

UNIVERSIDADE DO VALE DO RIO DOS SINOS – UNISINOS
UNIDADE ACADÊMICA DE GRADUAÇÃO
CURSO DE ENGENHARIA DA COMPUTAÇÃO

SAMUEL MOLLING

**DB Advisor: Recomendador de Banco de Dados Baseado em
Inteligência Artificial**

SÃO LEOPOLDO

2023

SAMUEL MOLLING

**DB Advisor: Recomendador de Banco de Dados
Baseado em Inteligência Artificial**

Trabalho de Conclusão de curso apresentado como requisito parcial para obtenção do título de Bacharel em Engenharia da Computação, pelo curso de Engenharia da Computação da Universidade do Vale do Rio dos Sinos (UNISINOS).

Orientador: Prof. Dr. Márcio Miguel Gomes

SÃO LEOPOLDO

2023

RESUMO

Nesta dissertação, abordamos um desafio comum no campo da tecnologia da informação: a seleção do banco de dados mais adequado para necessidades específicas. A pesquisa inicia com uma análise detalhada da literatura, explorando a aplicação de inteligência artificial na escolha de bancos de dados. O objetivo principal é desenvolver uma ferramenta, baseada em algoritmos de aprendizado de máquina, que auxilie os usuários a identificar o banco de dados mais apropriado para suas demandas específicas.

Para isso, coletamos dados de diversos bancos de dados disponíveis no mercado e exploramos diferentes técnicas de aprendizado de máquina, incluindo Llama2, K-Nearest Neighbors (KNN), Random Forest e Regressão Logística. O Llama2 foi o foco principal devido à sua eficácia notável. Os modelos foram treinados com os dados coletados, e utilizamos métricas como acurácia, precisão, revocação, ROUGE e METEOR para avaliar o desempenho.

Os resultados demonstraram a eficácia do Llama2 na categorização precisa dos bancos de dados e na geração de recomendações relevantes. Além disso, uma análise detalhada dos resultados forneceu insights sobre os pontos fortes e as áreas de melhoria de cada algoritmo. Assim, esta pesquisa contribui significativamente para o campo da seleção de bancos de dados, apresentando uma ferramenta eficiente e robusta com o Llama2. Este estudo abre caminho para futuras pesquisas e otimizações no domínio, destacando a importância de continuar explorando e aprimorando técnicas de aprendizado de máquina para aplicações práticas.

Palavras-chave: Inteligência artificial. Banco de dados. Algoritmo de recomendação.

ABSTRACT

In this dissertation, we address a common challenge in the field of information technology: selecting the most appropriate database for specific needs. The research begins with a detailed analysis of the literature, exploring the application of artificial intelligence in choosing databases. The main objective is to develop a tool, based on machine learning algorithms, that helps users identify the most appropriate database for their specific demands.

To do this, we collected data from several commercially available databases and explored different machine learning techniques, including Llama2, K-Nearest Neighbors (KNN), Random Forest and Logistic Regression. Llama2 was the main focus due to its remarkable effectiveness. The models were trained with the collected data, and we used metrics such as accuracy, precision, recall, ROUGE and METEOR to evaluate performance.

The results demonstrated the effectiveness of Llama2 in accurately categorizing databases and generating relevant recommendations. Additionally, a detailed analysis of the results provided insights into the strengths and areas for improvement of each algorithm. Thus, this research contributes significantly to the field of database selection by presenting an efficient and robust tool with Llama2. This study paves the way for future research and optimizations in the domain, highlighting the importance of continuing to explore and improve machine learning techniques for practical applications.

Keywords: Artificial intelligence. Database. Recommendation algorithm.

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura do transformers.	30
Figura 2 – Treinamento do Llama 2 e Llama2-Chat.	33
Figura 3 – Arquitetura final do sistema.	36
Figura 4 – Distribuição categórica para o conjunto de dados coletado	39
Figura 5 – Arquitetura de pré-processamento e extração de características. . .	42
Figura 6 – Arquitetura do modelo de RF.	43
Figura 7 – Esquema para fornecer a resposta a cada usuário.	45
Figura 8 – Esquema para fornecer a resposta a cada usuário com tradução. . .	46
Figura 9 – Diagrama de fluxo de como o LangChain funciona	51
Figura 10 – Nuvem de palavras do treino.	53
Figura 11 – Histograma das 20 palavras com maior frequência.	53
Figura 12 – Histograma das 20 palavras com maior frequência.	54
Figura 13 – O ID dos rótulos para cada classe de conjunto de dados.	55
Figura 14 – O resultado da incorporação de palavras para o conjunto de treina- mento.	55
Figura 15 – Fluxograma dos processos de pré-processamento e codificação. . .	56
Figura 16 – Pesquisa em grade para ajustes de hiperparâmetros.	57
Figura 17 – Código da utilização do Random Forest	57
Figura 18 – Mapeamento das perguntas e classes	58
Figura 19 – Pesquisa em grade com os melhores parâmetros	59
Figura 20 – Função para treinamento usando k-folders	60
Figura 21 – Função para validação da pergunta pela API OpenAI	61
Figura 22 – Análise de Componentes Principais	61
Figura 23 – Hiperparâmetros do KNN.	62
Figura 24 – Hiperparâmetros do Random Forest.	62
Figura 25 – Hiperparâmetros da Regressão Logística.	63
Figura 26 – Função de teste do modelo	63
Figura 27 – Validação das métricas	65
Figura 28 – Desempenho - RF e TF-IDF.	68
Figura 29 – Curva ROC - RF e TF-IDF.	69
Figura 30 – Matriz de confusão - RF e TF-IDF.	69
Figura 31 – Resultado final - RF e TF-IDF.	70
Figura 32 – Métricas do modelo AWS.	71
Figura 33 – Curva ROC do modelo AWS.	72
Figura 34 – Matriz de confusão do modelo AWS.	72
Figura 35 – Métricas do modelo GCP.	72

Figura 36 – Curva ROC do modelo GCP.	73
Figura 37 – Matriz de confusão do modelo GCP.	73
Figura 38 – Métricas do modelo Azure.	73
Figura 39 – Curva ROC do modelo Azure.	74
Figura 40 – Matriz de confusão do modelo Azure.	74
Figura 41 – Métricas do modelo Outros.	74
Figura 42 – Curva ROC do modelo Outros.	75
Figura 43 – Matriz de confusão do modelo Outros.	75
Figura 44 – Resultado dos modelos.	75
Figura 45 – Curva ROC do modelo de Regressão Logística.	76
Figura 46 – Matriz de confusão do modelo de Regressão Logística.	77
Figura 47 – Resultado da execução do modelo.	78
Figura 48 – Resultado da execução do modelo com LangChain.	79

LISTA DE TABELAS

Tabela 1 – Trabalhos relacionados	22
Tabela 2 – Conjunto de dados descrito e informações correspondentes para o desenvolvimento do algoritmo.	38
Tabela 3 – Exemplo do conjunto de dados para o banco de dados DynamoDB.	40
Tabela 4 – Hiperparâmetros com seus valores	65
Tabela 5 – Hiperparâmetros com seus valores	66
Tabela 6 – Resultado dos modelos	81
Tabela 7 – Porcentagens de distribuição de idiomas	81

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial
CAGR	Compound Annual Growth Rate (Taxa de Crescimento Anual Composta)
SQL	Structured Query Language (Linguagem de Consulta Estruturada)
NoSQL	Not Only SQL (Não somente SQL)
IMDBS	In-Memory Database (Bancos de Dados em Memória)
RDBMS	Relational Database Management System (Sistema de Gerenciamento de Banco de Dados Relacionais)
CRUD	Create, Read, Update and Delete (Criar, Ler, Atualizar e Apagar)
ACID	Atomicity, Consistency, Isolation and Durability (Atomicidade, Consistência, Isolamento e Durabilidade)
TPC-C	Transaction Processing Performance Council - C (Conselho de Desempenho de Processamento de Transações - C)
TPC-E	Transaction Processing Performance Council - E (Conselho de Desempenho de Processamento de Transações - E)
TPC-H	Transaction Processing Performance Council - H (Conselho de Desempenho de Processamento de Transações - H)
TPC-A	Transaction Processing Performance Council - A (Conselho de Desempenho de Processamento de Transações - A)
YCSB	Yahoo! Cloud Serving Benchmark (Yahoo! Referência de serviço em nuvem)
YCSB+T	Yahoo! Cloud Serving Benchmark + Transactions (Yahoo! Referência de serviço em nuvem + Transações)
TATP	Transaction Processing Performance Council - Benchmark C (Conselho de Desempenho de Processamento de Transações - Benchmark C)
AS3AP	ANSI SQL Scalable and Portable Benchmark (Benchmark ANSI SQL Escalável e Portátil)

AWS	Amazon Web Services
GCP	Google Cloud Platform
SaaS	Software as a Service (Software como Serviço)
IaaS	Infrastructure as a Service (Infraestrutura como Serviço)
PaaS	Platform as a Service (Plataforma como Serviço)
TI	Tecnologia da Informação
URL	Uniform Resource Locator (localizador uniforme de recursos)
DBA	Database Administrator (Administrador de banco de dados)
CSV	Comma-separated values (Valores separados por vírgulas)
KNN	K-Nearest Neighbors (K-vizinhos mais próximos)
SQuAD	Stanford Question Answering Dataset (Conjunto de dados de respostas a perguntas de Stanford)
RDS	Amazon Relational Database Service (Serviço de banco de dados relacional da AWS)
QLDB	Amazon Quantum Ledger Database (Banco de dados de contabilidade quântica)
PNL	Programação Neurolinguística
AUROC	Característica de Operação do Receptor
NQ	Natural questions
ML	Machine learning
LLM	Large Language Models (Modelos de linguagem de larga escala)
TF-IDF	Term Frequency-Inverse Document Frequency
ROUGE	Recall-Oriented Understudy for Gisting Evaluation (Substituto Orientado à Revocação para Avaliação de Resumos)
METEOR	Metric for Evaluation of Translation with Explicit Ordering (Métrica para Avaliação de Tradução com Ordenação Explícita)
PCA	Principal Component Analysis (Análise de Componentes Principais)
PEFT	Parameter Efficient Fine-Tuning (Ajuste fino eficiente de parâmetros)

QLoRA Efficient Fine-tuning of Quantized LLMs (Ajuste fino eficiente de LLMs
quantizados)

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Tema	12
1.2	Delimitação do tema	12
1.3	Problema	13
1.4	Objetivos	13
1.4.1	Objetivo Geral	13
1.4.2	Objetivos Específicos	13
1.5	Justificativa	13
2	TRABALHOS RELACIONADOS	15
2.1	Discussão dos Trabalhos Relacionados	20
3	FUNDAMENTAÇÃO TEÓRICA	23
3.1	Fundamentos de bancos de dados	23
3.2	Inteligência Artificial (IA) e Aprendizado de máquina	24
3.3	Processamento de Linguagem Natural (PLN)	25
3.3.1	Lematização	25
3.3.2	Stemização	26
3.4	Modelo	26
3.4.1	Random Forest	27
3.4.2	KNN	27
3.4.3	Regressão logística	28
3.5	Análise de Componentes Principais (PCA)	28
3.6	Term Frequency-Inverse Document Frequency (TF-IDF)	29
3.7	Transformadores	29
3.8	Representações Codificadoras Bidirecionais de Transformadores (BERT)	31
3.9	Modelos de Linguagem de Grande Escala (LLMs)	31
3.10	Modelos de Linguagem de Ajuste Fino (Fine-tuned Language Models)	32
3.11	Llama 2	33
3.12	LangChain	33
4	METODOLOGIA	35
4.1	Arquitetura	35
4.2	Conjunto de dados	36

4.3	Conjunto de dados de pesquisa	38
4.4	Pré-processamento de dados	40
4.5	Modelo	42
4.6	Treinamento	43
4.7	Avaliação	44
4.8	Aplicativo para o usuário final	44
4.9	LLM Llama 2 afinada	46
4.9.1	Fonte de dados e Preparação	46
4.9.2	Hiperparâmetros e Fine-Tuning	47
4.10	LLM Llama 2 com LangChain	48
4.10.1	Quantização	48
4.10.2	Tokenização e Pipelines	49
4.10.3	LangChain	50
5	DESENVOLVIMENTO	52
5.1	Dataset	52
5.2	Tradução	54
5.3	Random Forest e TF-IDF	54
5.4	Árvore de decisão e TF-IDF	58
5.5	Random Forest, KNN e Regressão Logística com Bert	59
5.6	LLM Llama 2 afinada	64
5.7	LLM Llama 2 com LangChain	64
6	RESULTADOS	68
6.1	Random Forest e TF-IDF	68
6.2	Árvore de decisão e TF-IDF	70
6.3	Random Forest, KNN e Regressão Logística com Bert	71
6.4	LLM Afinada com Llama2	77
6.5	LLM Llama 2 com LangChain	78
7	CONSIDERAÇÕES FINAIS	82
	REFERÊNCIAS	84

1 INTRODUÇÃO

A história dos sistemas de resposta a perguntas remonta à década de 1950, quando os computadores começaram a ser utilizados para tarefas de processamento de linguagem natural (QU et al., 2019). Desde então, diversos avanços têm sido realizados nessa área. A utilização de técnicas de análise semântica e sistemas especialistas baseados em regras adicionou complexidade e sofisticação aos sistemas de resposta a perguntas (QU et al., 2019). Com o desenvolvimento da web e tecnologias associadas, como a Web Semântica e a Web 2.0, surgiram sistemas mais complexos e sofisticados (YANG; YU; ZHOU, 2020).

Nos últimos anos, o aprendizado profundo (Deep Learning - DL) e as redes neurais artificiais (Artificial Neural Network - ANN) têm impulsionado avanços adicionais nos sistemas de resposta a perguntas, permitindo uma compreensão mais precisa de perguntas complexas e a geração de respostas mais precisas (QU et al., 2019). Além disso, técnicas estatísticas e probabilísticas têm sido aplicadas com sucesso nesse contexto, utilizando avanços em aprendizado de máquina (ML), processamento de linguagem natural (PLN) e recuperação de informações (QU et al., 2019).

A explosão de dados nos últimos anos impulsionou a necessidade de soluções eficientes para o armazenamento e gerenciamento dessas informações. A escolha adequada de um banco de dados tornou-se crucial para garantir o desempenho e a eficiência dos sistemas. Nesse contexto, a utilização de sistemas de inteligência artificial tem se mostrado uma abordagem promissora para auxiliar na seleção do banco de dados mais adequado às necessidades de cada aplicação de software.

1.1 TEMA

O tema deste trabalho é o desenvolvimento de um sistema de inteligência artificial para auxiliar na escolha do banco de dados mais adequado às necessidades de cada aplicação de software.

1.2 DELIMITAÇÃO DO TEMA

O foco deste trabalho está na criação de um sistema baseado em inteligência artificial e técnicas de processamento de linguagem natural para recomendar o banco de dados ideal, levando em consideração requisitos específicos, como segurança, disponibilidade, desempenho, escalabilidade, custo, entre outros.

1.3 PROBLEMA

O processo de seleção de um banco de dados adequado pode ser complexo e desafiador, dada a variedade de opções disponíveis e os diferentes requisitos de cada empresa. A falta de orientação nesse processo pode levar a escolhas ineficientes e impactar negativamente o desempenho dos sistemas.

1.4 OBJETIVOS

1.4.1 OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver um sistema de inteligência artificial capaz de recomendar o banco de dados mais adequado para cada aplicação de software, levando em consideração requisitos específicos.

1.4.2 OBJETIVOS ESPECÍFICOS

Para complementar o objetivo geral, a seguir encontram-se os objetivos específicos:

- a) Realizar uma revisão da literatura sobre técnicas de inteligência artificial aplicadas à sistemas de recomendação.
- b) Coletar dados e informações relevantes sobre diferentes bancos de dados disponíveis no mercado.
- c) Desenvolver uma técnica para avaliar as características dos bancos de dados e recomendar a melhor opção para cada caso.
- d) Avaliar a eficácia e a precisão do algoritmo proposto por meio de métricas apropriadas.

1.5 JUSTIFICATIVA

A escolha adequada de um banco de dados é essencial para garantir a eficiência, a segurança e o desempenho dos sistemas. No entanto, esse processo pode ser complexo e sujeito a erros, especialmente diante da diversidade de opções disponíveis. Portanto, a criação de um sistema de inteligência artificial para auxiliar nessa escolha pode trazer benefícios significativos, como otimização de recursos, redução de custos e melhoria na qualidade dos sistemas.

Além disso, este trabalho contribuirá para o avanço da área de inteligência artificial aplicada a bancos de dados, fornecendo uma solução prática e embasada em

dados para auxiliar os desenvolvedores, dbas e arquitetos de soluções na tomada de decisão.

2 TRABALHOS RELACIONADOS

Com o objetivo de realizar uma revisão abrangente e atualizada dos estudos relacionados ao tema abordado nesta monografia, foi adotada uma metodologia de revisão sistemática. Inicialmente, foram definidos os termos de pesquisa que serviram como base para a busca, entre eles: "NoSQL", "Banco de Dados Não-Relacional", "Bancos de Dados Distribuídos", "Banco de Dados para Big Data", "Escolha de Banco de Dados", "Comparação de Bancos de Dados", "Armazenamento de Dados em Nuvem", "Escalabilidade em Bancos de Dados", "Desempenho de Bancos de Dados", "Métricas de Bancos de Dados", "LangChain" e "Llama" que são domínios de maior interesse para este trabalho. Estes termos guiaram a pesquisa nas bases de dados do Google Scholar, IEEE Explorer e ScienceDirect, as quais foram escolhidas devido à sua abrangência e relevância acadêmica. Dessa forma é possível identificar técnicas utilizadas em cada situação e onde se encaixam, e os problemas que ainda existem e podem ser resolvidos.

Após a busca inicial, foi necessário aplicar critérios de inclusão e exclusão para filtrar os estudos. Foram considerados trabalhos publicados nos últimos 5 anos que abordassem diretamente os termos de pesquisa selecionados e que estivessem disponíveis integralmente. Por outro lado, trabalhos que estavam duplicados ou que não tivessem relação direta com o tema foram excluídos.

