categorias**, encontrei um [survey](#) para contemplar esses tópicos. Fiz uma leitura para me auxiliar a ter um ponto de partida na minha jornada para ser especialista em RAG. Esse estudo me proporcionou o entendimento de problemáticas, principais conceitos, comparações, técnicas e avaliação.

Link com todas as anotações e insights do survey: [EvellynNicole - Stage1](#)

Link com rascunhos das partes mais importantes realizados no próprio survey: [rascunho](#)

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

Pretendo, para a próxima semana, mergulhar mais nos conceitos chaves de RAG, como: Indexação, recuperação, embeddings, similaridade e geração. Com o intuito de entender as principais técnicas utilizadas dentro desses conceitos e como eles trabalham juntos para o todo. Além do mais, paralelo a isso, quero ter um olhar atento a quais e como os **frameworks** aplicam tais conceitos.

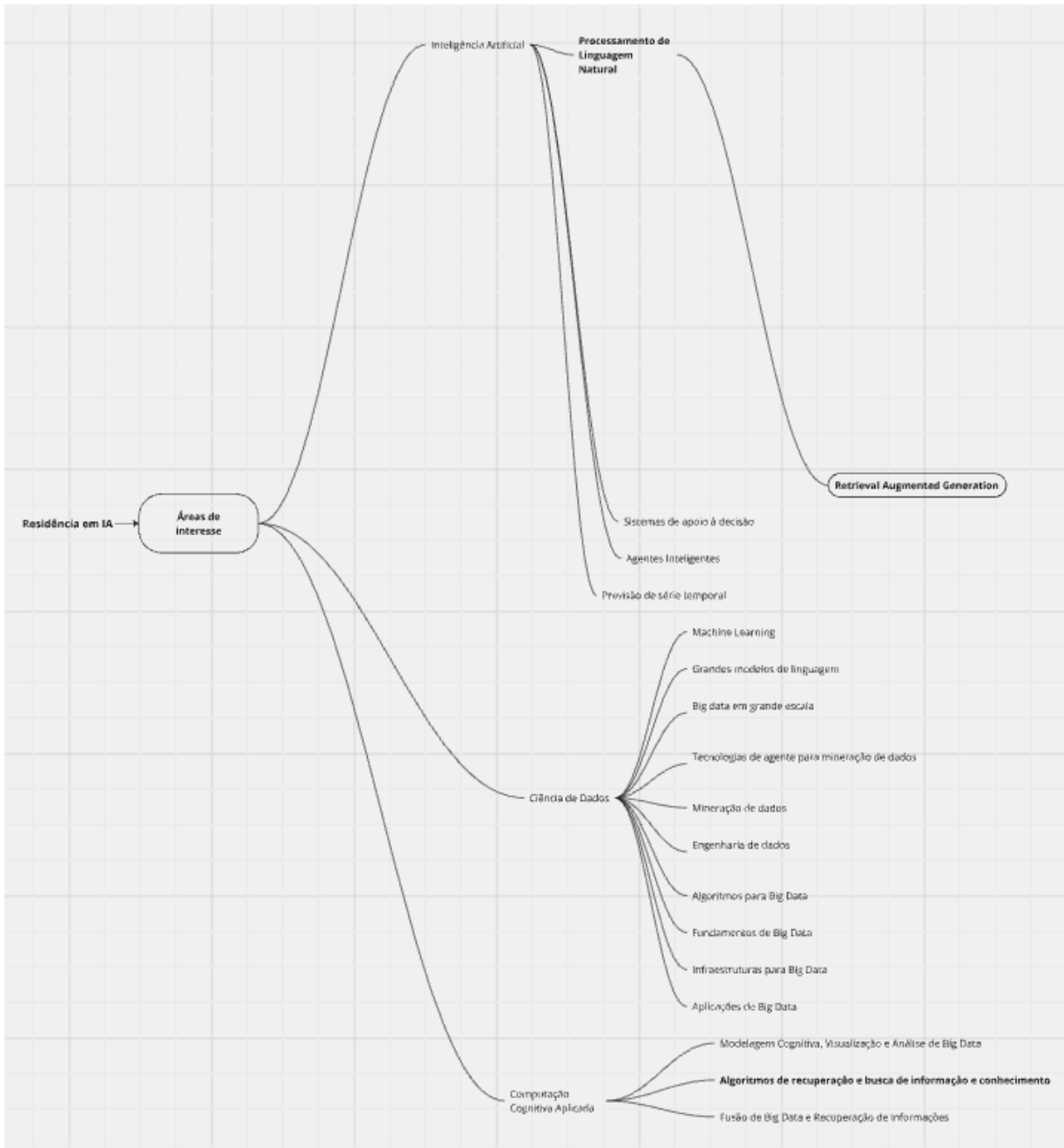
Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: [Go!](#)

[mapa mental citado no Termo de Aceite de Entrega de 18 de setembro]

Escolha Do tema



No mapa mental, eu quis elencar todas as áreas que eram do meu interesse para que eu pudesse ter uma visão mais clara de todas as possibilidades que eu tinha para me especializar durante a Residência. Com isso, defini que o tema principal no qual eu gostaria de me especializar é Retrieval-Augmented Generation (RAG).

[Survey citado no Termo de Aceite de Entrega de 18 de setembro]

Retrieval-Augmented Generation for Large Language Models: A Survey

Yunfan Gao^a, Yun Xiong^b, Xinyu Gao^b, Kangxiang Jia^b, Jinliu Pan^b, Yuxi Bi^c, Yi Dai^a, Jiawei Sun^a, Meng Wang^c, and Haofen Wang^{a,c}

^aShanghai Research Institute for Intelligent Autonomous Systems, Tongji University

^bShanghai Key Laboratory of Data Science, School of Computer Science, Fudan University

^cCollege of Design and Innovation, Tongji University

Anotações feitas do Survey:

1. O RAG surge de problemas do LLM, como: domínio específico, alucinações e consultas além de seus dados de treinamento.
2. O RAG é responsável por recuperar pedaços de documentos relevantes de uma base de conhecimento externa por meio do cálculo de similaridade semântica.
3. O RAG é categorizado em três estágios: Naive RAG, Advanced RAG e Modular RAG.
 - a. O Naive RAG segue um processo tradicional que inclui indexação, recuperação e geração, que também é caracterizado como uma estrutura "Retrieve-Read"
 - b. O advanced RAG refina suas técnicas de indexação por meio de uso de uma abordagem de janela deslizante, segmentação de granulação fina e incorporação de metadados.
 - c. O Rag modular incorpora diversas estratégias para melhorar seus componentes, como adicionar um módulo de busca para buscas de similaridade e refinar o recuperador por meio de FT.
4. RAG é frequentemente comparado com Fine Tuning e prompt engineering.

- a. Prompt engineering alavanca as capacidades inerentes de um modelo com necessidade mínima de conhecimento externo e adaptação do modelo.
 - b. RAG pode ser comparado a fornecer um modelo com um livro-texto personalizado para recuperação de informações.
 - c. FT é comparável a um aluno internalizando conhecimento ao longo do tempo.
5. A escolha entre RAG e Fine Tuning depende das necessidades específicas de dinâmica de dados, personalização e capacidades computacionais. RAG e Fine Tuning não são mutuamente exclusivos e podem se complementar, aprimorando as capacidades de um modelo em diferentes níveis.
6. Recuperação, Aumento e Geração são pontos cruciais.
7. A Recuperação possui ou pode possuir 5 pontos principais:
- a. Fonte de Recuperação
 - Estrutura de Dados
 - Granularidade de Recuperação
 - b. Otimização de Indexação
 - Estratégia de Chunking
 - Anexos de Metadados
 - c. Otimização de Consulta
 - Expansão da Consulta
 - Transformação de Consulta
 - Roteamento de Consultas
 - d. Incorporação
 - Recuperação mista/híbrida
 - Fine tuning do modelo de embeddings
 - e. Adaptador
8. A Geração deve ser ajustada conforme o conteúdo coletado e conforme o LLM:
- a. Curadoria de contexto
 - Reclassificação
 - Seleção/Compressão de Contexto
 - b. LLM - fine tuning
9. O processo de Aumento em RAG:
- a. Recuperação Iterativa

- b. Recuperação Recursiva
 - c. Recuperação Adaptativa
10. Há uma notável escassez de pesquisas dedicadas à avaliação das características distintas dos modelos RAG.
 11. Principais objetivos da avaliação:
 - a. Qualidade da Recuperação
 - b. Qualidade da Geração
 12. Aspectos de avaliação:
 - a. Pontuações de qualidade
 - b. Habilidades necessárias
 13. Usar LLMs para julgar.
 14. A presença de ruído ou informações contraditórias durante a recuperação pode afetar negativamente a qualidade da saída do RAG. “A desinformação pode ser pior do que nenhuma informação”. Mas alguns resultados de pesquisa revelam que incluir documentos irrelevantes pode aumentar inesperadamente a precisão em mais de 30%.
 15. Abordagens híbridas: Combinar fine tuning com RAG
 16. Ferramentas importantes: LangChain e LlamaIndex
 17. RAG multimodal

Perguntas a serem feitas:

- “O que recuperar”
- “Quando recuperar”
- “Como usar as informações recuperadas”

Revisão/Entendimento de conceitos citados:

1. aprendizado contrastivo:

No caso de LLMs e frameworks como o SANTA, o aprendizado contrastivo é aplicado para refinar embeddings de consultas e documentos. Isso significa que o modelo aprende a gerar representações vetoriais de consultas e documentos de forma que documentos relevantes para uma consulta específica fiquem próximos no espaço vetorial, enquanto documentos irrelevantes fiquem distantes. O contraste entre exemplos positivos e

negativos, nesse caso, guia o treinamento do modelo para melhorar sua capacidade de recuperação de informações.

2. Query Routing:

Query Routing é uma técnica usada em sistemas de Recuperação de Informações Aumentada por Geração (RAG, do inglês Retrieval-Augmented Generation) para direcionar consultas para diferentes pipelines ou mecanismos com base nas características da consulta.

3. RLHF:

RLHF é mencionado como uma inspiração para o processo de ajuste do retriever, onde o feedback gerado por LLMs pode ser usado para refinar a capacidade do sistema de recuperação de informações. O objetivo é melhorar a performance do modelo ao aliar o aprendizado com as preferências humanas ou o desempenho de LLMs pré-treinados, similar ao uso de feedback humano no RLHF.

Termo de Aceite de Entrega 2

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: 26 de set. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Durante a segunda semana do Stage 2, meu objetivo principal foi aprofundar o entendimento dos principais conceitos de Retrieval-Augmented Generation (RAG), que haviam sido abordados de maneira superficial no survey analisado na semana anterior. Para isso, direcionei meus estudos às três fases fundamentais do pipeline RAG: **indexação**, **recuperação** e **geração**. Ao final de cada fase, direcionei meus estudos para o framework do *langchain*, para entender o que é suportado e como é aplicado cada técnica das fases estudadas do pipeline.

Na fase de indexação, aprofundei nos seguintes aspectos:

- **Chunking:** Analisei diferentes técnicas de segmentação de texto, como chunking semântico e divisão por tamanho de caracteres, com o objetivo de otimizar o processamento de documentos longos.
- **Embeddings:** Explorei as diferenças entre vetores esparsos, densos e híbridos, focando em como essas representações capturam a semântica dos documentos.
- **Banco de dados vetorial:** Comparei várias soluções de bancos de dados vetoriais (como Pinecone, Chroma), focando em suas capacidades.

Na fase de recuperação, estudei métodos como:

- **Naive Retriever:** Um método simples que retorna os documentos mais semelhantes com base na similaridade de embeddings.
- **Reranking:** Uma técnica que reordena os documentos recuperados, priorizando os mais relevantes após a recuperação inicial

Na fase de geração, explorei os **tipos de modelos**, as **técnicas de geração** e os **desafios encontrados** ao gerar textos a partir de documentos recuperados.

Todo o processo dessa semana foi descrito no seguinte documento: [Anotacoes-260924-EvellynNicole](#)

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

Para a próxima semana, meu objetivo é encontrar artigos que explorem técnicas avançadas em

Retrieval-Augmented Generation (RAG). Meu foco será entender como e por que essas técnicas avançadas melhoram o pipeline básico do RAG. A intenção é entender o impacto dessas abordagens na performance do sistema e como elas otimizam os resultados em comparação com metodologias mais simples.

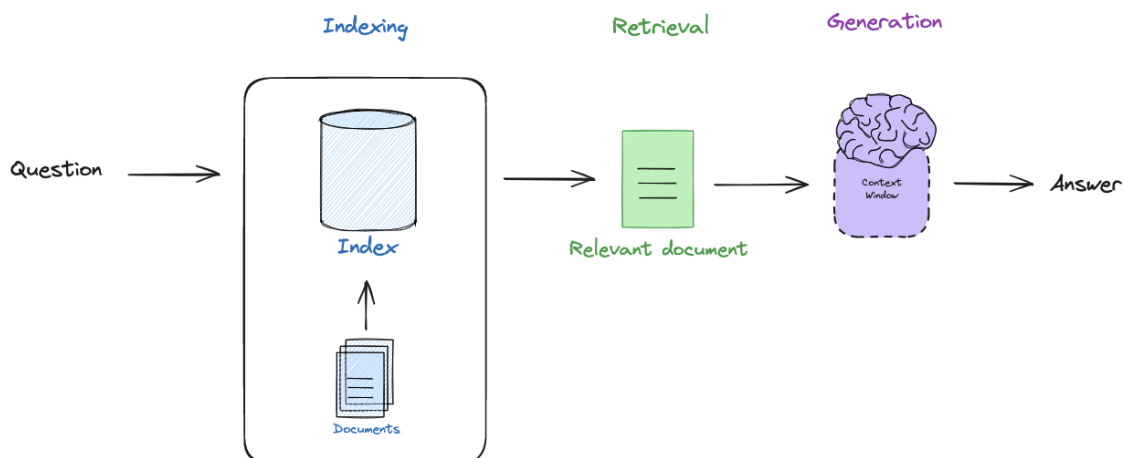
Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

[Fluxo básico do RAG citado nas anotações do termo de Aceite de Entrega de 26 de setembro]

Fluxo Básico



O fluxo representa um sistema RAG (Retrieval-Augmented Generation), onde uma pergunta do usuário é processada. Primeiramente, documentos são indexados para permitir buscas eficientes. Em seguida, o sistema recupera documentos relevantes com base na consulta

feita ao índice. Esses documentos são usados como contexto por um modelo de linguagem (LLM), que gera uma resposta informada e entrega ao usuário.

Anotações feitas na Semana 2:

Indexador:

Na fase de Indexação, os documentos serão processados, segmentados e transformados em Embeddings para serem armazenados em um banco de dados vetorial.

1. Chunking:

Os modelos de linguagem possuem janelas de contexto. Este é o comprimento do texto que eles podem processar em uma única passagem. Embora os comprimentos de contexto estejam cada vez maiores, foi demonstrado que os modelos de linguagem aumentam o desempenho nas tarefas quando recebem menos (mas mais relevantes) informações.

Uma maneira comum de fazer isso é agrupar ou subdividir seus grandes dados em partes menores. Para fazer isso, você precisa escolher uma estratégia de chunk. Idealmente, você deseja manter juntas as partes do texto semanticamente relacionadas [\[3\]](#).

a. Técnicas de Chunking:

- **Tamanhos fixos de caracteres:** Simples e direto, dividindo o texto em pedaços de um número fixo de caracteres.
- **Divisão de texto de caracteres recursivos:** Usando separadores como espaços ou pontuação para criar partes mais contextualmente significativas.
- **Divisão Específica do Documento:** Adaptando o método de chunking ao tipo de documento, como PDFs ou arquivos Markdown.
- **Divisão Semântica:** Usando embeddings para agrupar texto com base no conteúdo semântico.
- **Divisão Agente:** Empregar grandes modelos de linguagem para determinar a fragmentação ideal com base no conteúdo e no contexto.

Obs: Chunks podem ser enriquecidos com informações de metadados, como número de página, nome do arquivo, autor, carimbo de data/hora da categoria. [\[1\]](#)

A ferramenta Chunkviz pode ser usada para visualização de chunking [\[4\]](#).

No **framework do Langchain** existem essas técnicas listadas [\[5\]](#):

Nome	Classe	Descrição
Recursivo	RecursiveCharacterTextSplitter RecursiveJsonSplitter	Divide o texto recursivamente. Essa divisão tenta manter trechos de texto relacionados próximos uns dos outros. Este é o recommended way para começar a dividir o texto.
HTML	HTMLHeaderTextSplitter HTMLSectionSplitter	Divide o texto com base em caracteres específicos do HTML. Notavelmente, isso adiciona informações relevantes sobre a origem desse pedaço (com base no HTML)
Markdown	MarkdownHeaderTextSplitter	Divide o texto com base em caracteres específicos do Markdown. Notavelmente, isso adiciona informações relevantes sobre a origem desse pedaço (com base no Markdown)
Code	many languages	Divide o texto com base em caracteres específicos das linguagens de codificação. 15 idiomas diferentes estão disponíveis para você escolher.
Token	many classes	Divide o texto em tokens. Existem algumas maneiras diferentes de medir tokens.
Character	CharacterTextSplitter	Divide o texto com base em um caractere definido pelo usuário. Um dos métodos mais simples.
[Experimental] Semantic Chunker	SemanticChunker	Primeiras divisões em sentenças. Em seguida, combina uns ao lado do outro se forem semanticamente semelhantes o suficiente. Retirado de Greg Kamradt
AI21 Semantic Text Splitter	AI21SemanticTextSplitter	Identifica tópicos distintos que formam pedaços coerentes de texto e os divide ao longo desses.

2. Embeddings:

A representação do texto como um vetor é a parte mais crucial de qualquer problema de NLP. No Rag, convertemos pedaços de documentos nesses vetores para

recuperar documentos semelhantes conforme consulta do usuário [7]. Também temos duas opções de representação vetorial; vetores esparsos e vetores densos [6].

a. Vetores Esparsos:

Historicamente, os primeiros modelos de representação textual, como o TF-IDF (Term Frequency-Inverse Document Frequency) e o Bag of Words (BoW), utilizavam vetores esparsos. Esses vetores têm muitos zeros porque atribuem pesos somente a palavras específicas do documento, enquanto outras palavras do vocabulário inteiro ficam sem valor.

Características:

- Dimensionalidade muito alta: O vetor é do tamanho do vocabulário, podendo ter milhares ou milhões de posições.
- Muitos valores zero: Poucas palavras recebem valores diferentes de zero, resultando em muitos espaços vazios.
- Histórico: Esse método era amplamente utilizado nas primeiras aplicações de recuperação de informações e modelos de NLP.

Limitações:

- Incapacidade de capturar relações semânticas entre palavras, já que a proximidade vetorial entre duas palavras não reflete significados similares.
- Escalabilidade: Vetores esparsos podem se tornar ineficientes para processamento quando o vocabulário é muito extenso.

b. Vetores Densos:

Com o avanço dos word embeddings como Word2Vec, GloVe, e mais tarde transformers como BERT, surgiram os vetores densos. Ao contrário dos vetores esparsos, os densos contêm valores diferentes de zero em praticamente todas as posições e capturam melhor as relações semânticas entre palavras ou frases.

