UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Mateus Moro Torres

Utilizando a LLM GPT-3.5 Turbo para o desenvolvimento de uma ferramenta de busca de materiais por características técnicas

Florianópolis
2023

Mateus Moro Torres

Utilizando a LLM GPT-3.5 Turbo para o desenvolvimento de uma ferramenta de busca de materiais por características técnicas

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.
Orientador: Prof. Ricardo José Rabelo
Supervisor: Jairo Luis Eichendorf

Florianópolis
2023

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Torres, Mateus Moro
Utilizando a LLM GPT-3.5 Turbo para o desenvolvimento de uma ferramenta de busca de materiais por características técnicas / Mateus Moro Torres ; orientador, Ricardo José Rabelo, 2024.
69 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia de Controle e Automação, Florianópolis, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. IA Generativa. 3. Modelos de Linguagem. 4. Busca semântica. 5. RAG. I. Rabelo, Ricardo José . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Mateus Moro Torres

Utilizando a LLM GPT-3.5 Turbo para o desenvolvimento de uma ferramenta de busca de materiais por características técnicas

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis,13 de Dezembro de 2023.

Prof. Marcelo de Lellis Costa de Oliveira
Coordenador do Curso

Banca Examinadora:

Prof. Ricardo José Rabelo
Orientador
UFSC/CTC/DAS

Jairo Luis Eichendorf
Supervisor na Empresa WEG

Gustavo Claudio Karl Couto
Avaliador
UFSC/CTC/DAS

Prof. Eduardo Camponogara
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus colegas de turma e
aos meus queridos pais.

AGRADECIMENTOS

Agradeço à Deus, aos meus pais, à minha família e aos amigos que encontrei durante essa jornada que levarei para sempre comigo. Agradeço também à WEG pela oportunidade e aos meus colegas de trabalho que tornaram isso possível.

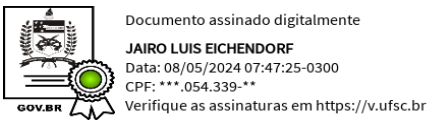
“You have brains in your head.
You have feet in your shoes.
You can steer yourself any direction you choose.”
(Seuss Theodor, 1991)

DECLARAÇÃO DE PUBLICIDADE

Florianópolis, 13 de Dezembro de 2023.

Na condição de representante da WEG na qual o presente trabalho foi realizado, declaro não haver ressalvas quanto ao aspecto de sigilo ou propriedade intelectual sobre as informações contidas neste documento, que impeçam a sua publicação por parte da Universidade Federal de Santa Catarina (UFSC) para acesso pelo público em geral, incluindo a sua disponibilização *online* no Repositório Institucional da Biblioteca Universitária da UFSC. Além disso, declaro ciência de que o Mateus Moro Torres, na condição de estudante da UFSC, é obrigado a depositar este documento, por se tratar de um Trabalho de Conclusão de Curso, no referido Repositório Institucional, em atendimento à Resolução Normativa n° 126/2019/CUn.

Por estar de acordo com esses termos, subscrevo-me abaixo.



Jairo Luis Eichendorf
WEG

RESUMO

Com a ascensão do ChatGPT e a crescente incorporação de tecnologias baseadas em modelos de linguagem na sociedade, como chatbots que simulam conversas com computadores, surgem diversas aplicações que se baseiam em modelos de linguagem grande, conhecidos como LLMs (Large Language Models). Esses modelos têm uma ampla gama de aplicações, uma vez que suas arquiteturas permitem a compreensão e utilização da linguagem natural para diversas finalidades. O presente trabalho tem como objetivo desenvolver um sistema de busca de materiais que utiliza esses modelos como sistemas de recomendação para os produtos e materiais utilizados pela empresa WEG. Com sua extensa linha de produtos, a WEG enfrenta o desafio de gerenciar milhões de materiais que seus funcionários precisam localizar nos sistemas internos da empresa, além de atender às necessidades de seus clientes que buscam produtos em seu site de comércio eletrônico. A combinação desses modelos de linguagem com técnicas de busca vetorial empregadas no processamento de linguagem natural está transformando a maneira como dados e informações são recuperados de diversas fontes. Nesse contexto, este trabalho se propõe a explorar o uso de modelos de linguagem e busca semântica para criar um sistema de busca inteligente e eficaz.

Palavras-chave: IA Generativa. Busca semântica. Modelos de linguagem.

ABSTRACT

With the rise of ChatGPT and the increasing integration of language model-based technologies in society, such as chatbots simulating conversations with computers, various applications relying on large language models (LLMs) have emerged. These models have a wide range of applications, as their architectures enable the understanding and utilization of natural language for various purposes. This work aims to develop a materials search system that utilizes these models as recommendation systems for the products and materials used by the company WEG. With its extensive product line, WEG faces the challenge of managing millions of materials that its employees need to locate within the company's internal systems, as well as meeting the needs of its customers searching for products on its e-commerce website. The combination of these language models with vector search techniques employed in natural language processing is transforming the way data and information are retrieved from various sources. In this context, this work seeks to explore the use of language models and semantic search to create an intelligent and effective search system.

Keywords: Generative IA. Semantic Search. Language Models

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	Approximate Nearest Neighbors
API	*Application Programming Interface*
BART	*Bidirectional and Auto-Regressive Transformers*
BERT	*Bidirectional Encoder Representations from Transformers*
CRISP-DM	Cross-Industry Standard Process for Data Mining
ERP	*Enterprise Resource Planning*
GPT	*Generative Pre-Trained Transformer*
GPT-3	*Generative Pre-Trained Transformer 3*
GPU	Graphics Processing Unit
IA	*Inteligência Artificial*
IoT	Internet das Coisas
LGPD	Lei Geral de Proteção de Dados Pessoais
LLM	*Large Language Model*
LLMs	*Large Language Models*
NLP	*Natural Language Processing*
PLN	*Processamento de Linguagem Natural*
PNL	*Processamento de Linguagem Natural*
RAG	*Retrieval-Augumented Generation*
RNN	Recurrent Neural Network
T5	*Text-to-Text transfer Transformer*
TF-IDF	Term Frequency–Inverse Document Frequency
TI	*Tecnologia da Informação*
UML	Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos ou até meses, o uso de modelos de linguagem se tem mostrado presente cada vez mais no nosso dia a dia. Desde o advento do *ChatGPT*, *chatbot* movido por inteligência artificial desenvolvido pela empresa *OpenAI*, tem aparecido inúmeras soluções que usam essa tecnologia revolucionária para ajudar pessoas e alavancar negócios usando o poder da inteligência artificial. A medida que a quantidade de dados disponíveis na internet e em repositórios digitais continua a crescer exponencialmente, avanços tecnológicos desse nível assumiram um papel fundamental em nossa sociedade

A tecnologia por atrás do *ChatGPT* é o uso de Modelos de Linguagem (*Large Language Models* (LLMs)). Tais modelos representam uma transformação significativa na forma como os computadores conseguem compreender e processar linguagem natural. Os LLMs, alimentados por avanços em *Deep Learning* e grandes volumes de dados de treinamento, permitem que as máquinas entendam o contexto, a semântica e a intenção por trás das consultas dos usuários, proporcionando resultados mais precisos e relevantes.

1.1 OBJETIVOS

Este trabalho tem como objetivo a implementação do *Large Language Model* (LLM) *GPT*-3.5 Turbo na estruturação de uma ferramenta de busca de materiais na empresa WEG. A crescente complexidade gerada pelo acúmulo exponencial de dados torna imperativa a necessidade de sistemas capazes de retornar resultados precisos e pertinentes em consultas de busca. Neste âmbito, é essencial que o sistema desenvolvido reconheça termos coloquiais ou equivalentes aos termos técnicos empregados nos produtos e materiais. A intenção é proporcionar aos usuários a facilidade de localizar itens desejados empregando uma linguagem acessível e descritiva, dispensando o uso de nomenclaturas técnicas específicas.

Um dos principais softwares utilizados pela empresa para gestão interna é o *Enterprise Resource Planning* (ERP) WEG, um sistema integrado de gestão empresarial que oferece uma gama extensa de funcionalidades. Entre estas, encontra-se a administração de estoque e um sistema de busca de materiais. Atualmente, esse sistema apresenta limitações em termos de funcionalidade, exigindo do usuário um conhecimento prévio específico do material desejado e um método de busca baseado em critérios fixos com vários campos para serem preenchidos, isso dificulta a busca por quem não tem todas essas informações disponíveis. Outro sistema importante na WEG é a plataforma de *e-commerce*, que, embora seja um canal onde os clientes têm acesso ao catálogo de produtos, não dispõe de um mecanismo de busca eficiente, não é possível encontrar um produto descrevendo-o por suas características técnicas, além

de que, quando se pesquisa um produto, é retornado milhares de resultados, muitas vezes não relevantes para a pesquisa.

Este trabalho, portanto, dada a identificação das limitações no sistema de gestão interna e *e-commerce* da WEG, especificamente no que se refere à busca de produtos e materiais, propõe desenvolver uma ferramenta de busca mais inteligente. Isso porque será utilizada tecnologias de processamento de *Processamento de Linguagem Natural* (PNL), oferecendo recursos como busca semântica e um *autocomplete*. O objetivo é facilitar a localização de produtos e materiais, tanto para usuários internos da empresa quanto para os clientes, por meio de um método mais intuitivo e eficaz. A inovação proposta visa superar as dificuldades enfrentadas com os métodos de busca baseados em critérios fixos e o preenchimento de múltiplos campos, permitindo buscas eficientes mesmo sem conhecimento prévio específico sobre os itens desejados.

1.2 ESTRUTURA DO DOCUMENTO

Este documento está dividido em sete capítulos. No capítulo 2, é feita uma contextualização da empresa, então introduz-se o que são os modelos de linguagem e a busca semântica de materiais. No capítulo 3, apresenta-se a fundamentação teórica, levantada a partir de estudos, conduzidos para aquisição do conhecimento necessário para efetuar o desenvolvimento. No capítulo 4, são apresentadas a metodologia de desenvolvimento empregada e especificação de requisitos. No capítulo 5, é apresentada a implementação efetiva da aplicação, relatando as diversas etapas do processo de desenvolvimento, com os resultados sendo apresentados no capítulo 6. Por fim, o capítulo 7 traz as conclusões finais e sugestões de tópicos a serem investigados em trabalhos futuros.

2 CONTEXTUALIZAÇÃO

2.1 A EMPRESA

A WEG, cuja sigla deriva dos nomes de seus fundadores Werner, Eggon e Geraldo, é uma empresa com sede em Jaraguá do Sul, Santa Catarina, e se destaca como uma das maiores fabricantes de equipamentos elétricos globalmente. Foi fundada em 16 de Setembro de 1961 com o empenho e habilidades dos seus três fundadores, um eletricista, um administrador e um mecânico. Seu foco principal reside na produção de motores elétricos, embora também ofereça uma vasta gama de produtos, como geradores, transformadores, disjuntores, tintas e vernizes, entre outros.

Figura 1 – Primeira sede da WEG em Jaraguá do Sul.

Fonte (WEG, 2023b)

A empresa tem como propósito "Desenvolver tecnologias e soluções para contribuir para a construção de um mundo mais eficiente e sustentável"(WEG, 2022a). A notável trajetória de sucesso da WEG é corroborada por alguns indicadores recentes; em 2022, a empresa alcançou um faturamento de R$ 29,9 bilhões e possui filiais em 37 países, além de fábricas distribuídas por 15 nações, assegurando sua presença em cinco continentes. Com um portfólio abrangendo mais de 1500 linhas de produtos e uma equipe de mais de 39.000 colaboradores, incluindo 4.300 engenheiros, a WEG se mantém como uma figura proeminente no mercado global (WEG, 2022b).

A WEG está em uma evolução contínua, com 59% de seus faturamentos originados pelo desenvolvimento de produtos nos últimos cinco anos. Um destaque é o WEG Digital, um ecossistema que integra equipamentos e sensores, incorporando conceitos cruciais para a industrial, tais como plataforma Internet das Coisas (IoT),

conectividade avançada, inteligência artificial e softwares alinhados à Indústria 4.0. Essa adaptabilidade é ainda evidenciada pelas aquisições recentes de *startups*, como a MVisia e a BirminD (WEG, 2023). Adicionalmente, a WEG fornece equipamentos para renomadas empresas globais, incluindo a *SpaceX* de Elon Musk e a Petrobras, a estatal brasileira de petróleo e gás. Com esta última, a WEG selou uma parceria para criar o maior aerogerador eólico do Brasil (FIESC, 2023).

2.2 MOTIVAÇÃO DO PROJETO

A presente seção visa elucidar as razões que fundamentam a iniciativa deste projeto. Ele surgiu de um funil de ideias de inovação com e teve como proposta aprimorar o processo de busca de materiais em sistemas da WEG. Um exemplo emblemático é o mecanismo de busca atualmente implementado pela WEG na plataforma de ERP. Este sistema exige a inserção de múltiplos dados, abrangendo desde o centro de material até o tipo de material, incluindo características específicas e código de idioma. Sua complexidade é tal que indivíduos sem treinamento regular e conhecimento especializado dos materiais e do ambiente ERP enfrentam dificuldades significativas para sua utilização eficaz.

Além disso, o atual sistema de busca de produtos dentro do *e-commerce* da WEG conta com um sistema de busca simples, não devolve os resultados pertinentes para uma pesquisa como no do caso da figura 2. Quando foi pesquisado "motor com 1/3cv com grau proteção ip21", o sistema retornou alguns conjuntos de proteções e motores para bomba de 0,5cv, além de mais de 22 mil produtos. Logo, ele não é preciso e não entende os termos utilizados, serve perfeitamente como busca quando já se tem o código do produto a ser pesquisado no catálogo.

O resultado correto da busca da figura 2 seria alguns motores com 0,33cv de potência e grau de proteção ip21, um possível resultado seria o da figura 3.

2.2.1 Consequências do *status* atual

A presença de um sistema de busca ineficaz em uma empresa como a WEG pode ter implicações negativas substanciais em várias dimensões do negócio. Primeiramente, tal ineficiência pode levar a uma redução significativa da produtividade, uma vez que os funcionários podem gastar um tempo excessivo na localização de informações necessárias, resultando em atrasos e frustração. Essa dificuldade na busca de informações pode também impactar negativamente a tomada de decisão, comprometendo a qualidade e a rapidez das decisões de negócios, já que dados relevantes podem não estar prontamente acessíveis ou podem ser imprecisos.

Do ponto de vista do cliente, especialmente no ambiente de *e-commerce*, um sistema de busca ineficaz pode deteriorar a experiência do usuário, dificultando a

Figura 2 – Sistema de busca dentro do *e-commerce* da WEG.

Fonte- WEG

Figura 3 – Motor com 0.33 cv com grau de proteção ip21.

Fonte- WEG

localização de produtos ou informações desejadas. Isso pode resultar em menores taxas de conversão, perda de vendas e potencial dano à imagem da marca. Além disso, desafios significativos podem surgir na gestão de estoque, em que a ineficiência na busca pode levar a problemas como excesso de estoque, escassez ou pedidos duplicados, aumentando os custos operacionais.

A longo prazo, um sistema de busca ineficaz pode aumentar os custos operacionais e necessitar de treinamento adicional ou suporte, além de colocar a empresa em desvantagem competitiva em um mercado que valoriza a rapidez e precisão na

obtenção de informações. Também podem existir riscos relacionados à segurança e privacidade dos dados. Por fim, a ineficácia no sistema de busca pode limitar a capacidade da empresa de inovar e crescer, restringindo sua resposta às mudanças do mercado e às oportunidades de desenvolvimento.

Portanto, a eficiência do sistema de busca é crucial não apenas para a operação diária, mas também para a estratégia global e o sucesso a longo prazo de uma empresa.

3 FUNDAMENTAÇÃO TEÓRICA

O propósito deste capítulo é fornecer um panorama das teorias e princípios subjacentes ao sistema de busca concebido neste trabalho. Será realizada a elucidação dos termos técnicos e conceitos-chave que constituem a base do projeto, garantindo uma fundamentação teórica para compreensão das decisões na execução do projeto. Esta explanação é essencial não apenas para a contextualização do desenvolvimento sistemático, mas também para assegurar um alinhamento com as tecnologias atuais no campo da inteligência artificial, modelos de linguagem e recuperação de informações.

3.1 INTRODUÇÃO AOS SISTEMA DE BUSCA TRADICIONAIS

Dada a era da informação e da imensidão de dados que se tem atualmente, os sistemas de busca tornaram-se componentes indispensáveis para navegar pelo vasto mar de informações disponíveis. Com o aumento exponencial do volume de dados gerados, armazenados e transmitidos, a capacidade de localizar informações relevantes de forma eficaz e eficiente tornou-se uma necessidade crítica. O valor da informação, nesse contexto, é diretamente proporcional à sua acessibilidade e relevância para tarefas específicas, sejam elas acadêmicas, empresariais ou pessoais (DAVENPORT; PRUSAK, 1998).

Tradicionalmente, a busca de informações estava limitada a bibliotecas e arquivos físicos. No entanto, com a digitalização e a crescente conectividade proporcionada pela Internet, os sistemas de busca evoluíram significativamente. Hoje, esses sistemas abrangem não apenas os motores de busca na web, como o Google e o Bing, mas também sistemas especializados em domínios como saúde, direito, manufatura e gestão de produtos e materiais (BAEZA-YATES; RIBEIRO-NETO, 2011).

Essa evolução foi acompanhada por avanços significativos em algoritmos e técnicas de recuperação de informações. Além dos modelos booleanos simples, métodos mais sofisticados, como o Modelo de Espaço Vetorial, TF-IDF e algoritmos baseados em aprendizado de máquina, como *Learning to Rank*, são amplamente utilizados (MANNING; RAGHAVAN; SCHÜTZE, 2008).

1. O Modelo Booleano é um dos métodos mais antigos e fundamentais em sistemas de recuperação de informação. Baseia-se na álgebra booleana e utiliza operadores lógicos como *AND*, *OR* e *NOT* para combinar termos de busca. Apesar de sua simplicidade e precisão para consultas específicas, este modelo carece de flexibilidade e não considera a relevância gradativa dos documentos.