Após essa seleção, os estudos remanescentes foram analisados em detalhe. As informações relevantes foram extraídas e organizadas em uma tabela. Os detalhes e a relação dos estudos selecionados podem ser visualizados na Tabela 1.

O estudo do autor (ZAGAN; DANUBIANU, 2021) apresenta os conceitos de data lake, uma tecnologia de armazenamento de grande volume de dados estruturados e não estruturados em um único lugar, com alta escalabilidade e baixo custo. O artigo discute os desafios associados ao armazenamento tradicional de dados e como a abordagem do data lake resolve esses desafios, permitindo que as organizações gerenciem e processem grandes volumes de dados com maior facilidade e eficiência. O autor também apresenta as vantagens de utilizar data lakes na nuvem, como a facilidade de rápida criação e a redução de custos operacionais. No entanto, o artigo destaca desafios relacionados a custo em longo prazo, uma vez que os provedores de nuvem cobram das empresas pelo armazenamento no prazo, o que pode tornar a nuvem mais cara no longo prazo. É importante notar que esses custos dependem de cada empresa, pois existem fatores que podem gerar ainda mais custos dentro de uma infraestrutura local.

O artigo também descreve os principais recursos de uma solução de cloud data lake e destaca os motivos para a migração dos dados para data lakes na nuvem, trazendo benefícios como a capacidade de lidar com diferentes fontes de dados, facilidade de escalabilidade, eficiência no processamento dos dados e gerenciamento do serviço. Por fim, o autor conclui que a abordagem de cloud data lake é uma solução promissora para gerenciar grandes volumes de dados em um ambiente de nuvem escalável, seguro e econômico.

A pesquisa realizada por (MONJARAS; BCNDEZÚ; RAYMUNDO, 2019) apresenta uma síntese da história dos bancos de dados, desde 1970, quando os bancos relacionais eram a opção mais comum. Contudo, com o crescimento exponencial dos dados, os bancos relacionais têm enfrentado problemas de performance em decorrência da grande quantidade de informações processadas. Os autores apresentam uma análise dos diferentes tipos de bancos de dados, como SQL, NoSQL e In-Memory, destacando suas características e modelos de dados. O problema apontado é que existem muitos bancos de dados com características distintas, o que torna difícil conhecer e compreender as suas particularidades. Isso pode levar o usuário a escolher um banco de dados inadequado, ocasionando problemas nas aplicações, como redução de desempenho, segurança e escalabilidade, entre outros fatores cruciais.

Dessa forma, o estudo propõe o conceito de árvore de decisão como uma ferramenta de classificação para representar um ponto de entrada com diversas saídas diferentes. Os autores sugerem um algoritmo que facilite a escolha do mecanismo de gerenciamento de dados mais apropriado, considerando o Microsoft SQL Server, MySQL, MongoDB, RedisDB, MariaDB e MonetDB. A árvore de decisão possui um ponto de entrada chamado de Caso de Uso, no qual existem diversas saídas, cada uma apontando para uma tecnologia de banco de dados diferente. A validação desse algoritmo foi realizada por meio da simulação das operações de CRUD em cada um dos bancos de dados, com os tempos de resposta para cada operação, considerando um volume inicial de um milhão de requisições. Tabelas comparativas foram elaboradas para analisar os resultados, e os autores concluíram que os bancos SQL são mais adequados para as propriedades ACID, uma vez que os bancos NoSQL não podem manter a propriedade das transações. No entanto, cada banco teve um resultado diferente. Os bancos SQL apresentaram uma precisão nas operações de leitura de 96,26%, enquanto os NoSQL tiveram 91,83% e os IMDBS ficaram em 93,87%. Na operação de inserção de dados, os resultados foram de 98,98%, 98,23% e 100% para SQL, NoSQL e IMDBS, respectivamente.

A pesquisa desenvolvida por (NAMDEO; SUMAN, 2022) apresenta os desafios enfrentados pelos bancos de dados devido ao aumento exponencial da quantidade de dados gerados. A cada sessenta segundos, bilhões de informações são criadas

e transformadas em dados. Os bancos de dados relacionais possuem limitações significativas, tanto em relação à performance quanto ao formato fixo de seus esquemas relacionais, onde os dados devem ser estruturados de maneira uniforme. Cada vez mais empresas estão optando pelos bancos de dados NoSQL devido às facilidades que eles proporcionam, o que torna necessária a migração de bancos de dados legados para essa nova tecnologia. Esse processo é conhecido como reengenharia, que consiste em converter softwares existentes em uma versão aprimorada, que solucione problemas identificados na versão anterior.

Os autores também apresentam o modelo de custo da reengenharia para bancos de dados, no qual diversos fatores devem ser considerados para a realização da reengenharia. Esses fatores incluem benefícios financeiros, custos fixos, custos variáveis, custos de redesenho do banco de dados, custos de tradução de consultas SQL do software antigo, custos de migração do banco de dados e um fator de risco. Com esses elementos de custo em mente, é possível elaborar uma equação que permite estimar o benefício da reengenharia do banco de dados. Os autores concluem que muitas empresas que utilizam RDBMS estão sofrendo com a degradação do desempenho, aumento no tempo de resposta e aumento dos recursos de hardware. A realização desse cálculo pode auxiliar as empresas na previsão dos benefícios da mudança de seu banco de dados legado para uma opção que atenda melhor às suas necessidades.

O objetivo do trabalho do autor (QU et al., 2022) é examinar os benchmarks atualmente disponíveis para avaliar bancos de dados transacionais distribuídos e questionar sua adequação para essa finalidade. Os autores iniciam com uma introdução sobre os bancos de dados distribuídos, discutindo suas vantagens e desafios. Eles destacam que, com o aumento da quantidade de dados gerados em aplicações como a bolsa de valores, há uma crescente necessidade de bancos de dados transacionais distribuídos. Em seguida, os autores apresentam dois tipos de bancos de dados e suas arquiteturas correspondentes, juntamente com seus pontos fracos. Além disso, apresentam uma revisão detalhada dos benchmarks existentes para bancos de dados distribuídos, como TPC-C, TPC-E, TPC-H, YCSB, entre outros. Eles discutem as limitações desses benchmarks na avaliação adequada dos bancos de dados transacionais distribuídos e como eles não levam em conta requisitos específicos, como escalabilidade, consistência, disponibilidade, latência, entre outros.

Compreendendo que os benchmarks atuais falham em testar corretamente os bancos de dados distribuídos, os autores propõem alguns benchmarks para esses tipos de bancos de dados. O AS3AP foi criado para compensar os problemas encontrados no benchmark de Wisconsin, cobrindo mais métricas como testes multiusuários, mais tipos de dados, distribuições de dados, entre outros. O YCSB+T, uma extensão do YCSB,

que pretende avaliar a capacidade transacional de bancos de dados NoSQL, também é mencionado. Além destes, foram adicionados mais cinco benchmarks clássicos já utilizados em projetos: DebitCredit, TPC-A, TATP, TPC-C e TPC-E, e também dois com propósitos específicos, o SmallBank e PeakBench, que avaliam o desempenho de diferentes protocolos serializáveis sob um isolamento de instantâneo e disputas entre esperas de transações, respectivamente. Por fim, após a realização de testes de benchmarks clássicos, os autores concluem que para medir o desempenho dos bancos de dados transacionais distribuídos, os benchmarks mais recomendados são o YCSB+T, TATP, SmallBank e PeakBench.

O estudo abordado em (SARASWAT; TRIPATHI, 2020) apresenta uma análise comparativa das soluções de computação em nuvem oferecidas pelos principais provedores de nuvem: Amazon Web Services (AWS), Microsoft Azure e Google Cloud (GCP). Os autores fornecem uma visão geral da arquitetura de computação em nuvem, incluindo os modelos de serviço SaaS, PaaS e IaaS e os modelos de implantação: Público, Privado, Híbrido e Comunitário. São apresentadas as características de cada um desses modelos e sua importância na escolha correta de um provedor de nuvem. O artigo também discute conceitos como computação de utilidade, arquitetura orientada a serviços, escalabilidade, confiabilidade, versatilidade, virtualização, computação autônoma e manutenção.

A metodologia utilizada no estudo é composta por alguns itens essenciais que um cliente deve considerar antes de decidir utilizar um provedor de nuvem, como serviços de infraestrutura e computação, serviços de tecnologias de rede, tecnologias de armazenamento, suporte à base de dados, serviços de backup e ferramentas-chave. Após validar vários serviços e itens dos provedores em questão, foram elencados os parâmetros gerais para avaliação, incluindo o tipo de modelo de serviço, foco, regiões de alta disponibilidade, pontos de venda e velocidade de resposta, segurança, natureza dos serviços, alcance, maior desvantagem e preço.

As conclusões do estudo indicam que a AWS é adequada para empresas maiores que necessitam de soluções versáteis e globais, a Azure é indicada para aqueles que estão usando a nuvem pela primeira vez, que possuem infraestrutura limitada e precisam de soluções híbridas, enquanto a GCP se concentra em aplicativos baseados na internet, com política de preços flexíveis e foco no modelo de contêineres. O estudo apresenta uma avaliação abrangente e comparativa dos principais provedores de nuvem, fornecendo informações valiosas para os clientes que procuram tomar decisões informadas na escolha de um provedor de nuvem.

O artigo de (CHEN; LEE, 2018) aborda o aumento no volume de dados coletados pelas empresas e como isso tem levado ao aumento do uso de bancos de dados NoSQL em detrimento de bancos de dados relacionais, os quais sofrem um gargalo

de aceleração devido às operações de junção de dados em massa. No entanto, como existem muitos tipos de bancos de dados NoSQL, torna-se crucial escolher um banco de dados apropriado para cada empresa específica, a fim de evitar impactos negativos no desempenho dos negócios.

O artigo apresenta uma comparação dos formatos e recursos de dados, além de listar os produtos reais para cada categoria de banco de dados NoSQL, propondo princípios para ajudar as empresas a escolher um banco de dados NoSQL adequado para seus problemas e desafios específicos de negócios. Segundo os autores, existem quinze diferentes tipos de bancos de dados NoSQL e para escolher um banco de dados NoSQL adequado, uma empresa deve entender seus problemas, metas e desafios atuais, determinar se continuará a usar um SQL atual ou mudará para um NoSQL, selecionar uma categoria apropriada de banco de dados NoSQL com base nos requisitos e recursos de negócios, descobrir quais bancos de dados NoSQL são mais discutidos na internet, de acordo com o DB-Engines Ranking, e, finalmente, selecionar o banco de dados NoSQL mais apropriado com base nas necessidades da empresa e nas vantagens e desvantagens dos bancos de dados disponíveis.

O artigo também apresenta alguns casos de uso e suas melhores opções de bancos de dados NoSQL. Portanto, a escolha adequada do banco de dados NoSQL pode ser determinante para o sucesso dos negócios, e as empresas devem levar em consideração esses princípios para escolher o banco de dados NoSQL mais apropriado para suas necessidades específicas.

O documento intitulado "Financial News Analytics Using Fine-Tuned Llama 2 GPT Model", de autoria de Bohdan M. Pavlyshenko, explora a adaptação e aplicação do modelo de linguagem Llama 2 GPT (LLM) no domínio específico das notícias financeiras. O Llama 2 GPT, uma evolução notável na área de modelos de linguagem, já provou ser eficiente em várias tarefas de processamento de linguagem natural. No entanto, para abordar desafios específicos relacionados às notícias financeiras, é essencial adaptá-lo ao contexto relevante. Para isso, o autor emprega uma técnica de fine-tuning utilizando o método PEFT/LoRA.

A análise de notícias financeiras é fundamental para tomadores de decisão, investidores e analistas, pois fornece insights valiosos sobre movimentos e tendências de mercado. No entanto, o volume maciço e a complexidade dessas notícias tornam a análise manual uma tarefa árdua e demorada. Neste estudo, o Llama 2 GPT é ajustado para desempenhar funções como análise de texto sob a perspectiva do mercado financeiro, destaque de pontos cruciais, sumarização e extração de entidades nomeadas com sentimentos associados. Os resultados iniciais indicam que o modelo ajustado não apenas facilita uma análise mais rápida e eficaz, mas também fornece estruturas de dados úteis (como JSON) que podem ser facilmente integradas a sistemas

de análise financeira. A capacidade do modelo de extrair sentimentos para entidades nomeadas oferece um potencial adicional, pois esses sentimentos podem ser usados como características preditivas em modelos de aprendizado de máquina supervisionado, auxiliando na previsão de tendências e movimentos de mercado.

Essa abordagem representa um avanço significativo na automação e precisão da análise de notícias financeiras, demonstrando o potencial dos modelos de linguagem de última geração quando ajustados para domínios específicos, na Tabela 1 é possível visualizar os trabalhos, cenários utilizados, algoritmos e métricas de cada um dos trabalhos relacionados comentados anteriormente.

2.1 DISCUSSÃO DOS TRABALHOS RELACIONADOS

A escolha do local apropriado para armazenamento de dados é uma decisão crítica que influencia diretamente a disponibilidade, segurança, escalabilidade e custo de gestão dos dados. Diversos estudos, como os mencionados na Tabela 1, têm explorado técnicas e algoritmos para otimizar essa escolha. No entanto, muitas dessas abordagens avaliam apenas um conjunto limitado de métricas ou se concentram em softwares específicos de armazenamento. Por exemplo, o estudo de (MONJARAS; BCNDEZÚ; RAYMUNDO, 2019) propõe uma árvore de decisão para facilitar a seleção de um banco de dados adequado, enquanto (NAMDEO; SUMAN, 2022) destaca os desafios dos bancos de dados relacionais e a necessidade de migração para bancos de dados NoSQL.

A complexidade da decisão é agravada pela variedade de bancos de dados disponíveis, cada um com suas características e vantagens. Além disso, a crescente tendência de armazenamento em nuvem, como discutido por (SARASWAT; TRIPATHI, 2020), introduz novas variáveis na equação, como custos operacionais e benefícios de escalabilidade. Diante desses desafios, este trabalho sugere a adoção de sistemas de inteligência artificial, que, através de técnicas de processamento de linguagem natural e aprendizado de máquina, podem fornecer recomendações precisas e personalizadas para cada cenário. A ideia é que, ao compreender profundamente as técnicas e soluções existentes e suas assertividades, possamos fundamentar melhor as decisões tomadas na proposta deste modelo e nas métricas a serem adotadas.

Adicionalmente, a utilização de sistemas de inteligência artificial permitirá a atualização constante e o aprimoramento contínuo do sistema, de forma a manter a relevância e a efetividade das recomendações ao longo do tempo. Dessa forma, espera-se que este trabalho contribua significativamente para a escolha adequada do local de armazenamento de dados, fornecendo soluções precisas e personalizadas que ajudarão as empresas a otimizar seus processos de armazenamento e gestão de

dados, garantindo maior eficiência e competitividade no mercado.

Tabela 1 – Trabalhos relacionados

Citação	Cenário	Algoritmo	Métrica
(ZAGAN; DANUBIANU, 2021)	Utilização do data lake na nuvem	Análise comparativa	Custo, performance, segurança
(MONJARAS; BCNDEZÚ; RAYMUNDO, 2019)	Algoritmo de árvore de decisão	Árvore de decisão	Inputs do usuário, performance no CRUD, tempo de resposta
(NAMDEO; SUMAN, 2022)	Modelo de custo de reengenharia de RDBMS para NoSQL	Reengineering = Reverse engineering + Δ + Forward engineering	Custo e benefício
(QU et al., 2022)	Benchmarks para bancos de dados transacional distribuídos	Benchmarks	Esquema de dados, carga de trabalho, métricas de desempenho em armazenamento, consulta e agendamento
(SARASWAT; TRIPATHI, 2020)	Análise comparativa entre os três principais provedores de nuvem: AWS, GCP e Azure	Análise comparativa	Tipo de serviço, foco, número de regiões, capacidade de responder rapidamente, segurança, natureza dos serviços, alcance, maior desvantagem, preço e pagamento
(CHEN; LEE, 2018)	Estudo para utilização de bancos de dados NoSQL e seus casos de uso	Análise comparativa	Não especificado
(PAVLYSHENKO, 2023)	Análise multitarefa de notícias financeiras	Fine-tuning do modelo de linguagem "Llama 2 GPT" usando o método PEFT/LoRA	Não especificado

Fonte: Elaborado pelo autor.

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo fornecer conceitos importantes para o entendimento do sistema de inteligência artificial (IA) voltado à escolha de banco de dados. Ao concluir a leitura deste capítulo, o leitor terá adquirido um embasamento teórico sólido sobre bancos de dados, suas classificações e requisitos, bem como uma compreensão dos conceitos fundamentais da inteligência artificial aplicada à escolha de banco de dados. Essa base teórica será fundamental para o desenvolvimento do sistema proposto no próximo capítulo, visando auxiliar organizações na seleção do banco de dados mais adequado às suas necessidades específicas.

3.1 FUNDAMENTOS DE BANCOS DE DADOS

Bancos de dados são essenciais para organizar e armazenar grandes quantidades de informações de forma estruturada. Existem vários tipos, como os relacionais, que organizam os dados em tabelas e são populares devido à sua eficiência em consultas e integridade dos dados (DATE, 2004). Os bancos de dados orientados a objetos, por outro lado, são mais adequados para sistemas de gerenciamento de conteúdo multimídia (RAMAKRISHNAN; GEHRKE, 2008). A linguagem SQL é uma ferramenta comum para interagir com bancos de dados relacionais, mas os bancos de dados NoSQL, que armazenam dados não estruturados ou semiestruturados, têm ganhado espaço por sua flexibilidade e escalabilidade (MACHADO, 2020).

A tendência atual é a utilização de sistemas distribuídos e armazenamento em nuvem, que proporcionam alta disponibilidade e desempenho (ROB; CORONEL, 2011). No projeto de um banco de dados, aspectos como estrutura dos dados, normalização e definição de restrições são fundamentais para garantir a integridade dos dados (RAMAKRISHNAN; GEHRKE, 2008). A segurança dos dados é outra preocupação crítica, exigindo medidas como criptografia e controle de acesso (MACHADO, 2020). Os sistemas de gerenciamento de banco de dados (SGBDs) facilitam a administração, fornecendo interfaces para várias funções e são classificados em várias categorias, como relacionais e NoSQL (ROB; CORONEL, 2011).