Características:

- Dimensionalidade mais baixa: Embora ainda altos, os vetores densos geralmente têm dimensões entre 100 a 1.000, muito menores que os vetores esparsos.
- Representação semântica rica: Vetores densos capturam similaridade semântica, permitindo que palavras ou frases com significados semelhantes fiquem próximas no espaço vetorial.
- Aplicações modernas: Modelos como BERT, GPT e embeddings gerados por redes neurais profundas tornaram-se populares por representar palavras de forma contextualizada e eficiente.

Vantagens:

- Melhor captura de significado: Dois textos semanticamente semelhantes terão vetores próximos, independentemente das palavras exatas utilizadas.
- Eficiência: Processar vetores densos é mais eficiente e escalável para muitas aplicações modernas de NLP, incluindo RAG.

c. Vetores Híbridos:

Atualmente, há uma crescente utilização de vetores híbridos, que combinam as forças de vetores esparsos e densos para melhorar a recuperação de informações. Isso se aplica especialmente em cenários onde ambos os aspectos — termos exatos e contexto semântico — são importantes.

Características:

- Combinação de modelos: Utiliza-se tanto vetores esparsos (por exemplo, TF-IDF ou BM25) para garantir a recuperação precisa de termos, quanto vetores densos (BERT, embeddings) para capturar similaridade semântica.
- Melhor performance: Sistemas híbridos conseguem melhores resultados na recuperação de documentos, equilibrando precisão (termos exatos) e recuperação de conteúdo semântico.

Embeddings langchain:

A incorporação de modelos cria uma representação vetorial de um trecho de texto. Você pode pensar em um vetor como uma matriz de números que captura o significado semântico do texto. Ao representar o texto dessa maneira, você pode realizar operações matemáticas que permitem fazer coisas como pesquisar outras partes do texto com significado mais semelhante. Esses recursos de pesquisa em linguagem natural sustentam muitos tipos de recuperação de contexto, onde fornecemos a um LLM os dados relevantes de que ele precisa para responder com eficácia a uma consulta [\[8\]](#).

O Embeddings class é uma classe projetada para fazer interface com modelos de incorporação de texto. Existem muitos provedores de modelos de incorporação diferentes (OpenAI, Cohere, Hugging Face, etc) e modelos locais, e esta classe foi projetada para fornecer uma interface padrão para todos eles.

A classe base Embeddings em LangChain fornece dois métodos: um para incorporar documentos e outro para incorporar uma consulta. O primeiro recebe vários textos como entrada, enquanto o último recebe um único texto. A razão para tê-los como dois métodos separados é que alguns provedores de incorporação têm métodos de incorporação diferentes para documentos (a serem pesquisados) versus consultas (a própria consulta de pesquisa).

Hybrid Langchain:

certifique-se de que o vectorstore que você está usando oferece suporte à pesquisa híbrida.

3. Banco de Dados Vetorial:

Após passar pelos processos de chunking e embedding, o próximo passo no pipeline de indexação em RAG (Retrieval-Augmented Generation) é o armazenamento dos embeddings em um banco de dados vetorial. Esse banco de dados é especializado em armazenar e recuperar representações vetoriais de dados, permitindo buscas rápidas e eficientes por similaridade. [\[9\]](#)

Comparação entre os Bancos de dados vetoriais [\[12\]](#):

	Pinecone	Qdrant	Chormadb	Milvus	PgVec tor	OpenSearch
Open Source	✗	✓	✓	✓	✓	✓
Self-host	✗	✓	✓	✓		✓
Gerenciamento de nuvem	✓	✓	✓	✓		✓
Serviço gratuito	2 GB de armazenamento	cluster de 1GB	-	0,5 milhões de vetores de 768 dimensões	-	-
Quanto custa?	aqui	aqui	-	aqui	-	-
Desenvolvido especificamente para vetores	✓	✓	✓	✓	✓	✗
Consultas por segundo (nytimes-256-angular dataset)	150 para p2	326	-	2406	141	-
Latência	1 para 4		-	1	8	-

(ms)	buscas em lote					
Pesquisa híbrida	✓	✓	✓	✓	✓	✓
Comprimento do vetor	20000	65536	ilimitado	ilimitado	2000	10000
Integrações	✓	✓	✓	✓	✓	✓
Github	-	19,5k	14,1k	28,9k	11,2k	9,2k

Banco de dados vetoriais suportados pelo langchain [\[10\]](#):

ActiveLoop Deep Lake, Aerospike, Alibaba Cloud OpenSearch, AnalyticDB, Annoy, Apache Doris, ApertureDB, Astra DB Vector Store, Atlas, AwaDB, Azure Cosmos DB Mongo vCore, Azure Cosmos DB No SQL, Azure AI Search, Bagel, BagelDB, Baidu Cloud ElasticSearch VectorSearch, Baidu VectorDB, Apache Cassandra, Chroma, Clarifai, ClickHouse, Couchbase, DashVector, Databricks, DingoDB, DocArray HnswSearch, DocArray InMemorySearch, Amazon Document DB, DuckDB, China Mobile ECloud ElasticSearch VectorSearch, Elasticsearch, Epsilla, Faiss, Faiss (Async), Google AlloyDB for PostgreSQL, Google BigQuery Vector Search, Google Cloud SQL for MySQL, Google Cloud SQL for PostgreSQL, Firestore, Google Memorystore for Redis, Google Spanner, Google Vertex AI Feature Store, Google Vertex AI Vector Search, Hippo, Hologres, Infinispan, Jaguar Vector Database, KDB.AI, Kinetica, LanceDB, Lantern, LLMRails, ManticoreSearch VectorStore, Marqo, Meilisearch, Amazon MemoryDB, Milvus, Momento Vector Index (MVI), MongoDB Atlas, MyScale, Neo4j Vector Index, NucliaDB, OpenSearch, Oracle AI Vector Search: Vector Store, Pathway, Postgres Embedding, PGVecto.rs, PGVector, Pinecone, Qdrant, Redis, Relyt, Rockset, SAP HANA Cloud Vector Engine, ScaNN, SemaDB, SingleStoreDB, scikit-learn, SQLite-VSS, StarRocks, Supabase (Postgres), SurrealDB, Tair, Tencent Cloud VectorDB, ThirdAI NeuralDB, TiDB Vector, Tigris, TileDB, Timescale Vector (Postgres), Typesense, Upstash Vector, USearch, Vald, Intel's Visual Data Management System (VDMS), Vearch, Vectara, Vespa, viking DB, vlite, Weaviate, Xata, Yellowbrick, Zep, Zep Cloud, Zilliz

Recuperador:

O retriever/recuperador em RAG desempenha um papel crucial ao buscar informações relevantes no banco de dados vetorial para auxiliar o modelo de geração de linguagem. Quando o modelo recebe uma pergunta ou instrução, o retriever localiza e recupera documentos ou dados pertinentes que possam conter a resposta ou informações úteis [\[11\]](#).

Métodos de Recuperação:

a. Naive Retriever:

Este método é o mais simples, aliás o seu nome o indica. Usamos este adjetivo para identificar este método pela simples razão de que ao inserir a consulta em nosso banco de dados, esperamos (ingenuamente) que ele retorne os documentos/pedaços mais relevantes.

Basicamente o que acontece é que codificamos a consulta do usuário com o mesmo embedding com o qual criamos o armazenamento de vetores. Uma vez obtida sua representação vetorial, calculamos a similaridade calculando o cosseno, a distância, etc.. E coletamos os K principais documentos mais próximos/semelhantes à consulta.

b. Parent Document Retriever:

Imagine que criamos um RAG para reconhecer possíveis doenças introduzindo alguns dos seus sintomas na consulta. Caso tenhamos um RAG Ingênuo, podemos coletar uma série de possíveis doenças que coincidem apenas em um ou dois sintomas, deixando nossa ferramenta em uma situação um pouco ruim. Este é um caso ideal para usar o Parent Doc Retriever. E o tipo de técnica consiste em cortar pedaços grandes (pedaço pai) em pedaços ainda menores (pedaço filho). Por possuírem pequenos pedaços, a informação que contém fica mais concentrada e, portanto, o seu valor informativo não se dilui entre os parágrafos do texto.

c. Self Query Retriever:

O Self Query Retriever é uma abordagem avançada para melhorar a eficiência e a relevância na recuperação de informações em sistemas que utilizam armazenamento de vetores. Sua principal característica é a capacidade de realizar pesquisas aplicando filtros baseados em metadados antes de calcular a similaridade entre a consulta e os documentos armazenados.

d. Contextual Compression Retriever (Reranking):

Digamos que esta técnica consiste em duas etapas muito diferentes:

etapa 1: Obtenha uma boa quantidade de documentos relevantes com base na entrada/pergunta. Normalmente definimos o K mais relevante.

etapa 2: Recalcule quais desses documentos são realmente relevantes. descartando os outros documentos que não são realmente úteis (Compressão).

Após a execução das duas etapas, temos uma lista refinada de documentos que não apenas estão relacionados à pergunta original, mas também foram submetidos a uma verificação mais detalhada para garantir sua relevância. O processo de reranking permite que o sistema selecione os documentos que são mais contextualmente adequados, eliminando informações redundantes ou irrelevantes.

Isso resulta em uma compressão dos dados, onde apenas os documentos mais úteis são mantidos para a próxima fase do pipeline

Recuperações Realizadas pelo langchain [\[13\]](#):

Vectorstore, ParentDocument, Multi Vector, Self Query, Contextual Compression, Time-Weighted Vectorstore, Multi-Query Retriever, Ensemble, Long-Context Reorder

Generation:

No pipeline de RAG (Retrieval-Augmented Generation), a fase de geração é a etapa final, onde o modelo de linguagem toma as informações recuperadas do banco de dados vetorial e as utiliza para produzir uma resposta ou completar uma tarefa baseada na consulta do usuário. O sucesso dessa etapa depende de como o modelo é capaz de integrar as informações relevantes recuperadas e aplicá-las de forma coerente na resposta.

a. Tipos de Modelos Utilizados na Geração:

- Modelos Pré-treinados: Modelos como o GPT são amplamente utilizados na geração de linguagem. Eles foram pré-treinados em grandes quantidades de dados e podem gerar respostas com base nas informações recuperadas. No entanto, sem uma boa etapa de recuperação, os modelos podem não ter o contexto necessário para fornecer uma resposta precisa.
- Fine Tuning: Em alguns cenários, é interessante fazer o fine-tuning dos modelos de linguagem para o domínio específico, como questões jurídicas, saúde, ou outros contextos especializados. O modelo passa a ser treinado com dados específicos daquele domínio, tornando suas respostas mais adequadas.

b. Técnicas de Geração:

- Geração baseada em Contexto Recuperado: O modelo usa os documentos recuperados para gerar respostas mais precisas e contextuais. O conteúdo extraído dos documentos serve como referência para a geração, garantindo que o modelo responda com informações mais relevantes e baseadas em fatos.
- Geração Abstrativa: Na geração abstrativa, o modelo não apenas extrai trechos dos documentos, mas também os reescreve, resumindo ou adaptando o conteúdo para formar uma resposta coerente. Esse método é especialmente útil em sumarizações ou respostas que necessitam ser mais sintéticas.
- Geração Extrativa: Aqui, o modelo se limita a selecionar partes dos documentos que foram recuperados, sem criar uma nova narrativa. Essa abordagem é útil para garantir que as respostas sejam exatamente o que está presente no texto, minimizando distorções.

c. Desafios na Geração:

- Coerência e Consistência: Um dos principais desafios da geração é garantir que as respostas sejam coerentes, especialmente quando o modelo precisa combinar informações de múltiplos documentos.
- Relevância: Mesmo com um bom sistema de recuperação, é crucial que o modelo selecione e priorize as informações mais relevantes para a resposta, eliminando detalhes irrelevantes ou repetidos.

Referências Semana 2:

[3] The Art of Chunking: Boosting AI Performance in RAG Architectures. Disponível em: <https://towardsdatascience.com/the-art-of-chunking-boosting-ai-performance-in-rag-architectures-acbdb8bdc2b>

[4] Chunkviz - Ferramenta de Chunking. Disponível em: <https://chunkviz.up.railway.app/>

[5] Text Splitters. Disponível em: https://python.langchain.com/v0.1/docs/modules/data_connection/document_transformers/

[6] Dense Vectors: Capturing Meaning with Code. Disponível em: <https://www.pinecone.io/learn/series/nlp/dense-vector-embeddings-nlp/>

[7] Embeddings: What they are and why they matter. Disponível em: <https://simonwillison.net/2023/Oct/23/embeddings/>

[8] Embedding models. Disponível em: <https://python.langchain.com/docs/concepts/#embedding-models>

[9] Which Vector Database Should You Use? Choosing the Best One for Your Needs. Disponível em: <https://medium.com/the-ai-forum/which-vector-database-should-you-use-choosing-the-best-one-for-your-needs-5108ec7ba133>

[10] Vectorstores. Disponível em: <https://python.langchain.com/docs/integrations/vectorstores/>

[11] Advanced Retriever Techniques to Improve Your RAGs. Disponível em: <https://towardsdatascience.com/advanced-retriever-techniques-to-improve-your-rags-1fac2b86dd61>

[12] Vector DB Comparison. Disponível em: <https://superlinked.com/vector-db-comparison>

[13] Retrievers. Disponível em: https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/

APÊNDICE 2

Termo de Aceite de Entrega 3

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: 2 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

No Stage 3, meu objetivo principal foi estudar as principais técnicas avançadas de RAG. Para isso, usei as melhorias implementadas pelo Langchain, já que elas são consolidadas, testadas e inspiradas em artigos já bem estabelecidos. As etapas de melhoria são:

1. **Query Translation:** Melhora a consulta do usuário com:
 - **Multi-Query:** Decomposição da query.
 - **Step Back:** Abstração da query.
 - **RAG Fusion:** Decomposição + Reranker.
2. **HyDE:** Geração de documentos hipotéticos antes de buscar na base, para capturar padrões de representação.
3. **Routing:** Direcionamento da consulta através do **Roteamento Lógico** ou **Semântico**.
4. **Query Construction:** Geração de instruções para interação com diferentes bancos de dados (SQL, grafos, vetores).
5. **Indexação:**
 - **Multi-representation indexing:** Armazenamento do documento
 - **Specialized Embeddings:** Fine Tuning/ColBERT
 - **Indexação Hierárquica:** Raptor
6. **Contextual Retrieval:** Adição de contexto ao chunk para melhorar a recuperação dentro do documento.

Todo o processo dessa semana foi descrito no seguinte documento: [Anotacoes-021024-EvellynNicole](#)

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

Para a próxima Semana pretendo estudar GraphRag que pretende solucionar o problema de queries mais generalistas e em recuperar dados relacionados através de grafos de conhecimento. Pretendo também pesquisar por métodos de avaliação dentro de RAG.

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

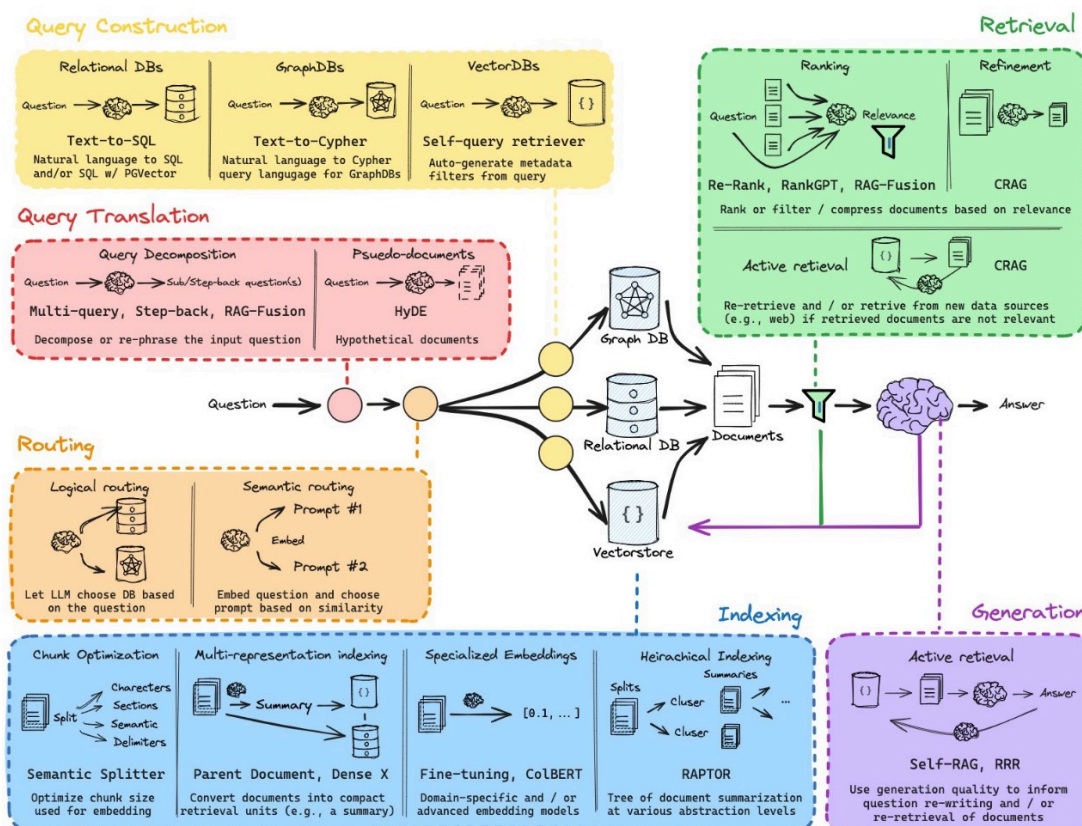
Acompanhei uma dissertação de doutorado sobre grafo conhecimento.

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

[Imagem que mostra tanto elementos básicos e avançados do RAG citado nas anotações do termo de Aceite de Entrega de 02 de outubro]

Técnicas Avançadas

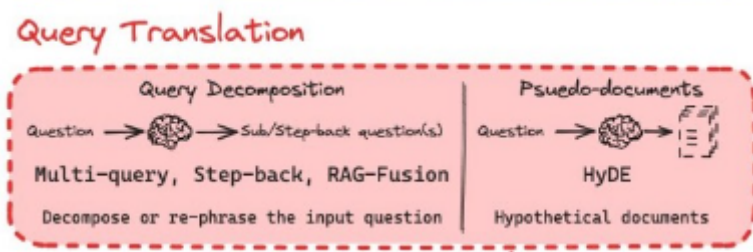


Este diagrama detalha o fluxo de um sistema avançado de RAG, e foi a partir dele que me guiei para estudar algumas das técnicas mais avançadas durante o fluxo.