2. O Modelo de Espaço Vetorial representa tanto documentos quanto consultas como vetores em um espaço multidimensional. A relevância de um documento é determinada pela similaridade do cosseno entre os vetores do documento e da

consulta. Este método oferece uma abordagem mais flexível e eficaz, permitindo a classificação dos documentos com base na similaridade com a consulta.

3. Term Frequency–Inverse Document Frequency (TF-IDF) é uma medida estatística usada para avaliar a importância de uma palavra em um documento dentro de um corpus. Esta técnica considera a frequência do termo no documento e a frequência inversa do documento no corpus, fornecendo uma ponderação que destaca termos importantes que são menos comuns no corpus como um todo.

4. *Learning to Rank* aplica técnicas de aprendizado de máquina ao problema de ordenação de documentos em sistemas de recuperação de informação. Esta abordagem utiliza algoritmos de aprendizado de máquina para criar um modelo que prediz a relevância dos documentos com base em suas características. É especialmente útil em motores de busca na web, onde a ordenação precisa dos resultados é fundamental.

Estes métodos representam etapas fundamentais no desenvolvimento de técnicas de busca e indexação de informações. No entanto, estes modelos tradicionais tendem a não capturar sinônimos ou nuances de linguagem, e frequentemente não conseguem entender a intenção ou o contexto mais amplo por trás das consultas dos usuários, limitando assim sua eficácia em situações onde a correspondência exata de palavras-chave não é suficiente para satisfazer as necessidades de busca do usuário.

3.2 *CRISP-DM*

O *CRISP-DM* é um modelo de processo amplamente aceito que oferece uma abordagem estruturada para projetos de mineração de dados. Desenvolvido em 1996 por um consórcio de empresas, incluindo SPSS, NCR Corporation e Daimler-Benz, *CRISP-DM* tornou-se um padrão de fato na indústria de análise de dados (CHAPMAN *et al.*, 2000).

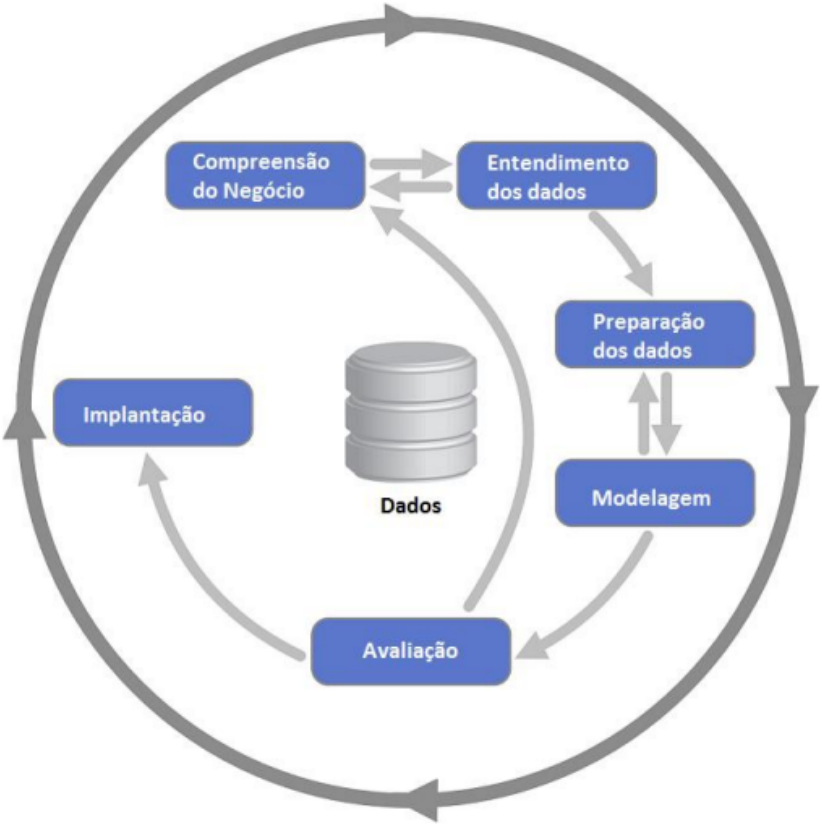
CRISP-DM é composto por seis fases principais:

1. Compreensão do Negócio: Esta fase inicial foca em entender os objetivos e requisitos do projeto a partir de uma perspectiva de negócios. Isso envolve a formulação de perguntas de mineração de dados que podem efetivamente abordar o problema de negócios.

2. Entendimento dos Dados: Envolve a coleta inicial dos dados e procede com atividades para se familiarizar com os dados, identificar problemas de qualidade dos dados, e descobrir primeiros *insights*.

3. Preparação dos Dados: Esta etapa inclui todas as atividades necessárias para construir o conjunto de dados final a partir dos dados brutos iniciais. As tarefas

Figura 4 – Fases do *CRISP-DM*.



Fonte: (SHEARER, 2000).

podem incluir limpeza de dados, construção de novas variáveis, e transformações.

4. Modelagem: Nesta fase, são selecionadas e aplicadas várias técnicas de modelagem de dados. Pode envolver a seleção de modelos, a definição de critérios de teste, e a construção de modelos.

5. Avaliação: Os modelos são avaliados no contexto dos objetivos do negócio. Esta fase é crucial para determinar se o modelo atende a um certo nível de aceitação ou se é necessário voltar e reformular o modelo.

6. Implementação: A última fase do processo é sobre implementar o modelo escolhido no ambiente operacional. Isso pode incluir a execução do modelo dentro de um sistema de tomada de decisão, apresentando os resultados do modelo para os *stakeholders* do negócio, ou simplesmente documentando a experiência e os conhecimentos adquiridos durante o projeto.

CRISP-DM é iterativo e flexível, permitindo ajustes e revisões em qualquer fase do projeto. Esta abordagem holística garante que os projetos de mineração de

dados sejam conduzidos de maneira eficiente e eficaz, com foco nas necessidades do negócio e na qualidade dos *insights* gerados. Como esse projeto não

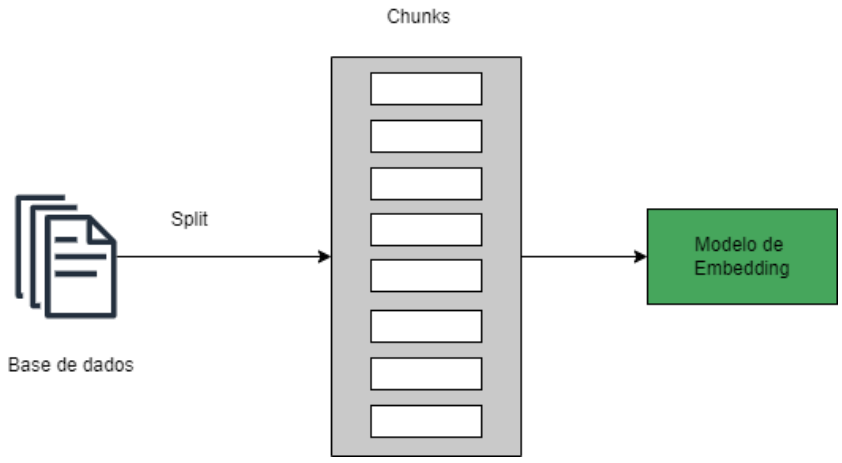
3.3 PRÉ-PROCESSAMENTO DE TEXTO

O pré-processamento de texto é uma etapa crítica na modelagem de linguagem e análise de dados de texto e está inserido dentro da fase de preparação dos dados dentro da metodologia do Cross-Industry Standard Process for Data Mining (CRISP-DM). Seu propósito é preparar dados linguísticos brutos para processamento posterior ou análise. Este processo envolve várias técnicas para limpeza e remoção de ruídos antes que o texto seja processado por modelos de linguagem. Tais procedimentos ajudam a reduzir a complexidade e melhoram a eficácia dos algoritmos de *Processamento de Linguagem Natural* (PLN), facilitando o reconhecimento de padrões e a recuperação de informações úteis.

3.3.1 *Chunking*

A técnica de *chunking* no pré-processamento de texto é um método usado em PLN para dividir um texto em segmentos menores, ou *chunks*, que contêm unidades de informação mais facilmente analisáveis e interpretáveis, figura 5. O objetivo é segmentar os dados em unidades menores e mais coesas, em que cada pedaço representa uma unidade semântica.

Figura 5 – Separação do texto em *chunks*.



Fonte - Elaborado pelo autor.

Esse técnica é importante pois facilita a análise semântica na extração de informações, já que em base de dados mais extensas, as informações estão dispersas em várias partes do texto ou documentos. Assim, a técnica facilita na hora do processo de

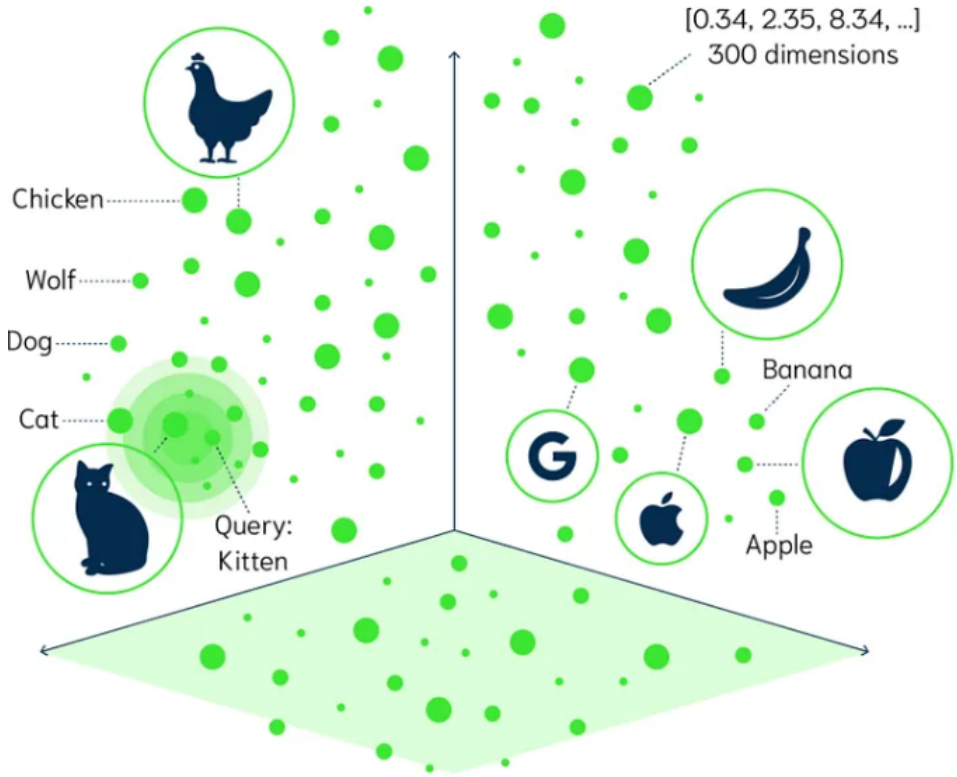
recuperação da informação encontrar cada uma dessas partes relevantes para dentro do contexto da busca.

3.3.2 Embeddings

Embeddings no contexto de PLN referem-se a representações vetoriais de palavras, frases, parágrafos ou documentos inteiros. Essas representações são feitas para capturar o significado, a semântica e a sintaxe dos elementos linguísticos de maneira a capturar as relações e contextos em que se encontram.

Em alto nível, *embeddings* são construídos de forma que palavras ou frases com significados ou contextos semelhantes estejam próximas umas das outras em um espaço vetorial, o que significa que elas teriam uma distância menor entre seus vetores correspondentes. Por outro lado, palavras ou frases com significados diferentes devem estar mais distantes neste espaço. A imagem 6 mostra essa representação vetorial, *kitten*, gatinho do inglês, esta mais perto perto de *cat*, gato, do que *chicken*, galinha.

Figura 6 – Representação vetorial de semântica.



Fonte: (HAM, 2022)

Esses *embeddings* capturam semelhanças semânticas e sintáticas entre palavras e podem ser usados para recuperar documentos que são contextualmente ou

semanticamente semelhantes a uma consulta, mesmo que não compartilhem palavras-chave idênticas. Por exemplo, uma busca por “motor com 12cv“ também poderia retornar produtos sobre "motor com 12 cavalos-vapor de potência", graças ao poder da busca semântica.

3.4 BANCO DE DADOS DE VETORES

Os Bancos de Dados de Vetores, ou *Vector Databases*, são sistemas de gerenciamento de banco de dados especializados que desempenham um papel vital no armazenamento, consulta e manipulação de dados vetoriais. Esses dados, fundamentais em aplicações de PNL e sistemas de busca baseados em *embeddings*, representam informações como vetores em espaços multidimensionais.

No âmbito do PNL, modelos como Word2Vec (MIKOLOV *et al.*, 2013), BERT (DEVLIN *et al.*, 2018), e as arquiteturas de *embeddings* disponibilizadas pela *Hugging Face* permitem transformar palavras ou frases em vetores. Nestes vetores, cada dimensão pode capturar um aspecto semântico do texto, possibilitando que operações matemáticas representem operações semânticas, tais como avaliar a semelhança entre termos.

Os bancos de vetores são especialmente otimizados para conduzir buscas de similaridade eficientes em grandes conjuntos de vetores de alta dimensão, empregando algoritmos de Approximate Nearest Neighbors (ANN) (AUMÜLLER; BERNHARDSSON; FAITHFULL, 2018). Esses algoritmos buscam encontrar os vizinhos mais próximos de um ponto de consulta de maneira aproximada, o que se traduz em consultas eficazes mesmo em espaços de dados extensos.

Considerando a demanda crescente por gerenciamento e consulta eficaz de dados complexos, que incluem vastos volumes de textos e imagens, os bancos de dados vetoriais tornam-se essenciais para o gerenciamento eficiente e para a recuperação ágil em repositórios de dados de alta dimensionalidade (JOHNSON; DOUZE; JÉGOU, 2017).

3.5 PROCESSAMENTO DE LINGUAGEM NATURAL

A área de *Natural Language Processing* (NLP) explora a interação entre computadores e linguagem humana, com o objetivo de capacitar máquinas para compreender, interpretar, e gerar texto ou fala de maneira que seja natural e útil para os usuários. O NLP engloba uma variedade de tarefas, desde a tradução automática e a análise de sentimentos até a resposta automática a perguntas e a geração de texto.

Com o advento das arquiteturas de *deep learning*, particularmente dos modelos *Transformer*, houve uma revolução significativa no campo do NLP. Os *Transformers*, introduzidos por Vaswani et al. em 2017 (VASWANI *et al.*, 2023), são modelos de

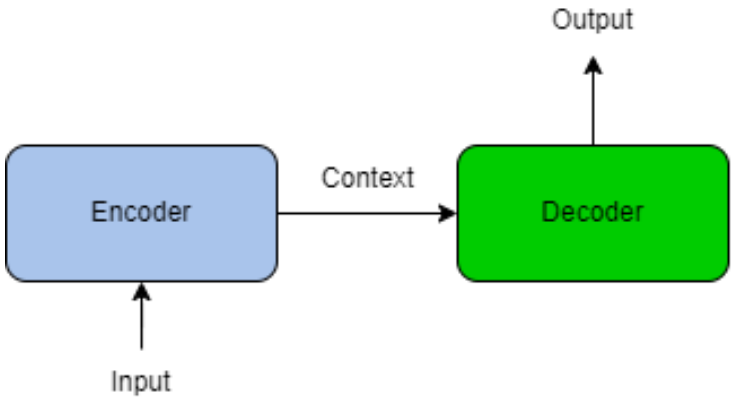
aprendizado de máquina que se baseiam inteiramente no mecanismo de atenção para processar sequências de dados. Diferentemente dos modelos anteriores que dependiam de Recurrent Neural Network (RNN), os *Transformers* permitem o processamento paralelo de dados e capturam as dependências de longo alcance mais eficientemente. Isso resulta em uma melhoria significativa no desempenho em uma vasta gama de tarefas de NLP, estabelecendo novos padrões para o que é possível em termos de compreensão e geração de linguagem natural por máquinas.

3.6 ARQUITETURA *TRANSFORMERS*

A arquitetura *Transformers* se baseia na arquitetura *encoder-decoder*, ela visa transformar uma sequência de entrada em uma nova sequência de saída, que pode ser de natureza diferente. Em tarefas como tradução automática, o *encoder* processa a sequência de entrada e cria uma representação de contexto. Esta representação contém a informação essencial da entrada, capturada em um vetor de características fixas, independentemente do comprimento da sequência de entrada.

O *decoder*, por sua vez, utiliza a representação de contexto para gerar a sequência de saída. A saída é construída passo a passo, e em cada passo, o *decoder* pode focar em diferentes partes da representação de contexto. Essa abordagem permite que a sequência de saída seja gerada com base no conteúdo completo da entrada, adaptando-se às necessidades específicas da tarefa em questão, como a geração de texto correspondente em outro idioma. A imagem 7 ilustra essa interação entre *encoder* e *decoder* de maneira simplificada, mostrando como o contexto é passado do *encoder* para o *decoder*, que então produz a saída.

Figura 7 – Diagrama geral da arquitetura *encoder-decoder*



Fonte - Elaborado do Autor

Os *Transformers* mantêm a estrutura básica *encoder-decoder*, mas introduzem

o mecanismo de atenção, especificamente o *Multi-Head Attention* para capturar as dependências dentro da sequência de entrada de forma mais eficiente. No *encoder* do *Transformer*, cada posição na sequência de entrada é capaz de atender diretamente a cada outra posição, facilitando a captura de dependências de longo alcance sem a necessidade de passar por várias etapas de processamento sequencial, como era o caso em arquiteturas baseadas em RNN.

Além disso, os *Transformers* aplicam o mecanismo de atenção no *decoder*, permitindo que ele se concentre em partes relevantes da representação de contexto enquanto gera a sequência de saída. Isso é especialmente útil em tarefas de geração sequencial, como tradução automática, onde a relevância das partes da entrada pode mudar ao longo da geração da saída.