No mundo atual, com grandes volumes de dados sendo processados em tempo real, a escalabilidade dos bancos de dados é vital. Técnicas como indexação e particionamento são usadas para otimizar consultas e melhorar a eficiência (RAMAKRISHNAN; GEHRKE, 2008). Protocolos como ODBC e JDBC ajudam na integração de bancos de dados com outras tecnologias (MACHADO, 2020). Bancos de dados in-memory representam um novo paradigma, oferecendo acesso mais rápido aos dados, enquanto

a computação em nuvem tem revolucionado a maneira como os bancos de dados são hospedados e gerenciados (DATE, 2004). A constante evolução e adaptação dos bancos de dados às mudanças nas necessidades são imperativas para sua relevância contínua (RAMAKRISHNAN; GEHRKE, 2008).

3.2 INTELIGÊNCIA ARTIFICIAL (IA) E APRENDIZADO DE MÁQUINA

A Inteligência Artificial (IA) representa um avanço na ciência da computação, buscando desenvolver sistemas que realizem tarefas típicas da inteligência humana, como raciocinar, aprender, decidir e solucionar problemas. Uma subárea em destaque é o Aprendizado de Máquina (ML), que foca no desenvolvimento de algoritmos e técnicas que permitem às máquinas aprender a partir dos dados e tomar decisões baseadas em padrões identificados (LUDERMIR, 2021). Existem diferentes abordagens de ML, como o Aprendizado Supervisionado, onde o algoritmo é treinado com dados rotulados (COZMAN; KAUFMAN, 2022); o Aprendizado Não Supervisionado, que busca padrões ocultos nos dados (REIS; MIRANDA; DAMY, 2019); e o Aprendizado por Reforço, aplicado principalmente em jogos e robótica (LUDERMIR, 2021).

Modelos de ML, como as redes neurais artificiais, são inspirados no funcionamento cerebral e têm sido aplicados em diversas áreas, como reconhecimento de padrões e processamento de linguagem natural (FIGUEIREDO; GIAMMELLA, 2018). O Aprendizado Profundo, que utiliza redes neurais com múltiplas camadas, é capaz de processar informações complexas, sendo eficaz em tarefas como reconhecimento de voz (REIS; MIRANDA; DAMY, 2019). O ML tem aplicações práticas em setores como medicina, onde auxilia no diagnóstico e análise de imagens, e no financeiro, com análise de risco e detecção de fraudes (COZMAN; KAUFMAN, 2022). A IA e o ML também são fundamentais em automação, otimização de cadeias de suprimentos e sistemas de assistência virtual (REIS; MIRANDA; DAMY, 2019).

Entretanto, a implementação bem-sucedida de IA e ML requer profundo entendimento dos dados e algoritmos, bem como um cuidadoso processo de coleta e preparação dos dados (LUDERMIR, 2021). Um desafio relevante é o "viés algorítmico", onde os algoritmos podem refletir preconceitos sociais. Assim, é essencial assegurar a imparcialidade e ética nas decisões automatizadas (COZMAN; KAUFMAN, 2022). A IA pode ser valiosa na seleção do banco de dados ideal, fornecendo recomendações baseadas em análises detalhadas e aprendendo com feedbacks dos usuários (LUDERMIR, 2021). Ainda assim, a IA deve ser vista como uma ferramenta complementar, não substituindo a análise e julgamento humanos (FIGUEIREDO; GIAMMELLA, 2018).

No contexto do treinamento de modelos de ML, os hiperparâmetros desempenham um papel crucial. Hiperparâmetros são parâmetros que não são aprendidos

durante o treinamento, mas são definidos previamente e influenciam diretamente o desempenho e a eficiência do modelo. Por exemplo, a taxa de aprendizado, o número de camadas em redes neurais e o tamanho do lote são hiperparâmetros comuns. A escolha adequada de hiperparâmetros pode melhorar significativamente a capacidade de um modelo de generalizar para dados não vistos. A otimização de hiperparâmetros é um processo iterativo, frequentemente realizado através de métodos como busca em grade ou busca aleatória, visando encontrar a combinação que maximize o desempenho do modelo (PROBST; WRIGHT; BOULESTEIX, 2019).

Além disso, a IA pode otimizar o desempenho dos bancos de dados, detectar anomalias e garantir a segurança dos dados (LUDERMIR, 2021). Em resumo, a união da expertise em bancos de dados com IA e ML pode oferecer soluções eficientes para a seleção e uso de bancos de dados em variados contextos, contudo, é vital considerar aspectos éticos e valorizar o conhecimento humano no processo (FIGUEIREDO; GIAMMELLA, 2018).

3.3 PROCESSAMENTO DE LINGUAGEM NATURAL (PLN)

PLN é uma abreviação de Processamento de Linguagem Natural. É um subcampo da inteligência artificial (IA) dedicado ao estudo e desenvolvimento de algoritmos e técnicas que permitem que os computadores entendam, interpretem e administrem a linguagem humana naturalmente (NADKARNI; OHNO-MACHADO; CHAPMAN, 2011).

O processamento de linguagem natural envolve a aplicação de técnicas computacionais para trabalhar com texto e dados linguísticos, permitindo que os computadores entendam o significado, a estrutura e o contexto da linguagem humana. Isso inclui tarefas como análise de sentimento, reconhecimento de entidade, recuperação de dados, tradução automática criando respostas computadorizadas e assim por diante.

Para executar essas tarefas, os sistemas PLN usam métodos como algoritmos de aprendizado de máquina, redes neurais, processamento estatístico, modelos de linguagem, etc. Essas técnicas permitem que os computadores processem e analisem de forma eficiente e automática grandes volumes de texto, permitindo a automação de diversas tarefas que envolvem a linguagem natural.

3.3.1 LEMATIZAÇÃO

A lematização é um processo linguístico utilizado no processamento de linguagem natural (PLN) que envolve a redução de palavras flexionadas às suas formas de base, chamadas de lemas. O lema de uma palavra é a sua forma canônica, que representa o seu sentido principal e é comumente encontrada em dicionários.

A lematização é importante porque permite tratar diferentes variações de uma palavra como sendo a mesma, simplificando assim a análise de textos. Por exemplo, as palavras "correr", "correndo", "corrida" e "correu" seriam todas reduzidas ao lema "correr" (KHYANI; S, 2021).

3.3.2 STEMIZAÇÃO

A stemização, também conhecida como stemming, é um processo utilizado no processamento de linguagem natural (PLN) para reduzir palavras à sua forma raiz ou radical, removendo os sufixos e prefixos adicionados a elas. O objetivo da stemização é simplificar as palavras para que diferentes variações de uma mesma raiz sejam tratadas como iguais.

Ao realizar a stemização, o algoritmo de stemming aplica uma série de regras heurísticas para remover os afixos das palavras. Por exemplo, as palavras "correr", "correndo", "corrida" e "correu" seriam todas reduzidas ao radical "corr".

A stemização é amplamente utilizada em diversas aplicações de PLN, como recuperação de informações, indexação de palavras-chave, agrupamento de documentos e classificação de textos. Ela ajuda a reduzir a dimensionalidade do espaço de palavras e a agrupar palavras relacionadas em um mesmo radical, facilitando a análise e o processamento de grandes volumes de texto (KHYANI; S, 2021).

3.4 MODELO

Um modelo em IA é uma representação computacional de um sistema, conceito ou fenômeno do mundo real. Ele é construído utilizando técnicas e algoritmos de aprendizado de máquina, processamento de linguagem natural, visão computacional ou outras áreas da Inteligência Artificial (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O modelo é treinado com dados de entrada para aprender padrões e relações, permitindo que faça previsões, tome decisões ou forneça respostas (RUSSELL; NORVIG, 2016).

Os modelos em IA são fundamentais para o desenvolvimento de sistemas inteligentes e têm sido amplamente aplicados em diversas áreas, como reconhecimento de voz, processamento de imagens, tradução automática, diagnóstico médico e recomendação de conteúdo. Eles podem ser construídos usando diferentes abordagens, como redes neurais, árvores de decisão, máquinas de vetores de suporte, entre outros (GOODFELLOW; BENGIO; COURVILLE, 2016).

3.4.1 RANDOM FOREST

Random Forest ou RF, é um método de aprendizado de conjunto supervisionado que possui diferentes casos de uso para tarefas de classificação e regressão. O algoritmo de RF consiste em diferentes algoritmos de árvore de decisão. A 'floresta' gerada pelo algoritmo de Random Forest é treinada por meio de ensacamento ou agregação de bootstrap. O RF estabelece o resultado com base nas previsões das árvores de decisão. Ele prevê tomando a média ou média da saída de várias árvores. Aumentar o número de árvores aumenta a precisão do resultado (SPEISER et al., 2019). O RF é superior aos algoritmos de árvore de decisão porque eliminará a desvantagem dos algoritmos de árvore de decisão, como problemas de sobreajuste. A principal vantagem do Random Forest é que ele reduz o overfitting e melhora a precisão do modelo, ao mesmo tempo em que lida bem com dados ausentes ou ruidosos. Além disso, ele é capaz de lidar com conjuntos de dados grandes e de alta dimensionalidade. A diversidade entre as forest trees permite que o modelo capture diferentes aspectos dos dados e reduza o viés.

Para o treinamento, o Random Forest têm vários hiperparâmetros que podem afetar seu desempenho. Esses incluem:

max_depth especifica a profundidade máxima de cada árvore na Random Forest.

min_samples_split especifica o número mínimo de amostras necessárias para dividir um nó interno na árvore.

max_features especifica o número de recursos a serem considerados ao procurar a melhor divisão.

n_estimators especifica o número de árvores na Random Forest.

3.4.2 KNN

O algoritmo K-Nearest Neighbors (KNN) é um algoritmo de aprendizado de máquina não paramétrico e supervisionado usado para classificação e regressão. Ele funciona encontrando os k pontos de dados mais próximos no conjunto de treinamento para um novo ponto de dados e usando a classe ou valor da maioria desses pontos para prever a classe ou valor do novo ponto de dados (XIONG; YAO, 2021). No algoritmo KNN, o valor de k determina o número de vizinhos a serem considerados para a classificação ou regressão. O algoritmo calcula a distância entre o novo ponto de dados e cada ponto de dados no conjunto de treinamento usando uma métrica de distância, como distância euclidiana ou distância de Manhattan. Os k vizinhos mais próximos são então selecionados com base em sua distância do novo ponto de dados (GAO; LI, 2020). Para problemas de classificação, a classe do novo ponto de dados é prevista

selecionando a classe mais comum entre os k vizinhos mais próximos. Para problemas de regressão, o valor do novo ponto de dados é previsto tomando a média dos valores dos k vizinhos mais próximos.

Para o treinamento, o KNN possui alguns hiperparâmetros para classificação. Esses dois hiperparâmetros são:

K este é o número de vizinhos que o algoritmo usará para fazer previsões.

Métrica de distância é assim que o algoritmo medirá a distância entre pontos no espaço de características. Algumas métricas de distância comuns usadas no KNN são distância euclidiana, distância de Manhattan e distância de Minkowski.

3.4.3 REGRESSÃO LOGÍSTICA

A Regressão Logística é um método estatístico utilizado para modelar a relação entre uma variável dependente binária (com dois possíveis resultados) e uma ou mais variáveis independentes. Diferentemente da Regressão Linear, que prevê um valor contínuo, a Regressão Logística estima a probabilidade de um evento ocorrer (ZOU et al., 2019).

Por exemplo, pode-se usar a Regressão Logística para prever se um e-mail é spam (1) ou não (0) com base na frequência de certas palavras no conteúdo. Neste caso, o resultado é binário: spam ou não spam. O modelo logístico usa a função logística (ou função sigmóide) para espremer os valores de saída entre 0 e 1, fornecendo assim uma probabilidade (ZOU et al., 2019). Se essa probabilidade for superior a um limiar, geralmente 0,5, o modelo prevê a classe 1; caso contrário, prevê a classe 0.

A Regressão Logística é amplamente utilizada em campos como medicina (para prever a ocorrência de uma doença com base em certos sintomas), finanças (para prever inadimplência) e marketing (para prever a probabilidade de um cliente comprar um produto).

3.5 ANÁLISE DE COMPONENTES PRINCIPAIS (PCA)

A Análise de Componentes Principais (PCA) é uma técnica de redução de dimensionalidade usada em aprendizado de máquina e análise de dados. O objetivo do PCA é encontrar uma representação de menor dimensão dos dados originais, mantendo a maior parte da variabilidade dos dados.

A ideia básica do PCA é transformar um conjunto de variáveis correlacionadas em um novo conjunto de variáveis não correlacionadas, chamadas de componentes principais. Esses componentes principais são ordenados de forma que o primeiro com-

ponente principal capture a maior parte da variância dos dados, o segundo componente principal capture a segunda maior parte da variância e assim por diante.

O PCA é amplamente utilizado em aprendizado de máquina para reduzir a dimensionalidade dos dados, remover correlações entre as variáveis e melhorar o desempenho de algoritmos de modelagem. Além disso, o PCA também pode ser usado para visualização de dados, permitindo representar dados multidimensionais em um espaço bidimensional ou tridimensional (UDDIN; MAMUN; HOSSAIN, 2021).

3.6 TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

O TF-IDF é uma técnica estatística utilizada no processamento de linguagem natural e recuperação de informações para quantificar a importância de uma palavra em um documento, em relação a um conjunto de documentos ou corpus. O valor do TF-IDF aumenta proporcionalmente à medida que a palavra aparece mais vezes no documento, mas é compensado pela frequência da palavra no corpus, o que ajuda a distinguir a importância real da palavra (LI; ZHANG, 2019).

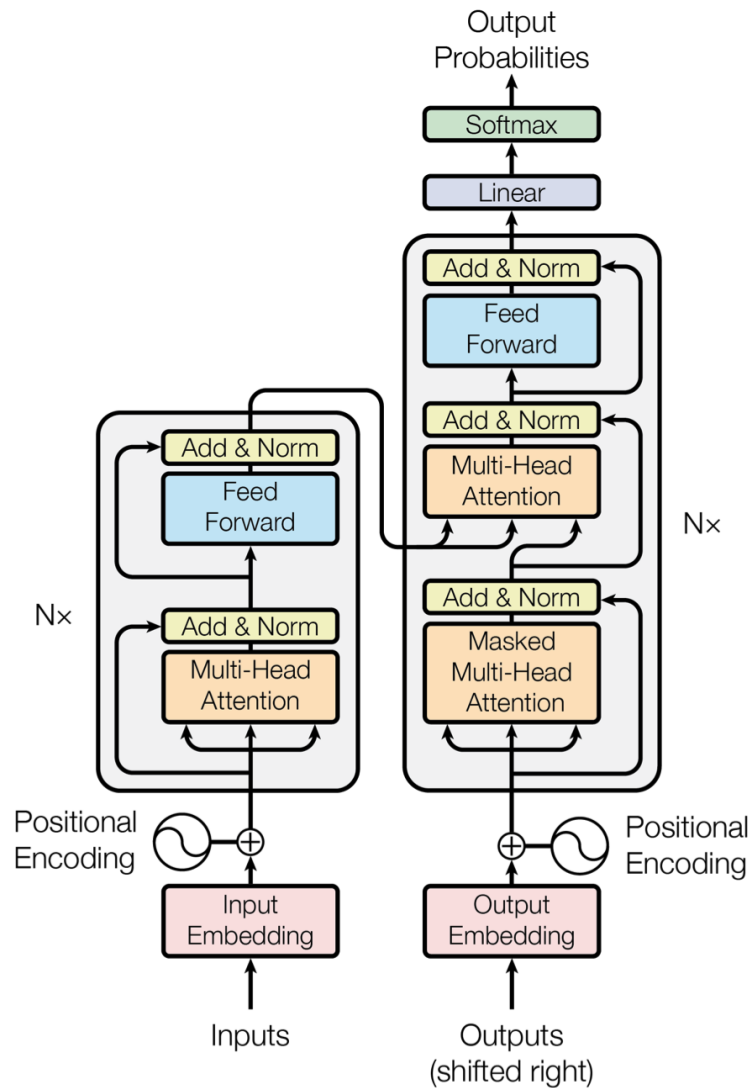
O Term Frequency (TF) refere-se à frequência de uma palavra em um documento. É calculado como o número de vezes que uma palavra aparece em um documento dividido pelo número total de palavras no documento. O Inverse Document Frequency (IDF) Mede a importância geral de uma palavra no corpus. É calculado como o logaritmo do número total de documentos dividido pelo número de documentos contendo a palavra. A combinação de TF e IDF reflete a importância de uma palavra não apenas no contexto de um documento específico, mas também em relação ao corpus inteiro. Isso torna o TF-IDF particularmente útil para tarefas como indexação de documentos e sistemas de recomendação.

3.7 TRANSFORMADORES

O modelo Transformer, introduzido em 2017 pelo artigo "Attention is All You Need" do Google (VASWANI et al., 2023), trouxe uma nova perspectiva para o processamento de sequências em tarefas de PLN. Diferentemente das abordagens tradicionais, como RNNs e LSTMs, o Transformer processa todos os tokens de entrada simultaneamente, permitindo uma eficaz paralelização.

O núcleo do Transformer é o mecanismo de "atenção", especialmente a "atenção multi-cabeça". Esta permite ao modelo focar em diferentes partes de uma sequência simultaneamente, ponderando sua relevância para cada token. Em termos simples, o mecanismo de atenção determina a importância de cada palavra em relação a

Figura 1 – Arquitetura do transformers.



Fonte: <https://arxiv.org/pdf/1706.03762.pdf>

outras palavras na sequência. Além disso, o Transformer utiliza redes feed-forward e camadas residuais. As redes feed-forward são responsáveis por realizar transformações lineares nos dados, enquanto as conexões residuais ajudam a evitar o problema do desaparecimento do gradiente em modelos profundos, garantindo que o sinal de entrada seja "carregado" através das camadas.