Anotações feitas na Semana 3:

Query Translation

[Imagem da etapa de Query Translation do termo de Aceite de Entrega de 02 de outubro]



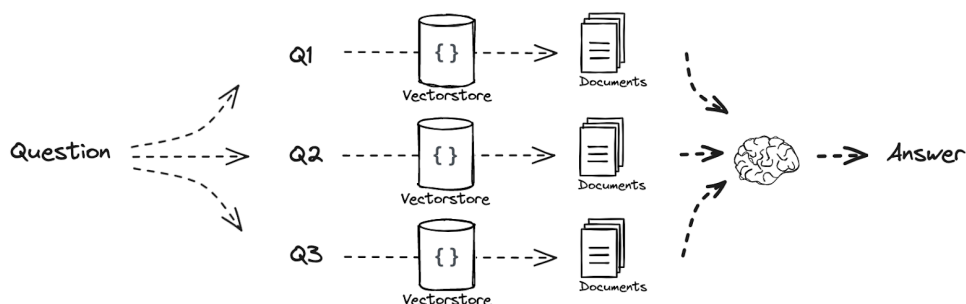
A etapa de **Query Translation** tem como objetivo reformular ou traduzir a consulta do usuário para torná-la mais clara e compatível com o sistema de RAG (Retrieval-Augmented Generation). Muitas vezes, as consultas do usuário não são suficientemente específicas ou estruturadas de maneira que o modelo possa processar com precisão. Assim, essa etapa visa melhorar a query original, seja decompondo-a em partes menores ou reescrevendo-a de forma otimizada, para facilitar o entendimento do modelo.

Esse processo garante que o sistema possa realizar uma recuperação de informações mais eficiente e precisa, alinhando os dados retornados com a intenção real do usuário e, consequentemente, gerando respostas mais relevantes.

A **Query Translation** é dividida em duas abordagens principais:

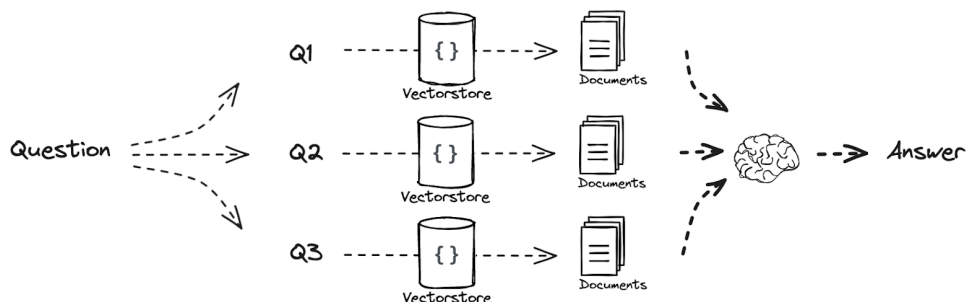
1. **Decomposição da Consulta:** Aqui, a consulta original é dividida em subconsultas menores e mais gerenciáveis. Essa estratégia facilita a busca de respostas específicas, tornando o processo de recuperação de informações mais direto e preciso [3].
 - **Multi-Query:** O conceito de multi-query refere-se a uma técnica de decomposição de consultas complexas em várias consultas menores e mais específicas. Em vez de tentar responder a uma consulta complexa de uma só vez, o sistema divide a consulta original em várias subconsultas ou reformulações. Cada uma dessas subconsultas pode ser respondida de forma independente, permitindo que o sistema encontre informações mais relevantes e completas [4].

[Imagem da técnica de Multi-Query do termo de Aceite de Entrega de 02 de outubro]



- **Step Back:** (Apresentada pelo google). A técnica de Step-Back Prompting é uma abordagem de prompting (geração de respostas) que permite que modelos de linguagem, como LLMs (Large Language Models), façam abstrações para responder de forma mais eficaz a perguntas complexas ou detalhadas. Ao invés de tentar responder diretamente a uma pergunta específica e possivelmente complicada, o modelo "dá um passo atrás" e tenta responder a uma pergunta mais abstrata ou princípios fundamentais que estão por trás da pergunta original. Esse processo ajuda o modelo a organizar seu raciocínio de forma mais clara e lógica [5].
- **Rag Fusion:** Semelhante ao Multy Query Translation, a diferença é que aplicamos a classificação desses documentos recuperados. Reclassificação (Re-ranking) é o ponto chave do RAG Fusion. Em vez de simplesmente combinar as respostas dos documentos recuperados como são, o sistema classifica os documentos ou partes de documentos com base na sua relevância para a consulta original. Isso garante que os resultados mais relevantes ou úteis sejam priorizados na resposta final. Após a reclassificação, o sistema combina os documentos em uma fusão de resultados mais relevantes em uma única resposta coesa. Esse processo de fusão prioriza as partes mais úteis e claras dos documentos recuperados, resultando em uma resposta de melhor qualidade [6].

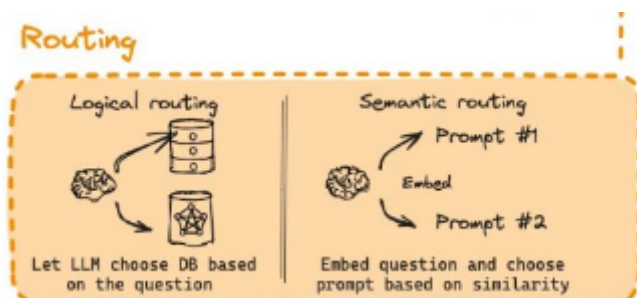
[Imagem da técnica de Rag Fusion do termo de Aceite de Entrega de 02 de outubro]



- Documentos Hipotéticos:** Nessa abordagem, são criados documentos baseados em respostas hipotéticas da pergunta, ajudando a prever informações que o usuário pode estar buscando, o que enriquece a etapa de recuperação e geração de conteúdo.
 - Hyde:** Utiliza a geração de documentos hipotéticos para resolver o problema de recuperação de informações em cenários de zero-shot learning. Quando uma consulta é feita, em vez de tentar imediatamente recuperar documentos relevantes no corpus, um modelo de linguagem, como o InstructGPT, é instruído a criar um documento fictício que responda ou se alinhe com a consulta. Esse documento gerado não é real e pode conter informações imprecisas ou falsas, mas é útil porque captura os padrões de relevância que o sistema está procurando. Por exemplo, se a consulta for "Como funcionam os painéis solares?", o modelo pode gerar um texto fictício que descreve como a luz solar é convertida em eletricidade por células fotovoltaicas, mesmo que os detalhes não estejam completamente corretos. Após a criação do documento hipotético, ele é codificado em um vetor de embedding utilizando um encoder não supervisionado, como o Contriever. Esse vetor funciona como um identificador de "bairro" no espaço de embedding do corpus, mapeando o documento fictício para uma área onde documentos reais, com relevância similar, possam ser encontrados. O passo seguinte é usar esse vetor para buscar documentos reais no corpus que sejam semelhantes ao documento hipotético. O encoder realiza essa busca com base na similaridade entre vetores, encontrando documentos reais que estão mais próximos do documento fictício no espaço de embedding [\[7\]](#).

Routing

[Imagem da etapa de Routing do termo de Aceite de Entrega de 02 de outubro]

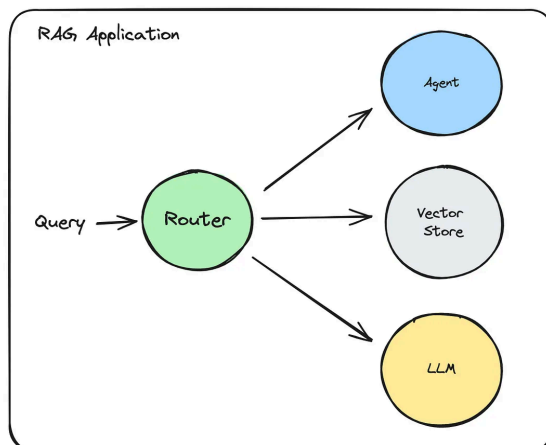


Roteamento, no contexto de RAG (Retrieval-Augmented Generation), é o processo de direcionar uma consulta para a fonte ou tratamento adequado, com base em regras predefinidas ou em técnicas de processamento da linguagem natural. Essencialmente, trata-se de encaminhar a pergunta ou consulta do usuário para a base de dados correta, que pode ser uma fonte diferente dependendo do conteúdo da pergunta. Em termos simples, é uma forma de condicionar a resposta, seja por meio de verificações lógicas ou por compreensão semântica [8].

1. Roteamento Lógico

O roteamento lógico utiliza verificações condicionais clássicas, como instruções if/else, para decidir o caminho que uma consulta deve seguir. Ele se baseia em variáveis ou condições discretas, como o comprimento de strings, nomes de arquivos ou valores numéricos específicos. Diferente do roteamento semântico, o roteamento lógico não precisa entender a intenção da consulta em linguagem natural; em vez disso, toma decisões baseadas em variáveis bem definidas que já estão presentes no código ou no fluxo de trabalho. É útil quando as regras de roteamento são claras e podem ser determinadas por dados ou condições lógicas fixas.

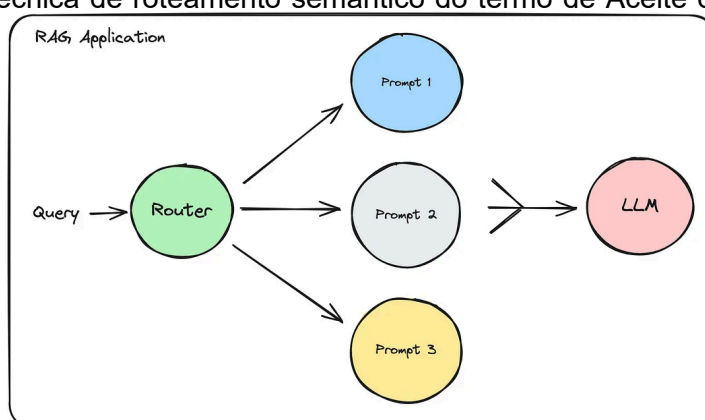
[Imagem da técnica de roteamento lógico do termo de Aceite de Entrega de 02 de outubro]



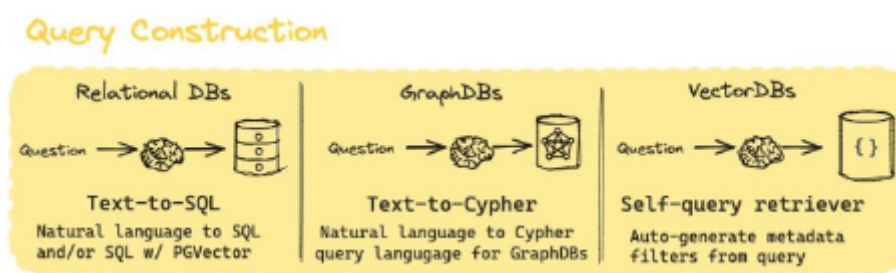
2. Roteamento Semântico

O roteamento semântico utiliza técnicas mais avançadas de processamento de linguagem natural, aproveitando embeddings (incorporações vetoriais) e buscas de similaridade. Nesse tipo de roteamento, a consulta do usuário é convertida em um vetor e comparada com exemplos de consultas já associadas a diferentes rotas. Essas consultas de exemplo são embutidas como vetores, e a rota com maior similaridade é selecionada para tratar a consulta. Esse tipo de roteamento é ideal para quando a intenção da consulta precisa ser compreendida de forma mais contextual e não pode ser determinada por simples verificações lógicas. Ele é frequentemente utilizado em situações em que há múltiplas categorias de perguntas e respostas e quando é necessário capturar nuances semânticas da linguagem.

[Imagem da técnica de roteamento semântico do termo de Aceite de Entrega de 02 de outubro]



[Imagem da etapa de Query Construction do termo de Aceite de Entrega de 02 de outubro]



Query Construction

A construção de queries refere-se ao processo de gerar instruções que interagem com diferentes tipos de bancos de dados, a partir de consultas em linguagem natural. A ideia é converter perguntas humanas em uma linguagem de consulta específica para o banco de dados subjacente. Isso permite que usuários sem conhecimento técnico profundo possam interagir diretamente com bancos de dados relacionais, de grafos e de vetores, utilizando perguntas comuns. Ferramentas e técnicas modernas permitem a tradução automática de consultas em linguagem natural para as linguagens de consulta apropriadas de cada tipo de banco de dados, como SQL, Cypher ou filtros de vetores [9].

1. Relational DBs

Nos bancos de dados relacionais (Relational DBs), a construção de queries envolve converter consultas em linguagem natural para SQL. SQL (Structured Query Language) é a linguagem padrão usada para gerenciar e consultar dados em sistemas de banco de dados relacionais, como PostgreSQL, MySQL ou SQL Server.

Text-to-SQL: Um mecanismo que interpreta perguntas em linguagem natural e gera consultas SQL correspondentes. Por exemplo, uma pergunta como "Quais são os cinco melhores clientes em receita?" pode ser traduzida para uma query SQL que retorna os resultados desejados.

PGVector: Quando utilizado com PGVector, uma extensão para PostgreSQL, queries SQL podem incorporar vetores, permitindo que bancos de dados relacionais também processem consultas vetoriais (como embeddings), além de dados tabulares. Isso amplia o uso de bancos de dados relacionais para suportar operações baseadas em aprendizado de máquina, como busca semântica ou recuperação de informações a partir de vetores.

2. GraphDBs

Nos bancos de dados de grafos (GraphDBs), a linguagem natural é convertida para Cypher, que é a linguagem de consulta mais utilizada para esses bancos de dados, como o Neo4j. Os bancos de dados de grafos são utilizados para representar e

consultar dados que têm fortes inter-relações, como redes sociais, rotas de transporte, ou sistemas de recomendação.

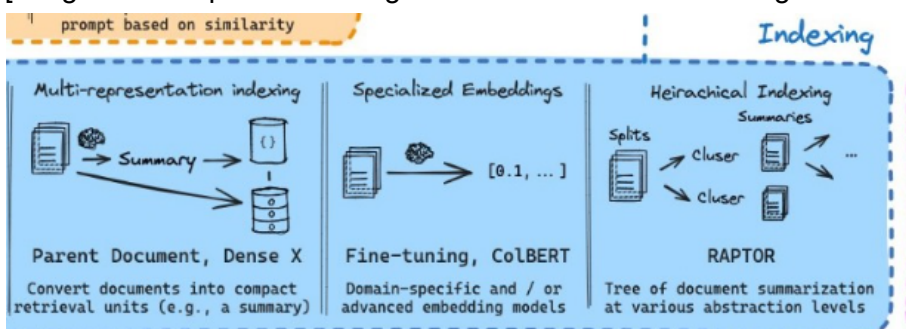
Text-to-Cypher: Esse processo interpreta uma pergunta em linguagem natural e a converte em uma query Cypher. Por exemplo, uma consulta como "Quais pessoas estão conectadas ao João através de amigos em comum?" poderia ser convertida em uma query Cypher que explora as conexões do grafo para encontrar essas pessoas.

3. Vector DBs

Nos bancos de dados de vetores (VectorDBs), o objetivo é realizar consultas baseadas em vetores, que são geralmente usados para representar embeddings de dados, como palavras, frases, ou imagens, gerados por modelos de machine learning.

Self-query retriever: Este método permite que filtros de metadados sejam gerados automaticamente a partir de uma consulta em linguagem natural. Em vez de usar uma linguagem de consulta estruturada como SQL ou Cypher, a ferramenta interpreta a consulta e gera filtros que podem ser aplicados diretamente no banco de dados vetorial. Isso é muito útil para tarefas de busca semântica, onde as consultas precisam ser mapeadas para vetores que capturam o significado semântico das palavras. O banco de dados então retorna os itens mais relevantes, baseando-se em similaridades entre os vetores.

[Imagem da etapa de Indexing do termo de Aceite de Entrega de 02 de outubro]



Indexing

Apesar de ter falando de indexação na semana passada, existem algumas técnicas mais avançadas durante esse processo, como:

1. Multi-representation indexing

Quebrar um corpus de recuperação em pedaços menores que sejam mínimos, auto-contidos, contextualizados e contenham fatos distintos, melhora a qualidade das tarefas de recuperação e QA. Com base no artigo Dense X Retrieval ou Proposition-based Retrieval, a indexação de múltiplas representações envolve a criação e o armazenamento de representações de cada documento dentro do sistema de recuperação. Representações aqui podem significar análise tradicional de

palavras-chave, compreensão semântica profunda ou resumo, e até mesmo elementos visuais, como imagens ou diagramas dos documentos.

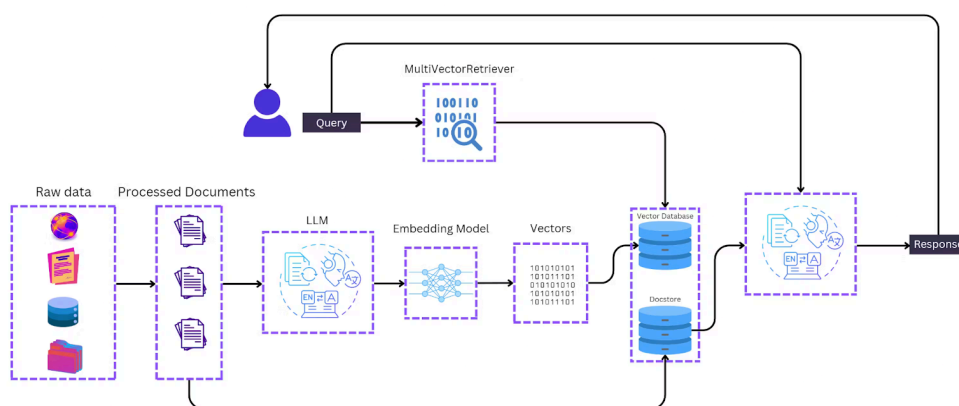
Armazenamos duas versões dos nossos documentos — Uma para indexação. Mas o outro é o documento real que recuperamos.

Depois de dividir nossos documentos, criamos uma versão otimizada para recuperação. Isso geralmente seria um resumo do pedaço original, que deve conter todas as informações vitais.

Armazenamos a versão otimizada em uma loja de vetores e o pedaço original em um arquivo de documentos. Nós ligamos os dois com a mesma chave para rastrear qual versão otimizada é para qual documento.

Agora, recuperamos como de costume. Pedimos ao repositório vetorial para buscar os documentos mais próximos de nossa entrada de texto no espaço vetorial. Mas à medida que recuperamos, usamos a chave que definimos antes para buscar o documento original em vez da versão otimizada [10].

[Imagem da técnica de Multi-representation indexing do termo de Aceite de Entrega de 02 de outubro]



2. Specialized Embeddings

- **Fine tuning:** Fine-tuning de embeddings refere-se ao processo de ajustar ou treinar novamente os embeddings (representações vetoriais de palavras, frases ou documentos) em um modelo pré-treinado, usando um conjunto de dados específico para uma tarefa ou domínio. Esse ajuste fino permite que os embeddings se tornem mais adequados para o contexto específico em que serão usados, melhorando o desempenho em tarefas específicas.