Em suma, a arquitetura *Transformer* aperfeiçoa a estrutura *encoder-decoder* ao incorporar atenção em larga escala, permitindo um processamento paralelo mais eficiente e uma melhor captura de dependências complexas em dados de sequência. A figura 8 ilustra a arquitetura detalhada do *Transformer*, destacando a interconexão entre as diferentes subcamadas e a passagem de informações através do modelo.

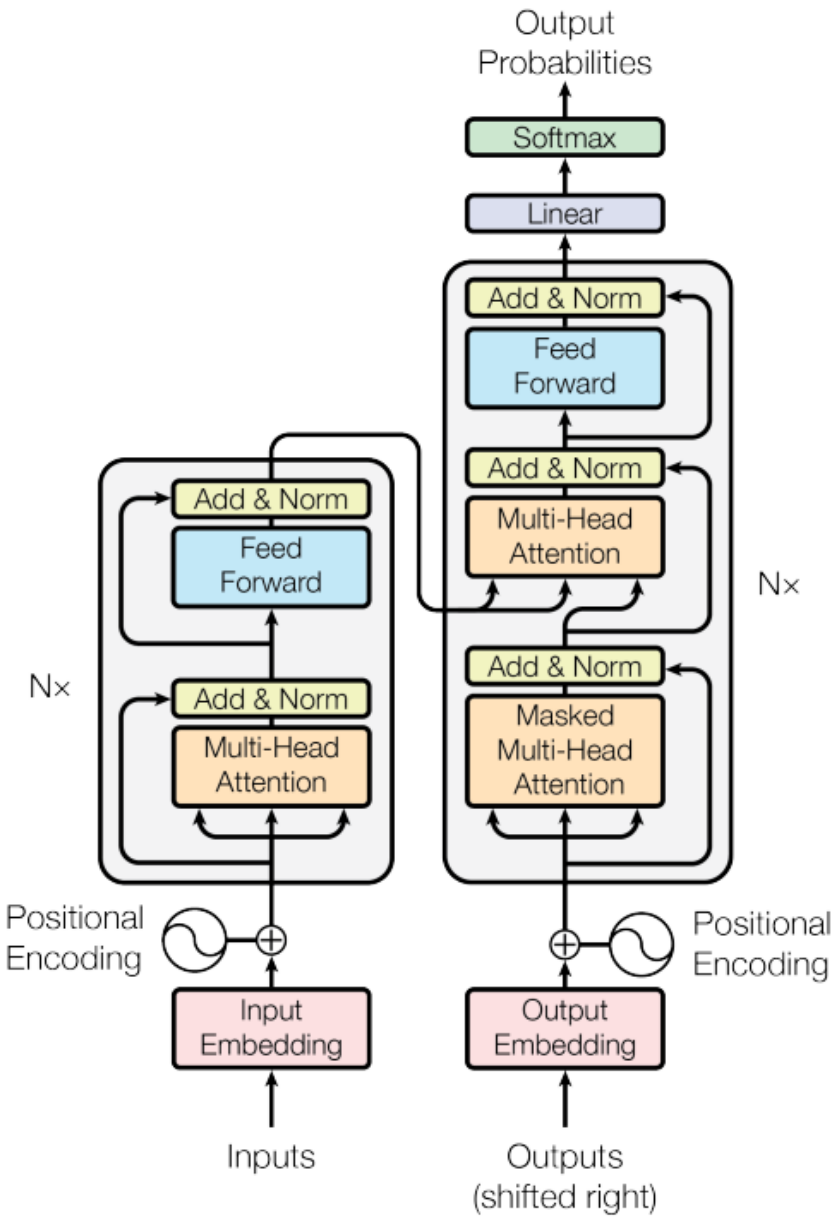
3.7 MODELOS DE LINGUAGEM

Os modelos de linguagem desempenham um papel crucial no avanço do PNL, e a arquitetura *Transformer* tem sido a base para alguns dos mais significativos dentre eles. A família de modelos *Generative Pre-Trained Transformer* (GPT), desenvolvida pela OpenAI, é notável por sua capacidade de gerar texto coerente e contextualmente relevante. O *Generative Pre-Trained Transformer 3* (GPT-3), em particular, é a terceira geração desta série e se destaca por seu tamanho massivo e sua abordagem de poucas ou nenhuma rotulação no aprendizado, conhecidos como *few-shot* e *zero-shot learning*, respectivamente. Com um vasto número de parâmetros e um extenso conjunto de dados para pré-treinamento, o GPT-3 é capaz de realizar uma ampla gama de tarefas de PNL com ajustes mínimos, um marco notável na área.

Além dos modelos GPT, o *Bidirectional Encoder Representations from Transformers* (BERT) da Google é outra inovação importante que utiliza a arquitetura *Transformer*. O BERT é inovador devido ao seu treinamento bidirecional, que lhe permite entender o contexto de um *token* dentro de uma sequência de texto, melhorando significativamente a qualidade da modelagem de linguagem em tarefas como classificação de texto e resposta a perguntas (DEVLIN *et al.*, 2019).

Outros modelos baseados em *Transformer* incluem o *Bidirectional and Auto-Regressive Transformers* (BART) e o *Text-to-Text transfer Transformer* (T5). O BART combina as características do BERT e do GPT, utilizando uma abordagem de *autoencoder* que é treinada para corrigir texto corrompido, o que o torna eficaz em tarefas de geração de texto e compreensão. O T5, por sua vez, adota uma abordagem única

Figura 8 – Arquitetura do *Transformers*

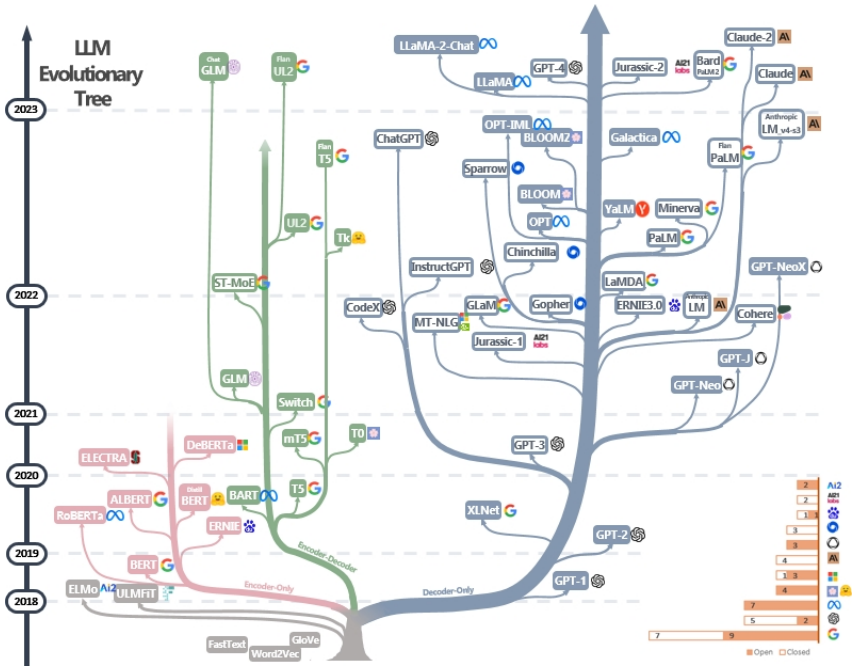


Fonte: (VASWANI *et al.*, 2023)

convertendo todas as tarefas de PNL em um problema de texto para texto, permitindo-lhe lidar com diversas tarefas com um único modelo generalista. A imagem 9 mostra a evolução dos modelos de linguagem desde seus primórdios com o modelo *Word2Vec* até os modelos atuais como o *GPT-4* e o *LLma-2* da empresa *Meta*.

Esses modelos são apenas o início das possibilidades inovadoras da arquitetura *Transformer* em PNL.Eles demonstram a flexibilidade e eficácia da arquitetura em capturar a complexidade da linguagem humana e em produzir modelos que podem entender e gerar linguagem de forma quase humana. A família GPT, em particular, tem

Figura 9 – Árvore de Evolução dos Modelos de Linguagem



Fonte: (YANG *et al.*, 2023)

impulsionado as fronteiras do que é possível em PNL, abrindo caminho para futuras inovações na área.

A *Inteligência Artificial* (IA) Generativa, que é essencialmente uma IA dotada da capacidade de criar, projetar ou gerar novos conteúdos que não existiam anteriormente, sendo o os LLMs seu exemplo mais notáveis, é uma área de interesse emergente, tem como aplicações em potencial variando desde setores manufatureiros, automotivos, aeroespacial e defesa. A inteligência artificial generativa facilita desde o design de peças com desempenho otimizado, levando em consideração restrições específicas, como materiais e métodos de fabricação (GARTNER, 2023), além de contribuir na geração de conteúdo e no aprimoramento de sistemas corporativos.

3.8 BUSCA SEMÂNTICA

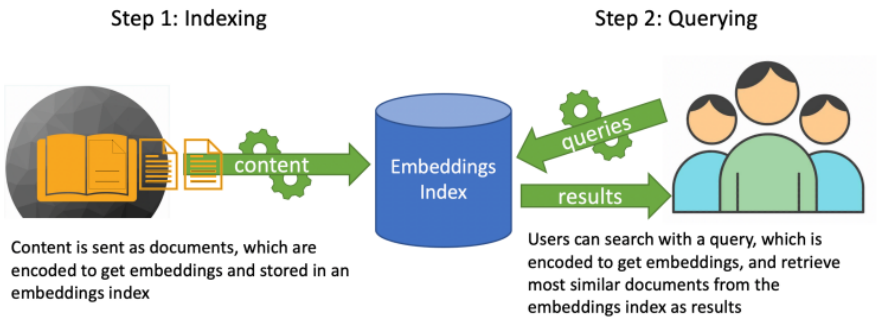
A busca semântica é um passo fundamental para se recuperar documentos relevantes de uma base de conhecimento específica. Nela são feitos os *embeddings* da consulta e dos documentos no conjunto de dados. Em seguida, é calculada a similaridade entre a consulta e os documentos para classificá-los de acordo com a relevância semântica. Esse cálculo é feito geralmente utilizando a similaridade de cossenos a qual mede o cosseno do ângulo entre dois vetores, o valor varia de -1 a 1, em que 1 indica vetores idênticos, 0 significa ortogonalidade e -1 sentidos opostos (DEERWESTER *et al.*, 1990). Na equação é calculado o produto vetorial dividido pelo

módulo do produto escalar dos vetores , equação1.

$$\text{Similaridade de Cosseno}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} \tag{1}$$

A figura 10 mostra esse fluxo de informações, os documentos são processados e indexados, a pergunta do usuário (*query*) é indexada também junto com os documentos e os resultados são os documentos mais similares retornados pelo sistema de *retrieval*.

Figura 10 – *Embeddings* e seu uso em sistemas de busca.



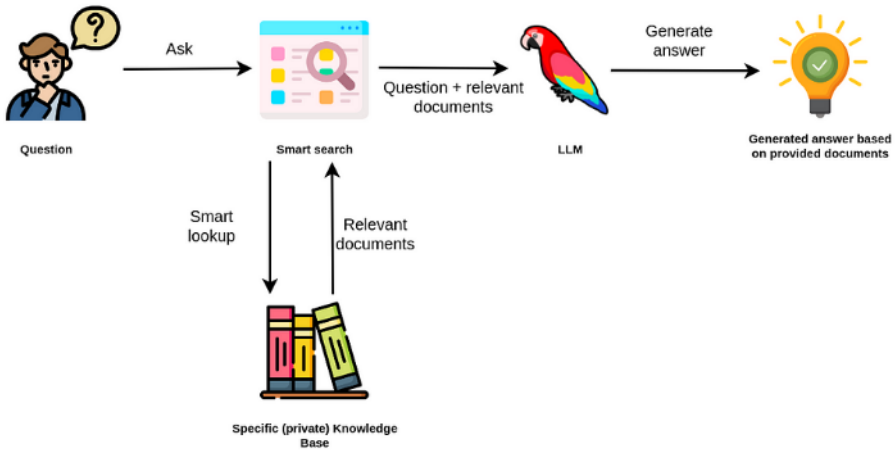
Fonte: (GRAINGER; TURNBULL; IRWIN, 2023)

3.9 *RETRIEVAL-AUGMENTED GENERATION*

O RAG é uma técnica usada no processamento de linguagem natural que combina a busca semântica com o poder dos LLMs para melhorar a qualidade e as informações do texto gerado. Nela o modelo pode ter acesso a uma base de conhecimento de vários tipos. Isso é vantajoso pois os modelos são treinados com um número de documentos específicos em um intervalo de tempo, de modo que informações e conhecimentos posteriores a essas datas o modelo não teve acesso, e portanto, não tem conhecimento sobre fatos ocorridos e conhecimentos posteriores. A figura 11 mostra esse processo, com a pergunta do usuário é feita uma busca semântica dentro da base de conhecimento específico e os documentos mais relevantes são passados para o modelo de linguagem junto com a pergunta do usuário para que ele elabore uma reposta adequada.

Nesse processo do RAG a busca semântica é capaz de recuperar documentos relevantes com base compreensão semântica da consulta. Em vez de depender estritamente de correspondência de palavras-chave, o sistema leva em consideração o significado das palavras e as relações entre conceitos.

Figura 11 – Arquitetura do *Retrieval-Augmented Generation*



Fonte:(BRATANIC, 2022)

Existem uma variedade de aplicações, pode ser usada em sistema de busca de documentos corporativos, assistentes virtuais e sistema de recomendação. Também pode ser usado em sistemas de pergunta e resposta (Q&A) e em ambientes de recuperação de informações complexas. Como é um tema relativamente novo, e com o avanço significativo dos LLMs tem-se um potencial enorme em melhorar significativamente a precisão e a eficácia dos sistemas de busca e recuperação de informações.

3.10 ENGENHARIA DE *PROMPT*

Engenharia de *prompt*, ou *prompt engineering*, é uma etapa fundamental para no uso de modelos de linguagem, especialmente no uso das LLMs. Ela envolve a criação de instruções de entrada, ou "prompts", para o modelo com o objetivo de obter respostas mais precisas e úteis. Com modelos como o GPT-3 da OpenAI, a forma como uma pergunta é estruturada pode influenciar significativamente a qualidade e a relevância da resposta gerada.

No contexto do RAG, a engenharia de *prompt* assume um papel crucial, pois *prompts* eficazes são necessários para instruir o modelo o que fazer com os dados recuperados do processo de busca semântica e como deve ser a reposta a partir deles. O design da *prompt* deve, portanto, ser suficientemente informativo e específico para guiar o modelo a acessar e integrar as informações corretas durante o processo de geração de resposta.

A literatura sobre engenharia de *prompt* é ainda incipiente, mas está crescendo à medida que mais pesquisadores e praticantes reconhecem a importância de interagir efetivamente com LLMs. Uma referência importante na área é o trabalho de Reynolds

e McDonell (REYNOLDS; MCDONELL, 2021) que exploram estratégias para otimizar *prompts* para modelos de linguagem, embora se concentrem no GPT-3.

Na área de aprendizado de máquina, particularmente no contexto de modelos de linguagem como os LLMs, as abordagens de *zero-shot*, *one-shot* e *few-shot* se referem ao número de exemplos fornecidos ao modelo para realizar uma tarefa sem treinamento adicional. Esses métodos são cruciais para entender como os modelos de IA podem ser efetivamente aplicados a problemas sem exigir uma extensa coleta de dados ou re-treinamento.

No *zero-shot learning* refere-se à capacidade do modelo entender e realizar uma tarefa sem ter recebido nenhum exemplo especifico da tarefa a ser executada, ele é instruído somente do que fazer. Já no *one-shot learning* é fornecido um exemplo para ajudar a compreender a tarefa antes de executar a tarefa, o exemplo serve como um modelo de reposta para o modelo seguir e ajuda ao desenvolver obter um resultado específico dentro da sua aplicação. Por último, o *few-shot learning* envolve fornecer ao modelo alguns exemplos, é usado em um contexto de aplicações mais complexas, em que o modelo deve captar padrões e regras implícitas da tarefa (BROWN *et al.*, 2020).

3.11 MÉTRICAS DE AVALIAÇÃO

Diversas métricas são empregadas para avaliar modelos de linguagem, sendo o *BLEU Score* e o *ROUGE* tradicionalmente usados para modelos pré-treinados em tarefas de tradução automática e sumarização, respectivamente. No entanto, o presente projeto requer métricas específicas que quantificam a acurácia dos resultados de busca fornecidos pelo sistema em comparação aos resultados esperados. Essas métricas mensuram a eficácia do sistema de recomendação em prover respostas precisas aos usuários. Para tal finalidade, duas métricas principais são utilizadas: o *Recall* e o *Precision*. Ambas avaliam aspectos cruciais em sistemas de recomendação e busca, embora por meio de perspectivas distintas (MANNING; RAGHAVAN; SCHÜTZE, 2008).

A métrica de *Precision* examina quais itens retornados são relevantes dentre um conjunto fixo de respostas recomendadas. A fórmula para calcular a *Precision*, apresentada na equação 2, é a razão entre o número de itens relevantes recomendados e o número total de itens recomendados:

$$\text{Precision} = \frac{\text{número de itens relevantes recomendados}}{\text{número de itens recomendados}} \tag{2}$$

Por outro lado, o *Recall*, foca na proporção de itens relevantes que foram recomendados em relação ao número total de itens relevantes existentes, como demonstrado na equação 3:

$$\text{Recall} = \frac{\text{número de itens relevantes recomendados}}{\text{número total de itens relevantes}} \tag{3}$$

Ambas as métricas são complementares: enquanto a *Precision* avalia a qualidade das recomendações dentro de um subconjunto limitado, o *Recall* considera a abrangência do sistema em identificar todos os itens relevantes disponíveis, fornecendo uma visão holística da performance do sistema de busca.

A métrica de *F1 Score* é uma medida de teste que considera tanto a precisão (*precision*) quanto a revocação (*recall*) para calcular a eficácia de um sistema de recuperação de informação. Esta métrica é particularmente útil quando se deseja encontrar um equilíbrio entre precisão e revocação, especialmente quando a distribuição das classes é desigual ou quando falsos positivos e falsos negativos têm consequências diferentes no contexto da aplicação.

