Aprimoramentos em sistemas RAG*

Matheus Oliveira Silva[1 [119180151]]

Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil
matheusflups8@gmail.com

Abstract. This project works on the integration of OpenAI's large language model (LLM) with a customized knowledge base to create a robust Retrieval-Augmented Generation (RAG) system. Utilizing the innovative 'llama_index' library, this integration enhances the LLM's data ingestion capabilities, enabling it to effectively utilize the specific data from the knowledge base. This synergy creates a dynamic RAG system that not only leverages the vast language understanding and generation capabilities of the LLM but also incorporates specialized knowledge from the custom database. This paper details the development process, challenges, and solutions encountered in integrating these technologies, and provides a comprehensive evaluation of the system's performance. The implications of this integration for future language model applications and knowledge-based systems are also explored, making this work a significant contribution to the field of AI and data management.

Keywords: RAG · information retrieval · chatbots.

1 Introdução

Nos últimos anos, os Modelos de Linguagem de Grande Escala (LLMs) têm revolucionado o campo da Inteligência Artificial, oferecendo capacidades impressionantes de compreensão e geração de linguagem. No entanto, a aplicação desses modelos em domínios específicos muitas vezes exige uma integração cuidadosa com dados personalizados para alcançar resultados verdadeiramente relevantes e precisos. Neste contexto, surge o desafio de como melhorar a eficácia dos LLMs para tarefas específicas, mantendo a riqueza e a diversidade de suas capacidades linguísticas.

Este trabalho apresenta um projeto inovador que aborda esse desafio, integrando um Modelo de Linguagem de Grande Escala da OpenAI com uma base de conhecimento customizada, resultando em um sistema de Geração Aumentada por Recuperação (RAG). A integração é realizada por meio do uso do llama_index, uma biblioteca avançada destinada à manipulação e recuperação eficiente de grandes conjuntos de dados.

O objetivo principal deste trabalho é demonstrar como a personalização de dados no contexto de LLMs pode enriquecer significativamente a qualidade e a aplicabilidade das respostas geradas, fornecendo um contexto mais preciso e uma

* Supported by COGG.AI

relevância aumentada em relação a domínios específicos. Para isso, exploramos a capacidade do sistema RAG de melhorar a contextualização das respostas do modelo, utilizando uma base de conhecimento que é tanto abrangente quanto específica ao domínio em questão.

Este artigo detalha a metodologia utilizada para a integração do LLM com a base de dados, discute as estratégias adotadas para enfrentar os desafios técnicos encontrados e analisa os resultados alcançados, demonstrando o impacto positivo dessa integração na performance do modelo em tarefas de processamento de linguagem natural.

2 Fundamentação Teórica

2.1 Modelos de Linguagem de Grande Escala (LLMs)

Modelos de Linguagem de Grande Escala (LLMs), como o GPT-4 da OpenAI, são ferramentas avançadas de IA que transformaram a maneira como interagimos e geramos conteúdo. Estes modelos são treinados em vastos conjuntos de dados de linguagem natural pública, abrangendo fontes como artigos da Wikipedia, discussões no Reddit, e perguntas relacionadas a programação similares às encontradas no Stack Overflow. Embora sejam extremamente capazes de fornecer acesso a uma grande quantidade de informações para busca e recuperação, eles muitas vezes carecem de otimização e treinamento em dados específicos de domínios ou organizações, o que pode ser uma barreira para a sua integração em contextos mais especializados.