Outro componente crucial é a camada de normalização, que mantém a saída das camadas do modelo dentro de uma escala definida, melhorando a estabilidade e a convergência durante o treinamento. O esquema da arquitetura do transformer é mostrado na Figura 1. Desde sua introdução, o Transformer tem sido a base para várias arquiteturas notáveis, como BERT, GPT, T5 e outras, que têm estabelecido novos padrões de desempenho em diversas tarefas de PLN.

3.8 REPRESENTAÇÕES CODIFICADORAS BIDIRECIONAIS DE TRANSFORMADORES (BERT)

BERT (Bidirectional Encoder Representations from Transformers) é um modelo de linguagem pré-treinado desenvolvido pela Google AI Language. Introduzido em 2018, teve um grande impacto no campo do processamento de linguagem natural (PLN).

O BERT é baseado na arquitetura Transformer, uma rede neural que se destaca no processamento de sequências e na captura de relacionamentos de longo alcance. Ao contrário dos modelos de linguagem anteriores que geralmente usavam arquiteturas unidirecionais, o BERT introduziu o conceito de pré-treinamento "bidirecional". Isso significa que o modelo é exposto a informações contextuais das palavras anteriores e posteriores da frase, permitindo que o sistema identifique rapidamente quais palavras estão relacionadas à pergunta e quais não estão, permitindo respostas mais rápidas e precisas. Além disso também pode ser usado para melhorar a precisão de recursos como reconhecimento de entidade e análise de sentimento, que são componentes importantes de um sistema de resposta a perguntas. Além disso, fornece uma maneira eficaz de representar palavras de uma maneira que permite o aprendizado de relacionamentos mais complexos e respostas mais precisas, resultando em melhores sistemas de resposta a perguntas.

Outro benefício de usar o BERT é a capacidade de tokenização de palavras. A tokenização de trecho de palavra permitirá que o modelo divida a palavra rara em prefixos e sufixos e capture a essência principal de cada palavra. Outro benefício de usar BERT como a palavra incorporada é o método de segmentação de posição. O método de segmentação de posição indica a posição das palavras na sequência (ALATAWI; ALHOTHALI; MORIA, 2020).

3.9 MODELOS DE LINGUAGEM DE GRANDE ESCALA (LLMs)

Modelos de Linguagem de Grande Escala (LLMs) são estruturas de inteligência artificial especializadas na compreensão e geração de linguagem. Eles registram padrões de uso da linguagem com base em grandes quantidades de texto da Internet. Ao introduzir um novo texto, o LLM pode prever ou elaborar extensões desse texto com base no que aprendeu. Esses modelos evoluíram significativamente nos últimos anos, liderados pelo GPT-3 da OpenAI, que possui notáveis capacidades de geração de texto. O LLM tem muitos usos, desde chatbots até geração de código, sendo uma subcategoria de IA Generativa. Porém, também apresentam desafios éticos, como a propagação de preconceitos ou geração de informações falsas. Embora avançados, LLMs carecem de consciência real, operando meramente por meio de padrões mate-

máticos e estatísticos aprendidos. Organizações líderes no campo, como OpenAI e Google, estão adotando medidas para garantir o uso ético e seguro desses modelos (GIORGI et al., 2023).

3.10 MODELOS DE LINGUAGEM DE AJUSTE FINO (FINE-TUNED LANGUAGE MODELS)

Modelos de linguagem ajustados são derivados de Modelos de Linguagem de Grande Escala (LLMs), mas são treinados adicionalmente em conjuntos de dados específicos ou para tarefas específicas. Este processo de ajuste fino permite que esses modelos se especializem em um domínio ou tarefa, tornando-os mais aptos em comparação com um LLM padrão (DUNN et al., 2022).

O ajuste fino é realizado para abordar dois principais desafios dos LLMs:

Especialização em Domínios Específicos Enquanto LLMs são treinados em vastos conjuntos de dados genéricos, eles podem não ter precisão em áreas muito específicas ou nichos. Por exemplo, um LLM padrão pode entender medicina em termos gerais, mas um modelo ajustado em literatura médica seria mais preciso ao responder perguntas específicas sobre doenças raras.

Correção de Vieses e Erros LLMs, por serem treinados em grandes volumes de texto da internet, podem herdar vieses e informações incorretas. O ajuste fino permite que os pesquisadores corrijam ou atenuem esses vieses, treinando o modelo em conjuntos de dados curados e bem controlados.

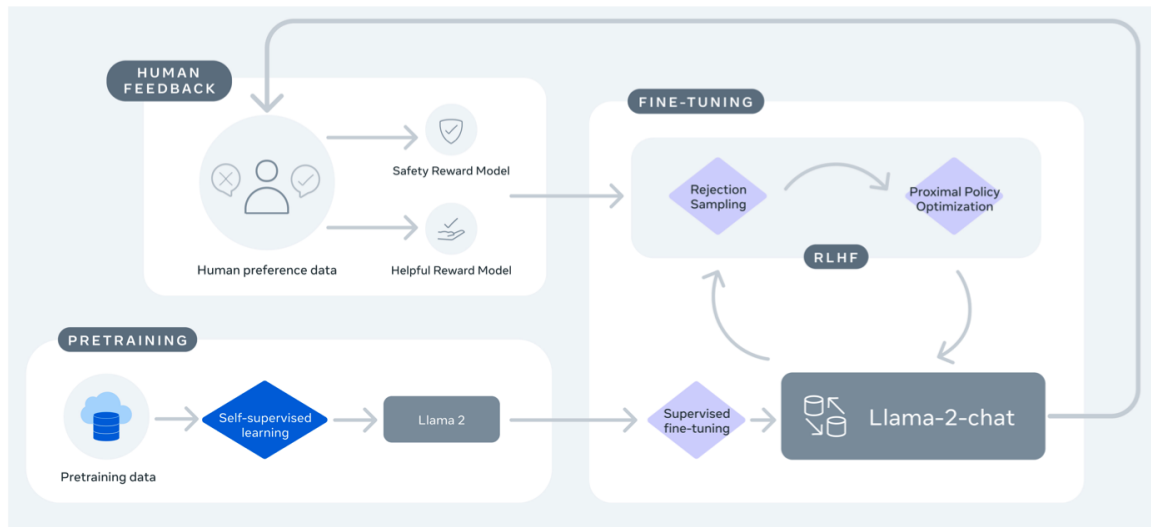
Apesar dos benefícios, o ajuste fino também apresenta desafios:

Perda de Generalização Ao se especializar, o modelo pode perder sua capacidade de generalizar em tarefas fora de seu domínio de especialização.

Sobreajuste Se o ajuste for muito específico ou o conjunto de dados for muito pequeno, o modelo pode se ajustar demais, tornando-o menos útil para variações da mesma tarefa.

Empresas líderes em IA, como OpenAI e Google, têm explorado extensivamente o ajuste fino para adaptar seus LLMs a aplicações práticas em diversos setores. O ajuste fino, portanto, serve como uma ponte entre a generalidade dos LLMs e as necessidades específicas de domínios e tarefas particulares.

Figura 2 – Treinamento do Llama 2 e Llama2-Chat.



Fonte: TOUVRON et al., 2023

3.11 LLAMA 2

O Llama 2 é um Modelo de Linguagem de Grande Escala (LLM) desenvolvido pelo grupo de IA da Meta, empresa-mãe do Facebook. Ele é otimizado para diálogos e supera diversos modelos de bate-papo de código aberto em benchmarks, variando em escala de 7 bilhões a 70 bilhões de parâmetros. Baseado na arquitetura de transformador do Google, o Llama 2 incorpora inovações como a pré-normalização RMSNorm, a função de ativação SwiGLU e a atenção de várias consultas. Seu treinamento envolveu um corpus de dois trilhões de tokens, e a Meta empregou recursos computacionais significativos, como GPUs Nvidia A100, durante o processo (TOUVRON et al., 2023). Em termos de segurança, o Llama 2 é posicionado pela Meta como um modelo confiável, mas, como todas as IAs generativas, exige cautela na interpretação de suas respostas. O processo de treinamento do Llama2 está representado na Figura 2.

3.12 LANGCHAIN

LangChain é uma estrutura destinada ao desenvolvimento de aplicações impulsionadas por modelos de linguagem. Seu principal objetivo é proporcionar uma integração fluida entre aplicações e modelos de linguagem, permitindo que essas aplicações sejam conscientes do contexto e possam raciocinar de acordo com ele (ASYROFI et al., 2023). Isso é realizado através da conexão do modelo de linguagem a várias fontes de contexto, como instruções de prompt, exemplos de poucos disparos, conteúdo para fundamentar sua resposta, entre outros. Um dos principais benefícios do LangChain reside nos seus componentes, que são abstrações projetadas para trabalhar com modelos de linguagem. Estes componentes são modulares e fáceis de

usar, e vêm com várias implementações para cada abstração. Além disso, o LangChain oferece "cadeias prontas para uso", que são conjuntos estruturados de componentes desenhados para realizar tarefas de alto nível. Essas cadeias facilitam o início do desenvolvimento, enquanto os componentes individuais permitem a personalização e a construção de novas cadeias para aplicações mais complexas.

4 METODOLOGIA

Esta seção apresenta os procedimentos realizados para a validação das soluções propostas neste trabalho, descrevendo o conjunto de dados utilizado, a coleta de dados, a definição do modelo, normalização e tratamento dos dados e os testes executados.

4.1 ARQUITETURA

A arquitetura de algumas das soluções propostas neste trabalho para a escolha de banco de dados segue um fluxo composto por etapas que envolvem coleta de dados, pré-processamento, treinamento do modelo, avaliação e aplicação para o usuário. Essa estrutura permite uma compreensão macro do funcionamento do algoritmo, facilitando a leitura da metodologia e a compreensão dos detalhes de cada etapa.

Na etapa de coleta de dados, são reunidas informações relevantes sobre os diferentes bancos de dados disponíveis, considerando suas características, desempenho, escalabilidade, recursos de segurança e outras métricas relevantes. Esses dados servirão como base para a tomada de decisão durante o processo de escolha.

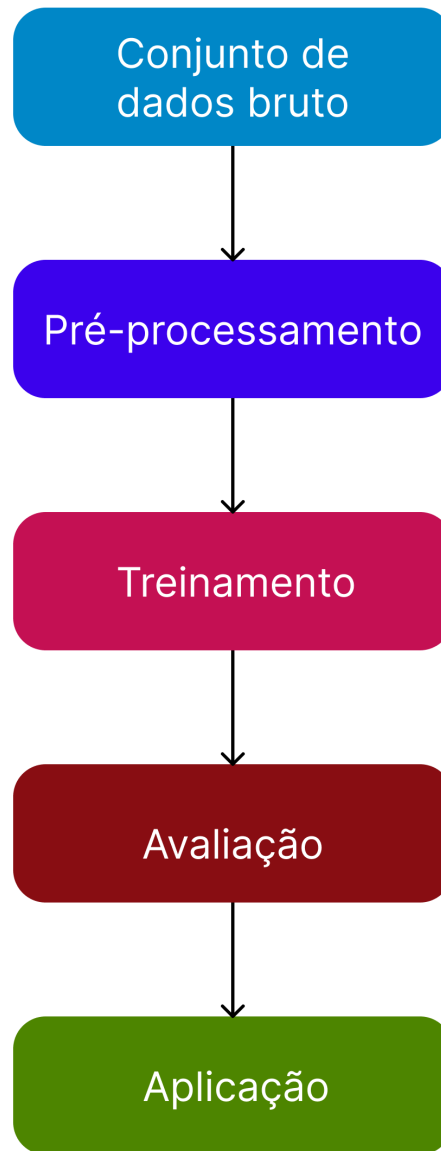
Após a coleta, ocorre o pré-processamento dos dados, onde é realizada a organização e o tratamento necessário para garantir a qualidade e a consistência dos dados. Nessa etapa, podem ser aplicadas técnicas de limpeza, transformação e normalização dos dados coletados, a fim de torná-los adequados para o treinamento do modelo.

O treinamento do modelo é uma etapa crucial, na qual é desenvolvida uma representação do conhecimento adquirido a partir dos dados coletados. Isso é feito por meio de algoritmos de aprendizado de máquina, que são aplicados aos dados pré-processados. Durante o treinamento, o modelo é ajustado para identificar padrões e relações entre os dados que permitam a classificação dos bancos de dados.

Após o treinamento, é realizada a avaliação do modelo. Essa etapa tem como objetivo verificar a eficácia e a precisão das classificações realizadas pelo modelo. São utilizadas métricas de desempenho, como acurácia, precisão, recall e F1-score, para avaliar a qualidade das previsões feitas pelo algoritmo.

Por fim, a aplicação para o usuário envolve a utilização do modelo treinado para realizar a escolha do banco de dados mais adequado. Com base nas informações fornecidas pelo usuário e nos critérios definidos durante o treinamento, o algoritmo faz uma classificação e fornece uma recomendação do banco de dados mais apropriado

Figura 3 – Arquitetura final do sistema.



Fonte: Elaborado pelo autor

para o contexto em questão.

A visualização da arquitetura pode ser vista na Figura 3.

4.2 CONJUNTO DE DADOS

Uma das etapas fundamentais para o desenvolvimento das soluções consiste na obtenção de um conjunto de dados a ser utilizado para treinar o modelo. Nesta

seção, são apresentados, de forma breve, alguns desses conjuntos de dados.

O Stanford Question Answering Dataset (SQuAD) é um conjunto de dados em grande escala para compreensão de leitura e resposta a perguntas, consistindo em mais de 100.000 pares de perguntas/respostas em um conjunto de artigos da Wikipedia. O conjunto de dados é dividido em dois subconjuntos: um conjunto de treinamento (mais de 80.000 pares de perguntas/respostas) e um conjunto de desenvolvimento (mais de 10.000). Cada pergunta no conjunto de dados está associada a um parágrafo da Wikipedia, e a resposta para a pergunta é um segmento de texto do mesmo parágrafo da Wikipedia (CARRINO; COSTA-JUSSÀ; FONOLLOSA, 2019).

A Hotpot Question Answering (HotPotQA) é um conjunto de dados de compreensão de leitura de máquina com perguntas complexas que exigem raciocínio além da simples correspondência de superfície. Ele contém mais de 111 mil pares de perguntas/respostas, abrangendo mais de 8,3 mil artigos da Wikipédia. O conjunto de dados é dividido em 10 tipos diferentes de perguntas (por exemplo ponte, comparação, correferência, etc.) e no conjunto há anotações de resposta (YANG et al., 2018). Ele é coletado por uma equipe de pesquisadores da Universidade Carnegie Mellon, Universidade de Stanford e Universidade de Montreal.

O bAbI é um conjunto de dados sintético baseado em texto para desenvolver e avaliar sistemas de IA para raciocínio básico. O conjunto de dados contém 20 tarefas diferentes, cada uma com um conjunto de mais de 1.000 pares de perguntas/respostas. As tarefas variam de perguntas simples de uma etapa a perguntas complexas de várias etapas envolvendo várias entidades e eventos (TAMARI et al., 2021).

WikiQA é um conjunto de dados para o desenvolvimento de sistemas de resposta a perguntas, criado por um grupo de pesquisadores do Laboratório de Inteligência Artificial e Visão Computacional da Microsoft Research. Consiste em mais de 108.000 pares de perguntas/respostas gerados a partir de perguntas e respostas do mundo real da web. O conjunto de dados é dividido em um conjunto de treinamento (mais de 86.000 pares de perguntas/respostas) e um conjunto de teste (mais de 22.000 pares de perguntas/respostas). Cada pergunta está associada a um artigo da Wikipédia, e a resposta é um trecho de texto do mesmo artigo da Wikipédia (GUO et al., 2020).

O conjunto de dados natural, Natural Questions (NQ), é uma coleção de mais de 100.000 perguntas reais de usuários, reunidas e anotadas pelo Google. Foi lançado em 2018 e é o maior conjunto de dados de resposta a perguntas públicas disponível. O NQ consiste em perguntas reais de usuários sobre artigos da Wikipédia, com contexto correspondente, respostas candidatas e anotações detalhadas da resposta certa e seu nível de confiança. As perguntas são de dificuldade variada, desde perguntas fáceis de uma única palavra até perguntas abertas complexas (KWIATKOWSKI et al., 2019). O NQ foi projetado para ajudar os pesquisadores a criar melhores sistemas de resposta a

perguntas, especialmente para configurações de domínio aberto.

Question Answering for Trivia (TriviaQA) é um conjunto de dados em grande escala para responder a perguntas de domínio aberto. Criado por um grupo de pesquisadores da Universidade de Washington e do Allen Institute for Artificial Intelligence (AI2). Consiste em mais de 650.000 pares de perguntas e respostas coletados de plataformas de trivia baseadas na web, bem como mais de 2 milhões de frases extraídas da Wikipedia. O conjunto de dados tem sido usado como referência para pesquisas em resposta a questões de domínio aberto. É dividido em conjuntos de treinamento, desenvolvimento e teste. As perguntas são tipicamente de natureza factual e as respostas são extraídas de passagens de texto. O TriviaQA foi projetado para permitir o desenvolvimento de modelos robustos à complexidade inerente ao controle de qualidade de domínio aberto (IZACARD; GRAVE, 2021).

Os conjuntos estão em formato CSV. Um resumo de cada um desses conjuntos de dados é apresentado na Tabela 2.

Tabela 2 – Conjunto de dados descrito e informações correspondentes para o desenvolvimento do algoritmo.

Nome do dataset	Tamanho do treinamento	Tamanho do teste	Origem
WikiQA	86,000	22,000	Artigos Wikipedia
bAbI	1,000	200	Gerado por inteligência artificial
HotPotQA	8,000	3,000	Artigos Wikipedia
SQuAD	80,000	10,000	Artigos Wikipedia
Natural Questions	80,000	20,000	Artigos Wikipedia
TriviaQA	550,000	100,000	Artigos Wikipedia e web

Fonte: Elaborado pelo autor.

