Implementação de Chatbot RAG para o Vestibular Unicamp 2025

Gabriel Sudo Enoki

1. Introdução

Este trabalho teve como objetivo a implementação de um chatbot baseado em Retrieval-Augmented Generation (RAG) para responder perguntas sobre o Vestibular Unicamp 2025. A solução envolveu o uso de diversas bibliotecas para processamento de texto, recuperação semântica e geração de respostas. Além disso, utilizou-se modelos de linguagem específicos para criar um sistema de perguntas e respostas eficiente. O foco principal foi integrar a recuperação de documentos com modelos generativos, a fim de melhorar a precisão e a relevância das respostas.

2. Bibliotecas Utilizadas e Suas Funções

Para a implementação do sistema, foram selecionadas diversas bibliotecas que facilitam a criação de embeddings, a recuperação de documentos e a geração de respostas. A biblioteca langchain_community foi fundamental para estruturar a pipeline de recuperação e geração. A classe HuggingFaceEmbeddings foi usada para gerar os embeddings de texto a partir do modelo sentence-transformers/paraphrase-multilingual-mpnet-base-v2, que é otimizado para tarefas de similaridade semântica, permitindo que o sistema compreenda e recupere documentos em múltiplos idiomas. A partir dos embeddings gerados, foi possível construir um banco de dados vetorial utilizando a ferramenta FAISS (Facebook AI Similarity Search). O FAISS possibilita a indexação dos embeddings e a busca eficiente por documentos relevantes, o que é essencial para a operação do chatbot. Para a divisão do texto em partes menores, foi utilizada a ferramenta RecursiveCharacterTextSplitter da biblioteca langchain.text_splitter. Essa ferramenta foi configurada para dividir o conteúdo do arquivo em chunks de 3000 caracteres, com uma sobreposição de 200 caracteres, garantindo que cada pedaço de texto mantivesse o contexto e fosse recuperável de maneira eficiente durante as consultas.

A parte de geração de respostas foi tratada pela API ChatGroq, que utiliza o modelo de linguagem llama3-8b-8192. Esse modelo foi configurado com uma temperatura de 0.7, permitindo que o chatbot gerasse respostas equilibradas, combinando coerência e criatividade nas respostas. O ChatGroq foi escolhido devido à sua capacidade de processamento rápido e de gerar respostas precisas em tempo real, aproveitando o hardware especializado da Groq. Para calcular a similaridade entre as respostas do chatbot e as respostas esperadas, foi utilizada a biblioteca sentence_transformers, com o modelo paraphrase-multilingual-mpnet-base-v2, que gerou os embeddings necessários para calcular a similaridade semântica utilizando a métrica de cosseno.

Essas bibliotecas e ferramentas foram escolhidas de forma estratégica para integrar todas as etapas do processo, desde a leitura do conteúdo até a geração das respostas, permitindo que o chatbot operasse de maneira eficiente e precisa em um sistema de RAG.

3. O Chatbot e a API Escolhida

O ChatGroq foi escolhido como modelo de linguagem devido à sua eficiência e capacidade de gerar respostas precisas rapidamente, especialmente em sistemas de RAG. O modelo llama3-8b-8192 foi utilizado, configurado com uma temperatura de 0.7, o que garante um equilíbrio entre respostas criativas e coerentes. O modelo se destaca por sua capacidade de processar rapidamente grandes volumes de dados, sendo ideal para tarefas de recuperação de documentos e geração de respostas em tempo real. A API FAISS foi usada para indexar os embeddings gerados a partir dos chunks de texto, o que permitiu a realização de buscas rápidas e precisas nas informações armazenadas.

A escolha do ChatGroq também se deu pela sua integração com o hardware especializado da Groq, o que proporciona um desempenho mais eficiente em tarefas de processamento pesado. A utilização dessa API garantiu a alta velocidade na geração das respostas, mantendo a qualidade necessária para um sistema de perguntas e respostas.

4. Desenvolvimento do Código

O código foi desenvolvido para construir um sistema de RAG, onde a recuperação de documentos relevantes e a geração de respostas acontecem de forma integrada. O fluxo de processamento é o seguinte:

Leitura e Processamento do Texto: O arquivo "normas_unicamp_noLines.txt" foi lido e o conteúdo foi dividido em chunks menores utilizando o RecursiveCharacterTextSplitter. Esses chunks foram segmentados de forma a preservar o contexto das informações, facilitando a busca por dados relevantes.