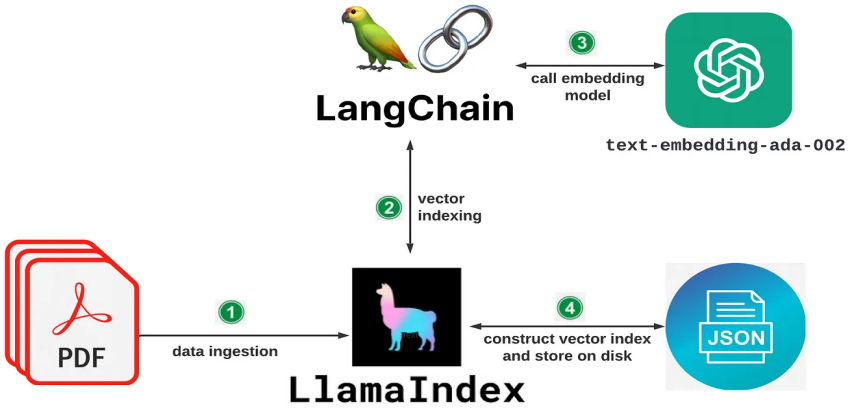
2.2 Llama Index

LlamaIndex, anteriormente conhecido como GPT Index, é uma estrutura de dados inovadora projetada para apoiar o desenvolvimento de aplicações baseadas em LLMs. Ele oferece uma interface avançada que permite aos desenvolvedores integrar uma variedade de fontes de dados aos LLMs.

2.3 Retrieval-Augmented Generation (RAG)

RAG é um método que combina a recuperação de informações relevantes (por exemplo, de uma base de conhecimento) com a geração de linguagem natural. Ele permite que o sistema produza respostas mais informativas e contextualizadas ao utilizar dados recuperados de fontes relevantes. O LlamaIndex, com suas capacidades avançadas de indexação e recuperação de dados, é particularmente adequado para a construção de sistemas RAG, oferecendo uma abordagem estruturada para a ingestão, organização e consulta de dados em combinação com LLMs. Uma imagem que representa bem como o Llama Index constrói um modelo RAG é a Figura 1.

Fig. 1. Pipeline RAG usando Llama Index



Pipeline RAG implementada usando Llama Index que ingere, vetoriza, indexa e faz embbeding de documentos.

3 Metodologia

3.1 Arquitetura

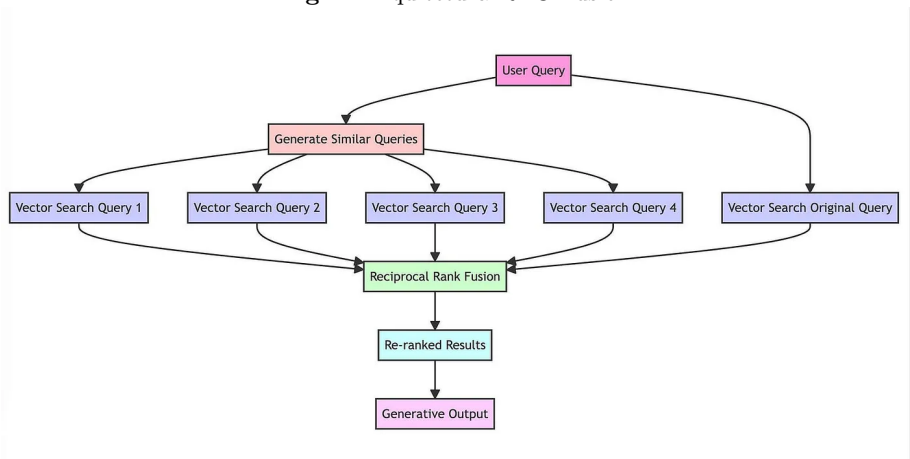
A proposta desse trabalho é implementar em Python uma arquitetura do tipo RAG-Fusion [1] que tem como proposta principal, pegar a query original do usuário e explorar diferente ângulos do que está sendo perguntado como podemos ver na Figura 2.

Essa proposta vem do fato de que muitas vezes o usuário não sabe de fato o que quer perguntar, o ser humano peca em ser sucinto e objetivo, principalmente quando busca informações sobre algo novo, e que portanto não domina. E como o contexto do uso do RAG, é de fazer perguntas acerca de documentos, é sensato de se pensar que o usuário vai ter uma experiência melhor se nossa pipeline do modelo RAG for mais robusta, para entender até mesmo perguntas mal feitas.

3.2 Dataset

Como vimos na Figura 1, modelos RAG tem sua fase de ingerir documentos, que vamos explicar mais para frente como funciona a fundo, então para testarmos nosso modelo, precisaremos de um documento para fazer essa ingestão, e de perguntas acerca desse documento, perguntas as quais nós já sabemos a resposta, para compararmos com o que vai ser respondido pelo modelo. Faremos então o uso do documento referente ao meu TCC que o tema é “Preenchimento de imagens usando U-Net” e fazer um número controlado de perguntas, as quais já sei a resposta, por conhecer esse documento por completo.

