Nome: Fernando Buligon Antunes **Data:** 02/06/2025

Introdução ao RAG

No início é contado uma história, em que há um jornalista que deseja escrever um artigo sobre um tópico que ele já tem um bom conhecimento geral sobre, mas como ele quer fazer mais completo ele vai até a biblioteca local onde há vários livros contendo o que ele procura, mas o jornalista não possui o conhecimento de como encontrar esses livros, então ele recorre ao bibliotecário, então o fluxo de trabalho ficaria jornalista pergunta ao bibliotecário > bibliotecário busca o que foi pedido > bibliotecário retorna os resultados. A junção dessas duas personalidades é poderosa, um sabe escrever mas não possui o conteúdo necessário e o outro possui o conteúdo mas não sabe escrever. Essa história foi feita como uma analogia ao processo do RAG (Retrieval Augmented Generation), onde grandes modelos de linguagens fazem a chamada de vetores de banco de dados para obter conhecimentos chaves para se chegar a resposta.

Para resolver certas questões, apenas a base de dados de treinamento das LLMs não são suficientes, por isso, existem os vetores de bancos de dados, que consistem em uma representação matemática de dados estruturados e não estruturados, podendo conter todo tipo de conteúdo, desde PDFs, imagens, outras aplicações e etc.

LangChain - RAG

Na introdução, é falada sobre uma das principais importâncias e utilidades do RAG, que é a possibilidade de você fazer uso de seus dados proprietários junto de uma LLM para responder perguntas, a relevância disso se baseia no seguinte ponto, nem todos os dados podem ser enviados para a API de alguma LLM, um dos exemplos mais comuns são dados sensíveis ou sigilosos como documentos. Então o RAG permite fazer a junção do contexto com a LLM, tornando possível extrair ao máximo a capacidade da LLM naquele momento.

Depois fomos para a parte prática, em que, novamente no tutorial é usada a API da OpenAI, que nos dias atuais não libera créditos para uso, ainda que o propósito não seja comercial. Optei por usar a API do Ollama, que não é tão boa quanto a da OpenAI, mas é gratuita.

O código base é bem semelhante ao que foi feito no card nove, a diferença é que agora além do conteúdo que a LLM já possuía, também foi adicionado novos dados, nesse caso, o conteúdo da página do filme Oppenheimer da WikiPédia, para isso, o texto foi transformado em embeddings, separado em partes menores e depois armazenados em um banco de dados vetorial.

Após ter a LLM e os dados externos armazenados em um banco de dados vetorial, usando RAG foi possível fazer a LLM responder tendo como contexto principal a página da WikiPédia que foi escolhida.

Retrieval - Augmented Generation

No artigo é contado que um dos principais problemas nos modelos de linguagem natural pré treinados é em relação aos dados, especificamente, os dados que não estão

presentes na base de dados. A falta desses dados provoca sérios problemas, como por exemplo, alucinações. Resolver essa questão é um dos maiores desafios dentro das LLMs, afinal, ficar atualizando os dados em tempo real não é uma tarefa fácil. Uma das tentativas que mais demonstraram serem capazes de contornar esse problema foi modelos que usam RAG.

O RAG, como já explicado nos pontos anteriores, permite com que o modelo acesse dados externos a base de dados, para que o modelo use como contexto base para formar suas respostas. Existem dois modelos de RAG distintos.

O de sequência, que basicamente usa os documentos disponíveis como base para gerar toda a sequência da saída. Esse modelo funciona da seguinte forma: primeiro, são recuperados os documentos mais relevantes com base na pergunta feita, segundo, o modelo gera uma resposta para cada documento, considerando apenas o contexto do documento da vez, e por fim, é feita uma média das respostas, levando em consideração todos os documentos, essa combinação resulta na resposta final.

O de token, a resposta é gerada token por token, para a escolha de cada token são consideradas todos os documentos, então a resposta final é feita com base em todos simultaneamente, contendo apenas o que o modelo calcula que mais tem probabilidade de estar correto.

Depois também foram apresentados os componentes de generator (BART) e o retriever (DPR). Esses dois componentes são usados em conjunto para a fase do treinamento sem nenhuma supervisão direta, o modelo vai aprendendo sozinho. Após isso foram apresentados os experimentos feitos, incluindo responder perguntas de domínios abertos, perguntas abstratas, perguntas no formato jeopardy (exemplo, “a copa do mundo” é resposta para “em 1986 o México bateu o marco de primeiro país a hospedar esse campeonato mundial duas vezes”) e verificação de fatos. E em todas, o modelo obteve ótimos resultados.

Na parte de trabalhos relacionados, o artigo mostra que a ideia de usar recuperação de informações já tinha sido aplicada com sucesso em várias tarefas de NLP, mas sempre de forma isolada. Isso inclui tarefas como perguntas e respostas abertas, verificação de fatos, tradução, geração de textos longos, criação de artigos da Wikipédia, diálogos e modelagem de linguagem. A contribuição do RAG é juntar essas ideias em uma arquitetura única e mais geral, que funciona bem em diferentes cenários. Na parte final, é reforçado um pouco as ideias gerais do RAG e porque é tão importante.