O *F1 Score* é a média harmônica da precisão e da revocação, fornecendo um único score que equilibra ambas as métricas. Ele atinge o seu melhor valor em 1 (precisão e revocação perfeitas) e o pior em 0. A fórmula para o cálculo do *F1 Score* é:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Aqui, tanto a precisão quanto o *recall* são diretamente proporcionalmente à medida do *F1 Score*: se um deles diminui, o *F1 Score* também diminui. Portanto, um *F1 Score* alto sugere que o sistema tem um desempenho robusto tanto em precisão quanto em revocação.

O *F1 Score* é particularmente útil em situações onde os falsos positivos e falsos negativos têm diferentes custos associados, ou quando as classes estão desequilibradas. Em sistemas de busca, como aqui usando o RAG, o *F1 Score* pode fornecer uma visão mais holística do desempenho ao considerar tanto os itens relevantes que foram recuperados corretamente quanto os itens irrelevantes que foram excluídos com sucesso.

4 O PROJETO

Este capítulo falará sobre os requisitos funcionais e não funcionais do sistema desenvolvido, os casos de uso, diagramas de sequência, *deployment*, arquitetura e a metodologia de desenvolvimento empregada.

4.1 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais do sistema são cruciais para assegurar a eficácia, confiabilidade e a usabilidade do sistema. Eles não tratam diretamente das atividades específicas que o sistema deve realizar, mas sim de como o sistema realiza essas atividades. Dentre alguns dos requisitos não-funcionais, pode-se citar:

- Desempenho: Tempo de resposta rápido para as consultas de busca, além de lidar com grande volume de dados.
- Segurança: Cumprimento de regulamentos de privacidade e proteção de dados como a Lei Geral de Proteção de Dados Pessoais (LGPD).
- Facilidade de modificar o sistema para correções, melhorias ou adaptações de novas exigências.
- Interface intuitiva e amigável ao usuário.
- Integração eficiente com outras ferramentas ou sistemas existentes.
- Ter custos de operações compatíveis com a tecnologia empregada.

4.2 REQUISITOS FUNCIONAIS

Os requisitos funcionais descrevem as funcionalidades específicas que o sistema deve ser capaz de realizar para atender às necessidades dos usuários. Esses requisitos são geralmente expressos em termos de tarefas e serviços que o sistema deve fornecer. Os requisitos do projeto são baseados na capacidade do sistema de buscar e encontrar os materiais requisitados pelo usuário através de seu *input*. O modelo deve retornar o código do produto e uma breve descrição do material ou produto.

No caso de o usuário digitar dois *inputs* diferentes com o mesmo sentido semântico, o sistema deve abstrair e retornar os produtos e materiais certos, independentemente da forma que o usuário digitou.

No caso de o usuário pesquisar: "motor à prova de explosão com 12cv"ou "motor 12cv de potência e proteção ex-db", o modelo deve retornar uma lista com o código seguido de uma breve descrição do produto à prova de explosão. Nesse caso "ex-db"é uma sigla técnica para motores aptos operar em Zona 1 e 2, Grupos de gás IIA, IIB

e IIC e classe de temperatura T4 (WEG, 2023a). Resumindo, tens-se os requisitos a seguir:

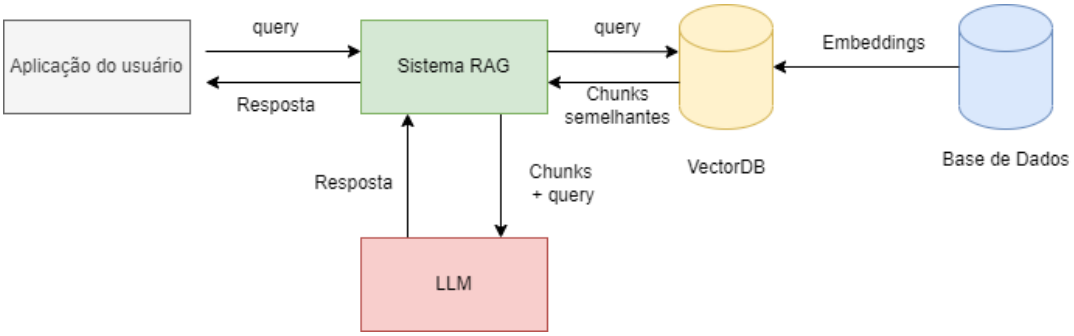
- Permitir que os usuários insiram termos de pesquisa.
- Retornar resultados relevantes da pesquisa.
- Oferecer suporte à busca por palavras-chave, frases e perguntas em linguagem natural.
- Mostrar os resultados de forma estruturada e com síntese das informações.

4.3 ARQUITETURA

A ferramenta desenvolvida consiste de uma API para um usuário fazer uma busca de materiais dentro de alguma aplicação própria. O intuito do projeto é de se fazer um protótipo simples para demostrar o uso e sua aplicabilidade.

A figura 12 mostra uma visão geral da arquitetura do sistema proposto. O banco de dados contem os nomes e as características dos materiais a serem usados pelo sistema. Para isso, deve-se fazer os *embeddings* desses dados que são armazenados em uma *vector store*, ou *vector database*. Com isso, dado um *input* do usuário, o sistema de *retrieval* faz uma busca dentro da *vector store* e retorna ao LLM os segmentos de textos mais semelhantes com a consulta. Por fim, o modelo de linguagem, com essas informações, elabora uma resposta a ser mostrada ao usuário na aplicação.

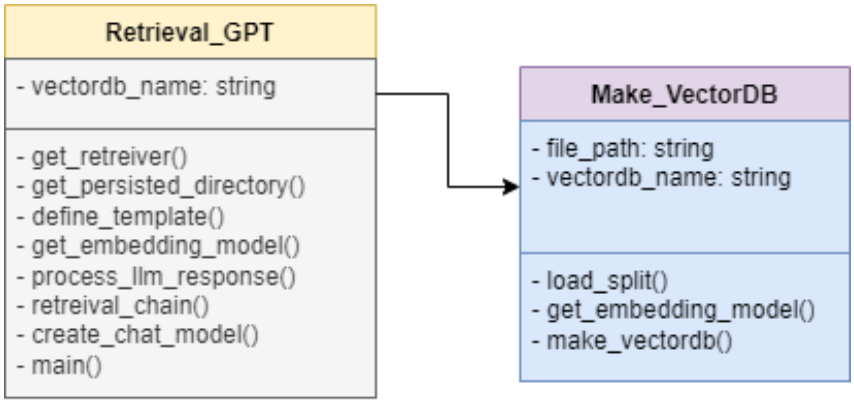
Figura 12 – Arquitetura geral do sistema.



Fonte - Elaborado pelo autor.

Complementando a arquitetura geral do sistema, o diagrama da figura 13 retrata em Unified Modeling Language (UML) as classes proposta para o sistema. A classe principal, *Retrieval_GPT* depende da execução da *Make_VectorDB* para ser realizado os *embeddings* dos dados. Feita uma vez, só é necessária sua execução caso aja uma alteração nessa base de dados.

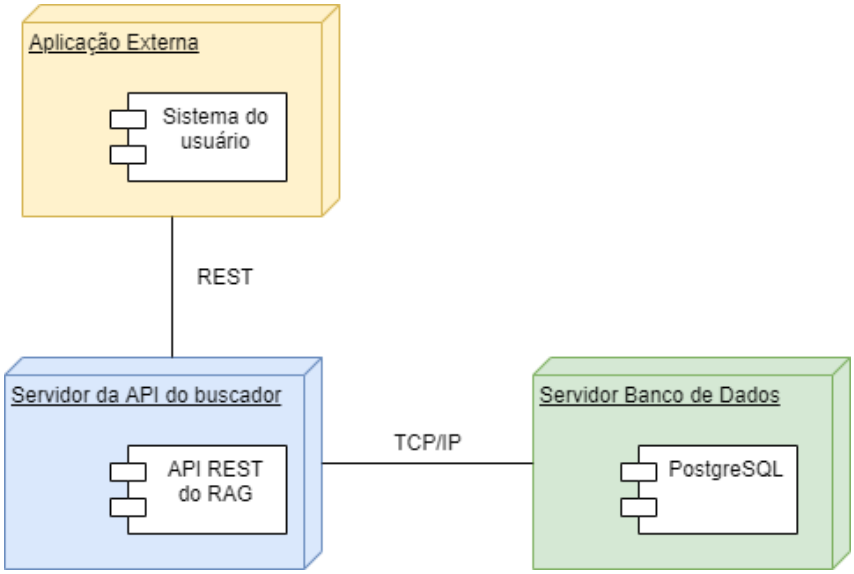
Figura 13 – Diagrama de classes da solução.



Fonte - Elaborado pelo autor.

Outro diagrama importante é o de implementação, retratado na figura 14. Ele mostra como a arquitetura proposta será executada, possibilitando a visualização de como os componentes de hardware e software do sistema estarão interligados.

Figura 14 – Diagrama de implementação do sistema.



Fonte - Elaborado pelo autor.

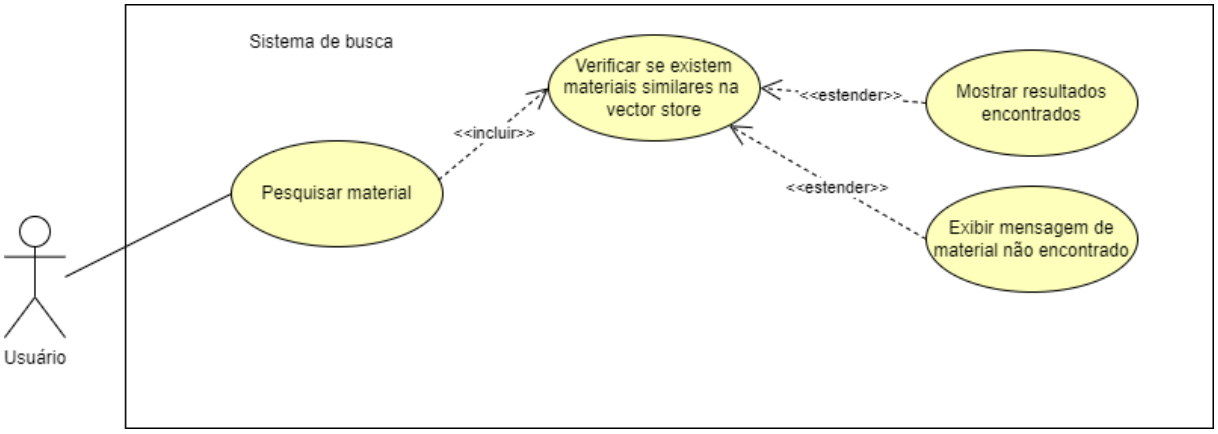
4.4 CASOS DE USO

Casos de uso UML constituem uma técnica de modelagem para capturar os requisitos funcionais de sistemas. Eles são uma ferramenta essencial no desenvolvimento de software e sistemas, permitindo a comunicação entre *stakeholders* técnicos e não técnicos (JACOBSON *et al.*, 1992; ROSENBERG; STEPHENS, 2007). A UML fornece uma forma padrão de visualizar a arquitetura de sistemas, incluindo aspectos como entidades, relações, atividades e processos.

Casos de uso são utilizados para especificar o comportamento esperado do sistema e como ele responde a solicitações externas. Essa abordagem é fundamental para garantir que todos os requisitos do usuário final sejam atendidos e que a funcionalidade do sistema esteja alinhada com os objetivos do negócio (KULAK; GUINEY, 2000).

O diagrama da figura 15 mostra as interações de um usuário com o sistema de busca de materiais, utilizando o modelo UML para representar os processos. Segue uma descrição dos casos de uso representados no diagrama:

Figura 15 – Casos de uso.



Fonte- Elaborado pelo autor.

1. Usuário: O ator principal do sistema, que interage com o sistema de busca de materiais.

2. Pesquisar material: Este é o caso de uso primário, onde o usuário inicia uma busca por materiais no sistema. Ele está ligado ao sistema de busca, que será responsável por processar a solicitação do usuário.

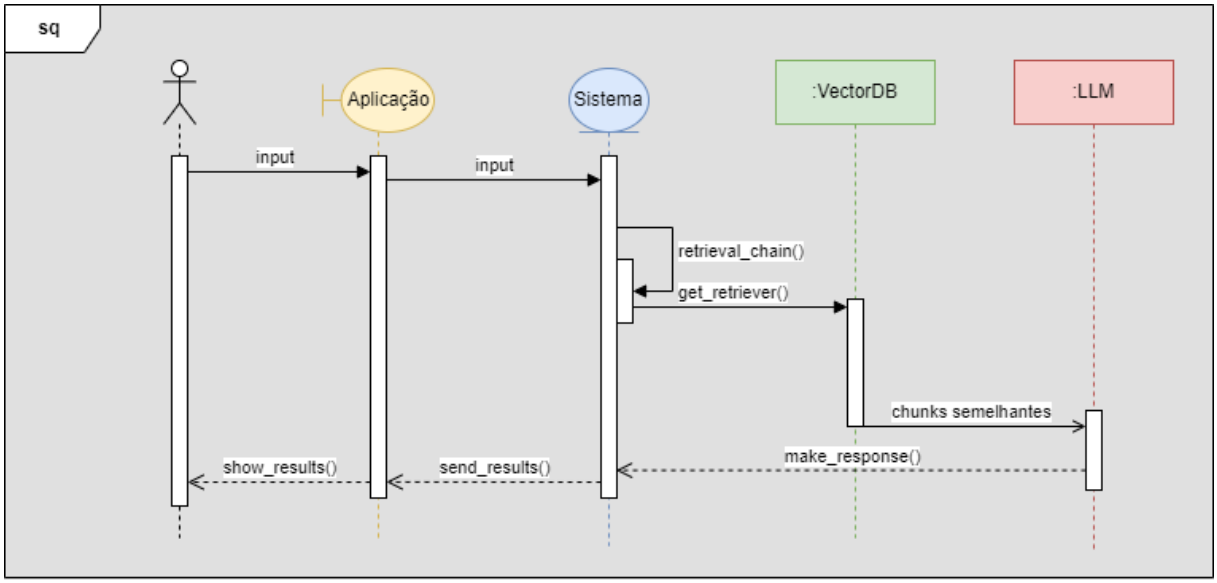
3. Verificar se existem materiais similares na *vector store*: Este caso de uso é uma funcionalidade do sistema de busca que verifica a existência de materiais similares dentro do banco de dados vetorial.

4. Mostrar resultados encontrados: Caso o sistema encontre materiais que correspondam à busca, este caso de uso descreve a ação de mostrar os resultados ao usuário.

5. Exibir mensagem de material não encontrado: Se o sistema de busca não conseguir encontrar materiais similares, ele exibirá uma mensagem indicando que o material não foi encontrado.

A partir dos casos de uso, construiu-se um diagrama de sequência, figura 16, o qual permite visualizar a sequência de eventos e respostas entre os autores e outros componentes do sistema. Na figura, a aplicação se refere a algum serviço que o usuário está interagindo e utilize o sistema RAG.

Figura 16 – Diagrama de sequência dos casos de uso do sistema.



Fonte- Elaborado pelo autor.

4.5 METODOLOGIA

O projeto foi desenvolvido utilizando a metodologia de desenvolvimento ágil *Scrum*, cuja principal característica é a sua abordagem adaptativa e flexível, entregas contínuas de valor, e a habilidade de responder rapidamente a mudanças (SCHWABER; SUTHERLAND, 2020). Nele se é dividido o projeto em ciclos curtos e gerenciáveis, conhecidos como *Sprints*, tipicamente durando de duas a quatro semanas. Cada *Sprint* começa com uma reunião de planejamento, onde se é selecionada as tarefas a serem desenvolvidas na *Sprint*.

Durante as *Sprints* são feitas reuniões diárias chamadas de *dailys* para monitorar o progresso. A *Sprint* se encerra com uma revisão e uma retrospectiva, visando uma melhoria contínua do processo.

Além das *dailys*, havia o líder da equipe que faz o papel de *Product Owner*. Com ele havia reuniões semanais para o acompanhamento do projeto, discussão do que já foi feito e quais as próximas etapas. Foi também utilizado um quadro *Kanban* com as tarefas dividas em *TO DO*, para fazer, *IN PROGRESS*, em andamento, e *DONE*, para tarefas já realizadas.

5 DESENVOLVIMENTO DO PROTÓTIPO

A Seção de Desenvolvimento é dedicada a detalhar a execução concreta do projeto. Será explicado brevemente as ferramentas utilizadas, a arquitetura sistêmica adotada e a aplicação específica das ferramentas que conduziram à solução final. Essa exposição compreende uma análise das escolhas técnicas e metodológicas, bem como uma descrição do processo de integração das diferentes tecnologias envolvidas. O intuito é fornecer uma compreensão de como os componentes individuais interagem e contribuem para o funcionamento do sistema desenvolvido.

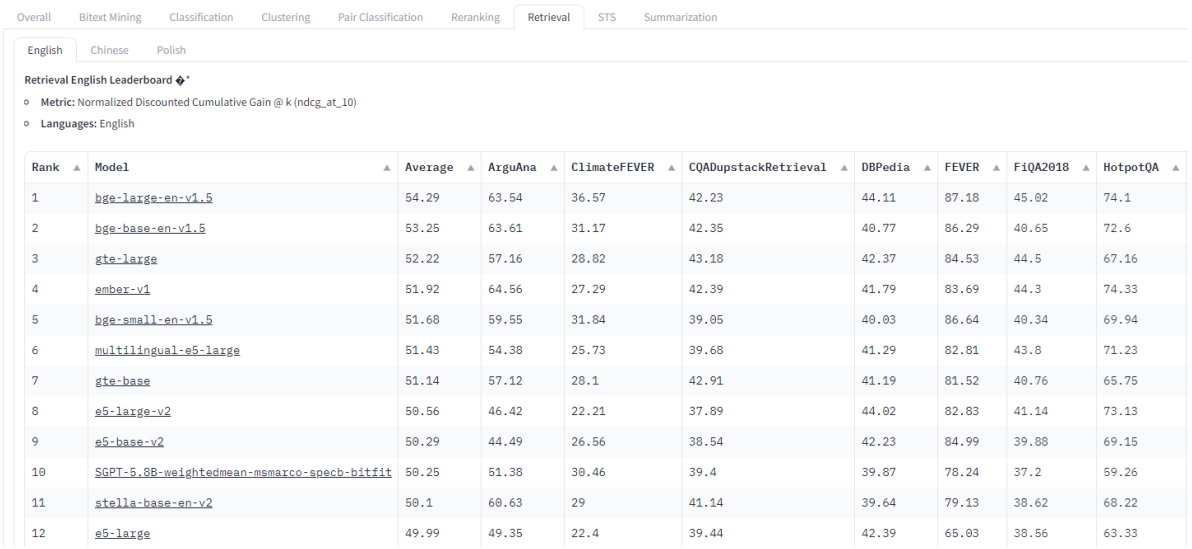
5.1 FERRAMENTAS

HuggingFace

Figura 17 – Símbolo do Hugging Face.