Fig. 2. Arquitetura RAG-Fusion



Fonte:
https://towardsdatascience.com/forget-rag-the-future-is-rag-fusion-1147298d8ad1

3.3 Metodologia de avaliação.

Vou utilizar de uma biblioteca chamada RAGAS, que é especializada na avaliação de desempenho de modelos RAG. Essa biblioteca possui uma função particular de avaliação de modelos, e podemos escolher as métricas que vamos focar a avaliação. Isso será melhor explicado em seções futuras.

4 Implementação

4.1 Modelo RAG

Import e configurações de variável de ambiente: No início do processo de implementação do sistema RAG, o código começa com a importação de bibliotecas e módulos cruciais. Bibliotecas como os.path e openai, juntamente com dotenv, são essenciais para configurar o ambiente de desenvolvimento. Essas importações permitem a manipulação de variáveis de ambiente e a integração com a API da OpenAI, fornecendo a base para interações futuras com o modelo de linguagem. A chave da API OpenAI, por exemplo, é carregada a partir de um arquivo .env, garantindo a segurança e a modularidade do acesso à API.

Leitura e Preparação dos Dados: A leitura e preparação dos dados são realizadas usando o SimpleDirectoryReader, um método do llama_index projetado para ler dados de um diretório local. Esta ferramenta é incrivelmente versátil, suportando uma ampla gama de tipos de arquivos, incluindo documentos em formatos como PDF, JPG, PNG, DOCX, entre outros. Esta flexibilidade é fundamental para construir uma base de dados diversificada e rica, que é um pré-requisito essencial para um sistema RAG eficaz.

Indexação e Embbeding: O processo de indexação e vetorização é realizado através do VectorStoreIndex. Este componente do llama_index é essencial para criar um índice dos dados lidos, armazenando cada pedaço de dado (ou 'nó') e uma correspondente representação vetorial (ou 'embedding'). Estes embeddings são cruciais para permitir que o sistema realize buscas rápidas e eficientes dentro da base de dados.

A indexação é o processo de transformar os dados brutos em uma forma que é fácil de pesquisar e recuperar. Durante este processo, os dados são convertidos em embeddings que representam semanticamente o conteúdo dos dados. Esta representação vetorial é o que possibilita que o sistema RAG localize rapidamente as informações relevantes na base de dados quando necessário.

A vetorização e indexação são fundamentais porque permitem que o sistema RAG recupere informações relevantes de forma eficiente. Esta eficiência é crítica para melhorar a precisão e a relevância das respostas geradas pelo sistema. Sem uma indexação eficaz, o sistema poderia se tornar lento ou fornecer respostas que não são tão precisas ou contextualmente relevantes quanto poderiam ser.

Após a primeira indexação e vetorização dos dados, um aspecto crucial do processo é o armazenamento local do índice criado. Este passo é fundamental, pois permite que o índice seja carregado em usos futuros do sistema, sem a necessidade de repetir todo o processo de vetorização. Essa abordagem é não apenas eficiente em termos de tempo, mas também tem implicações significativas em termos de custos operacionais.

Utilizar repetidamente o modelo de embedding do GPT para vetorizar os dados cada vez que o sistema RAG é utilizado pode ser proibitivamente caro. Modelos de linguagem de grande escala, como os da OpenAI, muitas vezes operam em uma estrutura de precificação baseada no uso, onde cada consulta ao modelo incide um custo. Ao armazenar localmente o índice após a primeira indexação, evita-se a necessidade de realizar consultas frequentes ao modelo de embedding, resultando em economias significativas.

Além disso, a reutilização do índice armazenado melhora a eficiência operacional do sistema. Uma vez que o índice é criado e armazenado, ele pode ser rapidamente carregado e utilizado para a recuperação de informações, agilizando o processo de resposta do sistema RAG. Essa abordagem garante que, mesmo em situações onde o acesso ao modelo de embedding da OpenAI não está disponível ou é restrito, o sistema RAG mantém sua capacidade de funcionar eficientemente.

Integração com LLM: A integração com o modelo de linguagem da OpenAI é realizada através do uso da API da OpenAI. Esta integração é crucial para a geração de linguagem baseada nas informações recuperadas do sistema RAG. O modelo de linguagem, ao receber dados do sistema RAG, gera respostas que são não apenas informativas mas também altamente relevantes ao contexto da consulta.