4.3 CONJUNTO DE DADOS DE PESQUISA

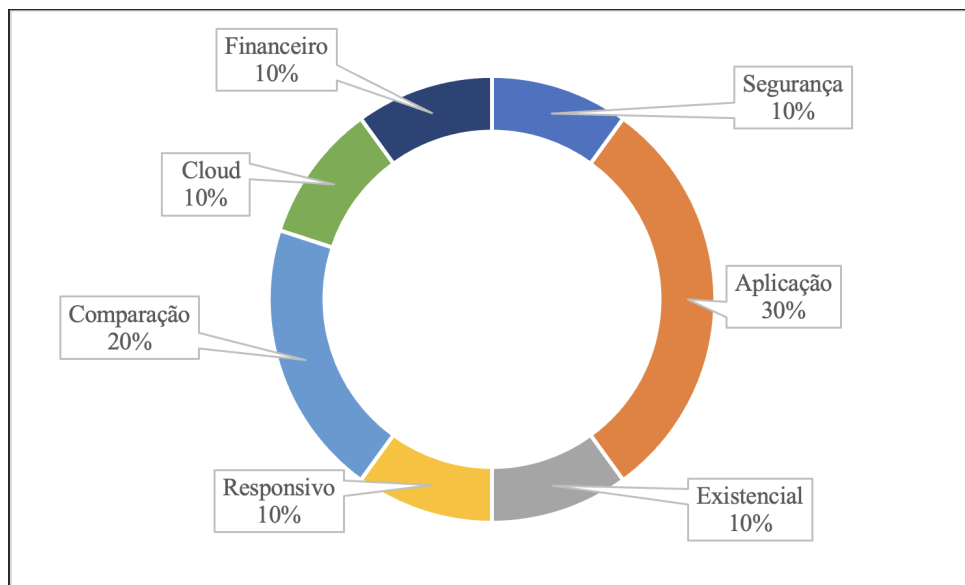
Como foi descrito a maior parte da base de conhecimento disponível para conjuntos de dados de resposta a perguntas, usou informações já disponíveis de artigos da Wikipedia para criar conjuntos de dados gerais em diferentes tópicos, como pontes, meteorologia ou IA (TAMARI et al., 2021).

Nesta pesquisa, é utilizado uma combinação de técnicas de resposta a perguntas baseadas em IA e contexto disponível em artigos da Wikipedia e as perguntas mais populares no processo de entrevista (OMAR et al., 2023). Todas as perguntas feitas estavam relacionadas a um dos tópicos do Amazon Aurora, DynamoDB, ElastiCache, DocumentDB, Keyspace, Neptune, Timestream, QLDB, CloudSQL, Datastore, MemoryStore, BigTable, Firestore, Firebase, Azure Database, Azure Cache, CosmosDB,

Azure, Time Series, Insights, Blockchain, PostgreSQL, Redis, MongoDB, Cassandra, Neo4j, InfluxDB, Hyperledger, Fabric, TimescaleDB, RDS, Redshift, AlloyDB, Spanner, MySQL e Cloud Storage.

Em cada tópico, são fornecidos vários conceitos, como diferentes tipos de conjuntos de dados e sua aplicação em IA, armazenamento de dados, telecomunicações, etc. Em geral, os tópicos que foram o foco principal para a coleta de informações são mostrados na Figura 4. Os tópicos estão divididos nos seguintes itens:

Figura 4 – Distribuição categórica para o conjunto de dados coletado



Fonte: Elaborado pelo autor

Financeiro Explora os aspectos financeiros dos bancos de dados, incluindo custos, licenciamento e despesas associadas a diferentes tecnologias de banco de dados.

Cloud Aborda as tecnologias de banco de dados relacionadas à computação em nuvem, como DBaaS, migração de dados para a nuvem e considerações específicas sobre armazenamento e gerenciamento de dados em ambientes de nuvem.

Comparação Realiza a análise e comparação de diferentes tecnologias de banco de dados em termos de recursos, desempenho, escalabilidade, custos e suporte à linguagem de consulta.

Responsivo Trata da capacidade dos bancos de dados lidarem com cargas de trabalho em tempo real, incluindo escalabilidade, otimização de desempenho e adaptabilidade a um ambiente de banco de dados dinâmico.

Existencial Explora questões mais amplas relacionadas ao que é o banco de dados e para qual finalidade ele foi criado, qual problema pretende resolver.

Tabela 3 – Exemplo do conjunto de dados para o banco de dados DynamoDB.

Pergunta	Resposta
What is the maximum size of an item in DynamoDB?	The maximum size of an item in DynamoDB is 400KB.
Can IAM permissions be used to control access to DynamoDB?	Yes, IAM permissions can be used to control access to DynamoDB.
Can DynamoDB be used for storing binary data?	Yes, DynamoDB can be used for storing binary data like images and videos
What type of data consistency is provided by DynamoDB?	DynamoDB provides eventual consistency and strongly consistent reads.

Fonte: Elaborado pelo autor.

Aplicação Envolve o uso prático de bancos de dados em aplicativos, incluindo design do esquema do banco de dados, modelagem de dados e otimização de consultas.

Segurança Foca nas medidas de segurança implementadas nos bancos de dados para proteger dados sensíveis, como autenticação de usuários, controle de acesso e criptografia.

As perguntas e as respostas são localizadas no conjunto de dados a partir de filtros utilizados com base nos tópicos e adicionadas no conjunto de dados final em seu tópico específico, conforme Tabela 3. As perguntas e respostas estão em inglês por conta da origem do conjunto de dados.

4.4 PRÉ-PROCESSAMENTO DE DADOS

Nesta seção, é fornecido uma explicação detalhada do pré-processamento dos dados. É descrito o processo de reprocessamento e extração de traços que envolve a eliminação de informações dialéticas como a pontuação no processo de pergunta-resposta. O pré-processamento dos dados de texto também inclui remover as pontuações das frases e converter as palavras em minúsculas. No final, aplica-se diferentes técnicas, como lematização, palavras de parada e stemização para obter apenas informações relevantes dos dados do texto (OMAR et al., 2023).

Com o conjunto de dados pronto, a primeira etapa do pré-processamento de dados consiste na remoção da pontuação, caracteres especiais, palavras de parada e caracteres relacionados a links. Essas etapas visam limpar os dados textuais, eliminando elementos irrelevantes que não contribuem para a análise e modelagem.

Para a remoção da pontuação e caracteres especiais, é aplicada uma técnica de filtragem, na qual são utilizados padrões ou expressões regulares para identificar e

remover todos os caracteres que não são letras ou números. Isso inclui símbolos de pontuação, como vírgulas, pontos, aspas, entre outros caracteres especiais.

Em seguida, é feita a remoção das palavras de parada (stop words), que são palavras comuns e frequentemente usadas na língua natural, mas que geralmente não contribuem para o significado da frase. Exemplos de palavras de parada são "o", "a", "e", "de", "para", entre outras. Essas palavras são removidas do texto, a fim de reduzir a dimensionalidade e melhorar o desempenho dos algoritmos de processamento e análise de texto.

Além disso, é realizada a remoção de caracteres relacionados a links, como URLs, endereços de e-mail ou qualquer outra informação que possa estar presente no conjunto de dados. As letras são convertidas de maiúscula para minúscula para facilitar. Esses caracteres não fornecem informações relevantes para a análise do texto em si e podem ser removidos com o objetivo de simplificar e limpar os dados.

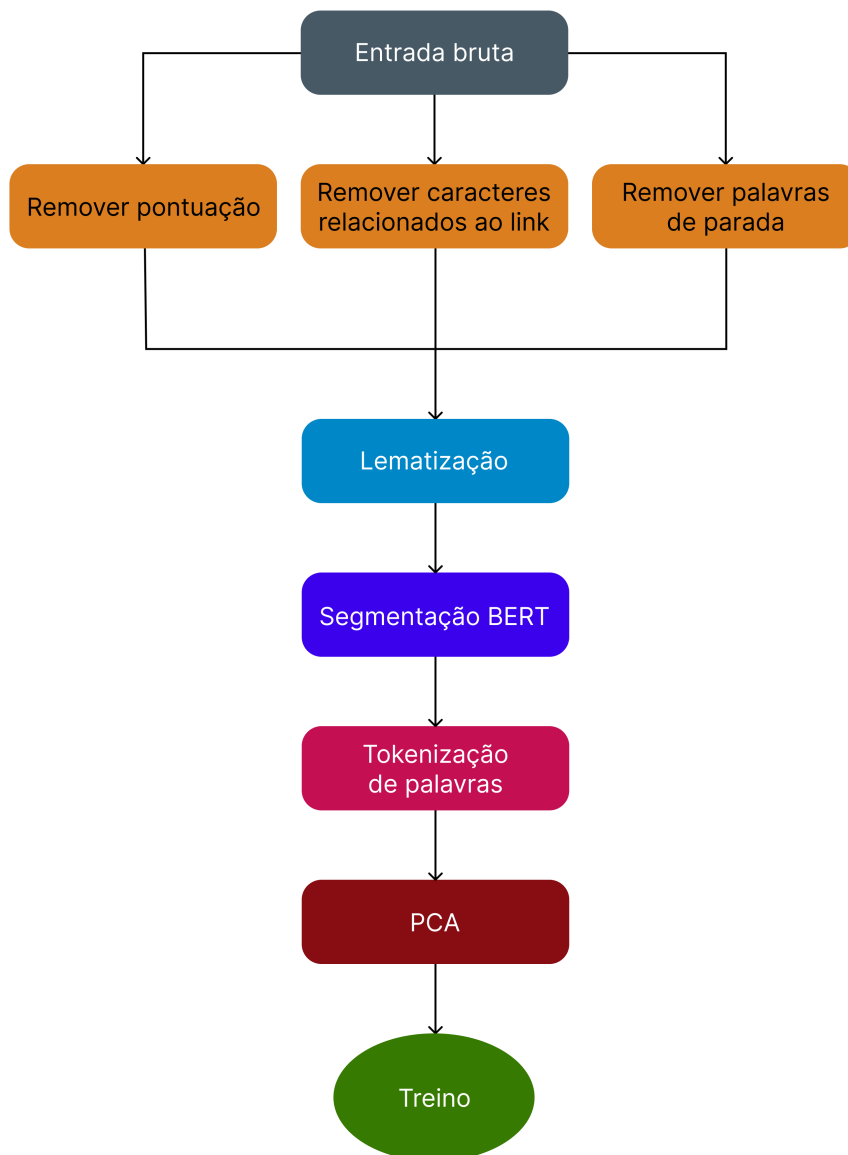
Essas etapas de pré-processamento de dados são essenciais para garantir a qualidade e a consistência dos dados utilizados na análise. Ao remover pontuação, caracteres especiais, palavras de parada e caracteres relacionados a links, é possível obter um conjunto de dados mais limpo e adequado para a aplicação de técnicas de análise de texto, como classificação, agrupamento ou extração de informações.

O estágio final é converter informações textuais em numéricas usando modelos de incorporação. A incorporação de palavras é uma técnica usada no processamento PNL que mapeia palavras ou frases de um vocabulário para vetores de números reais. A ideia por trás das incorporações de palavras é representar as palavras de uma maneira que capture suas propriedades semânticas e sintáticas, de modo que as palavras com significado semelhante sejam mapeadas para vetores próximos umas das outras no espaço de incorporação.

Para converter a informação textual em valor numérico, são utilizadas Representações Codificadoras Bidirecionais de Transformadores. (BERT). Incorporações de token, também conhecidas como embeddings, permitem a representação de palavras individuais em perguntas, além de toda a frase ou contexto (BIRUNDA; DEVI, 2021). Isso permite uma compreensão mais sutil da questão, permitindo que o sistema aprenda e reconheça padrões sutis que podem passar despercebidos pelos tokenizadores tradicionais, por exemplo, temos uma pergunta: o que é conjunto de dados? isso será codificado para "o que" = [1,0,0] "é" = [0,1,0]

Depois de converter as informações processadas em valor numérico, especificamos os recursos mais importantes usando a técnica de redução dimensional. Uma das principais técnicas para a escolha do melhor conjunto de características é a PCA. Este processo reduzirá a complexidade computacional durante o processo de treinamento.

Figura 5 – Arquitetura de pré-processamento e extração de características.



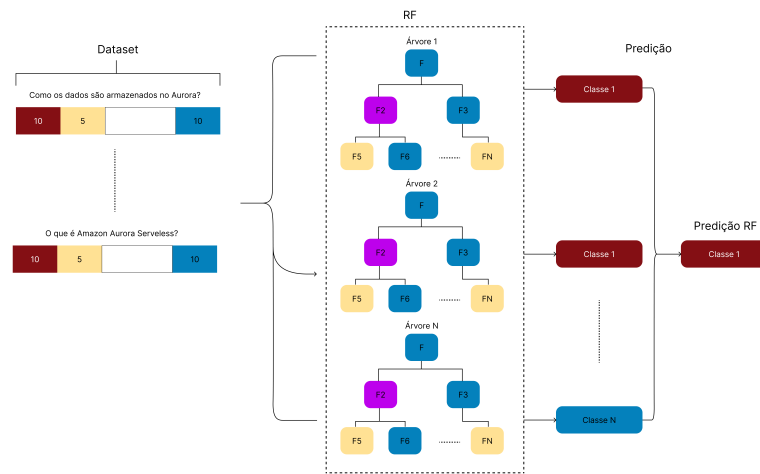
Fonte: Elaborado pelo autor

Um resumo dos métodos de pré-processamento propostos é mostrado na Figura 5.

4.5 MODELO

Na próxima etapa após a extração de recursos, o conjunto de dados passa por modelos de ML para classificação. Existem muitos algoritmos de ML que podem ser usados para classificação. Nesta pesquisa estão disponíveis 32 classes de informação para classificação e com base na informação contextual pré-processada. Serão utilizados os algoritmos de Random Forest, KNN, Regressão Logística com o propósito de prever qual o melhor local para armazenamento dos dados, conforme requisitos

Figura 6 – Arquitetura do modelo de RF.



Fonte: Elaborado pelo autor

elencados na pergunta. Após isso, será realizado o treinamento para adequar o modelo com os hiperparâmetro corretos, serão realizados diversos testes com valores diferentes, para entender o melhor hiperparâmetro de acordo com as avaliações feitas. O esquema do classificador de RF é mostrado na Figura 6.

4.6 TREINAMENTO

Nesta fase, o processo de treinamento é explicado. Para treinar o modelo IA, primeiro, o rótulo para cada pergunta deve ser indicado. No total, existem 32 classes diferentes de informações no conjunto de dados, portanto, o rótulo será de 1 a 32, começando na classe “Amazon Aurora” e terminando na classe “Cloud Storage”. Nesta pesquisa, são citados dois modelos com diferentes hiperparâmetros para treinamento.

Cada hiperparâmetro afetará o desempenho final do modelo. A escolha dos melhores conjuntos de hiperparâmetros levará a um desempenho preciso e, com a má escolha dos hiperparâmetros, o desempenho do modelo diminuirá drasticamente (SANDHA et al., 2020). Para ajustar todos os hiperparâmetros nesta pesquisa, a estratégia de pesquisa em grade é usada. A estratégia de ajuste de hiperparâmetros consiste em testar diferentes configurações para cada hiperparâmetro e, com base na avaliação de precisão do modelo, determinar a opção mais adequada para cada hiperparâmetro. Por exemplo, se testarmos uma profundidade máxima de 20 para a floresta de decisão e alcançarmos uma precisão de 99%, o processo de pesquisa em grade selecionará 20 como a profundidade máxima da floresta de decisão, considerando-a a melhor opção com base nos resultados obtidos. A otimização de hiperparâmetro está disponível na biblioteca de aprendizado de máquina scikit-learn Python.

4.7 AVALIAÇÃO

Para avaliar as métricas dos modelos ML e DL, como exatidão, precisão, recall, F1-score e matriz de confusão, AUROC é usado. Na classificação, a saída prevista é composta por Verdadeiro Positivo (TP), que se refere a amostras do conjunto de dados que são previstas corretamente como uma classe de destino específica, e Verdadeiro Negativo (TN), que se refere a amostras do conjunto de dados que são previstas corretamente como amostras que não pertencem à classe especificada. Falso positivo (FP) refere-se a amostras do conjunto de dados que não são previstas corretamente como uma classe de destino específica. Falso Negativo (FN) refere-se a amostras do conjunto de dados que não são previstas corretamente como amostras que não pertencem à classe especificada (AğBULUT; GüREL; BIÇEN, 2021). A definição para cada critério é definida nas Equações 4.1, 4.2, 4.3 e 4.4

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.1)$$

$$Precision = \frac{TP}{(TP + FN)} \quad (4.2)$$

$$Recall = \frac{TP}{(TP + FP)} \quad (4.3)$$

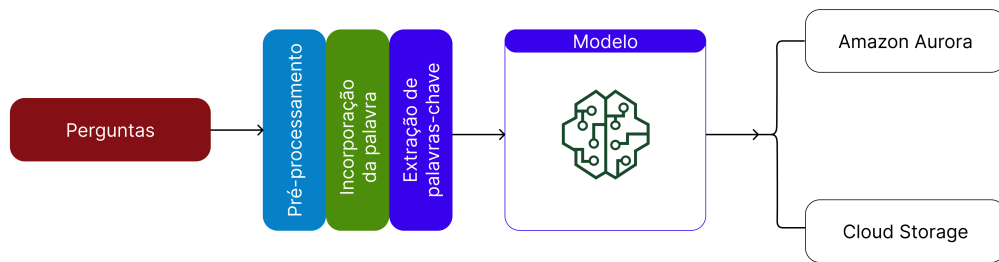
$$F1_score = \frac{(2 * TP)}{(2 * TP + FP + FN)} \quad (4.4)$$

Cada critério é usado para especificar o desempenho dos modelos de aprendizado profundo e de máquina em relação ao seu desempenho na previsão da taxa verdadeiramente positiva e da taxa negativa verdadeira. A alta precisão garante uma boa previsão em TP e TN, o recall é usada para garantir que a alta taxa de predição verdadeira não esconda a baixa taxa de verdadeiro negativo, enquanto F1_score garante que o resultado obtido (precisão e recall) do classificador IA seja válido. Os resultados serão aceitos a partir de 90% de precisão e recall.