O *Hugging Face* é uma plataforma de hospedagem de modelos de linguagens, atuando de forma parecida como o *Github* faz para códigos-fonte, nela se encontram os principais modelos de linguagem *open-source* disponíveis na internet. Qualquer usuário pode construir um modelo, fazer um *fine-tuning* com seus dados e armazená-lo na plataforma. Uma forma bastante popular de fazer modelos próprios não precisando de muita infraestrutura é fazendo a quantização desses modelos, essa técnica muda a precisão dos pesos que compõe a rede neural da LLMs fazendo com que o modelo reduza de tamanho e rode em *hardwares* mais baratos com pouca perda de desempenho a depender do seu tamanho, modelos grandes, com mais de 60 bilhões de parâmetros, tendem a ter pouco impacto no desempenho.

Além disso, disponibiliza vários modelos de *embeddings*, usados para converter o texto de entrada em *embeddings*, que são vetores de alta dimensão que representam o conteúdo semântico do texto de entrada. Neste trabalho foram usados alguns desses modelos de *embeddings* para verificar qual deles se comporta melhor no sistema de busca. A imagem 18 mostra o *ranking* dos modelos de *embeddings* (MUENNIGHOFF *et al.*, 2022), eles são classificados por métricas de desempenho em várias tarefas envolvendo linguagem natural, como classificação, *clustering*, *reranking*, *retrieval* (usado neste trabalho), entre outros.



Overall | Bitext Mining | Classification | Clustering | Pair Classification | Reranking | Retrieval | STS | Summarization

English | Chinese | Polish

Retrieval English Leaderboard

- **Metric:** Normalized Discounted Cumulative Gain @ k (ndcg_at_10)
- **Languages:** English

Rank	Model	Average	ArguAna	ClimateFEVER	CQADupstackRetrieval	DBPedia	FEVER	FiQA2018	HotpotQA
1	bge-large-en-v1.5	54.29	63.54	36.57	42.23	44.11	87.18	45.02	74.1
2	bge-base-en-v1.5	53.25	63.61	31.17	42.35	40.77	86.29	40.65	72.6
3	gte-large	52.22	57.16	28.82	43.18	42.37	84.53	44.5	67.16
4	ember-v1	51.92	64.56	27.29	42.39	41.79	83.69	44.3	74.33
5	bge-small-en-v1.5	51.68	59.55	31.84	39.05	40.03	86.64	40.34	69.94
6	multilingual-e5-large	51.43	54.38	25.73	39.68	41.29	82.81	43.8	71.23
7	gte-base	51.14	57.12	28.1	42.91	41.19	81.52	40.76	65.75
8	e5-large-v2	50.56	46.42	22.21	37.89	44.02	82.83	41.14	73.13
9	e5-base-v2	50.29	44.49	26.56	38.54	42.23	84.99	39.88	69.15
10	SGPT-5.8B-weightedmean-msmarco-specb-bitfit	50.25	51.38	30.46	39.4	39.87	78.24	37.2	59.26
11	stella-base-en-v2	50.1	60.63	29	41.14	39.64	79.13	38.62	68.22
12	e5-large	49.99	49.35	22.4	39.44	42.39	65.03	38.56	63.33

Figura 18 – Ranking dos modelos de *embeddings* do Hugging Face.

Chroma

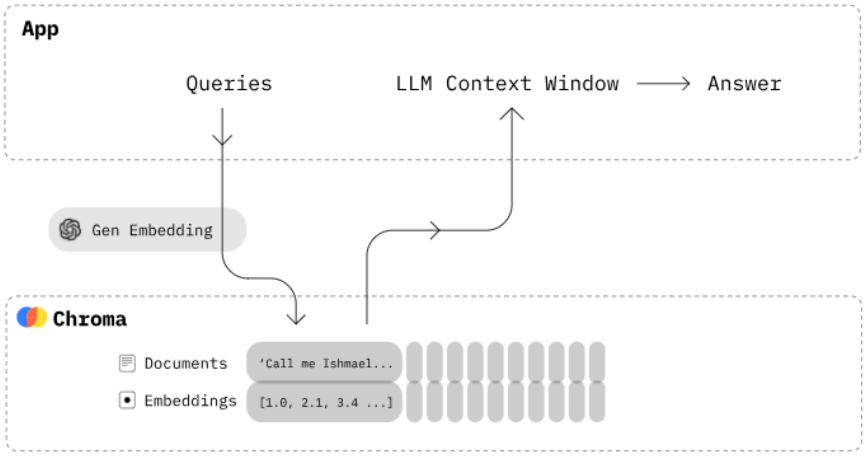
Figura 19 – Símbolo Chroma.

Chroma é uma banco de dados de vetores de alta dimensão. Após tirar os *embeddings*, eles são armazenados nessa *vector store*. Para otimizar o processo de busca dentro do banco, os vetores são indexados para otimizar o processo de *retrieval*, figura 20. Ele emprega estratégias avançadas de indexação para organizar e pesquisar rapidamente dados de alta dimensão. Essas estratégias podem incluir indexação baseada em árvore, *hash* ou até estruturas baseadas em grafos para facilitar a busca e recuperação rápidas.

Langchain

LangChain é uma biblioteca de código aberto para o desenvolvimento de aplicações que utilizam tecnologias de linguagem, particularmente modelos de linguagem grandes, como os da família GPT da OpenAI. A ideia por trás da LangChain é fornecer ferramentas e abstrações que facilitam a integração de modelos de linguagem em sistemas de software, permitindo que desenvolvedores e pesquisadores criem aplicações de IA conversacionais e interativas com mais facilidade.

Figura 20 – Arquitetura do Chroma dentro do processo de *retrieval.*



Fonte: (CHROMA. . . , 2023)

Figura 21 – Langchain

Essa biblioteca é utilizada como um ferramenta central no processo de RAG, já que ela vai fazer a integração entre a *vector store*, Chroma, a pergunta do usuário e o modelo de linguagem. Logo, funciona como um *hub* dentro do processo de *retrieval*.

Azure OpenAi

A Azure OpenAI API é uma colaboração entre a Microsoft e a OpenAI para disponibilizar os modelos de linguagem avançados da OpenAI, como o GPT, Codex e DALL-E, através da plataforma de computação em nuvem da Microsoft, o Azure. Esta integração tem como objetivo proporcionar as empresas e desenvolvedores uma maneira fácil e segura de acessar essas poderosas ferramentas de inteligência artificial com a infraestrutura e os recursos do Azure.

O uso da Azure OpenAI API permite que os usuários escalem seus aplicativos com base nas necessidades, aproveitando a capacidade da computação em nuvem da

Microsoft. Além disso, outro ponto fundamental é a questão de segurança da informação ao hospedar aplicações na Azure, os usuários e empresas podem se beneficiar das certificações de conformidade e das práticas de segurança da Microsoft, o que é particularmente importante para empresas que lidam com dados sensíveis, como a WEG. Assim, tem-se uma camada de segurança, já que os dados e *prompts* usados na *API* não vão ser usados para aprimorar os modelos da OpenAI (MICROSOFT, 2023). Por fim, a Azure oferece também todo um gerenciamento e monitoramento das aplicações, ajudando os usuários a controlar o desempenho, custos e utilização dos serviços de IA.

A Azure OpenAI API é um exemplo de como as capacidades de IA estão se tornando mais acessíveis e incorporadas em plataformas empresariais de *Tecnologia da Informação* (TI), com o objetivo de impulsionar a inovação e facilitar o desenvolvimento de novas aplicações que podem entender, interagir e gerar linguagem humana em nível natural.

5.2 PREPARAÇÃO DOS DADOS

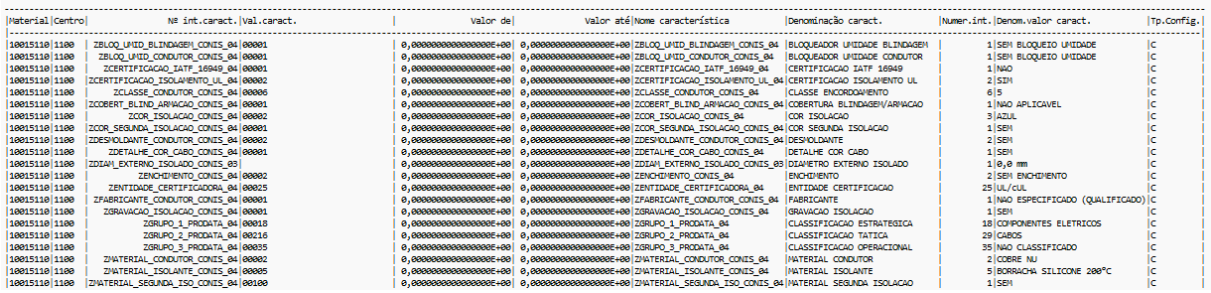
Para fazer o protótipo com base nos requisitos e diagramas da seção 4, foi utilizada uma base de dados para demonstrar e testar a solução proposta. Além disso, dada a metodologia do CRISP-DM da seção 3.2, essa capítulo trata das fases de entendimento e preparação dos dados que foram utilizados dentro do sistema de *retrieval* com LLM ou RAG.

5.2.1 Entendimento e preparação dos dados

Para esse protótipo foram usadas duas bases de dados, uma de materiais e outra de motores extraídas de fontes diferentes. Os dados de materiais foram extraídos do ERP WEG. Essa base de dados estava em formato de texto (.txt), figura 22, e continha colunas com características de materiais com seus respectivos valores, cada linha continha uma característica de cada material identificado por seu código. Para usá-la foi preciso converter os dados usando um expressão regular para extrair os dados e populá-los em um *dataframe* da biblioteca Pandas do *Python*. Assim, verificou-se que havia 5773 materiais e mais de 12 milhões de linhas.

Já no ambiente de desenvolvimento, para mapear o código do material com seus nomes, foi necessário concatenar essa base com outra tabela que continha os códigos e nomes. Além disso, como muitos materiais tinham centenas de caraterísticas e muitas vezes não tinham funcionalidades práticas no sistema de busca, foram selecionados materiais com poucas características, de no máximo 30, para validar a solução. Além disso, como na base havia milhares de motores, eles foram retirados, já que seriam inseridos com outra fonte mais atualizada. Essa base já estruturada e

Figura 22 – Base de dados de materiais.



Material	Centro	Nº int.caract.	Val.caract.	Valor de	Valor até	Nome característica	Denominação caract.	Numer.int.	Denom.valor caract.	Tp.Config.
10015110	1100	ZBLOQ_UMID_BLINDAGEM_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZBLOQ_UMID_BLINDAGEM_CONIS_04	BLOQUEADOR UMIDADE BLINDAGEM	1	SEM BLOQUEIO UMIDADE	C
10015110	1100	ZBLOQ_UMID_CONDUTOR_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZBLOQ_UMID_CONDUTOR_CONIS_04	BLOQUEADOR UMIDADE CONDUTOR	1	SEM BLOQUEIO UMIDADE	C
10015110	1100	ZCERTIFICACAO_IATF_16949_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZCERTIFICACAO_IATF_16949_04	CERTIFICACAO IATF 16949	1	NAO	C
10015110	1100	ZCERTIFICACAO_ISOLAMENTO_UL_04	00002	0,00000000000000E+00	0,00000000000000E+00	ZCERTIFICACAO_ISOLAMENTO_UL_04	CERTIFICACAO ISOLAMENTO UL	2	SIM	C
10015110	1100	ZCLASSE_CONDUTOR_CONIS_04	00006	0,00000000000000E+00	0,00000000000000E+00	ZCLASSE_CONDUTOR_CONIS_04	CLASSE ENCORDOAMENTO	6	5	C
10015110	1100	ZCOBERT_BLIND_ARMACAO_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZCOBERT_BLIND_ARMACAO_CONIS_04	COBERTURA BLINDAGEM/ARMACAO	1	NAO APLICAVEL	C
10015110	1100	ZCOR_ISOLACAO_CONIS_04	00002	0,00000000000000E+00	0,00000000000000E+00	ZCOR_ISOLACAO_CONIS_04	COR ISOLACAO	3	AZUL	C
10015110	1100	ZCOR_SEGUNDA_ISOLACAO_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZCOR_SEGUNDA_ISOLACAO_CONIS_04	COR SEGUNDA ISOLACAO	1	SEM	C
10015110	1100	ZDESMOLDANTE_CONDUTOR_CONIS_04	00002	0,00000000000000E+00	0,00000000000000E+00	ZDESMOLDANTE_CONDUTOR_CONIS_04	DESMOLDANTE	2	SEM	C
10015110	1100	ZDETALHE_COR_CABO_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZDETALHE_COR_CABO_CONIS_04	DETALHE COR CABO	1	SEM	C
10015110	1100	ZDIAM_EXTERNO_ISOLADO_CONIS_03		0,00000000000000E+00	0,00000000000000E+00	ZDIAM_EXTERNO_ISOLADO_CONIS_03	DIAMETRO EXTERNO ISOLADO	1	0,0 mm	C
10015110	1100	ZENCHIMENTO_CONIS_04	00002	0,00000000000000E+00	0,00000000000000E+00	ZENCHIMENTO_CONIS_04	ENCHIMENTO	2	SEM ENCHIMENTO	C
10015110	1100	ZENTIDADE_CERTIFICADORA_04	00025	0,00000000000000E+00	0,00000000000000E+00	ZENTIDADE_CERTIFICADORA_04	ENTIDADE CERTIFICACAO	25	UL/cUL	C
10015110	1100	ZFABRICANTE_CONDUTOR_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZFABRICANTE_CONDUTOR_CONIS_04	FABRICANTE	1	NAO ESPECIFICADO (QUALIFICADO)	C
10015110	1100	ZGRAVACAO_ISOLACAO_CONIS_04	00001	0,00000000000000E+00	0,00000000000000E+00	ZGRAVACAO_ISOLACAO_CONIS_04	GRAVACAO ISOLACAO	1	SEM	C
10015110	1100	ZGRUPO_1_PRODATA_04	00018	0,00000000000000E+00	0,00000000000000E+00	ZGRUPO_1_PRODATA_04	CLASSIFICACAO ESTRATEGICA	18	COMPONENTES ELETRICOS	C
10015110	1100	ZGRUPO_2_PRODATA_04	00216	0,00000000000000E+00	0,00000000000000E+00	ZGRUPO_2_PRODATA_04	CLASSIFICACAO TATICA	29	CABOS	C
10015110	1100	ZGRUPO_3_PRODATA_04	00035	0,00000000000000E+00	0,00000000000000E+00	ZGRUPO_3_PRODATA_04	CLASSIFICACAO OPERACIONAL	35	NAO CLASSIFICADO	C
10015110	1100	ZMATERIAL_CONDUTOR_CONIS_04	00002	0,00000000000000E+00	0,00000000000000E+00	ZMATERIAL_CONDUTOR_CONIS_04	MATERIAL CONDUTOR	2	COBRE NU	C
10015110	1100	ZMATERIAL_ISOLANTE_CONIS_04	00005	0,00000000000000E+00	0,00000000000000E+00	ZMATERIAL_ISOLANTE_CONIS_04	MATERIAL ISOLANTE	5	BORRACHA SILICONE 200°C	C
10015110	1100	ZMATERIAL_SEGUNDA_ISO_CONIS_04	00100	0,00000000000000E+00	0,00000000000000E+00	ZMATERIAL_SEGUNDA_ISO_CONIS_04	MATERIAL SEGUNDA ISOLACAO	1	SEM	C

Fonte - WEG.

limpa foi armazenada em um banco de dados relacional, o *PostgreSQL*.

A base de motores da empresa foi extraída de um banco de dados já pronto que foi feita a partir de um processo de *web-scrapping* do *e-commerce* da WEG, nele as características principais dos motores e sua descrição foram extraídas e armazenadas em um banco de dados não relacional, o *MongoDB*. Essa base de dados foi escolhida pois havia bastante dados disponíveis e com mais descrições dos motores do que havia nos dados provenientes do ERP WEG.

Como os dados de materiais e motores foram extraídos de base de dados diferentes, foram concatenados para o mesmo arquivo e armazenados. Esses dados, referidos aqui apenas como materiais, foram inicialmente manipulados para o formato *JSON*, já que é um formato de arquivo padrão para transferência e armazenamento de informação.

No entanto, após uma série de experimentações, constatou-se que o formato em questão não produz resultados satisfatórios no processo de *Retrieval*. Uma explicação plausível para esse fenômeno reside no fato de que arquivos em formato *JSON* contêm um número elevado de aspas para denotar cadeias de caracteres (*strings*), além das quebras de linhas contidas após cada descrição, o que pode resultar em representações vetoriais desprovidas de significado semântico coerente. Esse ruído nos dados poderia comprometer significativamente a eficiência do mecanismo de busca semântica na base de dados de vetores.

Para resolver esse problema o arquivo *JSON* foi transformado em um arquivo de texto (.txt) sem quebras de linhas entre as características de um mesmo material. Diminuindo, assim, o problema da representação de quebra de linha e eliminando as aspas da base de dados. Desse modo, obteve-se um resultado melhor.

Além disso, os dados foram normalizados, tirou-se os acentos e as letras foram passadas para minúsculas. A figura 24 mostra como ficou uma amostra da base de dados utilizada no processo de *embeddings*.