Este processo de geração de respostas é onde o verdadeiro poder do sistema RAG se manifesta. Utilizando as informações recuperadas da base de dados

indexada, o sistema é capaz de gerar respostas que são enriquecidas pelo contexto e pela especificidade dos dados. Isso não só melhora a qualidade das respostas mas também demonstra a eficácia da combinação de um sistema RAG com um modelo de linguagem avançado como o fornecido pela OpenAI.

5 Modelo RAG-Fusion

O sistema RAG Fusion representa uma evolução significativa da arquitetura RAG tradicional, incorporando avanços tecnológicos e metodológicos que aprimoram tanto o desempenho quanto a capacidade de geração de respostas contextualizadas. Nesta seção, exploramos as características distintivas do RAG Fusion, delineando as inovações que ele traz para o campo da geração de linguagem assistida por recuperação de informações.

5.1 Semelhanças com a arquitetura RAG

A arquitetura RAG-Fusion mantém várias das fundações estabelecidas pela arquitetura RAG tradicional. Ambas utilizam modelos de linguagem de grande escala (LLMs) da OpenAI para a geração de linguagem, empregam o LlamaIndex para indexação e vetorização de dados, e salvam o índice localmente após a primeira vetorização para melhor eficiência e redução de custos.

5.2 Inovações do RAG-Fusion

Geração de queries adicionais: Uma das principais inovações do RAG Fusion é a capacidade de gerar queries adicionais. Isso é implementado no código instânciando uma segunda LLM, que vai servir apenas para analisar a query original do usuário e expandir para outras k queries que englobam contextos mais específicos da pergunta do usuário. Por exemplo, a query do usuário foi "Fale sobre mudanças climáticas", e as queries adicionais geradas seriam "Quais impactos socioeconômicos das mudanças climáticas?", "Quais impactos ambientais das mudanças climáticas?", "O que vem causando mudanças climáticas?" e etc.

Uso do RRF usando multiplas queries: O Reciprocal Retrieval Fusion (RRF) é um componente avançado da arquitetura RAG Fusion, projetado para melhorar a precisão e a contextualização das respostas geradas.

Basicamente o que o RRF faz é calcular a pontuação de cada query (original e novas) em relação ao conteúdo dos documentos ingeridos. E depois que é feita a pontuação de similaridade entre as queries e os nós do documento ingerido, temos um rerankeamento, onde somamos as pontuações de cada query para cada nó, e o output do RRF é basicamente os nós mais relevantes, dados as queries.

É mais fácil de entender o que acontece se olharmos bem a fórmula na Figura 3, e a implementação do RRF em Python na Figura 4.

Fig. 3. RRF

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)},$$

Fig. 4. Implementação Python do RRF

```python
def fuse_results(results_dict, similarity_top_k: int = 2):

    k = 60.0  # k=60 é um hiperparametro testado pelos criadores do algoritmo
              # para controlar impacto de outliers.

    text_to_node = {}

    # Pegamos o resultado da busca vetorial e temos as pontuações de similaridade
    # de cada um dos nós do documento guardados em results_dict.
    for nodes_with_scores in results_dict.values():
        for rank, node_with_score in enumerate(
            sorted(
                nodes_with_scores, key=lambda x: x.score or 0.0, reverse=True
            )
        ):
            text = node_with_score.node.get_content()
            text_to_node[text] = node_with_score
            if text not in fused_scores:
                fused_scores[text] = 0.0
            fused_scores[text] += 1.0 / (rank + k)

    # Ordena do maior para o menor.
    reranked_results = dict(
        sorted(fused_scores.items(), key=lambda x: x[1], reverse=True)
    )

    # Ajusta para ficar typado no formato que o llama index espera.
    reranked_nodes: List[NodeWithScore] = []
    for text, score in reranked_results.items():
        reranked_nodes.append(text_to_node[text])
        reranked_nodes[-1].score = score

    return reranked_nodes[:similarity_top_k]
```

E após o algoritmo RRF retornar os nós ranqueados, nós podemos fazer uma melhor escolha de nó para entregar ao contexto da nossa LLM e produzir a resposta final para o usuário.