4.8 APLICATIVO PARA O USUÁRIO FINAL

Finalmente, treinando o modelo e avaliando seus resultados com base na precisão alcançada, recuperação e pontuação F1, o modelo treinado está pronto para aplicação pelo usuário. Com base no que foi coletado até o momento, o modelo proposto pode ser treinado usando os dois modelos de ML e, com base nos resultados obtidos, um deles será escolhido para o processo final de perguntas e respostas. Como

Figura 7 – Esquema para fornecer a resposta a cada usuário.



Fonte: Elaborado pelo autor

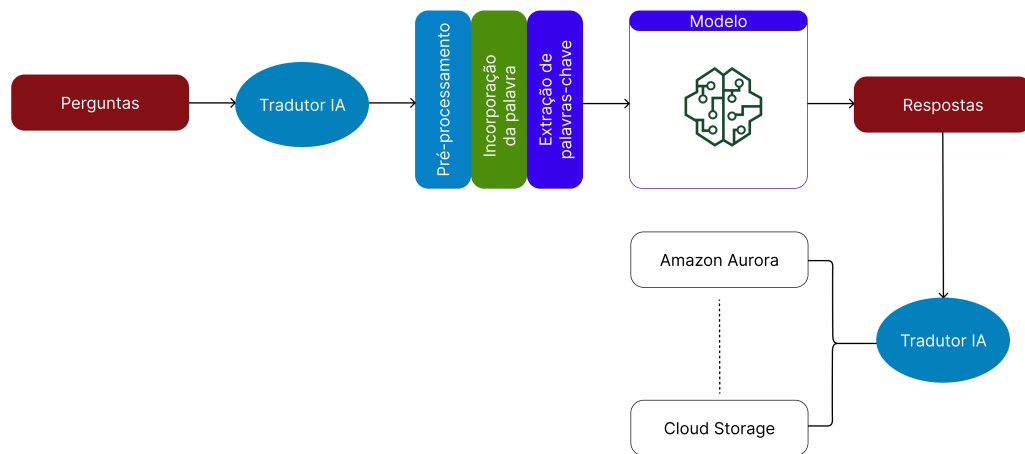
o modelo é treinado no conjunto de dados já disponível, ele capturará as palavras-chave de cada pergunta e, com base na semelhança das perguntas feitas no conjunto de dados de treinamento e no núcleo de conhecimento, o modelo treinado fornecerá uma solução ao usuário. A solução fornecida será apenas a melhor classe que pode ser usada para responder à pergunta do usuário. Um esquema para o processo de pergunta e resposta é fornecido para o usuário final, conforme mostrado na Figura 7.

A solução proposta para cada pergunta será diferente com base nas palavras-chave extraídas da pergunta. O modelo IA mostrado na Figura 7 é o peso salvo do modelo RF, KNN ou Regressão logística. Uma das vantagens de treinar o modelo com base na palavra-chave mais importante é a capacidade de encontrar semelhanças entre as perguntas e fornecer o melhor conjunto de soluções no final. A solução proposta usará um formato rápido e confiável de informações salvas para testar o modelo, portanto, o processo de classificação será rápido.

O método proposto pode recomendar o melhor tipo de banco de dados para vários tipos de questões em vários idiomas. Para se comunicar com o modelo principal de IA com aplicativos em diferentes idiomas, um tradutor de IA é adicionado ao início do processo de resposta a perguntas. O tradutor IA converte a pergunta feita de qualquer idioma para o idioma inglês (LI; NING; FANG, 0). No estágio final, depois de especificar a categoria principal do banco de dados para cada pergunta, um tradutor de IA pode descrever as perguntas respondidas de várias formas. Nesse formato, especificando o idioma convertido, a descrição da resposta pode ser convertida. O esquema para o método proposto é mostrado na Figura 8.

Em resumo, o sistema interage com o usuário, recebe uma pergunta, classifica-a com base nos modelos propostos e determina a melhor classe para responder à pergunta. Em seguida, ele utiliza o conjunto de dados para gerar uma resposta. A resposta final será uma classe de banco de dados das quais conta no dataset. Essa abordagem visa fornecer ao usuário uma resposta apropriada com base em um sistema de conhecimento prévio e, se necessário, complementado por informações externas.

Figura 8 – Esquema para fornecer a resposta a cada usuário com tradução.



Fonte: Elaborado pelo autor

4.9 LLM LLAMA 2 AFINADA

Nesta seção, são apresentados os métodos e abordagens adotados para otimizar os Modelos de Linguagem de Grande Escala (LLMs). Abordaremos em detalhe o processo de coleta de dados, as técnicas de ajuste fino empregadas, como QLoRA e PEFT. Além disso, será comentado sobre as ferramentas e plataformas utilizadas e a justificativa por trás de cada escolha metodológica.

4.9.1 FONTE DE DADOS E PREPARAÇÃO

Os dados utilizados para este projeto de ajuste fino foram obtidos a partir do dataset já criado anteriormente. Para tornar esses dados mais acessíveis ao ajuste fino do modelo de linguagem, deverá ser transformado para um formato mais adequado ao processamento por máquina. O processo de conversão resultou na criação de um novo conjunto de dados chamado `combined_data.jsonl`. Este conjunto de dados foi estruturado no formato JSON Lines (JSONL), que é um formato conveniente para trabalhar com dados textuais.

O conjunto de dados `combined_data.jsonl` é composto com componentes-chave essenciais para tarefas de processamento de linguagem natural. Esses componentes são:

Instrução Estas instruções representam consultas que os usuários poderiam fazer ao buscar informações ou orientações.

Contexto Quando disponível, o contexto é essencial para esclarecer a intenção por trás da pergunta, garantindo respostas mais precisas.

Resposta Ele oferece insights sobre quais bancos de dados ou serviços em nuvem são adequados para determinados casos de uso, sendo fundamental para gerar respostas informativas e relevantes.

Categoria Define a categoria ou tipo da resposta, auxiliando na identificação do banco de dados ou serviço em nuvem mencionado na resposta. Esta categorização é crucial para fornecer respostas mais direcionadas e conscientes do contexto às consultas dos usuários.

O conjunto de dados `combined_data.jsonl` atua como uma base de conhecimento abrangente, essencial para atender consultas de usuários, fornecer recomendações e disponibilizar informações sobre uma vasta gama de tópicos relacionados a bancos de dados, serviços em nuvem e muito mais. Ao interagir com este conjunto de dados, os usuários podem fazer perguntas e nosso sistema utiliza este recurso para fornecer respostas esclarecedoras e pertinentes.

4.9.2 HIPERPARÂMETROS E FINE-TUNNING

O ajuste fino é uma etapa fundamental que envolve a adaptação de um modelo pré-treinado a uma tarefa ou conjunto de dados específico. Os hiperparâmetros são variáveis definidas antes do início do treinamento, que influenciam a performance e a eficiência do modelo. Os hiperparâmetros essenciais utilizados em nosso processo incluem:

Taxa de Aprendizado (Learning Rate) Essa taxa determina o tamanho dos passos durante o processo de otimização.

Tamanho do Lote (Batch Size) Define a quantidade de entradas processadas simultaneamente.

Acúmulo de Gradiente (Gradient Accumulation Steps) Esta técnica otimiza o uso da memória durante o treinamento.

Passos de Aquecimento (Warmup Steps) Permitem o aumento gradual da taxa de aprendizado no início do treinamento, estabilizando-o.

FP16 (Precisão Mista) Utilizado para reduzir o consumo de memória, empregando números de ponto flutuante de menor precisão.

Passos de Registro (Logging Steps) Configurado para registrar o progresso a cada passo, monitorando a performance do modelo.

Algoritmo de Otimização (Optimization Algorithm) Algoritmo que será utilizado para otimização do modelo, diminuindo o custo de hardware, sacrificando um pouco da precisão.

Quanto à duração do treinamento, o modelo foi treinado por 15.000 passos, optando-se por passos em vez de épocas para maior controle do processo. Após o ajuste fino, o modelo foi submetido a testes rigorosos para avaliar sua performance e eficácia em tarefas de processamento de linguagem natural. Isso permitirá validar a eficiência do modelo e identificar áreas de melhoria potencial.

Para aferir a qualidade das respostas geradas em comparação com as respostas de referência, são utilizadas as métricas ROUGE (Recall-Oriented Understudy for Gisting Evaluation) e Meteor (Metric for Evaluation of Translation with Explicit Ordering). A métrica ROUGE fornece uma visão detalhada da qualidade das respostas geradas por meio de três variantes: ROUGE-1, ROUGE-2 e ROUGE-L, centradas em diferentes aspectos de sobreposição e similaridade de texto. Meteor, por sua vez, oferece uma perspectiva única sobre a qualidade das respostas, incorporando precisão, recall e heurísticas.

4.10 LLM LLAMA 2 COM LANGCHAIN

Nesta seção, é descrito a teoria subjacente à implementação do modelo Llama 2, discutindo técnicas, hiperparâmetros e as implicações desses componentes no desempenho do modelo.

4.10.1 QUANTIZAÇÃO

A quantização é uma técnica de otimização que envolve a conversão de pesos do modelo de uma representação de maior precisão (como ponto flutuante de 32 bits) para uma representação de menor precisão (como ponto flutuante de 16 ou 8 bits ou representações de 4 bits). Esta técnica é fundamental para o Llama, pois permite que o modelo seja carregado e operado com menos memória GPU, tornando o processo mais eficiente sem sacrificar significativamente a precisão. Abaixo segue a descrição de alguns parâmetros:

load_in_4bit Este parâmetro determina se os pesos do modelo devem ser carregados em uma representação de 4 bits. Quando definido como True, permite uma redução significativa no tamanho da memória necessária para armazenar os pesos, à custa de uma possível pequena perda de precisão.

bnb_4bit_quant_type Especifica o tipo de quantização a ser usado para a representação de 4 bits. O valor 'nf4' é uma especificação particular que determina o método exato de quantização. Diferentes métodos podem ter características variadas em termos de preservação de precisão e eficiência computacional.

bnb_4bit_use_double_quant Indica se a quantização dupla deve ser aplicada na representação de 4 bits. Quando definido como True, o processo de quantização é realizado em dois passos, o que pode oferecer uma melhor preservação da precisão em algumas situações.

bnb_4bit_compute_dtype Define o tipo de dados a ser usado durante os cálculos que envolvem quantização. O valor bfloat16 refere-se a um formato de ponto flutuante de 16 bits, que é uma alternativa ao formato tradicional de 16 bits (float16). O bfloat16 oferece uma maior faixa dinâmica, sacrificando a precisão em comparação com o float16.

4.10.2 TOKENIZAÇÃO E PIPELINES

A tokenização é um processo fundamental para converter texto bruto em uma sequência de tokens, que são unidades menores de texto que o modelo pode entender. Para o Llama, um tokenizador específico é usado para garantir que o texto seja interpretado de forma eficaz, mantendo nuances e significados. O modelo de tokenizador utilizado será o sentence-transformers/all-mpnet-base-v2 do Hugging Face.

No âmbito da biblioteca Hugging Face, os pipelines constituem-se como instrumentos sofisticados que consolidam o processo de tokenização e a inferência do modelo em um único procedimento. Para o modelo Llama, é implementado especificamente um pipeline destinado à geração textual, responsável por converter tokens em respostas textualmente geradas. Os parâmetros a serem empregados neste contexto incluem:

model Determina o modelo que será utilizado.

tokenizer Determina o tokenizador que será utilizado.

return_full_text Isso garante que o texto completo, e não apenas o texto recém-gerado, seja retornado.

task Define que o pipeline será utilizado para geração de texto.

temperature controla a aleatoriedade das saídas, onde 0 é o mínimo e 1 é o máximo.

max_new_tokens Número máximo de tokens na saída.

repetition_penalty Evita repetições no texto gerado.

stopping_criteria Define uma parada personalizada, neste caso será criada uma classe para definir até onde o modelo deve gerar a resposta.

4.10.3 LANGCHAIN

Nesta seção, delinea-se a metodologia proposta para a configuração de um agente conversacional, aproveitando as capacidades do framework LangChain. O LangChain disponibiliza diversas ferramentas que podem ser adotadas para aprimorar sistemas de resposta automáticos baseados em Modelos de Linguagem de Grande Escala (LLMs).

Inicialmente, sugere-se a integração da pipeline do Hugging Face com a LangChain. A pipeline do Hugging Face, conhecida por combinar eficientemente a tokenização e a inferência dos modelos, constitui-se como uma opção viável para a geração de respostas textuais. Ao integrar essa pipeline, espera-se que o modelo LLM consiga transformar consultas em respostas pertinentes.

O histórico de conversas é outra característica considerada na metodologia. Esse componente, apesar de opcional, permite armazenar as interações entre o usuário e o sistema. Ao manter um registro contextual, o sistema pode ter a habilidade de responder com base nas interações anteriores do usuário, oferecendo uma experiência de conversa mais fluida.

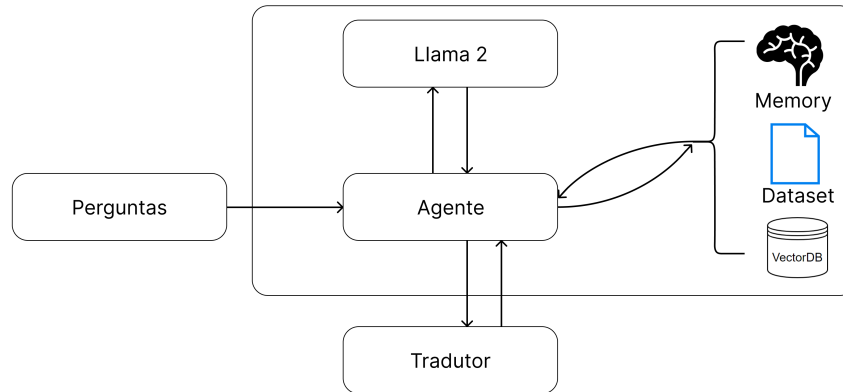
Outra ferramenta opcional é o VectorDB. Ele serve como um repositório para embeddings de segmentos de texto, facilitando buscas baseadas em similaridade semântica. O acesso rápido a essas representações vetoriais pode enriquecer a capacidade do sistema de fornecer informações pertinentes.

A figura central desta metodologia é o agente. Este, quando implementado, coordena a interação entre as diversas ferramentas, desde a pipeline até o VectorDB. O agente processa as consultas do usuário, consulta os recursos necessários e, finalmente, compila as respostas.

Adicionalmente, a metodologia destaca a importância de se trabalhar com prompts. O uso de prompts estruturados pode guiar o modelo LLM para gerar respostas mais alinhadas com as expectativas do usuário e os objetivos do sistema.

É válido mencionar que, embora algumas ferramentas sejam opcionais, a ideia é conceber a melhor arquitetura possível. A combinação otimizada dessas ferramentas e técnicas promete um agente conversacional que seja não só preciso, mas também eficaz em compreender e atender às necessidades do usuário. A Figura 9, representa o diagrama simplista do fluxo do sistema.

Figura 9 – Diagrama de fluxo de como o LangChain funciona



Fonte: Elaborado pelo autor.

5 DESENVOLVIMENTO

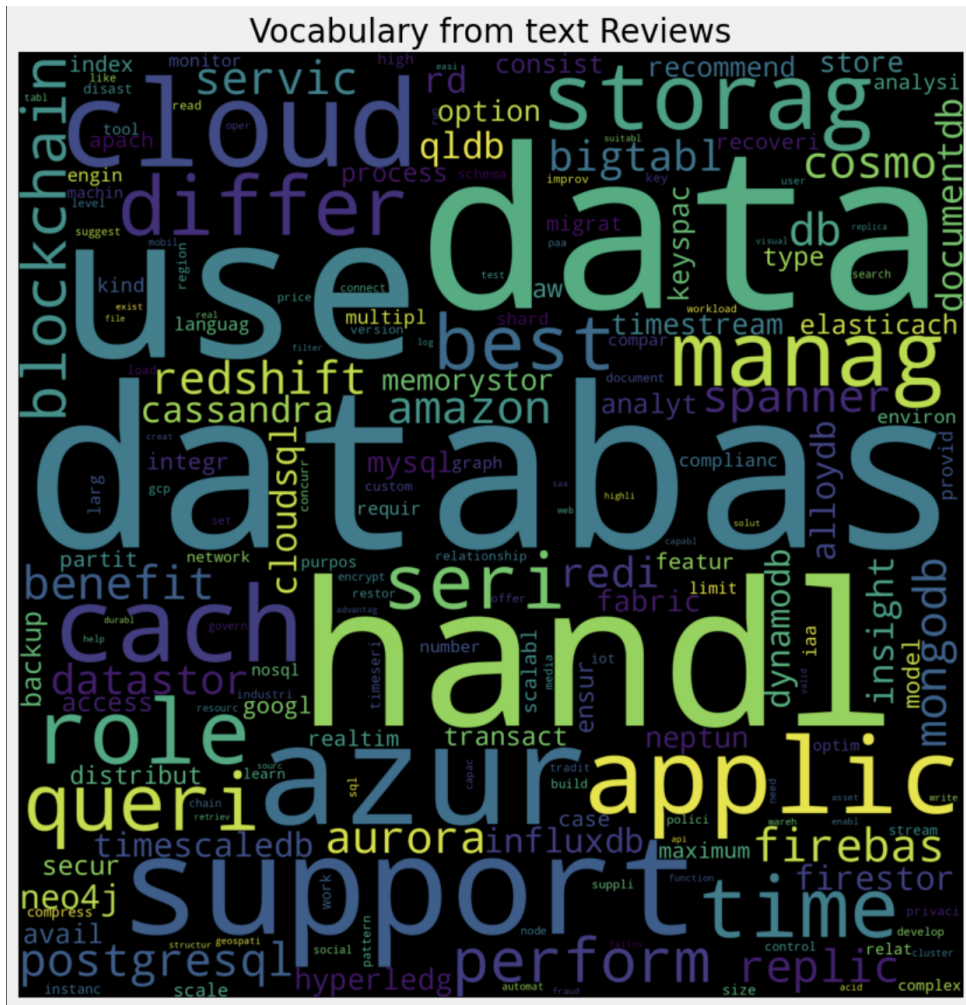
Nesta seção aborda-se a implementação de diversos modelos de processamento de linguagem natural empregados neste trabalho. Desde abordagens tradicionais, como TF-IDF e Unigram, passando por técnicas avançadas como BERT, até a integração de modelos de aprendizado de linguagem de última geração, como LLM com Llama2, esta seção desvenda o intrincado processo de transformar perguntas textuais em respostas precisas e relevantes. Ao explorar cada modelo e técnica, mergulhamos profundamente nas decisões, desafios e inovações que moldam este ambicioso projeto.