- **Colbert:** É uma abordagem que visa melhorar a precisão em sistemas de recuperação de informações, especialmente quando a consulta (query) e o documento (bloco ou "passage") são muito diferentes em tamanho e complexidade. Ele resolve alguns problemas que surgem em técnicas tradicionais de busca que não lidam bem com a diferença de tamanho entre consultas e documentos.

Na recuperação regular, o documento inteiro e a consulta são representados como vetores únicos (ou embeddings). A similaridade entre eles é medida para encontrar os documentos que melhor correspondem à consulta. Esse método é eficiente em termos de velocidade, mas tem um problema: ele trata o documento e a consulta como uma única unidade, o que é injusto, pois os documentos são muito maiores e mais complexos que as consultas.

Em outras palavras:

Consulta: é curta, pode conter apenas algumas palavras.

Documento: geralmente, é muito maior.

Essa diferença de tamanho faz com que comparar uma consulta curta a um documento longo nem sempre seja preciso, porque há muita informação irrelevante no documento que pode interferir no cálculo da similaridade.

O ColBERT adota uma abordagem tokenizada. Isso significa que, em vez de comparar um único vetor representando todo o documento com um vetor da consulta, o ColBERT compara os tokens individuais (ou seja, as palavras ou sub-palavras) da consulta com os tokens individuais do documento. A partir disso ele pega a melhor correspondência entre os tokens e atribui uma pontuação máxima (MaxSim) [\[11\]](#).

3. Hierarchical Indexing

- **Raptor:** O RAPTOR é um sistema de indexação e recuperação de informações que organiza textos de forma hierárquica, usando uma estrutura em árvore. Seu funcionamento se baseia na ideia de agrupar pedaços de texto em "clusters", que são então resumidos. A partir desses resumos, o RAPTOR repete o processo, criando uma hierarquia de resumos que vai do nível mais detalhado ao mais geral. Dessa forma, o sistema constrói uma árvore de informações, onde cada nível da árvore contém resumos que representam diferentes níveis de detalhe do texto original.

Essa estrutura hierárquica permite que grandes modelos de linguagem (como GPT-4) processem diferentes níveis de informação, desde resumos mais concisos até detalhes completos do texto, dependendo da necessidade. Isso melhora a eficiência e precisão ao responder perguntas sobre o conteúdo, pois o sistema consegue selecionar e carregar apenas as partes relevantes do texto, seja um detalhe específico ou uma visão geral.

RAPTOR foi projetado para otimizar o processo de recuperação de contexto em modelos de linguagem, especialmente em documentos longos. Ele foi testado em várias tarefas de perguntas e respostas e mostrou resultados superiores em relação a outras abordagens de recuperação, estabelecendo novos patamares em conjuntos de dados que envolvem textos narrativos, artigos científicos e perguntas de múltipla escolha baseadas em passagens textuais de comprimento maior [\[12\]](#).

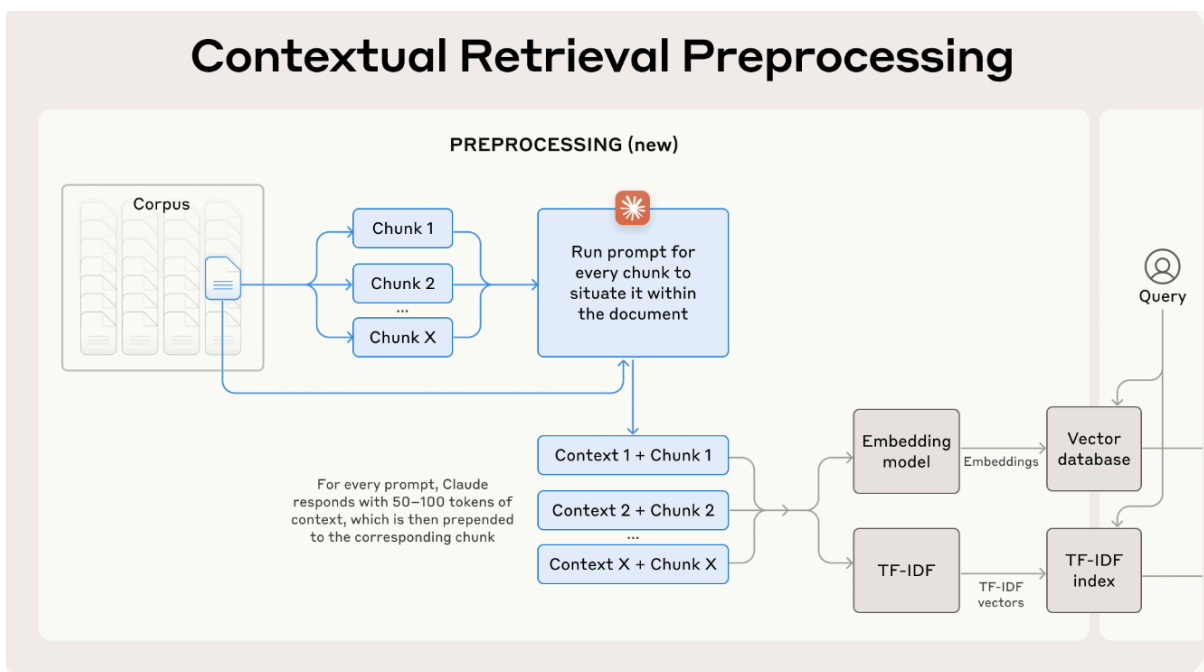
Introdução ao Contextual Retrieval

Lançado pela empresa Antropic no dia 19 de setembro de 2024, o Contextual Retrieval tem a ideia de dar um contexto sucinto e curto para situar o chunk dentro do documento geral com o propósito de melhorar a recuperação de pesquisa do chunk. Isso garante que os fragmentos recuperados tenham informações relevantes, evitando a ambiguidade causada pela falta de detalhes específicos [\[13\]](#).

Exemplo:

- Fragmento original: "A receita da empresa cresceu 3% em relação ao trimestre anterior."
- Fragmento contextualizado: "Esse trecho é de um relatório financeiro da empresa ACME no Q2 de 2023; a receita no trimestre anterior foi de \$314 milhões. A receita da empresa cresceu 3% em relação ao trimestre anterior."

[Imagem da técnica de Contextual Retrieval Preprocessing do termo de Aceite de Entrega de 02 de outubro]



Referências da Semana 3:

- [3] Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions. Disponível em: <https://arxiv.org/pdf/2212.10509>
- [4] How to use the MultiQueryRetriever. Disponível em: How to use the MultiQueryRetriever
- [5] TAKE A STEP BACK: EVOKING REASONING VIA ABSTRACTION IN LARGE LANGUAGE MODELS. Disponível em: <https://arxiv.org/pdf/2310.06117>
- [6] AG-Fusion: The Next Frontier of Search Technology. Disponível em: <https://github.com/Raudaschl/rag-fusion>
- [7] Precise Zero-Shot Dense Retrieval without Relevance Labels. Disponível em: <https://arxiv.org/abs/2212.10496>
- [8] Routing in RAG-Driven Applications. Disponível em: <https://towardsdatascience.com/routing-in-rag-driven-applications-a685460a7220>
- [9] Query Construction. Disponível em: <https://blog.langchain.dev/query-construction/>
- [10] Dense X Retrieval: What Retrieval Granularity Should We Use? Disponível em: <https://arxiv.org/pdf/2312.06648>
- [11] ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. Disponível em: <https://arxiv.org/pdf/2004.12832>

[12] RAPTOR: RECURSIVE ABSTRACTIVE PROCESSING FOR TREE-ORGANIZED RETRIEVAL. Disponível em: <https://arxiv.org/pdf/2401.18059v1>

[13] Introducing Contextual Retrieval. Disponível em: <https://www.anthropic.com/news/contextual-retrieval>

Termo de Aceite de Entrega 4

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: 10 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

No Stage 4, meu objetivo principal foi estudar GraphRag, uma maneira de utilizar grafos de conhecimento para representar as entidades e relações existentes em algum corpus ou documento. Para isso li o [paper](#) da Microsoft que publicou um estudo detalhado sobre GraphRag e fiz anotações sobre ele.

- O GraphRag surge como uma tentativa de resolver os “problemas” do Rag.
- Utiliza um processo de construir entidades e relações entre essas entidades.
- Geração de comunidades que agregam grupos de nós que estão densamente conectados.
- Pesquisa global e local.
- É caro.

Todo o processo dessa semana foi descrito no seguinte documento: [Anotacoes-101024-EvellynNicole](#)

Criei um desenho para me ajudar a entender de uma forma visual o processo de Rag e GraphRag: [link desenho](#).

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

Para a próxima Semana pretendo pesquisar mais eficientemente por métodos de avaliação dentro de RAG e começar a utilizar o Framework do Langchain para estruturar/testar algum código que utilize as etapas de Rag.

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

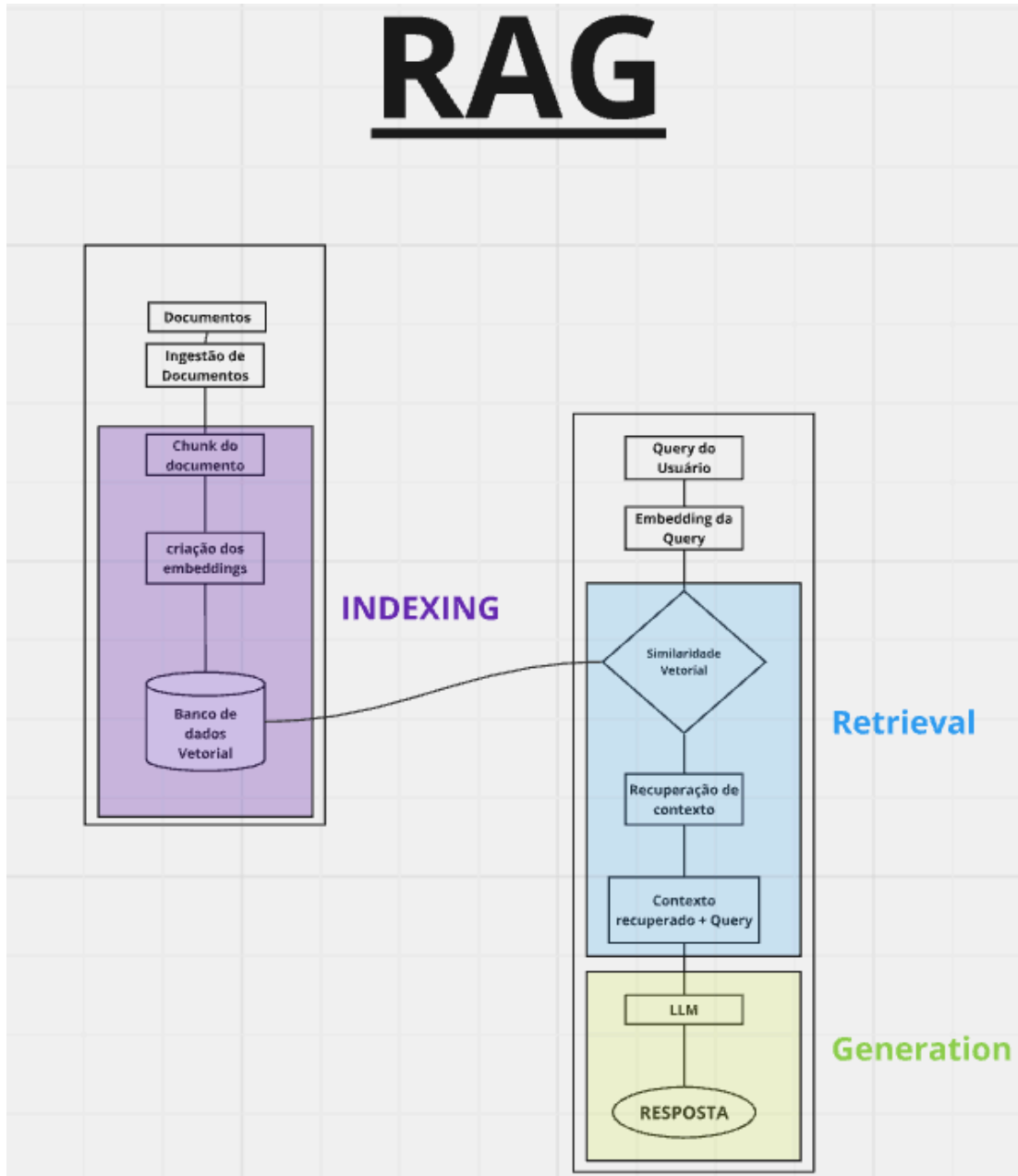
Disse que nessa Semana iria ver sobre métricas de avaliação, mas li apenas um survey sobre o assunto, já que GraphRag me tomou bastante tempo.

ACEITE DA ENTREGA:

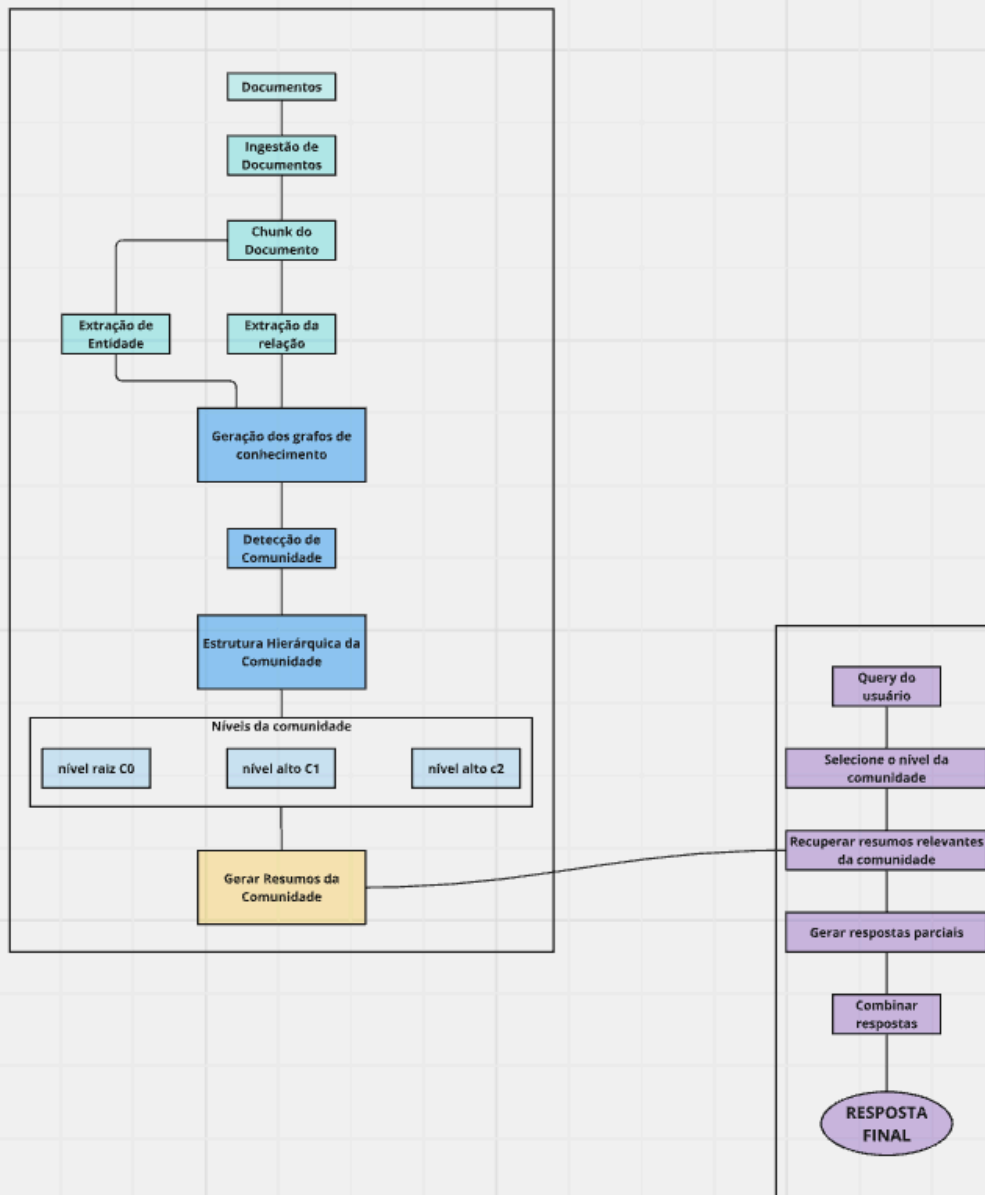
CEDRIC LUIZ DE CARVALHO: [Go!](#)

[Fluxos desenhados por mim para entender componentes, características e comparar o RAG e GraphRag citado nas anotações do termo de Aceite de Entrega de 10 de outubro]

Comparativo RAG vs GraphRAG



GraphRag



O GraphRag surge numa tentativa de resolver “problemas” do RAG.

- RAG falha em perguntas globais direcionadas a um corpus de texto inteiro.

- Ele falha em dados que têm relação. Como o retriever tenta achar somente os assuntos relacionados a query, ele não pega os assuntos que têm relacionados com aquilo que ele encontrou. Exemplo de textos que tem citações para outros textos, onde o RAG não busca as citações.
- O RAG pode também trazer vários chunks redundantes. Isso gera uma sobrecarga de texto que causa problemas.
- QFS (query focused summarization): O processo de geração de resumos especificamente adaptados para atender à consulta de um usuário.
- Algoritmo de Leiden (etapa 5):

Objetivo que o paper - Graph RAG Approach to Query-Focused Summarization - do GraphRag se propõe é conseguir responder a perguntas gerais sobre o texto inteiro utilizando um LLM para construir um índice de texto baseado em grafos em dois estágios: Primeiro para derivar um grafo de conhecimento de entidade dos documentos de origem, depois para pré-gerar resumos da comunidade para todos os grupos de entidades intimamente relacionadas. No artigo, é apresentada uma abordagem Graph RAG baseada na sumarização global de um grafo de conhecimento derivado de LLM.

O RAG (Retrieval-Augmented Generation) e o GraphRAG diferem principalmente na forma como estruturam os dados e realizam a recuperação e geração de informações. O **RAG** utiliza bancos de dados vetoriais para indexar documentos em chunks e criar embeddings representativos. A recuperação de contexto é feita com base em similaridade vetorial entre a consulta do usuário e os embeddings dos documentos. O contexto recuperado é combinado com a consulta e passado para um modelo de linguagem (LLM) que gera a resposta. Já o **GraphRAG** estrutura os documentos como grafos de conhecimento, extraindo entidades e suas relações. Esses grafos são organizados hierarquicamente em comunidades (níveis de granularidade, como C0, C1, e C2). A recuperação utiliza a estrutura de grafos para identificar comunidades relevantes e gerar resumos associados a esses níveis, que são utilizados para formar uma resposta final. Em resumo, o RAG é mais direto e baseado em embeddings vetoriais para busca e geração, enquanto o GraphRAG adiciona uma camada de conhecimento semântico e hierárquico, ideal para contextos onde a estrutura das relações entre os dados é importante.