6 Avaliação do modelo

6.1 Métricas usadas

Faithfulness (Fidelidade): Esta métrica avalia até que ponto as respostas geradas pelo sistema estão alinhadas com a verdade e os dados factuais. Em outras palavras, mede se as informações fornecidas são precisas e se baseiam em fontes confiáveis. No contexto de um sistema RAG, isso é especialmente importante, pois o sistema recupera informações de uma variedade de fontes antes de gerar uma resposta.

Answer Relevancy (Relevância da Resposta): Esta métrica determina o quão relevante é a resposta fornecida em relação à pergunta feita. Uma resposta pode ser factualmente correta, mas não relevante para a pergunta. Por exemplo, se alguém pergunta "Qual é a capital da França?" e o sistema responde com fatos precisos sobre Paris, mas não afirma explicitamente que Paris é a capital, a resposta pode ser considerada menos relevante.

Context Precision (Precisão de Contexto): Refere-se à capacidade do sistema de identificar e utilizar corretamente o contexto específico da pergunta. Isso é crucial em perguntas onde o contexto muda a natureza da resposta. Por exemplo, a pergunta "Quem é o presidente?" requer que o sistema reconheça o contexto de qual país está sendo perguntado.

Recall: É definido como a proporção de informações relevantes que são efetivamente recuperadas pelo sistema em relação a todas as informações relevantes disponíveis no conjunto de dados ou fonte de informação. Em termos de fórmula, é o número de itens relevantes recuperados dividido pelo número total de itens relevantes existentes.

Harmfulness (Prejudicialidade): Avalia se as respostas do sistema contêm ou promovem conteúdo prejudicial, como desinformação, viés, ou linguagem ofensiva. Esta métrica é especialmente importante para garantir que os sistemas RAG sejam seguros e éticos no fornecimento de informações aos usuários.

6.2 Estruturação dos dados

Como foi dito anteriormente, foi utilizado o documento do meu TCC para teste dos dois modelos, e para testar o desempenho deles, era necessário fazer manualmente perguntas e supostas respostas que o modelo deveria retornar. E a partir disso a função de avaliação da biblioteca RAGAS faria o resto do trabalho, fazendo essas queries ao nosso modelo, e perguntando para uma outra LLM instânciada internamente se as coisas faziam sentido em relação a cada uma das métricas que selecionamos para fazer a avaliação. Estruturei essas perguntas de forma hard-coded, porque eram poucas perguntas, mas se fosse algo mais profissional e bem trabalhado, haveria um arquivo .csv com uma coluna questions e outra coluna answers para estruturar tudo isso da melhor forma possível.

6.3 Discussão dos resultados

Os resultados (Figura 5) da avaliação do modelo RAG Fusion em comparação com o RAG tradicional revelam uma melhoria significativa em várias métricas, incluindo faithfulness (fidelidade), answer relevancy (relevância da resposta), context recall (recordação do contexto) e harmfulness (prejudicialidade). Estas melhorias são consistentes com as expectativas teóricas e práticas em torno das inovações implementadas no RAG Fusion.

Fig. 5. Resultados

	Faithfulness	Relevancy	Precision	Recall	Harmfulness
RAG-Fusion	0.72	0.87	0.8	1.00	0
RAG	0.533	0.76	0.9	1.00	0

Melhoria em Faithfulness e Answer Relevancy: A maior pontuação em faithfulness e answer relevancy no RAG Fusion pode ser atribuída à sua capacidade aprimorada de integração de dados e contexto. O processo iterativo de recuperação de informações, possibilitado pelo Recursive Retrieval Fusion (RRF), permite que o sistema refine a busca por dados relevantes. Isso resulta em respostas que são não apenas mais precisas, mas também mais alinhadas com o contexto da consulta, aumentando a fidelidade e relevância das respostas.

Melhoria em Recall: O aumento na pontuação de context recall no RAG Fusion pode ser justificado pela sua habilidade de gerar queries adicionais e processar múltiplas rodadas de recuperação. Isso ajuda o sistema a capturar uma gama mais ampla de informações pertinentes, melhorando a recordação do contexto.