Os códigos foram escritos em Python, aproveitando-se de diversas bibliotecas consagradas para assegurar eficiência e precisão, como pandas, numpy, nltk, sklearn, scikitplot, joblib, bitsandbytes, transformers, torch, peft. Para o desenvolvimento, edição e execução dos códigos, utilizamos o Jupyter Notebook, uma ferramenta interativa que facilita a visualização dos resultados e a iteração rápida. Adicionalmente, para garantir maior flexibilidade e acesso a recursos computacionais, os testes foram realizados no Google Colaboratory, uma plataforma de pesquisa em machine learning que oferece ambiente de execução gratuito na nuvem. Especificamente para o modelo Llama2, recorreremos à biblioteca da Hugging Face, uma referência na comunidade de processamento de linguagem natural, que proporciona uma vasta gama de modelos pré-treinados e ferramentas para facilitar o desenvolvimento de aplicações de ponta em PLN.

5.1 DATASET

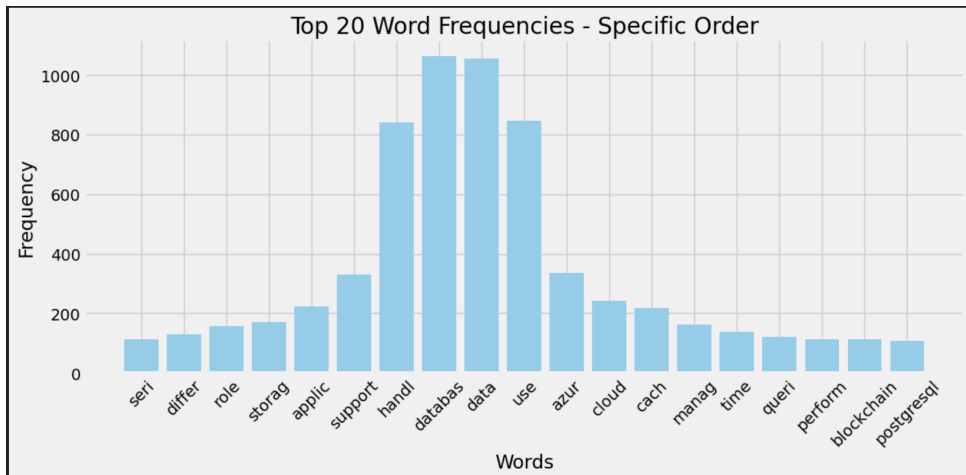
No presente estágio, foi compilado um conjunto de dados baseado nas informações provenientes dos conjuntos públicos descritos na metodologia. Este conjunto de dados está em formato CSV e contém 3.482 registros, distribuídos em colunas referentes a perguntas, respostas e a categoria alvo, que indica a classe de banco de dados mais apropriada. Procedeu-se a uma análise para identificar as palavras de maior ocorrência, empregando-se uma visualização em nuvem de palavras no dataset, conforme a Figura 10, um histograma destacando as 20 palavras mais frequentes, apresentado na Figura 11. Adicionalmente, realizou-se uma avaliação do equilíbrio entre as classes, a fim de assegurar que nenhuma classe de banco de dados estivesse sub-representada em relação às demais. Conforme evidenciado pela Figura 12, o conjunto de dados apresenta um balanceamento adequado, tornando desnecessária a aplicação de técnicas de aumento de dados.

Figura 10 – Nuvem de palavras do treino.



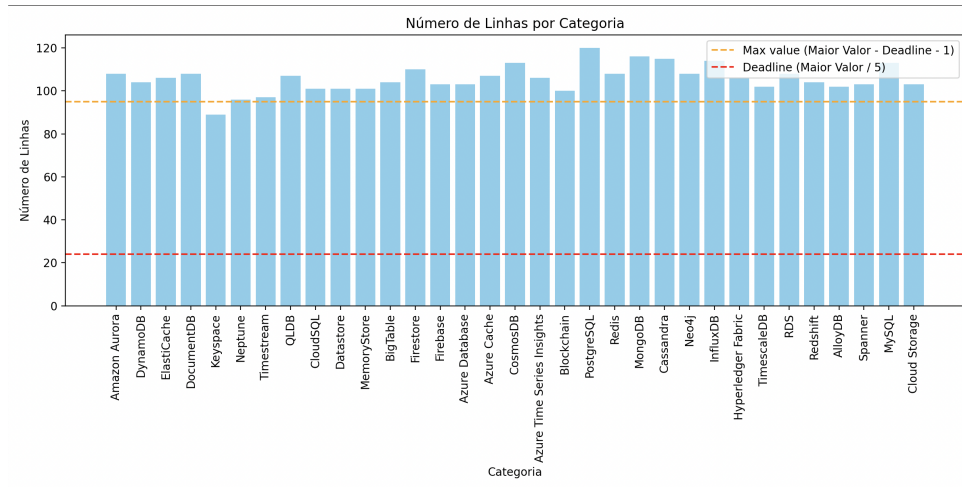
Fonte: Elaborado pelo autor

Figura 11 – Histograma das 20 palavras com maior frequência.



Fonte: Elaborado pelo autor

Figura 12 – Histograma das 20 palavras com maior frequência.



Fonte: Elaborado pelo autor

5.2 TRADUÇÃO

Na etapa inicial do modelo de resposta a perguntas, é essencial traduzir a consulta oriunda do português para o inglês. O algoritmo responsável por essa tradução foi desenvolvido em Python, utilizando a biblioteca EasyGoogleTranslate. Embora a implementação atual priorize a tradução do português para o inglês, o algoritmo é versátil e capaz de acomodar outros idiomas. A função de tradução é parametrizada, aceitando tanto a língua de origem quanto a de destino. Esta modularidade permite que a função seja invocada tanto no início do processo, para traduzir a pergunta para o inglês, quanto no final, revertendo a resposta traduzida para o português.

5.3 RANDOM FOREST E TF-IDF

O modelo começa lendo o conjunto de dados e as perguntas. As questões são armazenadas na variável All_question1 e categorizadas em 33 classes distintas, conforme evidenciado na Figura 13. Na sequência do processo de treinamento, a fase de pré-processamento é crucial. Para assegurar a extração de informações pertinentes do conjunto de dados original, foi realizado a remoção das quebras de linha, consolidando as sentenças em um único corpo textual. Em seguida, caracteres associados a URLs, especificamente padrões como "(?:@|https?:/)+", foram excluídos. Dado que muitos textos utilizam caracteres ASCII para expressar emoções, optamos por sua remoção. Ademais, pontuações presentes nos textos foram eliminadas. Posteriormente, foi removido palavras consideradas irrelevantes no contexto do idioma inglês, que não agregavam valor semântico ao conteúdo, as palavras de parada. Palavras com extensão superior a 35 caracteres também foram descartadas. Também foram utilizadas as técnicas de lematização e stematização para pegar somente o essencial da palavra.

Figura 13 – O ID dos rótulos para cada classe de conjunto de dados.

```
label_id = {"0": "Amazon_Aurora", '1': 'DynamoDB', '2': 'ElastiCache', '3': 'DocumentDB',
            '4': 'Keyspace', '5': 'Neptune', '6': 'Timestream', '7': 'QLDB',
            '8': 'CloudSQL', '9': 'Datastore', '10': 'MemoryStore', '11': 'BigTable', '12': 'Firestore',
            '13': 'Firebase', '14': 'Azure_Database', '15': 'Azure_Cache',
            '16': 'CosmosDB', '17': 'Azure_Time_Series_Insights', '18': 'Blockchain', '19': 'PostgreSQL',
            '20': 'Redis', '21': 'MongoDB', '22': 'Cassandra',
            '23': 'Neo4j', '24': 'InfluxDB', '25': 'Hyperledger_Fabric', '26': 'TimescaleDB', '27': 'RDS',
            '28': 'Redshift', '29': 'AlloyDB',
            '30': 'Spanner', '31': 'MySQL', '32': 'Cloud_Storage'}
```

Fonte: Elaborado pelo autor

Figura 14 – O resultado da incorporação de palavras para o conjunto de treinamento.

```
1 word_vectorizer = TfidfVectorizer(
2     sublinear_tf=True,
3     strip_accents='unicode',
4     analyzer='word',
5     ngram_range=(1, 1),
6     max_features=2000)
7
8 unigramdataGet = word_vectorizer.fit_transform(Procesed_questions)
9 unigramdataGet = unigramdataGet.toarray()
10
11 vocab = word_vectorizer.get_feature_names_out()
12 unigramdata_features=pd.DataFrame(np.round(unigramdataGet, 1), columns=vocab)
13 unigramdata_features[unigramdata_features>0] = 1
14 unigramdata_features.head()
```

	10	100	11	12	13	14	15	16	17	18	...	workload	wrangl	write	writebehind	writeheavi	writethrough	xml	xtradb	zeppelin	zone
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

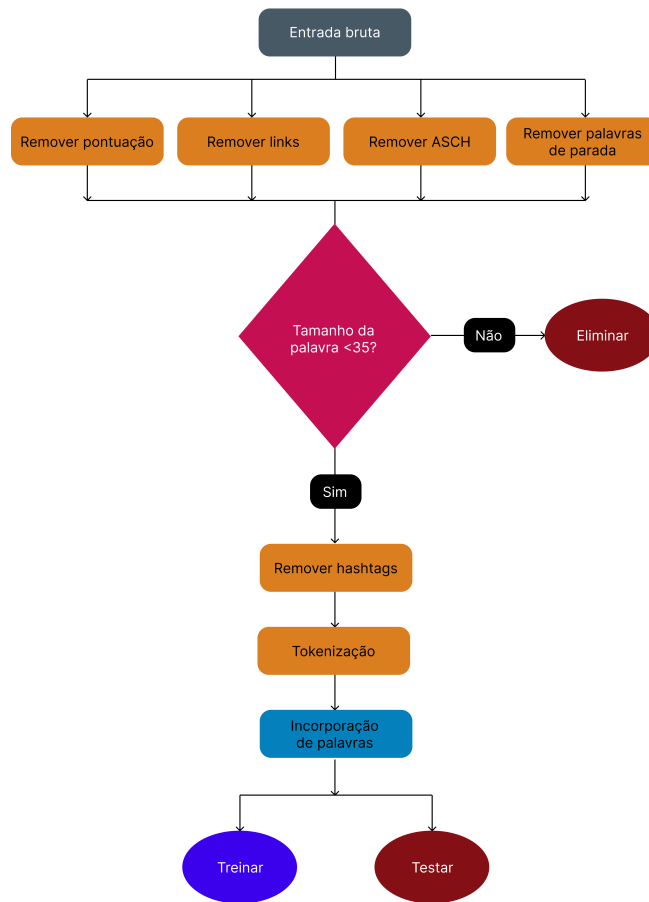
5 rows x 1247 columns

Fonte: Elaborado pelo autor

Após a adequada etapa de pré-processamento, foi realizado o estágio de codificação. Durante esta fase, recorre-se à técnica de word embedding para transformar cada termo das perguntas em vetores numéricos. Durante esta etapa, além desta técnica, também se utiliza a vetorização TF-IDF para transformar os dados de texto pré-processados em representações numéricas. É utilizado, adicionalmente, a técnica de junção unigrama, que considera uma única palavra como um token. Estes vetores são essenciais para alimentar e otimizar o desempenho dos algoritmos de aprendizado de máquina. A incorporação de word embeddings potencializou a precisão na análise textual, facilitando a interpretação da linguagem natural pelas máquinas. O resultado desse processo de codificação é visualmente representado na Figura 14. As palavras e os números são meticulosamente mapeados em vetores específicos, garantindo uma codificação uniforme para os conjuntos de treinamento e teste. Como ilustrado na Figura 15, após a codificação, é segmentado o conjunto de dados, destinando 70% para treinamento e 30% para testes, foi utilizado o módulo `sklearn.model_selection` importando o `train_test_split`. Concluída essa segmentação, foi implementado o modelo Random Forest para treinar o conjunto de dados.

A otimização de hiperparâmetros é uma etapa crucial na construção de modelos de aprendizado de máquina, uma vez que a seleção adequada desses parâmetros

Figura 15 – Fluxograma dos processos de pré-processamento e codificação.



Fonte: Elaborado pelo autor

pode influenciar significativamente o desempenho do modelo. No contexto do modelo Random Forest, diversos hiperparâmetros, tais como o número de estimadores, a quantidade máxima de características a serem consideradas, a profundidade máxima das árvores e o mínimo de amostras necessárias para efetuar uma divisão em um nó, devem ser meticulosamente ajustados.

Para efetuar essa otimização, empregou-se a técnica denominada Grid Search ou pesquisa em grade. Essa técnica consiste em uma abordagem exaustiva que avalia o modelo sob todas as combinações possíveis de valores de hiperparâmetros predefinidos. O principal benefício da pesquisa em grade reside na sua capacidade de realizar uma busca sistemática e objetiva, eliminando a necessidade de ajustes manuais ou dependência de intuições que podem não ser precisas, um exemplo do código do grid search está na Figura 16.

n-estimators 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

max-features 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

Figura 16 – Pesquisa em grade para ajustes de hiperparâmetros.

```
rf = RandomForestClassifier(max_depth=40,max_features=50,min_samples_split=12,n_estimators=40)
parameters = {
    'n_estimators': [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'max_depth': [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'max_features': [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'min_samples_split': [1, 2, 4, 6, 8, 10, 12]
}

from sklearn.model_selection import GridSearchCV
clf = GridSearchCV(rf, parameters, cv=5, verbose=1, n_jobs=-1, error_score='raise')
clf.fit(X_trainsm, y_trainsm)
print(clf.best_params_)
```

Fonte: Elaborado pelo autor

Figura 17 – Código da utilização do Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

[41] ✓ 0.0s

RandomForest=RandomForestClassifier(max_depth=40,max_features=50,min_samples_split=12,n_estimators=40)
RandomForest.fit(X_trainsm, y_trainsm)
y_pred_trainsm_rnd = RandomForest.predict(X_trainsm)
y_predsm_rnd = RandomForest.predict(X_testsm)
accuracy_trainsm = accuracy_score(y_trainsm, y_pred_trainsm_rnd)
accuracyism = accuracy_score(y_testsm, y_predsm_rnd)
print("Accuracyism of random_forest: %.2f%%" % (accuracyism * 100.0))
print("Accuracy trainsm of random_forest: %.2f%%" % (accuracy_trainsm * 100.0))

[42] ✓ 0.3s

... Accuracyism of random_forest: 96.52%
Accuracy trainsm of random_forest: 98.73%
```

Fonte: Elaborado pelo autor

max-depth 1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100

min-samples-split 1, 2, 4, 6, 8, 10, 12

Conforme detalhado acima, as possíveis configurações para o número de árvores, estimadores, dentro da floresta variam de 1 a 100. Durante a execução da pesquisa em grade, cada combinação de hiperparâmetros é avaliada, e sua performance é comparada com base em métricas predefinidas no conjunto de validação. Dentre as combinações avaliadas, identificou-se que a combinação ótima para o modelo Random Forest foi alcançada com uma profundidade máxima de 40, considerando 50 características máximas, com um mínimo de 12 amostras para divisão e composto por 40 estimadores, conforme Figura 17.

Após implementado o modelo é testado com uma pergunta e retorna um dos `label_ids` comentados na Figura 13. Para este modelo foram utilizados diversas bibliotecas entre elas `pandas`, `numpy`, `nlTK`, `sklearn`, `scikitplot`, `joblib`.

Figura 18 – Mapeamento das perguntas e classes

```

# Map the "target" values to the corresponding category names
df['Category'] = df['target'].map(label_to_category)

# Keep only the "Question" and "Category" columns
df_filtered = df[['Question', 'Category']]

# Now, the 'df' DataFrame contains "Question" and "Category" columns
print(df_filtered.tail())

```

	Question	Category
205	Which database is best for multimedia content ...	MySQL
206	I need a highly reliable relational database t...	Cloud_Storage
207	Which database is best for object storage?	Cloud_Storage
208	Which database is best for image data?	Cloud_Storage
209	Which distributed graph database for complex r...	Neo4j

Fonte: Elaborado pelo autor

5.4 ÁRVORE DE DECISÃO E TF-IDF

No desenvolvimento do modelo proposto, a primeira etapa consiste na leitura do conjunto de dados e nas respectivas perguntas. Uma subsequente operação de mapeamento é realizada, associando cada pergunta à categoria da resposta correspondente, conforme ilustrado na Figura 18.

Posteriormente, no processo, quatro arrays são estruturados, representando distintos provedores de serviços em nuvem: AWS, Azure, GCP e uma categoria denominada "Outros", este é utilizado no caso a pergunta não se encaixe em nenhum dos requisitos. Cada um desses arranjos engloba os bancos de dados associados ao respectivo provedor. Tal configuração foi meticulosamente adotada visando o treinamento particularizado do modelo para cada provedor. Assim, se uma pergunta contiver especificidades pertencentes unicamente a um provedor, o modelo tende a associar a pergunta a essa plataforma em particular.

No que tange à vetorização do texto contendo as perguntas e categorias de cada modelo, empregou-se a técnica TF-IDF. O algoritmo selecionado para essa tarefa foi uma Árvore de Decisão. Além disso, uma pesquisa em grade foi implementada para determinar os hiperparâmetros ideais. A Figura 19 exibe os parâmetros otimizados para o modelo, sendo eles: `max_depth` igual a 60, `max_features` igual a 20 e `min_samples_split` igual a 2. Adicionalmente, para a etapa de treinamento, optou-se pela estratégia de validação cruzada K-fold, com K definido como 10, conforme sugerido pela literatura. A Figura 20 apresenta a função empregada para este propósito.

Concluído o treinamento, quando um usuário consulta o sistema, a avaliação dos

Figura 19 – Pesquisa em grade com os melhores parâmetros