APÊNDICE 3

Termo de Aceite de Entrega 5

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: 17 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

No Stage 5, meu objetivo foi estudar métricas de avaliação para sistemas de RAG. Para isso, li o artigo *Evaluation of Retrieval-Augmented Generation: A Survey*, que explora as dificuldades em definir métricas eficazes devido à complexidade dos componentes interativos do RAG.

O artigo propõe o **Auepora**, um Processo Unificado de Avaliação de RAG, que visa padronizar a avaliação desses sistemas a partir de três perguntas:

- “O que avaliar?” (alvo)
- “Como avaliar?” (dataset de referência)
- “Como medir?” (métricas)

As métricas são divididas em duas categorias principais: **Recuperação**, que mede a pertinência dos documentos recuperados em relação à consulta do usuário, e **Geração**, que avalia a qualidade das respostas geradas com base nessa consulta.

Anotações: [Anotacoes-171024-EvellynNicole](#)

Além disso, desenvolvi um código simples para colocar na prática alguns conceitos aprendidos até aqui.

[171024-Stage4-residencia.ipynb](#)

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

Pretendo, para a próxima semana, tentar melhorar o código simples implementado durante essa Semana e documentar os resultados que obtive usando diferentes técnicas. Além disso, quero pesquisar sobre qual tema irei atacar nas semanas finais usando RAG.

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

[Figura que mostra o método de avaliação desenvolvido no artigo Evaluation of Retrieval-Augmented Generation: A Survey. Descrito no termo de aceite do dia 17 de outubro]

Métodos de Avaliação

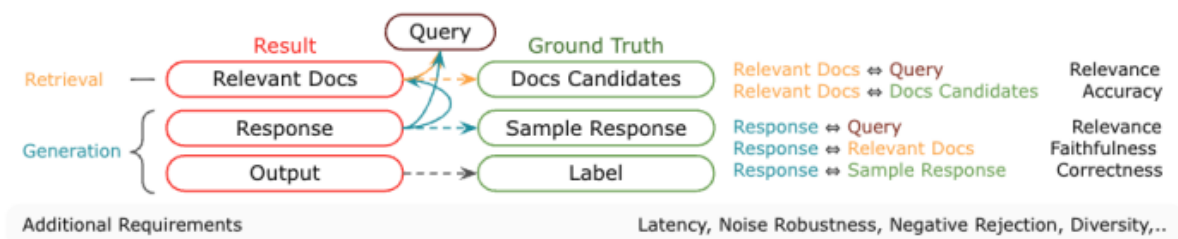


Fig. 2: The *Target* modular of the *Auepora*.

Introdução

Como qualquer sistema de engenharia, avaliar o desempenho é crucial para o desenvolvimento de aplicações RAG. O pipeline RAG é dividido em três componentes: 1. Indexação, 2. Recuperação e 3. Geração. A avaliação RAG é complicada devido à série de componentes interativos e ao esforço de coletar dados de teste. O RAG possui um banco de dados dinâmico externo, é composto por um sistema inteiro e é capaz de disponibilizar perguntas de domínio aberto. Todos esses fatores impedem uma métrica de avaliação única e completa.

No nível dos componentes, a avaliação do RAG geralmente se concentra na avaliação da qualidade do recuperador e do gerador, uma vez que ambos desempenham papéis críticos na produção de respostas precisas e relevantes.

O survey compilou 12 estruturas de avaliação distintas, trazendo benchmarks, analisando por meio de metas, conjuntos de dados e métricas.

Objetivo do Survey

O objetivo principal é oferecer uma visão abrangente sobre as métricas e benchmarks utilizados para avaliar componentes de recuperação e geração dentro dos sistemas RAG. O artigo também propõe um Processo Unificado de Avaliação de RAG (*Auepora*), que visa padronizar a avaliação desses sistemas.

Proposta do Survey

Auepora: A Unified Evaluation Process of RAG, que visa abordar de forma sistemática os problemas na avaliação de sistemas RAG. O Auepora é dividido em três módulos principais:

1. O que avaliar? (What to Evaluate?) Definir os alvos de avaliação, identificando os componentes de saída (outputs) e os "Ground Truths" (verdades de referência) contra os quais os sistemas serão comparados.
2. Como avaliar? (How to Evaluate?) Comparar os diferentes conjuntos de dados usados para avaliação e identificar as melhores práticas na construção de benchmarks.
3. Como medir? (How to Measure?) Apresentar as métricas apropriadas para avaliar os diferentes componentes, tais como a relevância e a fidelidade no componente de geração e a precisão e recall no componente de recuperação.

Auepora, aborda essas questões com uma estrutura modular:

1. Módulo alvo (Target): Foca em identificar quais são os resultados de interesse na avaliação do sistema. Isso pode incluir documentos relevantes para a recuperação e respostas adequadas para a geração.
2. Módulo dataset: Explora como os diferentes conjuntos de dados podem ser utilizados para avaliar adequadamente sistemas RAG, desde dados de perguntas-respostas até dados específicos de domínio.
3. Módulo métricas: Propõe métricas específicas para quantificar os resultados esperados em cada módulo. Inclui tanto métricas tradicionais como BLEU e ROUGE, quanto métricas adaptadas, como a avaliação por LLMs e métricas de robustez a informações errôneas.

O que avaliar?

RECUPERAÇÃO

- **RELEVÂNCIA**: Documentos relevantes -> Consulta - Avalia quão bem os documentos recuperados correspondem às informações necessárias expressas na consulta

- **ACURÁCIA**: Documentos relevantes -> Documentos candidatos - Avalia o quão preciso os documentos recuperados são comparados a um conjunto de documentos candidatos.

GERAÇÃO

- **RELEVÂNCIA**: Resposta -> Consulta - Mede o quão bem a resposta gerada se alinha com a intenção e o conteúdo da consulta inicial.

- **FIDELIDADE (Faithfulness):** Resposta -> Documentos Relevantes - Avalia se a resposta gerada reflete com precisão as informações contidas nos documentos relevantes e mede a consistência entre o conteúdo gerado e os documentos de origem.

- **CORREÇÃO (Correctness):** Resposta -> Resposta da Amostra - Mede a precisão da resposta gerada em relação a uma resposta de amostra, que serve como uma verdade fundamental.

Como avaliar?

Benchmarks distintos empregam estratégias variadas para construção de conjuntos de dados, que vão desde alavancar recursos existentes até gerar dados inteiramente novos adaptados para aspectos específicos de avaliação. Conjuntos de dados adaptados para métricas ou tarefas específicas melhoram a precisão da avaliação e orientam o desenvolvimento de sistemas RAG adaptáveis para necessidades de informações do mundo real.

Como quantificar?

Cada componente dentro de sistemas RAG requer uma abordagem avaliativa personalizada que reflita suas distintas funcionalidades e objetivos.

MÉTRICAS DE RECUPERAÇÃO: Para a avaliação de recuperação, o foco está em métricas que podem capturar com precisão a relevância, precisão, diversidade e robustez das informações recuperadas em resposta a consultas. Essas métricas não devem apenas refletir a precisão do sistema em buscar informações pertinentes, mas também sua resiliência em navegar no cenário dinâmico, vasto e às vezes enganoso de dados disponíveis. Essas métricas são focadas em avaliar a precisão, relevância e ordem dos documentos recuperados em resposta a uma consulta.

- **Misleading Rate**

Avalia a frequência com que o sistema recupera ou gera informações que são enganosas, ou seja, que parecem corretas, mas estão incorretas ou fora de contexto.

- **Mistake Reappearance Rate**

Avalia a frequência com que o sistema repete os mesmos erros em diferentes respostas ou consultas. Isso é útil para detectar padrões de erro persistentes..

- **Error Detection Rate**

Mede a capacidade do sistema de detectar e sinalizar erros internos, ou seja, identificar quando gerou ou recuperou informações incorretas.

- **MAP@K**

Média das pontuações de precisão média para cada consulta.

$$MAP = \frac{1}{Q} \sum_{q=1}^{|Q|} \frac{\sum_{k=1}^n P(K) \times rel(k)}{|documentos\ relevantes\ q|}$$

$rel(k) = 1$ se o documento na posição k for relevante, 0 caso contrário

Q = número de consultas

$P(K)$ = precisão no ponto de corte k

- **MRR@K**

Avalia a posição do primeiro documento relevante recuperado em uma lista ordenada de resultados. Dá mais peso a documentos relevantes que estão no topo.

$$MRR = \frac{1}{Q} \sum_{i=1}^{|Q|} \frac{1}{Rank\ i}$$

$|Q|$ = número de consultas

$Rank\ i$ = posição do primeiro documento relevante para a consulta i

- **Acurácia**

Proporção de resultados corretos (tanto verdadeiros positivos quanto verdadeiros negativos) entre o número total de casos examinados.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precisão**

Proporção de instâncias relevantes entre as instâncias recuperadas.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall at k**

Proporção de instâncias relevantes recuperadas sobre o total de casos relevantes, considerando apenas os primeiros k resultados.

$$Recall@k = \frac{|RD \cap TopK|}{|RD|}$$

$TopK$ = K primeiros recuperados

RD = documentos relevantes

- **LLM como juiz**

É um conceito emergente onde LLMs são usados para avaliar a qualidade das respostas recuperadas ou geradas por um sistema de RAG.

MÉTRICAS DE GERAÇÃO: No reino da geração, a avaliação transcende a mera precisão das respostas geradas, aventurando-se na qualidade do texto em termos de coerência, relevância, fluência e alinhamento com o julgamento humano.

- **Bleu**
BLEU calcula a precisão de n-gramas (grupos de palavras consecutivas) no texto gerado comparado com o texto de referência. Ele verifica a proporção de n-gramas no texto gerado que também aparecem no texto de referência. Além disso, ele aplica uma penalidade de brevidade para evitar que textos muito curtos obtenham uma pontuação alta.
- **Rouge**
ROUGE calcula a revocação de n-gramas entre o texto gerado e o texto de referência. Diferente do BLEU, o ROUGE se concentra na correspondência de palavras, frases ou subsequências do texto gerado em comparação com o texto de referência
- **BertScore**
Usa embeddings de modelos como BERT para avaliar a similaridade semântica entre o texto gerado e o de referência, considerando o contexto.
- **LLM como juiz**
Usa LLMs para avaliar automaticamente o texto gerado com base em critérios como coerência, relevância e fluência.

REQUISITOS ADICIONAIS: Latência, adversidade, ruído, rejeição negativa e robustez confractual.

Exemplo Básico - PPC BIA

Para começar a desenvolver o que foi aprendido nas semanas anteriores com o Fluxo básico, desenvolvi um código simples, no qual o intuito era indexar um documento - ppc do Bacharelado em Inteligência Artificial - e realizar perguntas para ver quais dados seriam recuperados e como funcionaria a parte da geração.

Para isso, utilizei um *chunk_size=2000* e *chunk_overlap=300* na fase de indexação. Além de utilizar *Qdrant* como banco de dados vetorial e *Langchain* como ferramenta principal. Na parte de recuperação, utilizei o modelo *text-embedding-3-large* e fiz com que fossem recuperados os 3 principais chunks. Na fase de geração foi utilizado o modelo *gpt-4o-mini* da *OpenAI* com o seguinte prompt: “*Você é um assistente útil que responde perguntas sobre o Projeto pedagógico do Bacharelado em Inteligência Artificial. Use o contexto abaixo para responder à pergunta. Não fuja do contexto; se a resposta não estiver no contexto, diga que você não sabe.*”

Contexto: *#dados recuperados*

{source_knowledge}

Pergunta: *{query} #pergunta do usuário*”

Com isso, obtive os seguintes resultados:

query: *Me fale sobre a disciplina de lógica matemática do BIA*

resposta: *A disciplina de Lógica Matemática do Bacharelado em Inteligência Artificial (BIA) é uma das disciplinas de formação básica obrigatórias. Ela não possui pré-requisitos e está classificada no Núcleo Comum (NC). A carga horária total da disciplina é de 64 horas, sendo 64 horas de teoria e nenhuma de prática. Em relação à bibliografia, a disciplina conta com referências básicas e complementares que incluem títulos como "Lógica para Ciência da Computação" de Souza, e "Introdução à Lógica" de Mortari, entre outros. Essas referências são voltadas para fornecer uma base sólida em lógica, que é fundamental para a compreensão de conceitos em computação e inteligência artificial.*

Termo de Aceite de Entrega 6

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: 30 de out. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

No Stage 6, meu objetivo foi melhorar com técnicas avançadas de RAG, o fluxo simples desenvolvido em código na semana passada, que era um RAG simples para analisar o PPC do BIA. Dessa forma, implementei 4 pontos de melhoria, que são:

- **Chunk Semântico**
- **Multi Query Retriever**
- **Pesquisa híbrida**
- **Reranker**

Também implementei uma estrutura de **Grafos** para entender como eram os nós e os relacionamentos dentro do documento.

Código:  301024-Stage6-residencia.ipynb

Além disso, fiquei de definir a minha área de atuação, que eu optei por usar RAG para construir um chatbot conversacional com dados do Campeonato Brasileiro de Futebol (Série A). Para isso, entrei em contato com o pessoal da plataforma [Transfermarkt](#), que me passaram um [documento](#) para preencher e assim me disponibilizarem os dados.

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

Estou aguardando os dados serem disponibilizados, mas preciso traçar uma estratégia caso isso demore. Minha ideia é procurar por outras plataformas ou coletar esses dados através de Scraping. Enquanto isso, também pretendo procurar por trabalhos que se assemelham ao meu para ter ideias na fase de implementação.

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

ACEITE DA ENTREGA:

LEONARDO ALVES: Go! ▾


[Figura que mostra o grafo criado a partir do ppc do Bacharelado em IA como produto no termo de aceite do dia 30 de outubro]

Melhorias no Exemplo Básico - PPC BIA

Diante do exemplo apresentado relacionado ao PPC do Bacharelado em Inteligência Artificial, me comprometi em melhorar com técnicas avançadas de RAG, o fluxo simples desenvolvido em código na Semana 5 que era um RAG simples. Com isso, apliquei algumas técnicas estudadas na Semana 3, que foram: Chunk Semântico, Multy Query Retriever, Busca Híbrida e Rerankar.

Notebook do Desenvolvimento Avançado.

[As imagens a seguir podem ser acessadas em formato de código no documento :

 301024-Stage6-residencia.ipynb]

Indexação

```
from PyPDF2 import PdfReader

def get_pdf_text(pdf_docs):
    text = ""
    for pdf in pdf_docs:
        pdf_reader = PdfReader(pdf)
        for page in pdf_reader.pages:
            text += page.extract_text()
    return text

text = get_pdf_text(['/content/ppc-bia-v1.2.pdf'])
```

- Chunk Semântico

```
from langchain_experimental.text_splitter import SemanticChunker
from langchain_openai.embeddings import OpenAIEmbeddings
from google.colab import userdata

openai_api_key = userdata.get('openai_api_key')

text_splitter = SemanticChunker(OpenAIEmbeddings(model="text-embedding-3-large", openai_api_key=openai_api_key))

chunks_semantico = text_splitter.create_documents([text])
```

- Chunk Splitter

```
from langchain.text_splitter import CharacterTextSplitter

def get_text_chunks(text):
    text_splitter = CharacterTextSplitter(
        separator="\n", chunk_size=2000, chunk_overlap=300,
        length_function=len
    )
    chunks = text_splitter.split_text(text)
    return chunks

chunks = get_text_chunks(text)
```

- Banco de dados Vetorial

```
from langchain_qdrant import QdrantVectorStore, RetrievalMode
from langchain_openai.embeddings import OpenAIEmbeddings
from langchain.docstore.document import Document
from langchain_qdrant import FastEmbedSparse
from google.colab import userdata

openai_api_key = userdata.get('openai_api_key')
url = userdata.get('url')
api_key = userdata.get('api_key')

embeddings = OpenAIEmbeddings(model="text-embedding-3-large",
openai_api_key=openai_api_key)

sparse_embeddings = FastEmbedSparse(model_name="prithvida/Splade_PP_en_v1")

docs_semantico = [Document(page_content=str(chunk), metadata={"source": f"Chunk {i+1}"})
for i, chunk in enumerate(chunks_semantico)]
docs = [Document(page_content=chunk, metadata={"source": f"Chunk {i+1}"}) for i, chunk in
enumerate(chunks)]

def vectordb(docs, embeddings, sparse_embeddings, url, collection_name: str):
    qdrant = QdrantVectorStore.from_documents(
        docs,
        embeddings,
        sparse_embedding=sparse_embeddings,
        url=url,
        prefer_grpc=True,
        api_key=api_key,
        collection_name=collection_name,
        retrieval_mode=RetrievalMode.HYBRID
    )

    return qdrant

qdrant = vectordb(docs, embeddings, sparse_embeddings, url, collection_name='ppc-chunk')
qdrant_semantico = vectordb(docs_semantico, embeddings, sparse_embeddings, url,
collection_name='ppc-chunk_semantico')
```

Recuperação

```
from langchain.retrievers.multi_query import MultiQueryRetriever
import logging
from langchain_openai import ChatOpenAI

logging.basicConfig()
logging.getLogger("langchain.retrievers.multi_query").setLevel(logging.INFO)

query = "Me fale como funciona o curso"

llm = ChatOpenAI(model='gpt-4o-mini', api_key=openai_api_key, temperature=0)

# chunk splitter
retriever = MultiQueryRetriever.from_llm(
    retriever=qdrant.as_retriever(search_kwargs = {'k':10}), llm=llm,
    include_original=True
)

unique_docs = retriever.get_relevant_documents(query=query)
len(unique_docs)

docs = retriever.invoke(query)
sources = [doc.metadata['source'] for doc in docs]

#chunk semantico
retriever_semantico = MultiQueryRetriever.from_llm(
    retriever=qdrant_semantico.as_retriever(search_kwargs = {'k':10}), llm=llm,
    include_original=True
)

unique_docs = retriever_semantico.get_relevant_documents(query=query)
len(unique_docs)

docs_semantico = retriever_semantico.invoke(query)
sources_semantico = [doc.metadata['source'] for doc in docs_semantico]
```

Exemplo de perguntas geradas com multy Query:

1. Quais são os principais tópicos abordados na disciplina de lógica matemática do BIA? ',
2. Como a lógica matemática é ensinada no curso de BIA? ',
3. Quais são os objetivos e a importância da disciplina de lógica matemática no BIA?