Harmfullness: O valor de 0 prejudicialidade observada tanto no RAG Fusion quanto no RAG, sugere que os modelos filtram ou evitam a geração de respostas potencialmente nocivas ou tendenciosas. Isso pode ser devido à capacidade do modelo de analisar e sintetizar informações de uma variedade mais ampla de fontes, permitindo uma avaliação mais equilibrada e diversificada das informações.

Precision: Apesar das melhorias em várias métricas, o RAG Fusion apresentou uma pontuação menor em precision (precisão) em comparação com o RAG tradicional. Há várias razões potenciais para isso, que podem ser exploradas para entender esse resultado. O RAG Fusion, com sua capacidade de processar e integrar uma quantidade maior de informações devido ao RRF e à geração de queries adicionais, pode às vezes incorporar informações mais amplas nas respostas. Enquanto isso é benéfico para a relevância e o contexto, pode, paradoxalmente, diluir a precisão focada de uma resposta específica. Isso pode ser particularmente evidente em um conjunto de dados de teste menor, onde variações na relevância dos dados podem impactar mais significativamente a precisão. Outro fator a considerar é o equilíbrio intrínseco entre precisão e abrangência. Enquanto o RAG Fusion é otimizado para abranger uma gama mais ampla de informações

e contexto, isso pode às vezes ocorrer à custa da precisão pontual. Em outras palavras, ao tentar cobrir um espectro mais amplo de informações relevantes, o modelo pode ocasionalmente incluir detalhes que, embora relacionados, não são estritamente necessários para a precisão da resposta específica

7 Conclusão

A comparação entre o sistema RAG Fusion e o RAG tradicional, conforme detalhado neste estudo, demonstra claramente o sucesso e a eficácia do RAG Fusion em várias métricas-chave de desempenho. Esta avaliação forneceu insights valiosos e confirmou a superioridade do RAG Fusion em termos de fidelidade, relevância da resposta, recordação do contexto e redução da prejudicialidade, apesar de uma ligeira redução na precisão pontual. O estudo foi uma experiência enriquecedora e significativa, contribuindo de forma substancial para um melhor entendimento dos sistemas de Geração Aumentada por Recuperação.

A implementação do RAG Fusion, com suas inovações como o Reciprocal Retrieval Fusion e a geração de queries adicionais, provou ser um avanço significativo em relação ao modelo tradicional. A habilidade de iterar na recuperação de informações e de sintetizar respostas mais contextualizadas e abrangentes demonstra um progresso notável na aplicação de modelos de linguagem de grande escala em tarefas de processamento de linguagem natural.

Este estudo não só testou e validou a eficácia do RAG Fusion, mas também forneceu uma compreensão mais profunda de como diferentes abordagens de recuperação de informações podem influenciar a geração de linguagem. A experiência acumulada neste processo é inestimável, contribuindo para o campo em crescimento da inteligência artificial e fornecendo direções claras para futuras pesquisas e desenvolvimentos.

Além de suas aplicações diretas em IA, o RAG Fusion emerge como uma ferramenta incrível para processos de aprendizado e descoberta. Sua capacidade de analisar, sintetizar e gerar informações de maneira contextual e precisa o torna ideal para aplicações educacionais e de pesquisa, onde a compreensão profunda e a precisão da informação são cruciais.

Com base nos resultados obtidos e nas lições aprendidas, há um caminho claro para a continuação da pesquisa e do aprimoramento do RAG Fusion. Isso inclui a exploração de estratégias para equilibrar ainda mais a precisão com a abrangência e o desenvolvimento de métodos para otimizar a eficiência em conjuntos de dados de diferentes escalas. O potencial de aplicação do RAG Fusion em diversos campos abre um leque de possibilidades para futuras inovações e descobertas.

References

1. Forget RAG, the Future is RAG-Fusion , Adrian H. Raudaschl, https://towardsdatascience.com/forget-rag-the-future-is-rag-fusion-1147298d8ad1

2. Building Your Own DevSecOps Knowledge Base with OpenAI, LangChain, and LlamaIndex, Wenqi Glantz, https://betterprogramming.pub/building-your-own-devsecops-knowledge-base-with-openai-langchain-and-llamaindex-b28cda15abb7
3. Relevance scoring in hybrid search using Reciprocal Rank Fusion (RRF), Janusz Lembicz, https://learn.microsoft.com/en-us/azure/search/hybrid-search-ranking