Figura 23 – Amostra dos dados em *JSON*.


```
},
{
    "nome": "rolamento 6315 c4",
    "anel retencao": "sem",
    "blindagem / vedacao": "sem blindagem / vedacao",
    "centralizacao gaiola": "corpo rolante",
    "classe precisao": "normal",
    "diametro externo": "160,000 mm",
    "diametro interno": "75,000 mm",
    "fabricante": "qualificado",
    "folga interna": "c4",
    "classificacao estrategica": "rolamentos",
    "classificacao tatica": "rolamentos grandes",
    "classificacao operacional": "nao classificado",
    "isolamento eletrico": "sem",
    "largura": "37,000 mm",
    "lubrificante": "sem lubrificante",
    "material gaiola": "aco carbono",
    "quantidade carreira": "1",
    "ranhura fixacao": "sem",
    "serie rolamento": "6.315",
    "temperatura trabalho": "temperatura normal (120\u00b0c)"
},
{
```


Fonte - WEG.

Figura 24 – Amostra da base de dados utilizada em formato de texto.


```
nome: motor 5cv 2p 80m/ms wfs1, code: 10019960, carcaca comercial: 80m/ms, frequencia: 60 hz, potencia: 5 cv, rotacao: 3470 rpm, tensao: 220/380 v, grau de protecao: ip54, polos: 2, material: ferro
nome: motor 1cv 4p ex61g wbx1, code: 10021932, carcaca comercial: ex61g, frequencia: 60 hz, potencia: 1 cv, rotacao: 1740 rpm, tensao: 110/220 v, grau de protecao: ip44, polos: 4, material: chapaa
nome: motor 1.5hp 4p d56 wca1 st mot tri, code: 10021960, carcaca comercial: d56, frequencia: 50 hz, potencia: 1.5 hp, rotacao: 1410 rpm, tensao: 220/380 v, grau de protecao: ip21, polos: 4, mater
nome: motor 7.5cv 4p 901/ms wfs1, code: 10022551, carcaca comercial: 901/ms, frequencia: 60 hz, potencia: 7.5 cv, rotacao: 1730 rpm, tensao: 220/380 v, grau de protecao: ip54, polos: 4, material:
nome: motor 10cv 4p 901/ms wfs1, code: 10083997, carcaca comercial: 901/ms, frequencia: 60 hz, potencia: 10 cv, rotacao: 1715 rpm, tensao: 220/380 v, grau de protecao: ip54, polos: 4, material: fe
nome: motor 1/2cv 2p e56j wca1, code: 10084191, carcaca comercial: e56j, frequencia: 60 hz, potencia: 0.5 cv, rotacao: 3480 rpm, tensao: 127/220 v, grau de protecao: ip21, polos: 2, material: chap
nome: motor 1/2hp 4p a56 wca1 st mot tri, code: 10084736, carcaca comercial: a56, frequencia: 50 hz, potencia: 0.5 hp, rotacao: 1430 rpm, tensao: 220/380 v, grau de protecao: ip21, polos: 4, mater
nome: motor 50cv 4p 200l wfx1, code: 10345416, carcaca comercial: 200l, frequencia: 60 hz, potencia: 50 cv, rotacao: 1775 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 4, material: ferro,
nome: motor 1cv 4p 90s wfx1, code: 10371655, carcaca comercial: 90s, frequencia: 60 hz, potencia: 1 cv, rotacao: 1760 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 4, material: ferro, fas
nome: motor 5cv 2p 100l wfx1, code: 10483115, carcaca comercial: 100l, frequencia: 60 hz, potencia: 5 cv, rotacao: 3505 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro, f
nome: motor 3cv 4p 901/ms wfs1, code: 10483210, carcaca comercial: 901/ms, frequencia: 60 hz, potencia: 3 cv, rotacao: 1745 rpm, tensao: 220/380 v, grau de protecao: ip54, polos: 4, material: ferr
nome: motor 10cv 2p 132m wfx1, code: 10579387, carcaca comercial: 132m, frequencia: 60 hz, potencia: 10 cv, rotacao: 3540 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro,
nome: motor 0.5cv 2p 90s wfx1, code: 10592195, carcaca comercial: 90s, frequencia: 60 hz, potencia: 0.5 cv, rotacao: 3495 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro,
nome: motor 6cv 2p 112m wfx1, code: 10624327, carcaca comercial: 112m, frequencia: 60 hz, potencia: 6 cv, rotacao: 3490 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro, f
nome: motor 2cv 2p 90s wfx1, code: 10628286, carcaca comercial: 90s, frequencia: 60 hz, potencia: 2 cv, rotacao: 3475 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro, fas
nome: motor 1cv 2p 90s wfx1, code: 10637689, carcaca comercial: 90s, frequencia: 60 hz, potencia: 1 cv, rotacao: 3490 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro, fas
nome: motor 3cv 2p 901 wfx1, code: 10702694, carcaca comercial: 901, frequencia: 60 hz, potencia: 3 cv, rotacao: 3440 rpm, tensao: 220/380 v, grau de protecao: ip55, polos: 2, material: ferro, fas
```


Fonte - WEG.

5.2.2 Preparação para *embeddings* da base de materiais

Antes da extração dos *embeddings*, os dados são primeiramente segmentados em unidades menores e mais coesas, frequentemente referidas como *chunks*. Esta fase de pré-processamento é crucial no contexto do RAG, pois estabelece o tamanho e a qualidade dos *chunks* que serão empregados na subsequente fase de *retrieval*. A seleção de um delimitador inapropriado pode comprometer a integridade semântica dos *chunks*, afetando negativamente o processo de recuperação do modelo. A estratégia ótima identificada foi a utilização do separador de linha, “\n”. Com esta abordagem, cada material e suas respectivas especificações são isolados em linhas

distintas, assegurando que não ocorra perda informativa durante a operação de *split*.

O processo de separação de documentos em *chunks* e a subsequente transformação desses em *embeddings* é um procedimento fundamental nos sistemas de RAG. A seção seguinte vai discutir a arquitetura da solução a partir dos dados já prontos.

5.3 ARQUITETURA DA SOLUÇÃO

A Figura 25 ilustra o esquema da solução proposta. Será descrito uma visão mais geral do sistema e será detalhado posteriormente alguns componentes que precisam de mais especifidades técnicas.

5.3.1 Visão Geral

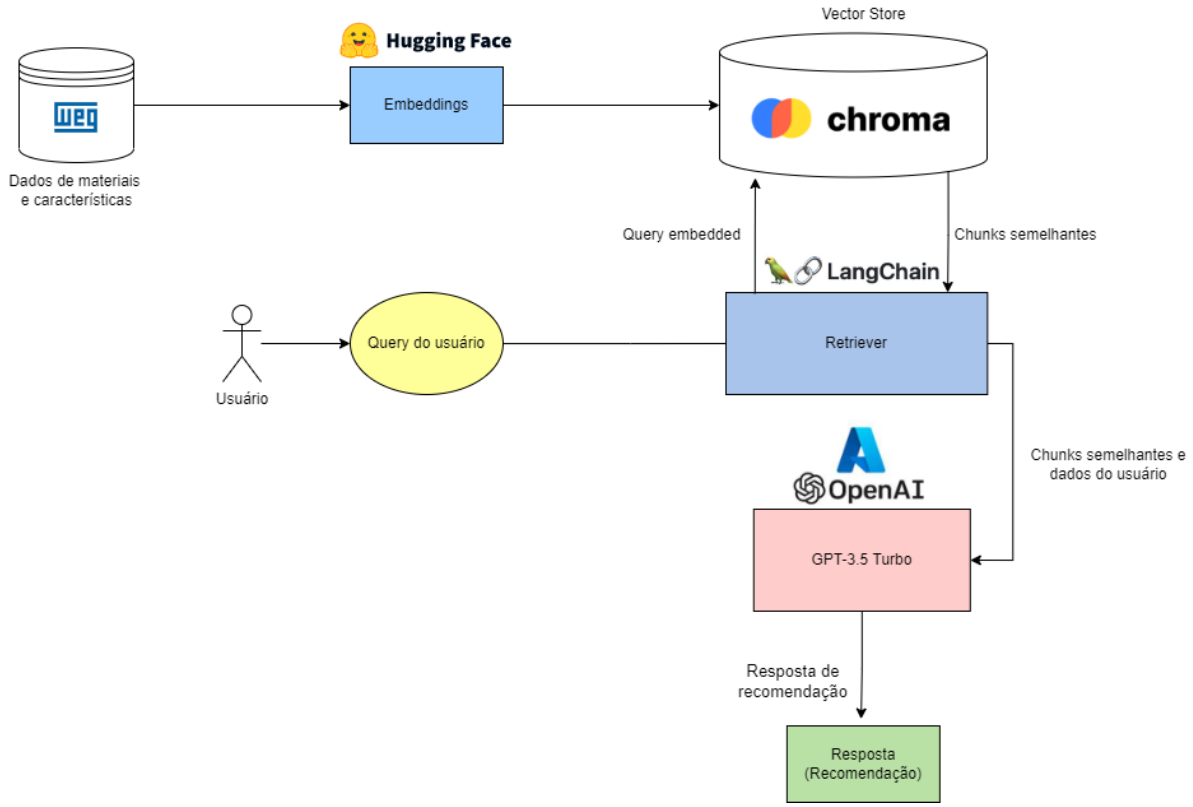
Inicia-se o processo tirando os *embedding* da base de dados dos materiais, onde os dados já estão segmentados em *chunks*. Este processo facilita a subsequente criação de índices para o mecanismo de *retrieval*. Para a transformação desses dados em representações vetoriais, empregou-se um modelo de *embedding* do *Hugging Face*. Este modelo atua analogamente a uma função de mapeamento, convertendo conjuntos de texto em vetores num espaço vetorial contínuo e de dimensionalidade reduzida. Os vetores resultantes, que codificam as propriedades semânticas e as relações inerentes aos dados, são então armazenados em um repositório denominado *vector store*, aqui usado o Chroma, servindo como base para o *retrieval*.

No processo de busca, o usuário introduz um comando de entrada (*input*) no sistema, o qual é processado pelo algoritmo de *retrieval* incorporado ao *LangChain*. Este algoritmo realiza uma consulta à *vector store* para identificar os *chunks* que exibem a maior correspondência semântica com a consulta realizada. A similaridade entre os vetores é determinada por meio do cálculo da similaridade de cossenos. Os ’n’ *chunks* com valores mais altos de similaridade são selecionados e repassados ao *retriever*, que, por sua vez, fornece ao modelo de *LLM* os segmentos de texto mais relevantes juntamente com o *input* original do usuário.

Ao processar essas entradas — o *input* do usuário e os *chunks* selecionados — o modelo elabora uma resposta com a recomendação dos materiais que satisfazem a entrada do usuário.

Essa estrutura da solução permite uma facilidade em se modificar os componentes e fazer melhorias no sistema. É possível alterar o modelo de linguagem ou a *vector store* sem maiores problemas, por exemplo. Tal fato, satisfaz os requisitos não funcionais de facilidade de modificação do sistema e de integração com outras ferramentas existentes.

Figura 25 – Arquitetura do sistema



Fonte - Elaborado pelo autor

5.3.2 *Embeddings*

No desenvolvimento do sistema de busca, optou-se pelo modelo de *embeddings* *text-embedding-ada-002* fornecido pela OpenAI. Esta escolha fundamentou-se na capacidade do modelo de gerar representações vetoriais de alta dimensionalidade, com especificamente 1536 dimensões no vetor de saída. Tal característica é presumida para oferecer uma captura mais rica da semântica dos dados representados (OPENAI, 2021). Em contraste, outros modelos de *embeddings*, incluindo o *BAAI/bge-large-en-v1.5*, foram avaliados, mas não alcançaram desempenho comparável (ARTIFICIAL INTELLIGENCE, 2020). A alta dimensionalidade dos vetores do *text-embedding-ada-002* é um fator contribuinte importante para uma mapeamento semântico mais detalhado, o que é crucial para a eficácia do sistema de busca em identificar e recuperar informações relevantes com precisão.

5.3.3 Processo de *Retriever*

A função de *Retrieval* do LangChain funciona como um orquestrador dentro do sistema de RAG. Ele vai receber o *input* do usuário e usar uma função de similaridade

para recuperar quais os *chunks* mais parecidos com a entrada do usuário. Nessa função foi usado o método chamado de *Similarity Search* para recuperar os *chunks* com maior similaridade de cosseno entre a entrada do usuário e os *embeddings* dos *chunks* presentes na *vector store*. Foi ainda usado outro parâmetro, "k", que define o número de segmentos de textos a serem passados para o modelo de linguagem, foi usado o valor de 10 neste último.

5.3.4 Modelo de Linguagem

O modelo de linguagem utilizado nesse sistema foi o *GPT-Turbo 3.5* disponibilizado pela *API* da Azure. Ele foi configurado com uma temperatura zero, esse é um parâmetro de configuração padrão de modelos de linguagem que controla a aleatoriedade das respostas geradas. Uma temperatura de zero significa que o modelo escolherá sempre a resposta mais provável, sem introduzir variação ou aleatoriedade. Isso geralmente leva a respostas mais consistentes e previsíveis, mas pode reduzir a criatividade ou a variedade nas respostas geradas. Como o sistema lida com dados assertivos e não precisa de criatividade em sua resposta, a temperatura zero foi a definida a ser usada.

A sua função é elaborar uma resposta a partir dos dados recebidos do *retriever* com a orientação do *prompt* que será explicado a seguir.

5.4 PROMPT DO RAG

Um ponto fundamental ao se trabalhar com LLM é o uso de um *prompt* adequado para o uso esperado do modelo, já que ele serve como uma instrução ao modelo. Esse processo envolve a criação de um *template*, que são instruções específicas de como o modelo deve se comportar ao realizar a tarefa requisitada pelo sistema, neste caso, é de recomendar materiais e produtos com base na requisição do usuário.

A biblioteca do Langchain oferece funções que fazem essa integração com o processo de *retrieval*. No *template* é passado um contexto para o modelo, instruindo o que ele vai receber de dados e o que se deve retornar a partir dele com base em exemplos de perguntas e respostas, *few-shot learning*. Além disso é passado também a pergunta do usuário. Com todas essas informações, o modelo consegue trazer resultados relevantes.

A figura 26 mostra um exemplo de um *prompt* utilizado para instruir o sistema a retornar o resultado correto, é dito qual a função dele e o que vai receber de dados, além do que deve retornar, nesse caso o código do material e uma breve descrição dele. É utilizado a técnica de *few-shot learning* para mostrar ao modelo o formato da resposta esperada, são utilizados alguns exemplos de requisições de usuários e modelos de respostas. O *prompt* da figura 41 inserido no apêndice A mostra a versão

completa e implementada no sistema.

Figura 26 – Exemplo de *template* do sistema de busca de materiais.

```
template = """

Você é um end-point de uma API de um sistema de busca de materiais,
    deve retornar uma lista com o código dos materiais dado pela
   chave "código" seguido de uma breve descrição do material.
   Somente retorne materiais que satisfazem as características
   pedidas pelo usuário.

Exemplo:
Input: motor 10cv, 2 polos, carcaça comercial 132m.
Output: [13007222 : motor com 350cv, 2 polos, carcaça do tipo 355m/
   l, frequência de 60hz, grau de proteção ipw66, trifásico,
   material de ferro.Com o motor w22xtb sua empresa estara operando
    com seguranca e confiabilidade em areas classificadas como zona
    21 e 22, grupos iiia, iiib e iiic, classe de temperatura t125c.
    É sinonimo de economia e seguranca para aplicacoes em
   processamento de graos, cereais, fibra textil, tinta po,
   polimeros, entre outros.]
{context}
Pergunta: {question}
Resposta:  """
```

Fonte - Elaborado pelo autor

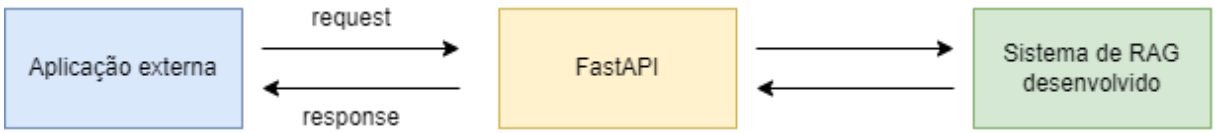
5.5 DESENVOLVIMENTO DA API

O sistema de RAG foi desenvolvido em um servidor com unidade de processamento gráfico, ou Graphics Processing Unit (GPU), para ter mais poder computacional e, assim, atender o requisito não funcional de desempenho com respostas rápidas. Para disponibilizar a ferramenta para outras aplicações foi desenvolvida uma API que recebe a entrada do usuário e retorna a resposta. Ela foi desenvolvida usando o *FastAPI*, que é um *framework Python* para desenvolvimento de APIs. A figura 27 mostra o esquema da API fazendo a interconexão entre a aplicação externa e o sistema RAG aqui proposto. A aplicação externa aqui desenvolvida foi uma interface *web* que será abordada na próxima seção.

5.6 INTERFACE

A interface *web* foi desenvolvida utilizando o *framework Flask* da linguagem de programação *Python*. Esta interface serve como uma demonstração conceitual do projeto e de suas capacidades funcionais que consome a API desenvolvida. A interface

Figura 27 – Relação da API com a aplicação e o sistema RAG



Fonte - Elaborado pelo autor

é composta por uma tela principal que apresenta um campo de busca, figura 28. Neste campo, o usuário pode inserir o nome do produto ou material de interesse. Foi integrado um mecanismo de *autocomplete* que, à medida que o usuário digita, exibe uma lista de seleção dinâmica com os produtos que correspondem aos caracteres inseridos. Independentemente de o usuário completar a digitação com um nome de material diretamente sugerido pelo *autocomplete* ou inserir uma descrição mais detalhada do item desejado, a ativação do botão inicia o processo de RAG, redirecionando para uma nova página que exibe os resultados pertinentes à consulta realizada.

Figura 28 – Interface do sistema de busca.