Criação de Embeddings e Indexação: Os chunks de texto foram transformados em embeddings vetoriais usando o modelo paraphrase-multilingual-mpnet-base-v2. Em seguida, os embeddings foram indexados utilizando a ferramenta FAISS para garantir a recuperação rápida e eficiente durante as consultas ao chatbot.

Integração entre Recuperação e Geração: A integração entre a recuperação de documentos e a geração de respostas foi feita por meio da classe RetrievalQA. Esta classe permite que o modelo ChatGroq utilize as informações recuperadas do banco de dados FAISS para gerar respostas precisas e contextualizadas.

Interação com o Chatbot: O código foi estruturado para permitir que o usuário faça perguntas sobre o Vestibular Unicamp 2025, com respostas geradas com base no conteúdo extraído do arquivo de texto. O sistema foi configurado para funcionar em um loop contínuo de interações.

5. Uso do ChatGPT para Criar Datasets de Teste

O ChatGPT foi utilizado para criar datasets de teste para avaliar o desempenho do chatbot. O processo envolveu a geração de perguntas sobre o Vestibular Unicamp 2025, com base em tópicos relevantes extraídos do arquivo de texto, como datas de inscrição, valores das taxas, número de vagas e requisitos de cotas. O ChatGPT gerou as perguntas, e as

respostas esperadas foram formuladas de acordo com as informações exatas presentes no conteúdo do arquivo.

Além de ajudar na criação das perguntas e respostas, o ChatGPT também foi utilizado para organizar o conteúdo de forma a garantir que o chatbot fosse testado com dados reais e relevantes. Essa colaboração foi fundamental para a criação de um conjunto de testes realistas, que permitiram avaliar a precisão do chatbot durante a interação com os usuários.

6. Uso do ChatGPT para Verificar e Formatá-lo

O ChatGPT foi essencial para verificar a estrutura do arquivo de texto. O modelo ajudou a garantir que o conteúdo estivesse organizado de maneira adequada para ser utilizado em um sistema de RAG. A partir da análise feita pelo ChatGPT, foi possível identificar a necessidade de ajustes no formato do texto, como a divisão de parágrafos longos e a inclusão de subtítulos claros, o que facilitou a recuperação semântica. Além disso, o modelo sugeriu melhorias para garantir que as informações no texto estivessem organizadas de forma que pudessem ser facilmente recuperadas durante as consultas.

7. Resultados dos Testes

Os testes realizados no chatbot mostraram uma acurácia entre 54% e 60%. Embora o chatbot tenha fornecido respostas geralmente corretas, a acurácia foi impactada pela geração de informações adicionais que não estavam presentes nas respostas esperadas. Durante os testes, observou-se que o modelo frequentemente incluía detalhes ou palavras extras, o que afetou a métrica de similaridade semântica.

Por exemplo, ao responder à pergunta "Qual é o valor da taxa de inscrição?", a resposta do chatbot era correta, mas frequentemente incluía informações como datas de inscrição ou detalhes adicionais, que não faziam parte da resposta esperada. Como resultado, as pontuações de similaridade semântica entre a resposta do chatbot e a resposta esperada ficaram abaixo do esperado.

Para melhorar a acurácia, seria necessário ajustar o modelo para evitar a geração de informações irrelevantes e tornar as respostas mais concisas e diretamente relacionadas à pergunta.

8. Conclusão

A implementação do chatbot RAG utilizando ChatGroq e FAISS foi bem-sucedida, mas apresentou desafios relacionados à precisão das respostas. O modelo gerou respostas corretas na maioria das vezes, mas a inclusão de informações adicionais impactou a acurácia. A utilização do ChatGPT foi crucial tanto para a geração de datasets de teste quanto para a verificação da estrutura do texto, garantindo que o conteúdo estivesse bem formatado para o uso em um sistema de RAG. O sistema tem grande potencial para melhorias, especialmente no ajuste da geração de respostas para evitar a inclusão de detalhes irrelevantes, o que poderia aumentar a precisão e a qualidade das respostas fornecidas.