- Reranker

```
# reranker semantico e splitter

from langchain.retrievers import ContextualCompressionRetriever
from langchain.retrievers.document_compressors import CrossEncoderReranker
from langchain_community.cross_encoders import HuggingFaceCrossEncoder

def create_compression_retriever(base_retriever):
    model = HuggingFaceCrossEncoder(model_name="BAAI/bge-reranker-base")
    compressor = CrossEncoderReranker(model=model, top_n=3)
    return ContextualCompressionRetriever(base_compressor=compressor,
    base_retriever=base_retriever)

compression_retriever = create_compression_retriever(retriever)
compression_retriever_semantico = create_compression_retriever(retriever_semantico)
```

```
def retrieve_documents(query, retriever, retriever_semantico):
    docs = retriever.invoke(query)
    docs_semantico = retriever_semantico.invoke(query)
    return docs, docs_semantico

docs, docs_semantico = retrieve_documents(query, compression_retriever,
compression_retriever_semantico)
```

Geração

```
def custom_prompt(query, source_knowledge):  
    return f"""Use o contexto abaixo para responder à pergunta. Não fuja do contexto; se  
a resposta não estiver no contexto, diga que você não sabe.  
  
Contexto:  
{source_knowledge}  
  
Pergunta: {query}"""
```

```
from langchain.schema import (  
    SystemMessage,  
    HumanMessage,  
    AIMessage  
)  
from langchain_openai import ChatOpenAI  
  
def generate_response(query, docs, api_key):  
    source_knowledge = "\n".join([x.page_content for x in docs])  
  
    prompt = HumanMessage(  
        content=custom_prompt(query, source_knowledge)  
    )  
  
    messages = [  
        SystemMessage(content="Você é um assistente útil que responde perguntas sobre o  
Projeto pedagógico do Bacharelado em Inteligência Artificial."),  
        prompt  
    ]  
  
    chat = ChatOpenAI(model='gpt-4o-mini', api_key=api_key, temperature=0)  
    response = chat.invoke(messages)  
    return response.content  
  
content_splitter = generate_response(query, docs, openai_api_key)  
content_semantico = generate_response(query, docs_semantico, openai_api_key)  
  
print(f'Chunk splitter:\n{content_splitter}\n')
```

- Output com chunk splitter:

Query: Me fale como funciona o curso

Resposta: O curso de Bacharelado em Inteligência Artificial (BIA) é estruturado para proporcionar aos alunos um aprendizado abrangente sobre os diversos paradigmas e técnicas relacionadas à Inteligência Artificial e à Ciência de Dados ao longo dos sete primeiros semestres. Durante esse período, os alunos são incentivados a desenvolver habilidades como trabalho colaborativo, proatividade, empreendedorismo, ética e inovação, preparando-os para atuar como agentes transformadores na sociedade.

Após esses sete semestres, os alunos participam do Programa de Residência em Inteligência Artificial, onde têm a oportunidade de se dedicar integralmente à solução de um problema concreto. Essa experiência visa aprofundar suas capacidades cognitivas e raciocínio lógico, permitindo que dialoguem com os docentes de forma autônoma.

A avaliação dos alunos considera não apenas o cumprimento de requisitos técnicos, mas também a postura ética e profissional. Cada disciplina possui condições mínimas que os alunos devem atender para serem considerados aprovados, embora se espere que eles realizem atividades em níveis mais exigentes. As disciplinas específicas de Inteligência Artificial têm suas avaliações orientadas por condições mínimas que definem as habilidades que os alunos devem demonstrar.

Em resumo, o curso combina teoria e prática, com um forte foco em desenvolvimento de habilidades práticas e éticas, preparando os alunos para enfrentar desafios reais na área de Inteligência Artificial.

- Output com chunk semântico:

Query: Me fale como funciona o curso

Resposta: O curso de Bacharelado em Inteligência Artificial (BIA) é oferecido pelo Instituto de Informática (INF) da Universidade Federal de Goiás (UFG) e tem como objetivo formar profissionais capacitados para desenvolver sistemas inteligentes e autônomos, seguindo princípios éticos e uma postura profissional adequada.

Estrutura do Curso:

- **Modalidade:** Presencial

- ****Duração****: Mínimo de 8 semestres e máximo de 14 semestres, com uma carga horária total de 3.200 horas.
- ****Número de Vagas****: 40 vagas no primeiro semestre do ano letivo.
- ****Turno de Funcionamento****: Integral.

Currículo:

O curso é dividido em quatro blocos geradores:

1. ****Fundamentos em Matemática e Computação****
2. ****Mentalidade****
3. ****Técnicas****
4. ****Integração****

A matriz curricular inclui disciplinas obrigatórias e optativas, com foco em formação básica em Computação e Matemática nos primeiros períodos, e disciplinas específicas de Inteligência Artificial nos períodos seguintes. O curso também inclui uma disciplina de "Residência em Inteligência Artificial", que é um componente prático de 384 horas, onde os alunos trabalham em projetos integradores.

Metodologia:

- ****Atividades Práticas****: O curso enfatiza a prática desde o início, com a "Residência em Inteligência Artificial" sendo um ponto central onde os alunos aplicam o conhecimento adquirido em problemas reais.
- ****Interdisciplinaridade****: O curso promove a interação entre diferentes áreas do conhecimento, permitindo que os alunos escolham disciplinas do Núcleo Livre (NL) para complementar sua formação.
- ****Avaliação****: O desempenho dos alunos é avaliado continuamente, levando em consideração não apenas o conhecimento técnico, mas também a postura ética e profissional.

Objetivos:

- Formar profissionais que possam contribuir com a demanda da sociedade por sistemas inteligentes e autônomos.
- Promover a inovação e o empreendedorismo.
- Fomentar a participação dos alunos em atividades de pesquisa, desenvolvimento e extensão.

Perfil do Egresso:

O egresso do BIA deve ter uma formação sólida em Inteligência Artificial, com habilidades para identificar, compreender, analisar e modelar problemas, além de propor soluções inteligentes utilizando técnicas e ferramentas adequadas. Em resumo, o curso de Bacharelado em Inteligência Artificial da UFG é estruturado para oferecer uma formação abrangente e prática, preparando os alunos para atuar em um mercado em constante evolução e demanda por profissionais qualificados na área de IA.

GraphRag

```
import os

from langchain_community.graphs import Neo4jGraph

os.environ["NEO4J_URI"] = "neo4j+s://81a62548.databases.neo4j.io"
os.environ["NEO4J_USERNAME"] = "neo4j"
os.environ["NEO4J_PASSWORD"] = ""

graph = Neo4jGraph()

from google.colab import userdata
from langchain_experimental.graph_transformers import LLMGraphTransformer
from langchain_openai import ChatOpenAI

openai_api_key = userdata.get('openai_api_key')

llm = ChatOpenAI(model='gpt-4o-mini', api_key=openai_api_key, temperature=0)
llm_transformer = LLMGraphTransformer(llm=llm)

from langchain_core.documents import Document

documents = [Document(page_content=text)]
graph_documents = llm_transformer.convert_to_graph_documents(documents)
print(f"Nodes:{graph_documents[0].nodes}")
print(f"Relationships:{graph_documents[0].relationships}")

graph.add_graph_documents(
    graph_documents,
    baseEntityLabel=True,
    include_source=True
)
```


Estrutura do Grafo:

Node properties:

Course {id: STRING}

Entity {id: STRING}

Degree {id: STRING}

Institution {id: STRING}

Person {id: STRING}

Location {id: STRING}

Document {id: STRING}

Date {id: STRING}

Registration number {id: STRING}

Organization {id: STRING}

Department {id: STRING}

University {id: STRING}

Country {id: STRING}

System {id: STRING}

Relationship properties:

The relationships:

(:Course)-[:HAS_DEGREE]->(:Degree)

(:Course)-[:OFFERED_BY]->(:Institution)

(:Course)-[:REGISTERED_UNDER]->(:Registration number)

(:Course)-[:IS_A]->(:Degree)

(:Entity)-[:STRUCTURES]->(:Course)

(:Person)-[:DIRECTOR]->(:Entity)

(:Person)-[:DIRECTOR_OF]->(:University)

(:Person)-[:VICE_DIRECTOR]->(:Entity)

(:Person)-[:VICE_DIRECTOR_OF]->(:University)

(:Person)-[:COORDINATOR]->(:Course)

(:Person)-[:COORDINATOR_OF]->(:Course)

(:Person)-[:PRESIDENT]->(:Entity)

(:Person)-[:PRESIDENT_OF]->(:Organization)
(:Location)-[:LOCATED_IN]->(:Location)
(:Location)-[:LOCATED_IN]->(:Country)
(:Document)-[:CREATED_BY]->(:Course)
(:Date)-[:MEETING_DATE]->(:Entity)
(:Date)-[:DIRECTOR_MEETING_DATE]->(:Course)
(:Date)-[:ESTABLISHED_ON]->(:Course)
(:Organization)-[:STRUCTURES]->(:Course)
(:Department)-[:OFFERS_COURSE]->(:Course)

Exemplo de pergunta para o grafo:

> Entering new GraphCypherQChain chain...

Generated Cypher:

```
MATCH (p:Person)-[:COORDINATOR]->(c:Course) RETURN p, c
```

Full Context:

```
[{'p': {'id': 'Anderson Da Silva Soares'}, 'c': {'id': 'Inteligência Artificial'}}]
```

> Finished chain.

```
{'query': 'Quem é o coordenador do curso?',
```

```
'result': 'O coordenador do curso é Anderson Da Silva Soares.'}
```

Área de atuação

Na Semana 6, decidi minha área de atuação: a construção de um chatbot conversacional utilizando dados do Campeonato Brasileiro de Futebol - Série A. Para dar início ao projeto, comecei a buscar informações relevantes para sua implementação e descobri a plataforma Transfermarkt, conhecida por suas análises detalhadas sobre futebol.

Entrei em contato com os responsáveis pela plataforma e obtive um documento necessário para solicitar acesso a alguns dados. Após preenchê-lo, finalizei o processo para garantir as informações necessárias ao desenvolvimento do chatbot. Três dias depois, recebi o material solicitado, mas percebi que não era satisfatório, pois se restringia apenas a dados de jogadores, sem contemplar o campeonato como um todo. Diante disso, decidi recorrer a uma API de futebol, que disponibiliza dados mais abrangentes em formato JSON. Esses dados, por serem estruturados, permitiram que eu os transformasse em um banco de dados, facilitando consultas complexas. Assim, tive a ideia de estabelecer uma conexão ao banco e acessar as informações utilizando queries SQL geradas pelo modelo de linguagem (LLM), a partir de dados recuperados em um banco de dados vetorial que continha detalhes sobre minha base de dados. Dessa forma, o chatbot poderia fornecer respostas mais completas e contextuais sobre o Campeonato Brasileiro Série A.

APÊNDICE 4

Termo de Aceite de Entrega 7

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 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Tema: ChatBot conversacional com dados Campeonato Brasileiro Série A.

Abordagem: Utilizar RAG + SQL para que o LLM possa formular queries adequadas de acordo com o banco de dados.

Objetivo da Semana: Organizar dados que serão utilizados na entrega final da Residência em IA.

Descartes e Redefinição de Fonte de Dados:

- Descartei os dados da [Transfermarkt](#), que eram focados apenas em atletas e não incluíam informações abrangentes sobre o Campeonato Brasileiro.
- Encontrei uma alternativa na [API Futebol](#), que fornece dados completos sobre campeonatos de futebol do Brasil e do mundo.

Aquisição e Estruturação de Dados:

- Realizei requisições na [API Futebol](#) para obter dados em formato JSON, incluindo:
 - Artilharia, Fases do campeonato, partidas, próximas partidas, rodadas, times e tabelas

Conversão para Banco de Dados:

- Converti os arquivos JSON em tabelas estruturadas dentro de um Banco de Dados SQLite, já que irei trabalhar com perguntas estruturadas, como “Quantos gols fulano marcou no campeonato?”.

Indexação e Preparação para RAG + SQL:

- Iniciei o processo de indexação das informações do banco de dados para realizar a recuperação de informações e enviar para o LLM.

Anotações:  Anotacoes-061124-EvellynNicole

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

Para a próxima Semana pretendo já ter essas informações todas indexadas e começar a construir o fluxo de recuperação, que vai envolver indexação das informações do banco de dados + recuperação das informações de acordo com a query do usuário + geração de sql por parte do modelo + resposta

estruturada.

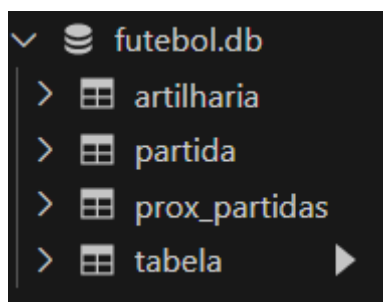
Observação: [caso precise fazer alguma observação, de qualquer “natureza”]

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: Go! ▾

[Figura que mostra as tabelas criadas no banco de dados. Descrito no termo de aceite do dia 06 de novembro]

Banco de Dados



- **Artilharia:** contém os nomes dos jogadores e dados detalhados sobre seus gols, compondo uma tabela de artilheiros do campeonato.
- **Partida:** armazena todos os dados de cada partida do campeonato, incluindo substituições, escalações, placar, cartões amarelos, gols, entre outros detalhes relevantes.
- **Prox_Partidas:** contém informações sobre as próximas partidas do campeonato, como data, horário, equipes envolvidas, time mandante, entre outros.
- **Tabela:** representa a classificação do campeonato, com a posição de cada time, pontos acumulados e outros dados que indicam a situação atual de cada equipe.

[Figura que mostra as tabelas artilharia e partida e suas respectivas colunas. Descrito no termo de aceite do dia 06 de novembro]

```

futebol.db
├── artilharia
│   ├── atleta_id : integer
│   ├── nome_popular : text
│   ├── posicao_nome : text
│   ├── posicao_sigla : text
│   ├── time_id : integer
│   ├── time_nome_popular : text
│   ├── time_sigla : text
│   └── gols : integer
├── partida
│   ├── partida_id : integer
│   ├── campeonato_id : integer
│   ├── campeonato_nome : text
│   ├── time_mandante_id : integer
│   ├── time_mandante_nome : text
│   ├── time_visitante_id : integer
│   ├── time_visitante_nome : text
│   ├── placar_mandante : integer
│   ├── placar_visitante : integer
│   ├── disputa_penalti : integer
│   ├── status : text
│   ├── data_realizacao : text
│   ├── hora_realizacao : text
│   ├── estadio_id : integer
│   ├── estadio_nome : text
│   ├── rodada : text
│   ├── estatisticas : text
│   ├── escalacoes : text
│   ├── gols : text
│   ├── substituicoes : text
│   └── cartoes : text

```

A tabela **artilharia** contém informações sobre jogadores (atletas) que marcam gols, possivelmente relacionadas a um campeonato ou competição específica.

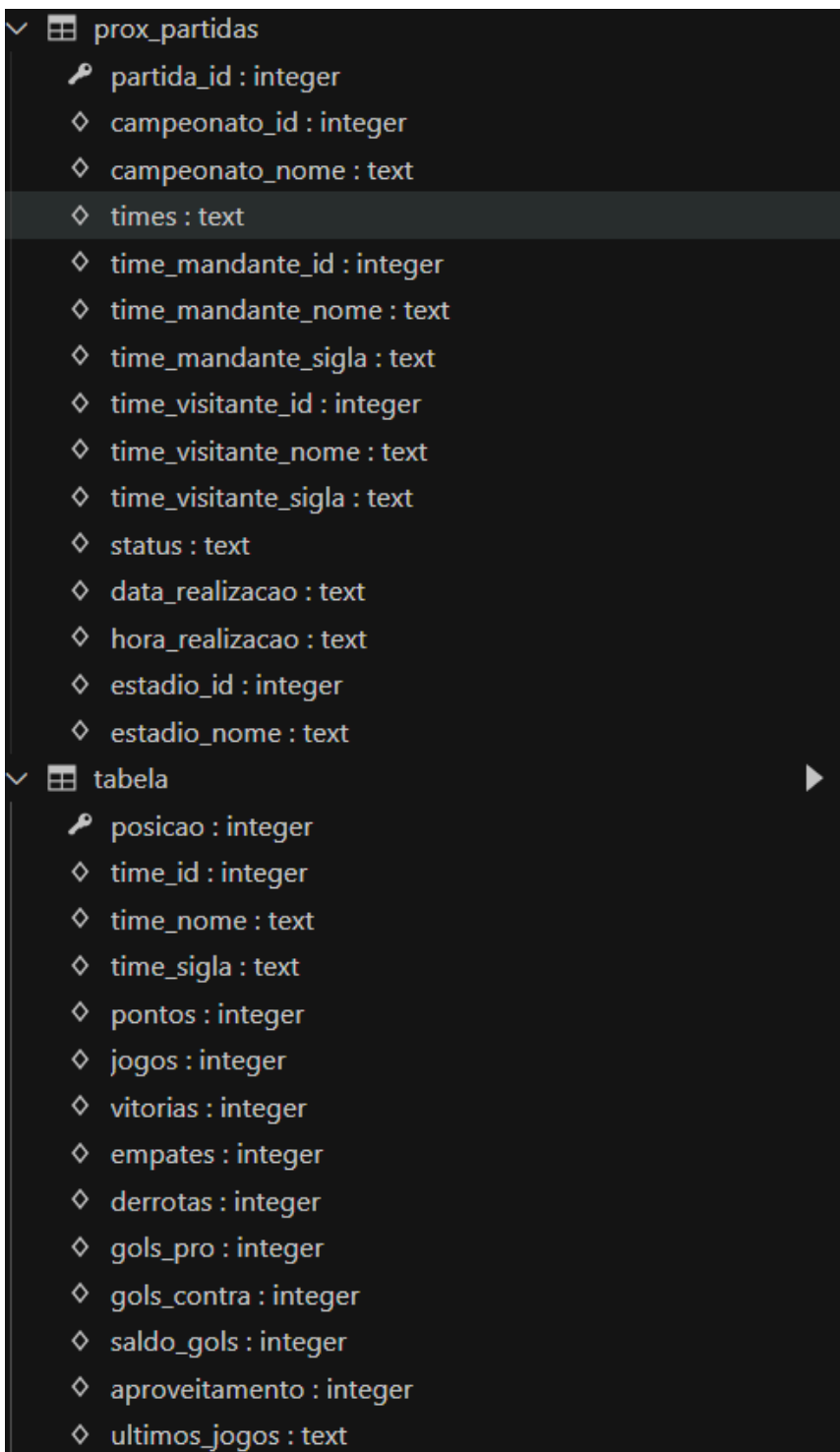
- **atleta_id** (integer):
Identificador único do atleta (jogador). É utilizado para relacionar o artilheiro a seus dados pessoais ou a outras informações do atleta em outras tabelas.
- **nome_popular** (text):
Nome popular do jogador, normalmente o nome pelo qual ele é conhecido no meio esportivo (por exemplo, um apelido ou nome de destaque).
- **posicao_nome** (text):
Nome da posição em que o atleta atua (ex: "Atacante", "Meia", "Zagueiro").
- **posicao_sigla** (text):
Sigla ou abreviação utilizada para representar a posição do atleta (ex: "ATA" para atacante).
- **time_id** (integer):
Identificador único do time ao qual o atleta pertence. Permite a ligação com informações da equipe, possivelmente em outra tabela.
- **time_nome_popular** (text):
Nome popular do time a que o atleta pertence (ex: "Flamengo", "Palmeiras"). É a forma usual de se referir ao clube.
- **time_sigla** (text):
Sigla do time, geralmente utilizada para identificação rápida (ex: "FLA", "PAL").
- **gols** (integer):
Número de gols marcados pelo atleta no contexto analisado (por exemplo, no campeonato em questão).