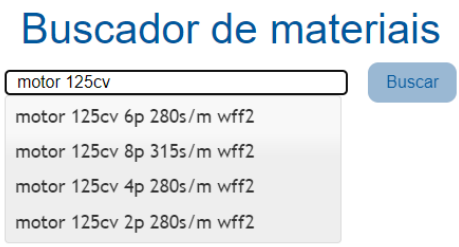
Fonte - Elaborado pelo autor

5.6.1 Modulo de autocomplete

O módulo de *auto-complete* foi implementado utilizando a biblioteca *jQuery*. Esta biblioteca processa uma lista pré-definida contendo os nomes de todos os materiais empregados no protótipo. Assim, quando um usuário inicia a digitação de um termo, o sistema automaticamente sugere uma caixa de seleção que exibe os produtos correspondentes à entrada do usuário, facilitando a busca e seleção dos itens desejados. A

imagem 29 mostra um caso da funcionalidade, em que é digitado “motor 125cv" e é listado os nomes dos motores que possuem tal potência.

Figura 29 – Modulo de autocomplete.



Fonte - Elaborado pelo autor

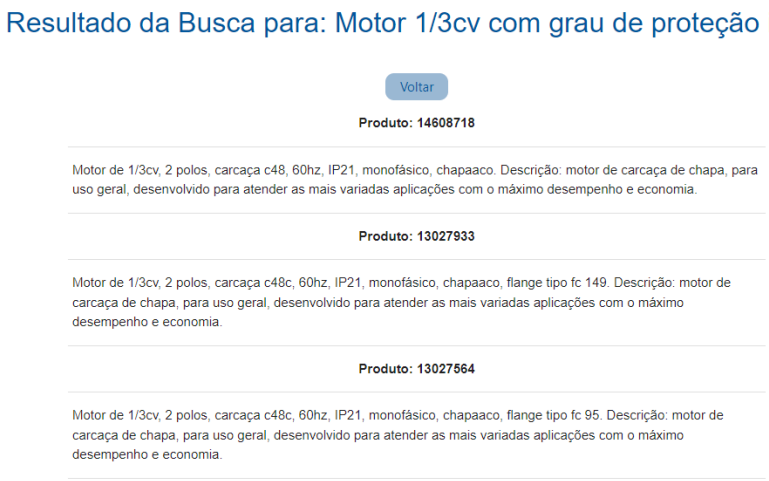
5.6.2 Tratamento do *input* do usuário

Antes de se passar o *input* do usuário ao processo de RAG é feito um tratamento simples do texto. Ele é transformado para letras minúsculas, os acentos são retirados, além de que pontuações sem sentido semântico, como pontuações duplicadas. Isso é feito pois como a busca semântica é feita entre os *embeddings* do *input* do usuário e dos dados da base, é necessário que eles estejam no mesmo padrão de texto. Como os dados seguiram esses mesmos tratamentos antes de se tirarem os *embeddings*, é necessário fazer o mesmo procedimento com o *input* do usuário.

5.6.3 Tela de resultados

A figura 30 mostra a tela com os resultados. Na parte superior da tela aparece a pesquisa do usuário, seguido dos resultados. Foi instruído ao modelo retornar código do produto seguido de uma breve descrição do material, com essa estrutura de retorno foi possível manipular os dados para serem exibidos em formato de tabela e centralizados na página. Com isso, foi atendido o requisito de exibição das informações de forma estruturada e uma interface intuitiva e amigável ao usuário.

Figura 30 – Tela com o um exemplo de resultado de busca.



Fonte - Elaborado pelo autor

6 RESULTADOS

Esta seção apresenta os resultados obtidos com a implementação do sistema de busca de materiais utilizando LLMs no contexto de um processo de RAG. Discute-se o desempenho do sistema completo, a qualidade das respostas geradas pelo modelo de LLM em cenários de uso real. Os resultados são quantificados utilizando casos testes desenvolvidos com base nos dados de materiais.

6.1 MÉTRICAS UTILIZADAS

Como o modelo utilizado, o *GPT-3.5 Turbo* da OpenAI, é pré-treinado e disponibilizado para uso geral, métricas típicas de avaliação de Modelos de Linguagem de Grande Escala (LLMs), como o *BLEU Score* ou o *ROUGE*, não são diretamente aplicáveis. Essas métricas são mais adequadas para tarefas de tradução automática e sumarização, onde a semelhança textual é fundamental. No entanto, na aplicação em questão, é essencial considerar tanto a precisão da informação recuperada quanto a eficácia do sistema em lidar com variações nos inputs de consulta. Portanto, métricas como Precisão no Top-k (*Precision*), Revocação (*Recall*), e a Medida F1 (*F1 Score*) são mais apropriadas para avaliar a adequação dos resultados fornecidos pelo sistema RAG.

6.2 CASOS TESTE

Foram feitos vários casos de testes utilizando a base de motores e de materiais. Desde casos mais gerais até casos mais complexos de busca. Foram pegos alguns materiais como exemplos para serem usados como testes e de seus resultados foram feitas as métricas.

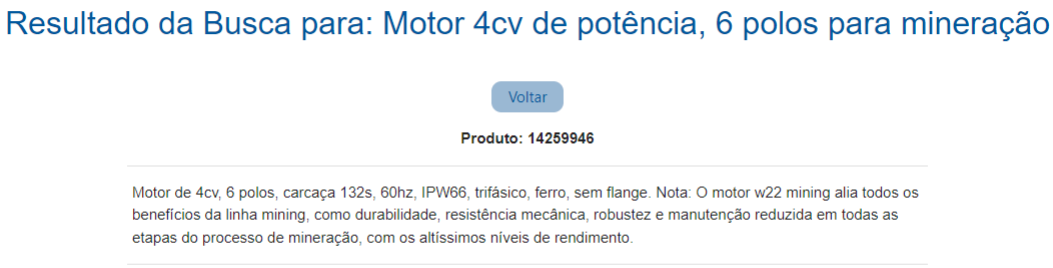
Para averiguar como o sistema se comporta com variações de *input*, foram feitos 2 variações de entrada para pesquisar um mesmo material, uma sendo extensa, com uma maior caracterização e descrição das características e outra com elas resumidas, usando-se os valores das características e pouca descrição textual, como no exemplo a seguir:

1. Motor 4cv de potência, 6 polos para mineração.
2. Motor 4cv, 6P da linha mining.

Os primeiros casos testes foram feito usando uma linguagem mais descritiva, em vez de dizer 4*cv* de potência e 6 polos, foram abreviados para 4cv e 6P. Esse exemplo é um caso simples, existe apenas um motor com essas características no *dataset* utilizado, nesse caso o sistema retorna corretamente os resultado com as

duas variações de descrição do motor. As figuras 31 e 32 mostram os resultados de duas formas de pesquisa diferentes para o mesmo material. Nesse caso tanto o *recall* quanto *Precision* seriam no valor de 1.

Figura 31 – Resultado do busca por motor



Fonte - Elaborado pelo autor

Figura 32 – Resultado do busca por motor

Fonte - Elaborado pelo autor

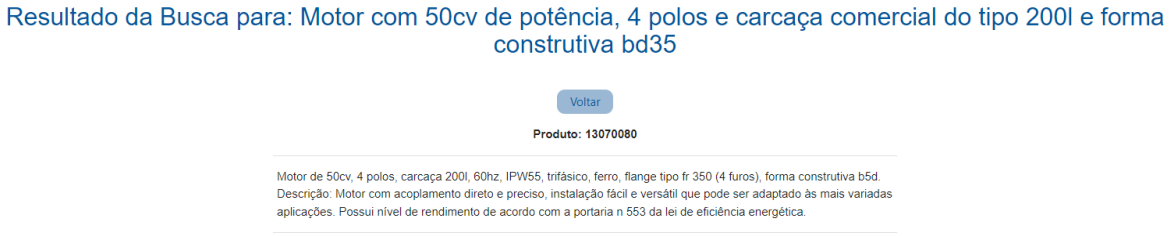
Já no caso de pesquisar uma carcaça específica de um motor, onde já existem dezenas com o mesmo nome, apenas a carcaça com o tipo diferente, o processo de busca se torna bem mais complexo e não retorna algumas vezes um valor correto. Foram feitos alguns casos de testes mais complexos como o a seguir:

1. Motor com 50cv de potência, 4 polos e carcaça comercial do tipo 200l e forma construtiva bd35.
2. Motor com 50cv, 4P e carcaça do tipo 200l e forma construtiva bd35.

A imagem 33 mostra o resultado para uma busca mais descritiva, retornou um motor com a potência, número de polos e carcaça correta, porém a forma construtiva

não, retornou um motor com forma construtiva b5d, em vez de bd35. Logo, o *recall* quanto *Precision* foram considerados como zero, apesar de serem motores praticamente iguais.

Figura 33 – Resultado do busca por motor



Fonte - Elaborado pelo autor

Já o segundo caso teste retornou um dos dois resultados corretos, figura 34. Assim, o *Precision* é de 1, já que retornou o resultado correto de apenas um resultado que foi retornado, enquanto *recall* é de 0,5, pois retornou 1 dos 2 resultados corretos existentes na base de dados.

Figura 34 – Resultado do busca por motor

Fonte - Elaborado pelo autor

Já para os materiais, excluindo motores, eles seguiram a mesma estratégia, um com uma linguagem mais descritiva e outro resumida.

Para o exemplo a seguir foi feita uma variação quanto as dimensões físicas da tampa.

1. Tampa vedação rolamento 72mm diâmetro e 9mm de altura.
2. Tampa vedação 72x9.

Os dois casos o modelo retornou o único material da base de dados, figuras 35 e 36, de modo que ambas as métricas ficaram com o valor 1

Figura 35 – Resultado do busca por tampa

Fonte - Elaborado pelo autor

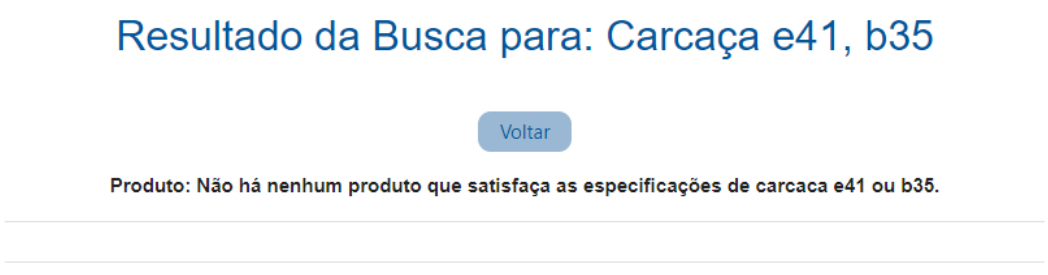
Figura 36 – Resultado do busca por tampa

Fonte - Elaborado pelo autor

Um caso que não retornou o material foi a pesquisa por um carcaça da família MAS com forma construtiva b35 e41.

1. Carcaça família, mas com forma construtiva b35 e41.
2. Carcaça e41, b35.

Figura 37 – Resultado não encontrado da carcaça do tipo MAS.

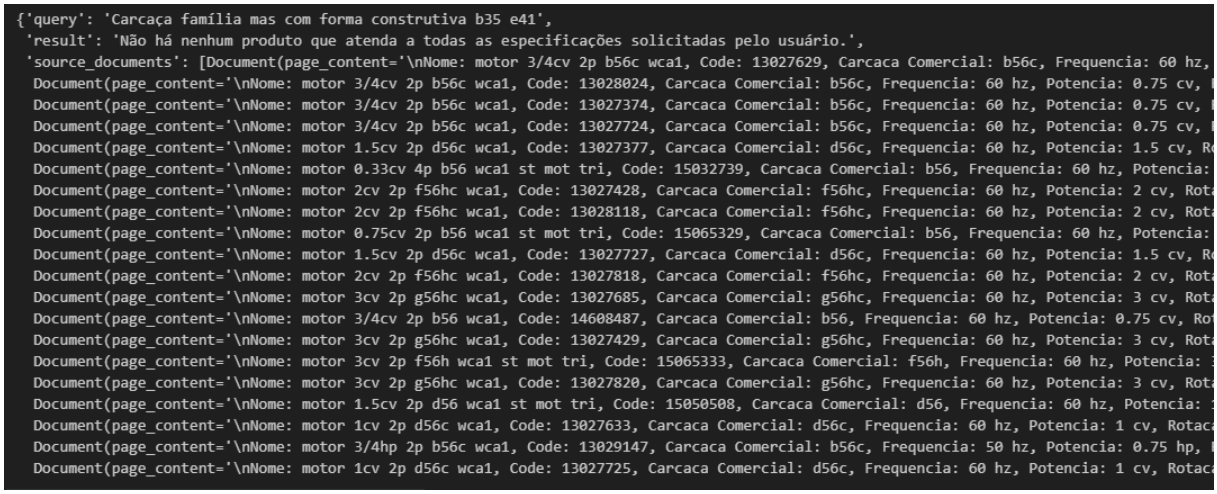


Fonte - Elaborado pelo autor

A figura 37 mostra a mensagem quando o modelo não encontra o material. Na verdade, é o processo de *retrieval* que não passa para o modelo o *chunk* em que está

o material carcaça, por algum motivo, talvez pelos cálculos de similaridade, esse *chunk* não um dos mais bem *rankeados* para ser enviado para a LLM elaborar a resposta. A figura 38 mostra os chunks retornados para o modelo. Percebe-se que nenhum deles está presente a carcaça pesquisada.

Figura 38 – Resultado do processo de *retrieval* do material carcaça.



Fonte - Elaborado pelo autor

Uma possível explicação seria que, como existem bastantes motores na base de dados, todos tem uma característica chamada carcaça, o que deve dificultar o processo de recuperação do exato *chunk* que contem o material carcaça especificado. Além de que, como essa similaridade estabelecida por meio de representações vetoriais, talvez a palavra, carcaça, seja mal presentada pelo modelo de *embeddings* selecionado do processo de *retrieval*.

6.3 ANÁLISE DOS RESULTADOS

Nesta seção, serão apresentados os resultados obtidos a partir da análise de métricas específicas aplicadas em casos de teste, comentários e *feedbacks* coletados durante a apresentação e avaliação do sistema por usuários, além de uma discussão dos resultados e das limitações do sistema.

6.3.1 Comentário dos usuários

Como esse projeto é uma prova de conceito e foi originado de um funil de inovação da WEG, após a finalização desse entregável, foi feita uma apresentação e testes com alguns gestores interessados no projeto para se ter um *feedback* útil de quem poderá utilizá-lo. Nela foram feitos alguns comentários do uso do sistema e possíveis aplicações que estavam de fora do escopo inicial do projeto.

Inicialmente, os gestores destacaram a eficiência do sistema em realizar buscas sem a necessidade de seguir critérios fixos ou uma ordem específica para inserir características de produtos ou materiais. Além da habilidade do sistema em fornecer resultados consistentes, independentemente da ordem das palavras inseridas na busca, e a capacidade de localizar um produto apenas por suas características principais, mesmo sem mencionar seu nome.

Adicionalmente, as possíveis aplicações inicialmente previstas para a ferramenta — o sistema de busca integrado ao ERP e o portal de *e-commerce* da WEG — foram validadas como opções viáveis para integração. Outras plataformas internas da WEG também foram identificadas como candidatas potenciais para incorporar a ferramenta, incluindo um sistema para configuração de produtos e materiais e outro sistema de busca que utiliza um método tradicional baseado em características categóricas.

Por último, os usuários sugeriram novas funcionalidades para o sistema de busca, como a possibilidade de ordenar os produtos por relevância e uma funcionalidade para buscar materiais ou produtos baseados em tipos específicos de aplicação. Esses comentários e sugestões dos usuários indicam um reconhecimento do potencial do sistema e oferecem direções para futuras melhorias e expansões do projeto.

6.3.2 Métricas dos casos testes

Para se ter uma forma de mensurar o desempenho do sistema foram feitos 30 casos testes com 2 variações de *input* para cada caso. Assim, pode-se ter uma avaliação mais geral da eficácia do sistema. Para cada caso foi anotada a precisão e o *recall* nas duas variações do mesmo material.

Caso	Precision	Recall	F1 Score
1 - Descrição extensa	0.66	0.59	0.623
2 - Descrição resumida	0.62	0.54	0.577

Tabela 1 – Comparação das métricas de avaliação para descrições extensas e resumidas.

A tabela 1 mostra quatro valores, representando métricas de avaliação para dois casos de teste diferentes: um com descrição mais extensa (caso número 1) e outro com descrição mais resumida (caso número 2). As métricas apresentadas são *precision* e *recall*.

Para o caso número 1 (com descrição mais extensa):

A precisão é de 0,66, o que indica que 66% dos materiais recomendados estavam corretos. O *recall* foi de 0,59, o que sugere que 59% materiais corretos totais foram identificados nas buscas.

Para o caso número 2 (com descrição mais resumida):

A precisão foi de 0,62, o que implica que 62% dos materiais recomendados estavam corretos. O *recall* foi de 0,54, o que significa que 54% dos itens corretos totais foram identificados nas buscas.

Analisando essas métricas, pode-se inferir que a descrição mais extensa, caso 1, resultou em um *precision* e *recall* ligeiramente superiores em comparação com a descrição mais resumida do caso 2. Isso pode indicar que uma descrição mais detalhada ajuda o sistema a recuperar com mais precisão os materiais corretos. No entanto, a diferença não é substancial, sugerindo que o sistema tem um bom desempenho em ambas as configurações para os 30 casos testados.

Foi calculado ainda uma métrica final que é o *F1 Score*, que combina o *precision* e *recall* em uma única métrica, sendo calculada como uma média harmônica da precisão e *recall*. Tem-se que os caso 1, de descrição maior, tiveram uma métrica um pouco maior, indicando que uma descrição mais extensa das características retorne um resultado melhor.

Apesar de o projeto ser uma prova de conceito, os resultados foram feitos seguindo métricas tradicionais e calculadas a partir de critérios bem rígidos. Sobre o valor das métricas em torno de 0,6, isso mostra que o sistema já pode ser utilizável na prática. Há de se considerar ainda que foram usados casos testes considerados difíceis, já que exigiam um alto nível de diferenciação de materiais, como no caso de uma forma construtiva específica de um motor. Outro ponto, é que avaliar esse sistema é uma tarefa subjetiva, uma pequena alteração no *input* do usuário pode trazer resultados diferentes, como uma descrição a mais ou escrita de forma diferente com mais semelhança com o que foi utilizada no banco de dados.