A tabela **partida** armazena dados sobre jogos específicos (partidas) de um campeonato ou torneio.

- **partida_id** (integer):
Identificador único da partida.
- **campeonato_id** (integer):
Identificador do campeonato ou competição ao qual a partida pertence.
- **campeonato_nome** (text):
Nome do campeonato ou torneio (ex: "Campeonato Brasileiro", "Copa do Mundo").
- **time_mandante_id** (integer):
Identificador do time mandante (o time que joga em casa).
- **time_mandante_nome** (text):
Nome do time mandante da partida.
- **time_visitante_id** (integer):
Identificador do time visitante (o time que joga fora de casa).

- **time_visitante_nome (text):**
Nome do time visitante da partida.
- **placar_mandante (integer):**
Número de gols marcados pelo time mandante no final da partida.
- **placar_visitante (integer):**
Número de gols marcados pelo time visitante no final da partida.
- **disputa_penalti (integer):**
Indicador se houve disputa de pênaltis após o tempo regulamentar, geralmente indicando quantas penalidades cada time converteu, ou 1/0 se apenas houve ou não houve pênaltis.
- **status (text):**
Estado atual ou final da partida (ex: "Encerrada", "Em andamento", "Agendada").
- **data_realizacao (text):**
Data programada ou em que a partida ocorreu (formato possivelmente "YYYY-MM-DD").
- **hora_realizacao (text):**
Horário programado ou em que a partida começou (formato possivelmente "HH:MM").
- **estadio_id (integer):**
Identificador do estádio onde a partida foi disputada.
- **estadio_nome (text):**
Nome do estádio onde a partida foi realizada.
- **rodada (text):**
Informações sobre a rodada da competição (por exemplo: "Rodada 15").
- **estatisticas (text):**
Campo contendo dados estatísticos em formato serializado (JSON ou outro), como posse de bola, finalizações, escanteios etc.
- **escalacoes (text):**
Campo que contém informações sobre as escalações dos times (possivelmente em formato JSON ou outro estruturado), listando atletas, numeração, etc.
- **gols (text):**
Detalhes sobre gols marcados na partida (marcador do gol, tempo, etc.).
- **substituicoes (text):**
Informações sobre substituições realizadas durante a partida (quem saiu, quem entrou, em qual minuto).
- **cartoes (text):**
Informações sobre cartões amarelos e vermelhos distribuídos na partida (para qual jogador, em que minuto, etc.).

[Figura que mostra as tabelas próximas partidas e tabela e suas respectivas colunas. Descrito no termo de aceite do dia 06 de novembro]



```

└─ prox_partidas
  ├── partida_id : integer
  ├── campeonato_id : integer
  ├── campeonato_nome : text
  ├── times : text
  ├── time_mandante_id : integer
  ├── time_mandante_nome : text
  ├── time_mandante_sigla : text
  ├── time_visitante_id : integer
  ├── time_visitante_nome : text
  ├── time_visitante_sigla : text
  ├── status : text
  ├── data_realizacao : text
  ├── hora_realizacao : text
  ├── estadio_id : integer
  └── estadio_nome : text

└─ tabela
  ├── posicao : integer
  ├── time_id : integer
  ├── time_nome : text
  ├── time_sigla : text
  ├── pontos : integer
  ├── jogos : integer
  ├── vitorias : integer
  ├── empates : integer
  ├── derrotas : integer
  ├── gols_pro : integer
  ├── gols_contra : integer
  ├── saldo_gols : integer
  ├── aproveitamento : integer
  └── ultimos_jogos : text

```

Tabela: prox_partidas

- partida_id (integer): Identificador único da partida.
- campeonato_id (integer): Identificador do campeonato ao qual a partida pertence.
- campeonato_nome (text): Nome do campeonato.
- times (text): Lista ou conjunto de informações sobre os times envolvidos na partida.
- time_mandante_id (integer): Identificador do time mandante.
- time_mandante_nome (text): Nome do time mandante.
- time_mandante_sigla (text): Sigla do time mandante.
- time_visitante_id (integer): Identificador do time visitante.
- time_visitante_nome (text): Nome do time visitante.
- time_visitante_sigla (text): Sigla do time visitante.
- status (text): Status da partida (por exemplo, "agendada", "encerrada").
- data_realizacao (text): Data programada para a realização da partida.
- hora_realizacao (text): Horário programado para a realização da partida.
- estadio_id (integer): Identificador do estádio onde a partida será realizada.
- estadio_nome (text): Nome do estádio onde a partida será realizada.

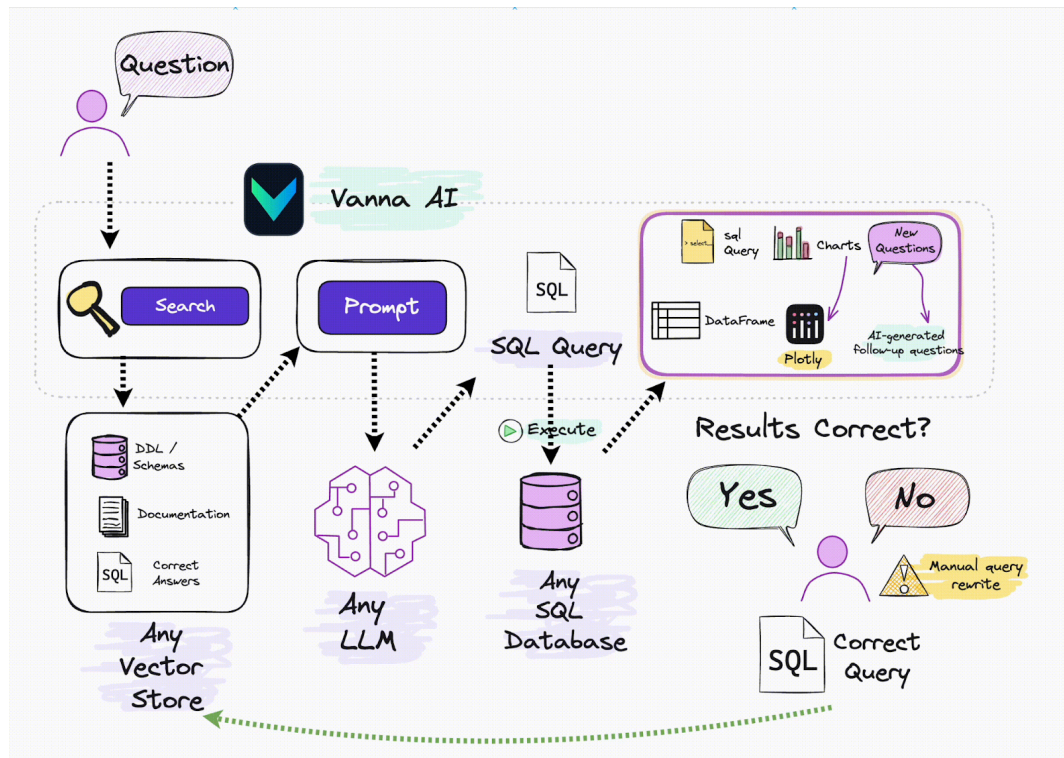
Tabela: tabela

- posicao (integer): Posição do time na tabela de classificação.
- time_id (integer): Identificador do time.
- time_nome (text): Nome do time.
- time_sigla (text): Sigla do time.
- pontos (integer): Pontos acumulados pelo time no campeonato.
- jogos (integer): Número total de jogos disputados pelo time.
- vitorias (integer): Número total de vitórias do time.
- empates (integer): Número total de empates do time.
- derrotas (integer): Número total de derrotas do time.
- gols_pro (integer): Número de gols marcados pelo time.
- gols_contra (integer): Número de gols sofridos pelo time.
- saldo_gols (integer): Diferença entre gols marcados e gols sofridos.
- aproveitamento (integer): Percentual de aproveitamento do time no campeonato.
- ultimos_jogos (text): Sequência dos últimos resultados do time, representada por letras que indiquem vitória, empate ou derrota.

Vanna AI

Testes preliminares foram realizados utilizando a ferramenta Vanna AI. No entanto, os resultados não foram satisfatórios, o que me fez sair desse ambiente.

[Figura que mostra o fluxo do VannaAI. Descrito no termo de aceite do dia 06 de novembro]



APÊNDICE 5

Termo de Aceite de Entrega 8

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: 14 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Tema: ChatBot conversacional com dados do Campeonato Brasileiro Série A.

Abordagem: Utilizar RAG + SQL para que o LLM possa formular queries adequadas de acordo com o banco de dados.

Objetivo da Semana: Indexar os dados e começar testes iniciais.

Os dados foram indexados no [Qdrant](#), que é um banco de dados vetorial. Para fazer essa indexação utilizei 3 coleções: **esquema** (esquema do banco de dados), **documentation** (documentação das tabelas e colunas do banco de dados) e **sql** (exemplos de consultas sql).

Com isso, já comecei a testar a geração de sql para ver quais são os acertos e as dificuldades do modelo e como a recuperação ajudou e pode ajudar. Notei que está tendo uma dificuldade para gerar queries que envolvem datas, número de vitórias, derrotas e empates consecutivos e queries que envolvem mais de uma pergunta.

Planilha comparativa: [dataset_sql_futebol](#)

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

Para as próximas Semanas pretendo melhorar a documentação e gerar mais sqls de exemplo para resolver os problemas observados. Além disso, quero montar um dataset maior para servir de avaliação e validação. Pretendo também testar fora do framework do Vanna AI, para que eu possa ter mais controle da minha aplicação.

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

ACEITE DA ENTREGA:

CEDRIC LUIZ DE CARVALHO: [Go!](#)

Indexação dos Dados no Banco de Dados Vetorial

A indexação é uma etapa fundamental para viabilizar o uso de **RAG (Retrieval-Augmented Generation)** no desenvolvimento do chatbot conversacional sobre o Campeonato Brasileiro Série A. Nesse caso, o **Qdrant**, um banco de dados vetorial, foi utilizado para armazenar e organizar informações que auxiliam o modelo na formulação de queries SQL.

Estrutura das Coleções Indexadas

Três coleções foram criadas no Qdrant para atender diferentes necessidades de recuperação de informações:

1. Esquema (Schema):

- Contém o esquema do banco de dados, incluindo a estrutura das tabelas, os nomes das colunas e seus tipos.
- Ajuda o modelo a compreender como os dados estão organizados e quais tabelas ou colunas podem ser consultadas para responder uma pergunta.

[Imagem dos dados indexados no banco de dados vetorial referente ao esquema. Descrito no termo de aceite do dia 14 de novembro]

```
page_content      CREATE TABLE prox_partidas ( partida_id INTEGER PRIMARY KEY, campeonato_id INTEGER,
campeonato_nome TEXT, times TEXT, time_mandante_id INTEGER, time_mandante_nome TEXT,
time_mandante_sigla TEXT, time_visitante_id INTEGER, time_visitante_nome TEXT,
time_visitante_sigla TEXT, status TEXT, data_realizacao TEXT, hora_realizacao TEXT, estadio_id
INTEGER, estadio_nome TEXT );

metadata          ~{ 2 Items
  "table_name": "prox_partidas"
  "type": "schema"
}
```

2. Documentação:

- Inclui descrições detalhadas das tabelas e colunas do banco de dados.

- Por exemplo, uma coluna como *placar_mandante* pode conter uma descrição indicando que armazena o número de gols marcados pelo time mandante em uma partida.
- Isso auxilia o modelo a interpretar corretamente o propósito de cada elemento no banco.

[Imagens dos dados indexados no banco de dados vetorial referente a documentação de colunas. Descrito no termo de aceite do dia 14 de novembro]

Point 0a81c4ef-253e-4f75-93fd-3f44b718d3ea

Payload:

page_content	A coluna vitórias da tabela 'tabela' armazena o número total de vitórias do time.
metadata	<pre>~{ 3 Items "table_name": "tabela" "type": "documentation" "column_name": "vitorias" }</pre>

Point 0efdb1c4-05fe-4295-a93d-8964687e9623

Payload:

page_content	A coluna estatísticas dentro da tabela partida armazena as estatísticas gerais da partida, como posse de bola, chutes a gol, etc.
metadata	<pre>~{ 3 Items "table_name": "partida" "column_name": "estatisticas" "type": "documentation" }</pre>

3. SQL (Exemplos de Consultas Bem-Sucedidas):

- Contém exemplos de queries SQL válidas e bem-formuladas.
- Essas consultas fornecem um padrão que o modelo pode seguir para estruturar queries adequadas, principalmente para casos de uso comuns.

[Imagens dos dados indexados no banco de dados vetorial referente a sqls de exemplo. Descrito no termo de aceite do dia 14 de novembro]

```
metadata      ~{ 2 Items
  "type": "sql"
  "sql": " SELECT time_nome FROM tabela WHERE derrotas >= 15 "
}

page_content  Quais times tiveram 15 ou mais de 15 derrotas?
```

```
page_content  Qual time fez mais gols na rodada 1 do Brasileirão?

metadata      ~{ 2 Items
  "sql":
  "SELECT time_nome, SUM(total_gols) AS total_gols FROM ( SELECT time_mandante_nome
  AS time_nome, placar_mandante AS total_gols FROM partida WHERE rodada = '1ª
  Rodada' UNION ALL SELECT time_visitante_nome AS time_nome, placar_visitante AS
  total_gols FROM partida WHERE rodada = '1ª Rodada' ) AS gols_por_time GROUP BY
  time_nome ORDER BY total_gols DESC LIMIT 3;"
  "type": "sql"
}
```

Fluxo de Recuperação e Integração com o LLM

A ideia principal do processo é usar a recuperação para fornecer ao modelo de linguagem (LLM) o contexto necessário antes de formular a resposta. O fluxo funciona assim:

1. Pergunta do Usuário:

O chatbot recebe uma pergunta, como:

- "Quantas vitórias consecutivas o Palmeiras teve no campeonato?"

2. Recuperação:

- A pergunta é usada para buscar informações relevantes nas coleções do Qdrant.
- Isso inclui partes do esquema, documentação das colunas relevantes e exemplos de consultas SQL que envolvem lógicas similares.

3. Contextualização:

- As informações recuperadas são enviadas como contexto para o LLM.
- Esse contexto orienta o modelo a gerar queries SQL mais precisas, mesmo em perguntas complexas.

Observações Iniciais

Durante os testes, foi observado que o modelo enfrentou dificuldades em:

- Queries relacionadas a datas:
Como calcular eventos em um intervalo de tempo específico.
- Consultas complexas:
Como identificar vitórias, derrotas ou empates consecutivos, que exigem manipulação avançada de dados.
- Perguntas compostas:
Quando a consulta SQL precisa responder a mais de uma questão simultaneamente.

APÊNDICE 6

Termo de Aceite de Entrega 9

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: 27 de nov. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Tema: ChatBot conversacional com dados do Campeonato Brasileiro Série A.

Abordagem: Utilizar **RAG** para que o LLM possa formular queries SQL adequadas de acordo com informações do banco de dados.

Objetivos da Semana:

1. Expandir o banco de dados vetorial com mais exemplos de consultas SQL, ajustando conforme as dificuldades observadas nas respostas do modelo.
2. Desenvolver a solução em um ambiente externo ao VannaAI, utilizando Qdrant e Langchain como principais ferramentas.
3. Validar a geração de SQL com um [dataset](#) de teste ampliado, verificando a precisão e adequação das consultas geradas.
4. Implementar um modelo conversacional que ofereça respostas contextualizadas e em linguagem fluida ao usuário.
5. Criar um [repositório no GitHub](#) para controle de versão e compartilhamento do código.

Exemplo da parte conversacional:

pergunta do usuário:

“Quanto ficou a partida entre Flamengo e São Paulo e quando ela foi realizada?”

resposta do modelo:

A partida entre Flamengo e São Paulo teve dois encontros recentes. No primeiro, realizado no dia 17 de abril de 2024, o Flamengo foi o mandante e venceu por 2 a 1. Já no segundo confronto, que ocorreu em 3 de agosto de 2024, o São Paulo jogou em casa e ganhou por 1 a 0.

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

Para a próxima Semana pretendo realizar mais testes, construir uma interface e desenhar o fluxo final do projeto.

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

LEONARDO ANTÔNIO ALVES: Go!