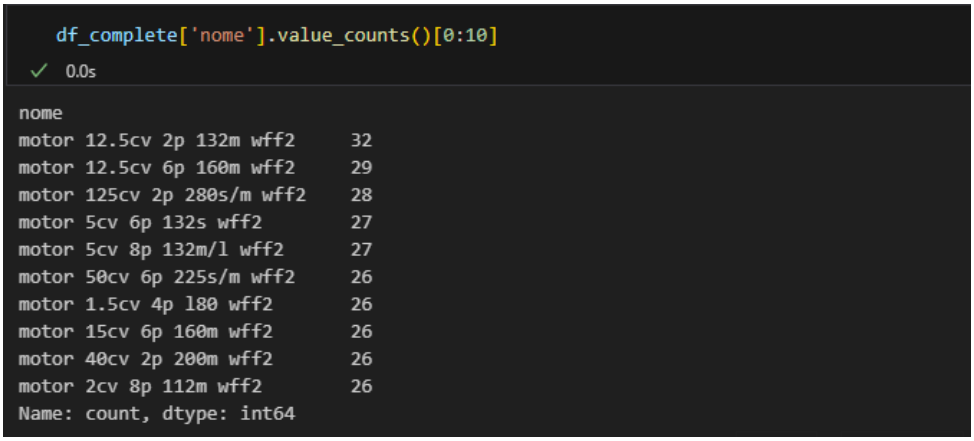
Além disso, tem a questão da qualidade dos dados, não foi utilizada uma base com uma descrição mais textual de certas características dos materiais, e sim, com mais códigos e siglas. Ainda nesse sentido, os casos testes não são representativos da quantidade total de materiais utilizados, teria que se encontrar uma forma de automatizar os testes, uma vez que foram feitos com 30 materiais, com 2 variações cada, de uma base de em torno de 5 mil. A seção a seguir discute com mais detalhes e exemplos algumas das limitações encontradas.

6.4 LIMITAÇÕES DO SISTEMA

Como apresentado anteriormente, o sistema de *retrieval* tem suas limitações. Como a base de materiais contem muitos dados e muitas vezes, no caso de motores, têm o mesmo nome para motores com diferenças na carcaça, flange, ou níveis de tensão, o sistema de *rankeamento* das respostas mais similares com a *query* do usuário sofre uma perda de informação por não encontrar o *chunk* correto, talvez uma pequena diferença em alguma característica não tenha um grau de representação suficiente nos *embeddings* quando se usa a métrica de similaridade. A imagem 39

uma amostra da quantidade de motores com mesmo nome.

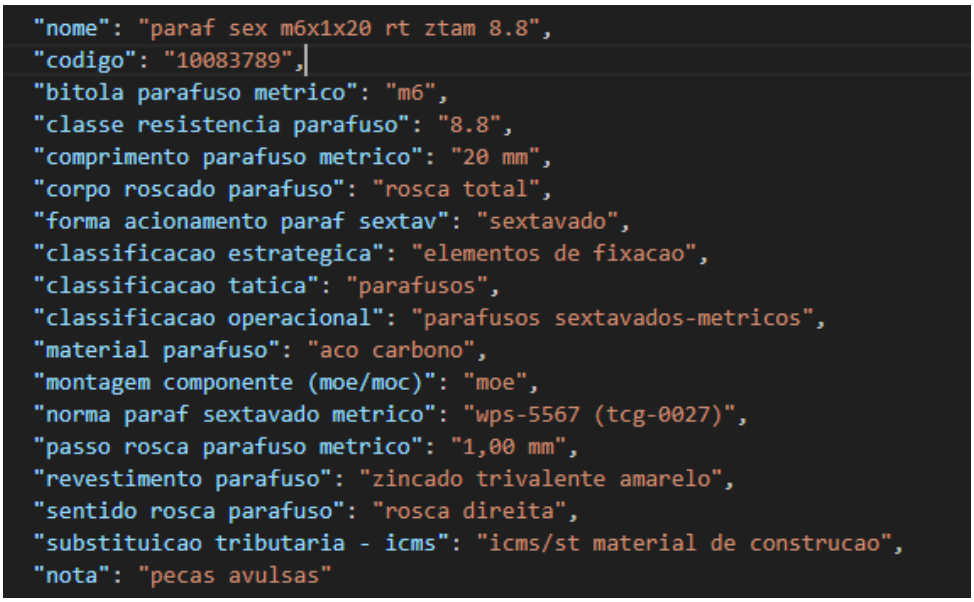
Figura 39 – Contagem de motores com mesmo nome na base de dados.



Fonte - Elaborado pelo autor

Nos dados de materiais, excluindo motores, as informações utilizadas contêm termos e abreviações que dificultam a localização de materiais pelo sistema se o usuário desconhece os termos técnicos utilizados na base de dados. A imagem 40 mostra um desses casos, o ideal seria um breve descrição do material em linguagem natural e não em formato de dicionário com chave e valor. Além disso, o nome do material é abreviado, o que dificulta o processo de recuperação da informação e entendimento das características pelo modelo de linguagem.

Figura 40 – Exemplo de um material da base de dados.



Fonte - WEG.

Outro ponto é a questão dos *embeddings*, a representação da informação em vetores de baixa dimensão é fundamental para o processo de RAG. Como esse modelo

utilizado, o *text-embedding-ada-002*, é treinado com textos em diversas linguagens, incluindo o português, o tipo de texto e documentos utilizados influenciam no mapeamento dos dados utilizados neste trabalho. Por serem dados técnicos, a representação deles em espaço multidimensional deve ser mais precisa do que algum texto descritivo ou narrativo. Variações mínimas, como a adição ou omissão de uma letra ou número, como no caso de um tipo específico de carcaça em um motor, pode significar um material diferente, mostrando a necessidade de uma representação vetorizada altamente precisa. Nesse contexto, surge a necessidade de desenvolver ou adaptar modelos de *embeddings* especificamente treinados para mapear descrições técnicas com maior precisão em representações vetoriais.

Além disso, importante comentar de um fenômeno inerente dos LLMs chamado de *hallucination* (alucinação) que é quando o modelo gera respostas sem sentido ou completamente fora do contexto esperado. Essas alucinações ocorrem principalmente quando há limitações na compreensão do contexto ou falta de dados para embasar a resposta. Além disso, modelos de linguagem de grande escala podem produzir alucinações devido à natureza estatística e probabilística de sua geração de texto (MANAKUL; LIUSIE; GALES, 2023). Em alguns testes realizados se observou que o RAG “inventa” alguns códigos de materiais ou troca alguma característica de certo motor ou material em raras ocasiões.

7 CONCLUSÃO

Neste trabalho, foram realizados o projeto e o desenvolvimento de um prova de conceito de uma ferramenta de busca de materiais por características técnicas na empresa WEG com o objetivo de facilitar a pesquisa de materiais e produtos no dia a dia tanto de colaboradores quanto de consumidores da empresa. Como o número de materiais e produtos é da ordem de milhares de itens, surgiu a necessidade de criar um sistema fora das regras rígidas e complexas para localizar um material nos sistemas internos na WEG e no *e-commerce* da empresa.

Para isso, com a relevância cada vez maior do uso de IA generativa, em particular do uso do famoso *ChatGPT* da empresa OpenAI, foi utilizada uma abordagem que alia o poder generativo do modelo de linguagem do GPT-3.5 Turbo com a busca semântica de uma base externa de dados de características de materiais. Esse método é chamado de *Retrieval Augmented Generation*, ou simplesmente (RAG), foi utilizado como um ferramenta de busca para os materiais da WEG.

Por ser uma abordagem nova, ela se mostrou bastante eficaz no sentido de proporcionar a busca por algum item descrevendo-o em linguagem natural e sendo possível encontrá-lo com mais de um comando de pesquisa. Porém, dada a complexidade da base de dados, muitos materiais parecidos e um extenso número de características pouco descritivas, ela se mostrou falha em certos casos específicos.

Um possível trabalho futuro seria de investigar avanços na representação vetorial de base de dados de conhecimento específico, ou modelos de *embeddings*, já que ela se mostrou não suficientemente representativa dos dados em algumas situações. Além disso, como é um campo tecnológico com avanços muito rápidos, pode-se aparecer novas soluções para problemas deste tipo com outro método ou até mesmo uma combinação de algum sistema de busca mais preciso combinado com o poder dos LLMs.

REFERÊNCIAS

ARTIFICIAL INTELLIGENCE, Beijing Academy of. Introducing BAAI's large English text-embedding. **BAAI**, v. 1, n. 5, 2020.

AUMÜLLER, Martin; BERNHARDSSON, Erik; FAITHFULL, Alexander. **ANN-Benchmarks: A Benchmarking Tool for Approximate Nearest Neighbor Algorithms**. [*S.l.*: *s.n.*], 2018. arXiv: 1807.05614 [cs.IR].

BAEZA-YATES, Ricardo; RIBEIRO-NETO, Berthier. **Modern Information Retrieval**. [*S.l.*]: ACM Press Books, 2011.

BRATANIC, Tomaz. **Knowledge Graphs LLMs: Fine-Tuning vs. Retrieval-Augmented Generation**. Accessed: 2023-11-02. 2022. Disponível em: https://neo4j.com/developer-blog/fine-tuning-retrieval-augmented-generation/.

BROWN, Tom B. *et al.* **Language Models are Few-Shot Learners**. [*S.l.*: *s.n.*], 2020. arXiv: 2005.14165 [cs.CL].

CHAPMAN, Peter; CLINTON, Julian; KERBER, Randy; KHABAZA, Tom; REINARTZ, Thomas; SHEARER, Colin; WIRTH, Rüdiger. **CRISP-DM 1.0: Step-by-step data mining guide**. [*S.l.*]: SPSS Inc, 2000.

CHROMA Documentation. Accessed: 2023-11-05. 2023. Disponível em: https://docs.trychroma.com/.

DAVENPORT, Thomas H.; PRUSAK, Laurence. **Working knowledge: How organizations manage what they know**. [*S.l.*]: Harvard Business Press, 1998.

DEERWESTER, Scott; DUMAIS, Susan T; FURNAS, George W; LANDAUER, Thomas K; HARSHMAN, Richard. Indexing by latent semantic analysis. **Journal of the American society for information science**, v. 41, n. 6, p. 391, 1990.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **arXiv preprint arXiv:1810.04805**, 2018.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *In*: PROCEEDINGS of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, jun. 2019. P. 4171–4186. DOI: 10.18653/v1/N19-1423. Disponível em: https://aclanthology.org/N19-1423.

FIESC. **Petrobras Weg**. 21th October, 2023. 2023. Disponível em: https://fiesc.com.br/pt-br/imprensa/petrobras-e-weg-firmam-parceria-para-desenvolvimento-do-maior-aerogerador-de-energia.

GARTNER. **Beyond ChatGPT: The Future of Generative AI for Enterprises**. Accessed: 2023-11-07. 2023. Disponível em: https://www.gartner.com/en/articles/beyond-chatgpt-the-future-of-generative-ai-for-enterprises.

GRAINGER, T.; TURNBULL, D.; IRWIN, M. **AI-Powered Search**. [*S.l.*]: Manning, 2023. ISBN 9781617296970. Disponível em: https://books.google.com.br/books?id=0EJMzwEACAAJ.

HAM, Laura. **A Gentle Introduction to Vector Search**. Accessed: 2023-10-28. 2022. Disponível em: https://opendatascience.com/a-gentle-introduction-to-vector-search/.

JACOBSON, Ivar; CHRISTERSON, Magnus; JONSSON, Patrik; ÖVERGAARD, Gunnar. **Object-Oriented Software Engineering: A Use Case Driven Approach**. [*S.l.*]: Addison-Wesley, 1992.

JOHNSON, Jeff; DOUZE, Matthijs; JÉGOU, Hervé. **Billion-scale similarity search with GPUs**. [*S.l.*: *s.n.*], 2017. arXiv: 1702.08734 [cs.CV].

KULAK, Daryl; GUINEY, Eamonn. **Use Cases: Requirements in Context**. [*S.l.*]: ACM Press/Addison-Wesley Publishing Co., 2000.

MANAKUL, Potsawee; LIUSIE, Adian; GALES, Mark J. F. **SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models**. [*S.l.*: *s.n.*], 2023. arXiv: 2303.08896 [cs.CL].

MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. **Introduction to Information Retrieval**. [*S.l.*]: Cambridge University Press, 2008.

MICROSOFT. **Política de privacidade de dados do Azure OpenAI**. [*S.l.*: *s.n.*], 2023. Acessado em: 2023-11-05. Disponível em: https://learn.microsoft.com/pt-br/legal/cognitive-services/openai/data-privacy?context=%2Fazure%2Fai-services%2Fopenai%2Fcontext%2Fcontext.

MIKOLOV, Tomas; CHEN, Kai; CORRADO, Greg; DEAN, Jeffrey. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013.

MUENNIGHOFF, Niklas; TAZI, Nouamane; MAGNE, Loic; REIMERS, Nils. MTEB: Massive Text Embedding Benchmark. **arXiv preprint arXiv:2210.07316**, arXiv, 2022. DOI: 10.48550/ARXIV.2210.07316. Disponível em: https://arxiv.org/abs/2210.07316.

OPENAI. **Introducing text-embedding-ada-002**. [*S.l.*: *s.n.*], 2021. Disponível em: https://platform.openai.com/docs/guides/embeddings/what-are-embeddings.

REYNOLDS, Laria; MCDONELL, Kyle. **Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm**. [*S.l.*: *s.n.*], 2021. arXiv: 2102.07350 [cs.CL].

ROSENBERG, Doug; STEPHENS, Matt. **Use Case Driven Object Modeling with UML: Theory and Practice**. [*S.l.*]: Apress, 2007.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game**. [*S.l.*]: Scrum.org, 2020.

SHEARER, Colin. The CRISP-DM Model: The New Blueprint for Data Mining. **Journal of Data Warehousing**, v. 5, n. 4, p. 13–22, 2000.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. **Attention Is All You Need**. [*S.l.*: *s.n.*], 2023. arXiv: 1706.03762 [cs.CL].

WEG. **Motores Elétricos Trifásico - Baixa Tensão Atmosferas Explosivas à Prova de Explosão (Ex-db)**. Portuguese. Acessado em: 2023-11-05. WEG. 2023a. Disponível em: https://www.weg.net/catalog/weg/BR/pt/Motores-

El%C3%A9tricos/Trif%C3%A1sico---Baixa-Tens%C3%A3o/Atmosferas-Explosivas/%C3%80-Prova-de-Explos%C3%A3o-%28Ex-db%29/c/BR_MT_3PHASE_LV_EXD. Acesso em: 5 nov. 2023.

WEG. **WEG History**. 28th October, 2023. 2023b. Disponível em: https://www.weg.net/institutional/BR/pt/history.

WEG. **Birmind Weg**. 21th October, 2023. 2023. Disponível em: https://www.weg.net/institutional/BR/pt/news/resultados-e-investimentos/weg-adquire-totalidade-das-acoes-da-birmind.

WEG. **Thisis Weg.** 21th October, 2023. 2022a. Disponível em: https://www.weg.net/institutional/BR/pt/this-is-weg.

WEG. **Weg in Numbers.** 21th October, 2023. 2022b. Disponível em: https://www.weg.net/institutional/BR/pt/weg-in-numbers.

YANG, Jingfeng; JIN, Hongye; TANG, Ruixiang; HAN, Xiaotian; FENG, Qizhang; JIANG, Haoming; YIN, Bing; HU, Xia. Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond, 2023. arXiv: 2304.13712 [cs.CL].

APÊNDICE A – INSTRUÇÃO USADA NO MODELO DE LINGUAGEM

Figura 41 – *Template* utilizado no sistema de busca de materiais.

```
template = """
Você é um end-point de uma API de um sistema de busca de materiais.
    Receberá informações sobre vários produtos, incluindo nome, có
   digo, características principais e uma descrição geral.Devolva
   uma lista contendo o código e uma descrição breve de cada
   produto. Apenas inclua os produtos que satisfaçam todas as
   especificações do usuário. Confira sempre se os produtos
   satisfazem TODAS as especificações do usuário. Confira se cada
   motor satisfaz TODAS as especificações do usuário das seguintes
   características: potência, polos, rotação, tensão, grau de prote
   ção, material, fases, flange e carcaça. Se tiver mais de um
   material, separe-os por colchetes.

Exemplo:
Se o usuário solicita: 'motor 10cv, 2 polos, carcaça comercial 132m
   ',
A resposta seria:
[10579387 : Motor de 10cv, 2 polos, carcaça 132m, 60hz, IP55, trifá
   sico, ferro, flange tipo c. Nota: Design à prova de explosão
   para ambientes com atmosferas explosivas.][13937209 : Motor de
   10cv, 2 polos, carcaça 132m, 60hz, IP55, monofásico, ferro. Nota
   : Adequado para energia doméstica ou rural com alto torque
   inicial.]
Outro exemplo:
Se o usuário solicita: 'motor 50cv 4p 200l wfx1',
A resposta seria:
[10579387 : Motor de 50cv, 4 polos, carcaça 200l, 60hz, IP55, trifá
   sico, ferro, flange tipo ff. Nota: Design à prova de explosão
   para ambientes com atmosferas explosivas.]
Outro exemplo:
Se o usuário solicita: 'kit centrifugador agua 36x100'
A resposta seria:
[10164711: Kit centrifugador de água da classe de partes e peças de
    motor elétrico, slinger nbr 36x100mm.]
Outro exemplo:
Se o usuário solicita: 'flange adaptador para motor c120'
Resposta esperada:
[14443293: Flange adaptador motor c120, fr200 da familia wcg20 do
   grupo motoredutor, modelo c-460 do tipo e tamanho da flange
   motor c120, fr200.]
{context}
Pergunta: {question}
Resposta: """
```

Fonte - Elaborado pelo autor