Dataset de Validação

[Tabela de algumas amostras do dataset de validação. Descrito no termo de aceite do dia 27 de novembro]

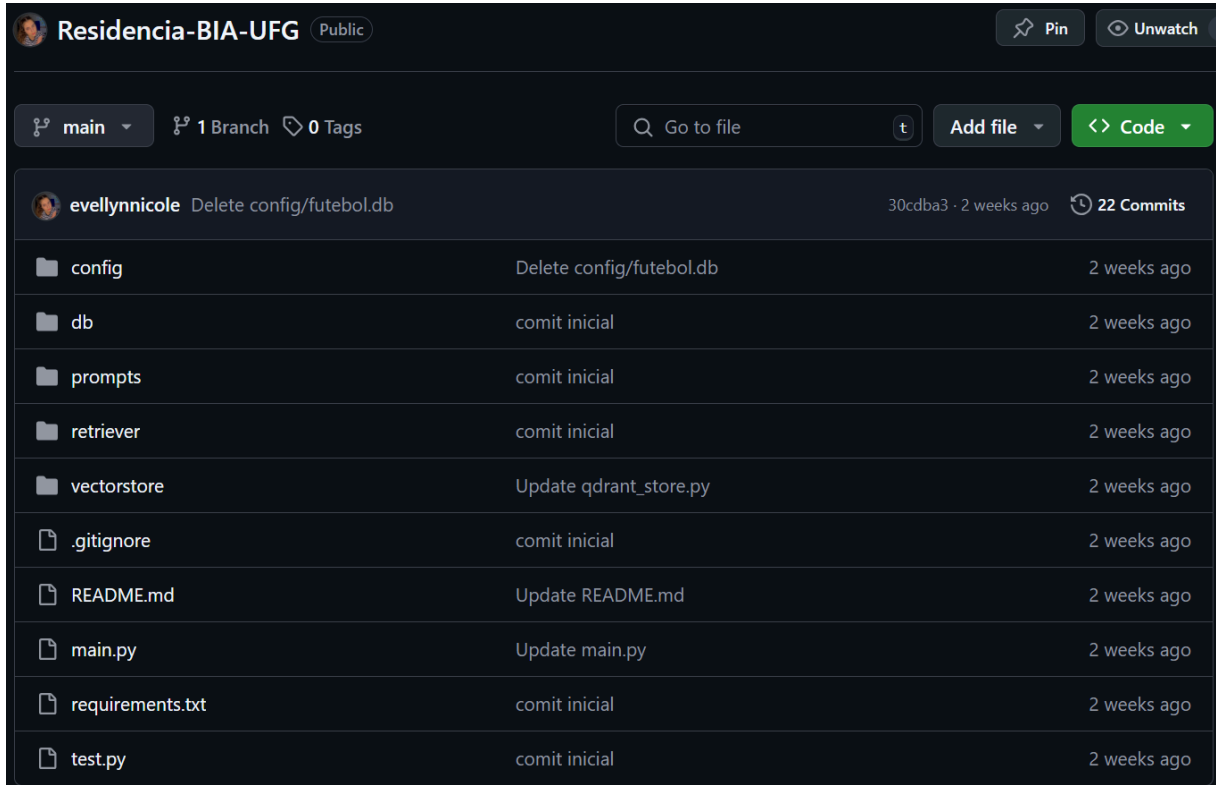
Sql Correto	Resultado - sql correto	Question	Sql gerado	Resultado - sql gerado
<pre>SELECT time_mandante_ nome, time_visitante_no me FROM partida WHERE data_realizacao = '14/04/2024';</pre>	<pre>[('Cruzeiro', 'Botafogo'), ('Vitória', 'Palmeiras'), ('Vasco', 'Grêmio'), ('Corinthians', 'Atlético-MG'), ('Athletico-PR', 'Cuiabá'), ('Atlético-GO', 'Flamengo')]</pre>	Quais são os nomes dos times mandantes e visitantes das partidas realizadas no dia 14 de abril?	<pre>SELECT time_mandante_ nome, time_visitante_no me FROM partida WHERE data_realizacao = '14/04/2024';</pre>	<pre>[('Cruzeiro', 'Botafogo'), ('Vitória', 'Palmeiras'), ('Vasco', 'Grêmio'), ('Corinthians', 'Atlético-MG'), ('Athletico-PR', 'Cuiabá'), ('Atlético-GO', 'Flamengo')]</pre>
<pre>SELECT estadio_nome, COUNT(*) AS num_partidas FROM partida GROUP BY estadio_id HAVING COUNT(*) > 15</pre>	<pre>[(None, 50), ('Heriberto Hülse', 18), ('Maracanã', 33), ('Nilton Santos (Engenhão)', 16), ('Alfredo Jaconi', 16), ('Nabi Abi Chedid', 16), ('Arena Pantanal', 16), ('Barradão', 17), ('Castelão (CE)', 16), ('Antônio Accioly', 16)]</pre>	Quais estádios tiveram mais de 15 partidas?	<pre>SELECT estadio_nome, COUNT(partida_i d) AS total_partidas FROM partida GROUP BY estadio_nome HAVING COUNT(partida_i d) > 15;</pre>	<pre>[(None, 50), ('Alfredo Jaconi', 16), ('Antônio Accioly', 16), ('Arena Pantanal', 16), ('Barradão', 17), ('Castelão (CE)', 16), ('Heriberto Hülse', 18), ('Maracanã', 33), ('Nabi Abi Chedid', 16), ('Nilton Santos (Engenhão)', 16)]</pre>
<pre>SELECT</pre>	<pre>[('Pedro',),</pre>	Quais são os	<pre>SELECT</pre>	<pre>[('Pedro',),</pre>

nome_popular FROM artilharia WHERE gols > 10	('Estevão',)]	nomes dos jogadores que fizeram mais de 10 gols?	nome_popular FROM artilharia WHERE gols > 10;	('Estevão',)]
SELECT data_realizacao, COUNT(*) AS partidas_no_dia FROM partida GROUP BY data_realizacao ORDER BY partidas_no_dia DESC	[('24/11/2024', 10)	Qual foi o dia com mais partidas realizadas?	SELECT data_realizacao, COUNT(partida_id) AS total_partidas FROM partida GROUP BY data_realizacao ORDER BY total_partidas DESC LIMIT 1;	[('24/11/2024', 10)]
SELECT time_mandante_n ome, time_visitante_no me, placar_mandante, placar_visitante FROM partida WHERE rodada = '10ª Rodada' AND (placar_mandante > placar_visitante OR placar_visitante > placar_mandante)	[('Internacional', 'Corinthians', 1, 0), ('Cruzeiro', 'Fluminense', 2, 0), ('Vitória', 'Atlético-MG', 4, 2), ('Flamengo', 'Bahia', 2, 1), ('Palmeiras', 'Bragantino', 2, 1), ('São Paulo', 'Cuiabá', 0, 1), ('Fortaleza', 'Grêmio', 1, 0), ('Atlético-GO', 'Criciúma', 1, 2), ('Juventude', 'Vasco', 2, 0)]	Quais partidas da 10ª rodada não terminaram em empate?	SELECT partida_id, time_mandante_n ome, time_visitante_no me, placar_mandante, placar_visitante FROM partida WHERE rodada = '10ª Rodada' AND placar_mandante != placar_visitante;	[(17037, 'Internacional', 'Corinthians', 1, 0), (17039, 'Cruzeiro', 'Fluminense', 2, 0), (17041, 'Vitória', 'Atlético-MG', 4, 2), (17043, 'Flamengo', 'Bahia', 2, 1), (17047, 'Palmeiras', 'Bragantino', 2, 1), (17048, 'São Paulo', 'Cuiabá', 0, 1), (17049, 'Fortaleza', 'Grêmio', 1, 0), (17050, 'Atlético-GO', 'Criciúma', 1, 2), (17051, 'Juventude', 'Vasco', 2, 0)]

GitHub

[A imagen a seguir podem ser acessada no seguinte link:

<https://github.com/evellynicole/Residencia-BIA-UFG/tree/main>. Descrito no termo de aceite do dia 27 de novembro]



The screenshot shows the GitHub interface for the repository 'Residencia-BIA-UFG'. The repository is public and has 22 commits. The commit history is as follows:

Commit Message	Author	Date
Delete config/futebol.db	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago
Update qdrant_store.py	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago
Update README.md	evellynicole	2 weeks ago
Update main.py	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago
comit inicial	evellynicole	2 weeks ago

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 dez. de 2024

Participantes da Entrega [matriculados em Residência em IA]:

EVELLYN NICOLE MACHADO ROSA

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

Tema: ChatBot conversacional com dados do Campeonato Brasileiro Série A.

Abordagem: Utilizar **RAG** para que o LLM possa formular queries SQL adequadas de acordo com informações do banco de dados.

Objetivos da Semana:

1. Realizar mais testes.
2. Indexação de mais informações
3. [Criar interface](#)
4. [Desenho do fluxo do projeto](#)

Cheguei no meu objetivo final e no que me propus durante a minha especialização.

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

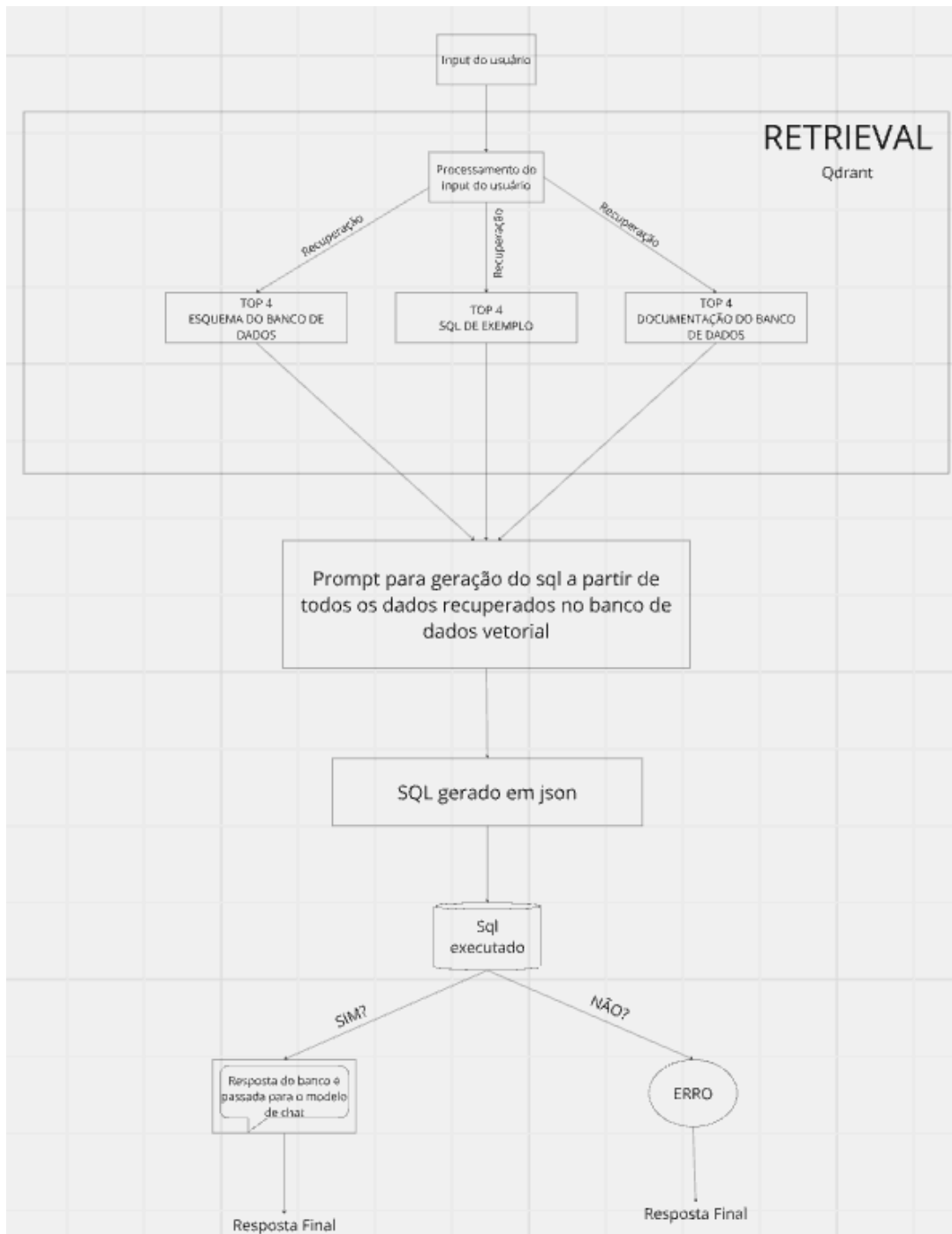
Atualizar meu banco de dados com a última rodada do Brasileirão.

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

CEDRIC LUIZ DE CARVALHO: Go! ▾

Fluxo Projeto Final

[Imagem do desenho do fluxo final desenvolvido. Descrito no termo de aceite do dia 04 de dezembro]



1. Input do usuário:

O processo inicia quando o usuário fornece uma entrada textual, que pode ser uma pergunta ou uma requisição de informação que exige a consulta ao banco de dados.

2. Processamento do input do usuário (Retrieval):

O sistema recebe o input e realiza uma etapa de "retrieval", ou seja, busca contextos e informações relevantes num banco de dados vetorial (usando Qdrant). Durante essa busca, são identificados documentos ou trechos mais relevantes, baseados no conteúdo fornecido pelo usuário.

3. Recuperação de Dados (Top 4):

A partir do processamento, o sistema seleciona:

- Top 4 esquema do banco de dados: Quatro melhores resultados que descrevem a estrutura do banco (nome de tabelas, colunas, tipos de dados).
- Top 4 SQL de exemplo: Quatro melhores exemplos de consultas SQL já existentes ou pré-definidas, que sirvam de modelo para a consulta a ser gerada.
- Top 4 documentação do banco de dados: Quatro melhores trechos da documentação, descrevendo o significado das tabelas, colunas ou relacionamentos.

4. Esses três conjuntos de dados são resultados da etapa de recuperação e fornecem insumos de contexto para a geração da consulta SQL final.

5. Geração do prompt para SQL:

Com base nos insumos obtidos (esquema do banco de dados, exemplos de SQL e documentação), o sistema constrói um prompt. Esse prompt é essencialmente uma instrução clara e contextualizada para um modelo de linguagem, que irá gerar a consulta SQL desejada.

6. SQL gerado em JSON:

A partir do prompt construído, o modelo gera o SQL correspondente em formato

JSON. Esse JSON encapsula a consulta SQL que deverá ser executada contra o banco de dados.

7. Execução do SQL:

A consulta SQL, extraída do JSON, é então executada diretamente no banco de dados real.

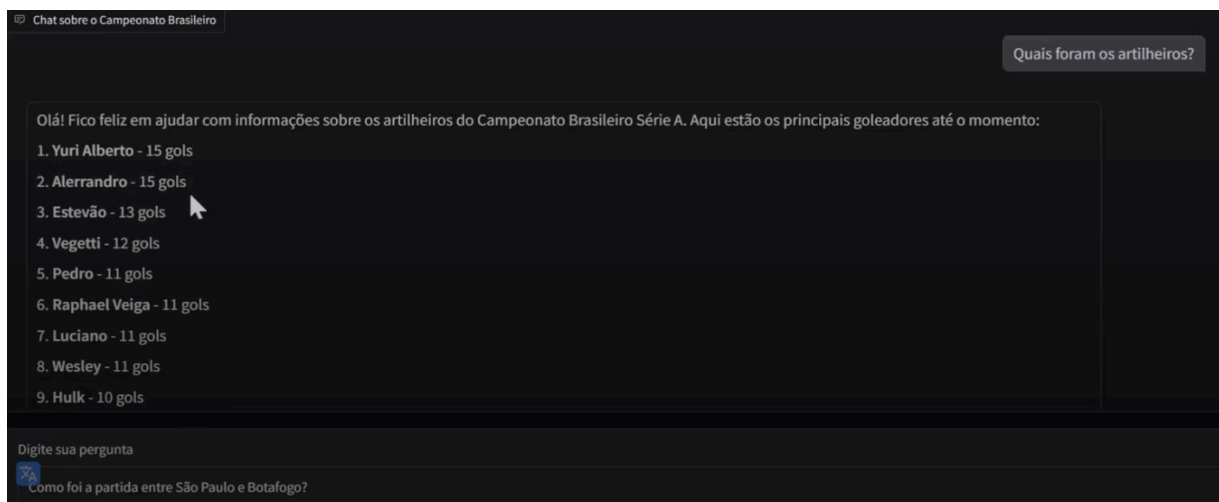
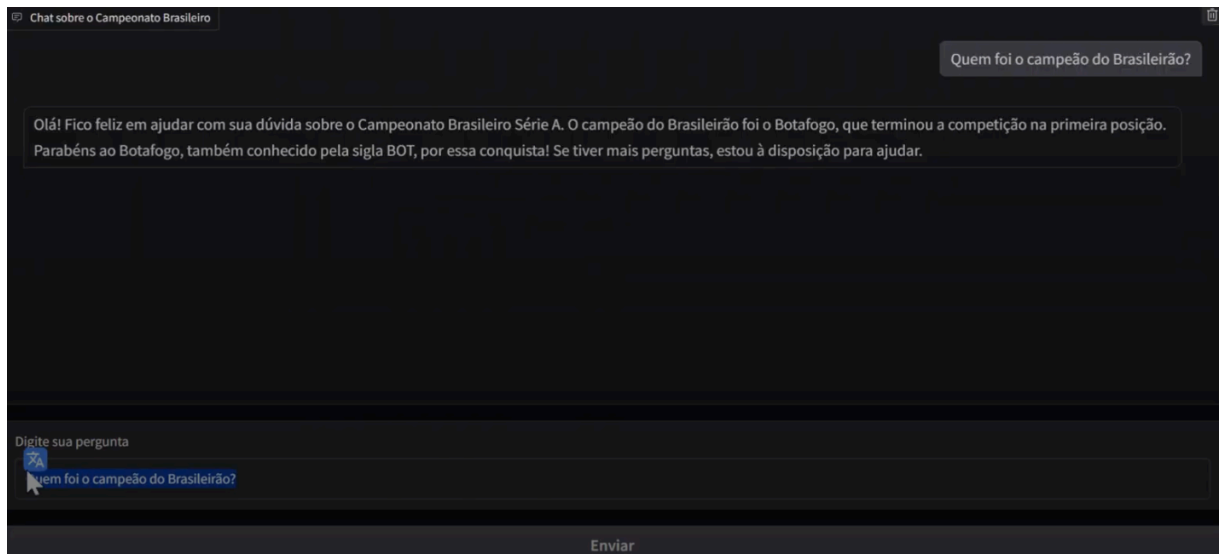
- Se a execução for bem-sucedida, o resultado retornado pelo banco de dados (linhas, colunas e dados) é então passado para o modelo.
- Se ocorrer um erro na execução (por exemplo, sintaxe inválida, tabela inexistente), o fluxo segue para a rota de erro.

8. Resposta Final:

- Caso a execução do SQL seja bem-sucedida, a resposta retornada pelo banco (os dados resultantes da consulta) são fornecidos ao modelo, que os utiliza para produzir a resposta final ao usuário em linguagem natural.
- Se houver erro na execução SQL, o sistema retorna uma mensagem de erro.

Em resumo, o fluxo do projeto é uma cadeia de etapas: usuário envia uma requisição → informações contextuais são recuperadas do banco vetorial → um prompt é criado a partir desses dados → o prompt gera um SQL → o SQL é executado → a resposta do banco é integrada e formatada → o usuário recebe a resposta final. Caso haja falha em alguma etapa (especialmente na execução do SQL), uma mensagem de erro é retornada ao usuário.

[Imagens da interface final. Descrito no termo de aceite do dia 04 de dezembro]



Chat sobre o Campeonato Brasileiro

Como foi a partida entre São Paulo e Botafogo?

Olá! Vamos falar sobre os confrontos entre São Paulo e Botafogo no Campeonato Brasileiro Série A de 2024.

1. Primeiro Jogo:

- Data: 24 de julho de 2024
- Local: Estádio do Morumbi
- Resultado: São Paulo 2 x 2 Botafogo

Foi uma partida emocionante no Morumbi, onde o São Paulo, jogando em casa, empatou com o Botafogo. Ambos os times mostraram um bom desempenho, resultando em um jogo equilibrado.

2. Segundo Jogo:

- Data: 8 de dezembro de 2024
- Local: Estádio Nilton Santos (Engenhão)
- Resultado: Botafogo 2 x 1 São Paulo

Digite sua pergunta

Faça sua pergunta aqui...

Enviar

Chat sobre o Campeonato Brasileiro

Espero que essas informações tenham ajudado! Se precisar de mais detalhes, estou à disposição. 😊

quais foram os 4 times rebaixados?

Olá! Vamos falar sobre os times que infelizmente foram rebaixados no Campeonato Brasileiro Série A. De acordo com a classificação final, os quatro times que terminaram nas últimas posições e foram rebaixados são:

1. Cuiabá, que ficou na 20ª posição.
2. Atlético-GO, que terminou em 19º lugar.
3. Criciúma, que ficou na 18ª posição.
4. Athletico-PR, que terminou em 17º lugar.

Espero que essas informações tenham ajudado! Se precisar de mais alguma coisa, é só me chamar. 😊

Digite sua pergunta

quais foram os 4 times rebaixados?

Enviar