```
# Define a parameter grid for hyperparameter tuning
param_grid = {
    'max_depth': [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'max_features': [1, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
    'min_samples_split': [1, 2, 4, 6, 8, 10, 12]
}

# Create a GridSearchCV object
grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5)

# Fit the grid search to the training data
grid_search.fit(X_resampled, y_resampled)

# Get the best hyperparameters
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Train the model with the best hyperparameters
best_model = DecisionTreeClassifier(**best_params, random_state=42)

✓ 8.2s
Best Hyperparameters: {'max_depth': 60, 'max_features': 20, 'min_samples_split': 2}
```

Fonte: Elaborado pelo autor

requisitos é executada por meio da API da OpenAI. A consulta é formulada como um prompt, e a API, por sua vez, retorna um JSON contendo a característica do provedor de serviços em nuvem. Com base nessa resposta, o sistema direciona a consulta para um dos submodelos específicos relacionados ao provedor de nuvem indicado, um exemplo é indicado pela Figura 21. Finalmente, o modelo fornece uma resposta categorizada em uma das classes de banco de dados.

5.5 RANDOM FOREST, KNN E REGRESSÃO LOGÍSTICA COM BERT

Neste modelo, inicia-se traduzindo perguntas do português para o inglês, em seguida, os dados das perguntas foram carregados de várias categorias, como Amazon Aurora e DynamoDB, do dataset. Para consolidar essas informações, todas as perguntas de diferentes categorias foram concatenadas em uma única variável.

O próximo passo envolveu uma série de etapas de pré-processamento de dados aplicadas a cada pergunta, incluindo limpeza de texto, remoção de caracteres especiais, descontração de palavras, lematização e stematização. As perguntas processadas são armazenadas na lista `Processed_questions`.

Com os dados preparados, o modelo BERT da biblioteca Hugging Face Transformers foi empregado para extrair embeddings de cada pergunta processada. Estes embeddings foram utilizados como características para treinar e testar os modelos. Para lidar com a alta dimensionalidade dos embeddings BERT, utilizou-se o PCA como uma opção para redução de dimensionalidade, conforme Figura 22.

A fase subsequente envolve o treinamento de vários modelos, incluindo o

Figura 20 – Função para treinamento usando k-folders

```

# Define the number of folds for cross-validation
n_splits = 10

# Initialize variables to keep track of overall accuracy
total_accuracy = 0.0

# Create a KFold cross-validation object
kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)

for train_index, test_index in kf.split(X_resampled):
    # Split the data into training and testing sets for this fold
    X_train, X_test = X_resampled[train_index], X_resampled[test_index]
    y_train, y_test = y_resampled[train_index], y_resampled[test_index]

    # Train the Decision Tree model on the resampled data
    best_model.fit(X_train, y_train)

    # Make predictions on the test data
    y_pred = best_model.predict(X_test)

    # Calculate accuracy for this fold
    fold_accuracy = accuracy_score(y_test, y_pred)
    print("Fold Accuracy:", fold_accuracy)

    # Add the accuracy for this fold to the total accuracy
    total_accuracy += fold_accuracy

# Calculate the average accuracy across all folds
average_accuracy = total_accuracy / n_splits
print("Average Accuracy (across all folds):", average_accuracy)

```

```

Fold Accuracy: 0.9848484848484849
Fold Accuracy: 0.9772727272727273
Fold Accuracy: 0.9545454545454546
Fold Accuracy: 0.9696969696969697
Fold Accuracy: 0.9393939393939394
Fold Accuracy: 0.9772727272727273
Fold Accuracy: 0.9772727272727273
Fold Accuracy: 0.9621212121212122
Fold Accuracy: 0.9621212121212122
Fold Accuracy: 0.9924242424242424
Average Accuracy (across all folds): 0.9696969696969697

```

Fonte: Elaborado pelo autor

Figura 21 – Função para validação da pergunta pela API OpenAI

```

input_question = input("Por favor, faça sua pergunta: ")
print(input_question)

response = get_features_from_openai(input_question, prompt)
print(response)

predicted_category = predict_category(input_question, json.dumps(response))
print("Predicted Category:", predicted_category)

{
  "cloud": "String",
  "service": "String",
  "datatypes": "Array",
  "features": "Array",
  "application": "String",
  "observations": "Array"
}
Predicted Category: MemoryStore

```

Fonte: Elaborado pelo autor

Figura 22 – Análise de Componentes Principais

```

from sklearn.decomposition import PCA

# Assuming bert_embeddings is a NumPy array containing BERT embeddings for your training data**
# Define the number of components you want to retain
n_components = 50

# Initialize the PCA model
pca = PCA(n_components=n_components)

# Fit and transform the BERT embeddings using the PCA model
reduced_bert_embeddings = pca.fit_transform(bert_embeddings)

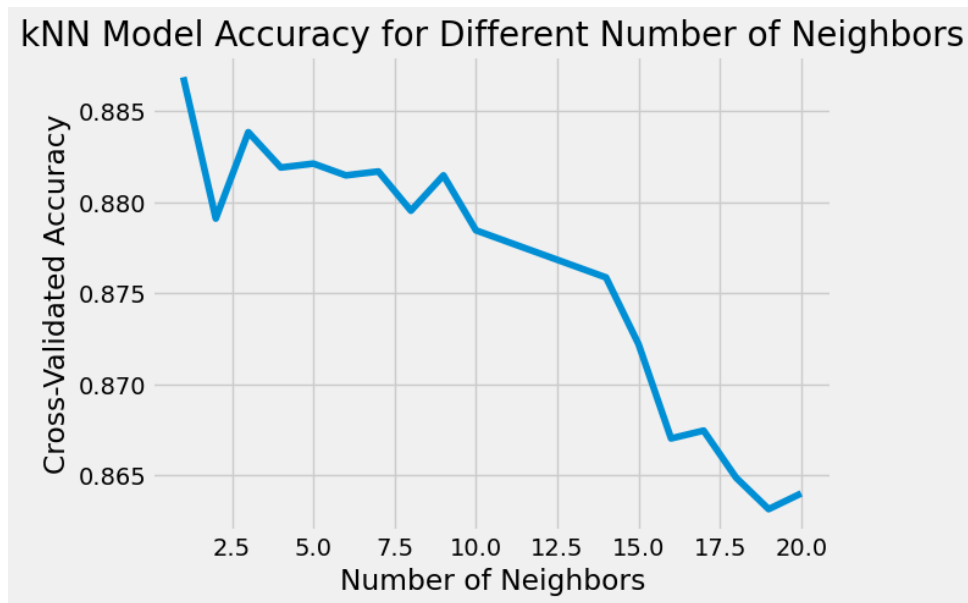
import joblib

# Save the PCA model to a file
joblib.dump(pca, 'pca_model.joblib')

```

Fonte: Elaborado pelo autor

Figura 23 – Hiperparâmetros do KNN.



Fonte: Elaborado pelo autor

Figura 24 – Hiperparâmetros do Random Forest.

```

RandomForestClassifier
RandomForestClassifier(max_depth=20, max_features='log2', min_samples_leaf=3,
min_samples_split=4, n_estimators=97)

```

Fonte: Elaborado pelo autor

Random Forest, KNN (K-Nearest Neighbors), Regressão Logística e um modelo híbrido que une KNN com Random Forest. Um elemento-chave neste processo é o ajuste e validação dos hiperparâmetros, pois isso é determinante para otimizar a performance de cada modelo. Para esta otimização, o script utiliza o `RandomizedSearchCV`, uma técnica que realiza buscas aleatórias nos espaços de hiperparâmetros visando encontrar as combinações mais propícias. Esta abordagem se diferencia da pesquisa de grade (Grid Search) pois não explora todas as combinações possíveis, tornando-se mais eficaz, especialmente quando o espaço de hiperparâmetros é extenso.

Especificamente, o modelo KNN foi ajustado com um parâmetro de K recebeu 1 e a métrica de distância escolhida foi a minkowski, conforme a Figura 23. Já o modelo Random Forest teve seus hiperparâmetros ajustados, cujos resultados estão detalhados na Figura 24. A Regressão Logística, por sua vez, foi otimizada com certos parâmetros, e seus detalhes estão na Figura 25.

Uma vez treinados e otimizados, os modelos são avaliados utilizando métricas de classificação, como acurácia, precisão, recall e F1-score. Estas métricas fornecem uma visão abrangente do desempenho do modelo em diferentes aspectos. O treinamento com validação cruzada de 10 pastas garante que o modelo seja robusto. Este esquema

Figura 25 – Hiperparâmetros da Regressão Logística.

```

▼ LogisticRegression
LogisticRegression(C=0.3728566270161285, max_iter=230, solver='newton-cg')

```

Fonte: Elaborado pelo autor

Figura 26 – Função de teste do modelo

```

# Load pre-trained model tokenizer and model
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased')
model.eval()

# Load the trained Logistic Regression model
loaded_model = joblib.load("LogReg_Bert_Question_Answering.joblib")

# Load PCA model if necessary
# Load the PCA model from a file
pca = joblib.load('pca_model.joblib')

# Define the function to test the model
def testing_the_model(question):
    # Preprocess the question
    question = deep_clean(question)

    # Convert preprocessed question to BERT embeddings
    inputs = tokenizer(question, return_tensors='pt', truncation=True, max_length=512, padding='max_length')
    with torch.no_grad():
        outputs = model(**inputs)
        sentence_embedding = torch.mean(outputs.last_hidden_state, dim=1).squeeze().numpy()

    # Apply PCA if it was used during training
    reduced_sentence_embedding = pca.transform(sentence_embedding.reshape(1, -1))

    # Make a prediction using the Logistic Regression model
    predicted_label = loaded_model.predict(reduced_sentence_embedding)[0]

    # Convert the numerical label to category name
    predicted_category = label_id[str(predicted_label)]

    return predicted_category

```

Fonte: Elaborado pelo autor

divide o conjunto de dados em 10 partes e realiza o treinamento 10 vezes, utilizando 9 para treinamento e 1 para teste em cada ciclo.

Após a etapa de treinamento, os modelos foram armazenados em arquivos, assegurando que possam ser utilizados posteriormente. Uma função específica foi estabelecida para avaliar o modelo com novas questões, garantindo que entradas subsequentes sejam processadas e classificadas corretamente, conforme ilustrado na Figura 26. Adicionalmente, recorre-se à função de tradução para converter a resposta obtida para o idioma português.

Concluindo, enquanto o uso de embeddings BERT e o classificador Random Forest proporcionou uma abordagem robusta para a tarefa, também se revelou complexo e exigente em termos de recursos, levantando questões sobre sua viabilidade em cenários práticos.

5.6 LLM LLAMA 2 AFINADA

Para desenvolver este modelo, iniciou-se pela instalação de bibliotecas essenciais, incluindo `huggingface_hub`, `bitsandbytes`, `datasets` e `transformers`. A seguir, foi feito o login na plataforma Hugging Face, permitindo o download do modelo escolhido, `meta-llama/Llama-2-7b-hf`, que se destaca por seus 7 bilhões de parâmetros, otimizados para conversação.

A fase de coleta de dados envolveu a reutilização de um dataset previamente empregado em outros modelos. Reconhecendo a importância de um formato de dados otimizado para o ajuste fino, foi realizada uma etapa de transformação dos dados. O resultado foi o conjunto de dados `combined_data.jsonl`, estruturado no formato JSON Lines (JSONL).

Dentro do `combined_data.jsonl`, encontram-se mais de 500 registros, cada um contendo campos como "instrução", "contexto", "resposta" e "categoria". Esses campos desempenham funções vitais, desde armazenar consultas típicas dos usuários até oferecer respostas e recomendações adequadas. Este conjunto de dados tornou-se uma fonte abrangente de conhecimento, essencial para responder a consultas de usuários sobre bancos de dados e serviços em nuvem.

Com os dados preparados, foi feito o processo de ajuste fino. Durante este processo, várias configurações e hiperparâmetros foram empregados. A taxa de aprendizagem foi fixada em 0.0002 após testes empíricos, enquanto o tamanho do lote foi estabelecido em 1. Para otimização da memória, utilizamos acumulação de gradiente com 4 etapas e, visando estabilizar o treinamento, definimos 2 etapas de aquecimento. A precisão mista foi empregada através do `fp16`, e o progresso do treinamento foi monitorado e registrado a cada etapa. Por fim, o modelo ajustado foi salvo no diretório `outputs`.

Durante o treinamento do modelo, é desenvolvido um código que verifica cada resposta produzida. No entanto, devido à vasta quantidade de dados, a validação em uma GPU T4 é demorada. Portanto, optou-se por validar uma amostra de 500 registros em vez da totalidade dos dados, conforme comentado no início da seção. Um trecho deste código pode ser visto na Figura 27

5.7 LLM LLAMA 2 COM LANGCHAIN

No início do desenvolvimento, foi dada prioridade à importação de bibliotecas e definição de parâmetros essenciais. A biblioteca `transformers` da Hugging Face, em particular, foi fundamental devido à sua capacidade de interface com modelos de linguagem de grande escala, como o Llama. Bibliotecas como `accelerate`, `einops`, `langchain`,

Figura 27 – Validação das métricas

```

import nltk
from nltk.translate.meteor_score import meteor_score
import statistics

# Define lists to store Meteor scores and other statistics
meteor_scores = []

# Define reference answers
references = test_data["response"]

for i in range(len(references)):
    instruction = test_data["instruction"][i]
    generated_answer = get_response(instruction) # You should use your get_response function here
    reference = references[i]

    # Check if generated_answer is a valid string
    if isinstance(generated_answer, str):
        # Tokenize the generated_answer and reference
        generated_tokens = nltk.word_tokenize(generated_answer)
        reference_tokens = nltk.word_tokenize(reference)

        # Calculate Meteor score
        meteor = meteor_score([reference_tokens], generated_tokens)

        # Append the Meteor score to the list
        meteor_scores.append(meteor)

        print(f"Instruction: {instruction}")
        print(f"Generated Answer: {generated_answer}")
        print(f"Reference Answer: {reference}")
        print(f"Meteor Score: {meteor}")
        print()
    else:
        # Handle cases where generated_answer is not a valid string
        print(f"Instruction: {instruction}")
        print("Generated Answer is not a valid string.")
        print(f"Reference Answer: {reference}")
        print(f"Meteor Score: N/A")
        print()

# Calculate summary statistics
mean_meteor = statistics.mean(meteor_scores)
median_meteor = statistics.median(meteor_scores)
std_deviation_meteor = statistics.stdev(meteor_scores)

print("Summary Statistics:")
print(f"Mean Meteor Score: {mean_meteor}")
print(f"Median Meteor Score: {median_meteor}")
print(f"Standard Deviation Meteor Score: {std_deviation_meteor}")

```

Fonte: Elaborado pelo autor

xformers e bitsandbytes proporcionaram funcionalidades adicionais, otimizando o fluxo de trabalho.

Inicialmente, é empregado o método BitsAndBytesConfig para configurar a quantização. Esta ferramenta é uma parte do pacote bitsandbytes e permite definir parâmetros específicos para a quantização, garantindo que os vetores sejam comprimidos de acordo com as necessidades do sistema. A configuração adequada é crucial, pois determina como os vetores originais são reduzidos e quanta informação é retida no processo. A Tabela 4 contém os parâmetros utilizados.

Tabela 4 – Hiperparâmetros com seus valores

Hiperparâmetro	Valor
load_in_4bit	True
bnb_4bit_quant_type	nf4
bnb_4bit_use_double_quant	True
bnb_4bit_compute_dtype	bfloat16

Fonte: Elaborado pelo autor.

A autenticação na plataforma Hugging Face foi realizada, permitindo o acesso e carregamento do modelo Llama e suas configurações associadas. Para tal, foram utili-

zados os métodos AutoConfig e AutoTokenizer. O AutoConfig carrega as configurações do modelo, que especificam características como o número de camadas e tamanho de embeddings, garantindo que o modelo seja inicializado corretamente. O AutoTokenizer, por sua vez, fornece um tokenizer adequado ao modelo escolhido, essencial para a conversão de sequências textuais em tokens.

Uma vez configurado o modelo e o tokenizer, foi estabelecido um pipeline de geração de texto. Esse pipeline, oferecido pela Hugging Face, unifica as operações de tokenização e inferência, otimizando a geração de respostas. Diversos hiperparâmetros foram calibrados, assegurando respostas de alta qualidade. Os valores de cada hiperparâmetro estão descritos na Tabela 5.

Tabela 5 – Hiperparâmetros com seus valores

Hiperparâmetro	Valor
model	meta-llama/Llama-2-13b-chat-hf
tokenizer	sentence-transformers/all-mpnet-base-v2
return_full_text	True
task	text-generation
stopping_criteria	stopping_criteria (função personalizada)
temperature	0.1
max_new_tokens	512
repetition_penalty	1.1

Fonte: Elaborado pelo autor.

Para refinar a geração de texto, foram implementados critérios de parada personalizados. Estes utilizam tokens específicos para sinalizar ao modelo o momento adequado para finalizar a resposta, evitando textos excessivamente longos ou truncados.

No que se refere à obtenção de dados, uma série de URLs, sobretudo relacionados ao Databricks, foram integrados ao sistema através do WebBaseLoader. Esta ferramenta carrega e extrai informações dessas fontes web. Posteriormente, os dados recuperados foram processados pelo RecursiveCharacterTextSplitter, que segmentou os documentos em unidades menores para facilitar o manuseio.

Os documentos segmentados, após passarem por um processo de vetorização usando embeddings da Hugging Face, foram armazenados no VectorDB. Esse armazenamento vetorizado é crucial para uma busca rápida e precisa, permitindo ao sistema localizar informações relevantes em questão de milissegundos. A quantização foi empregada nesse contexto para otimizar ainda mais o armazenamento e a recuperação de dados, comprimindo os vetores sem perder significativamente a qualidade.

O agente conversacional, implementado através do LangChain, atua como o núcleo interativo do sistema. Ele gerencia as interações dos usuários, processa as consultas e formula respostas informadas, utilizando o modelo Llama, as informações do VectorDB e o dataset fornecido.

Um aspecto final foi a incorporação de uma funcionalidade de tradução. Assim, se um usuário formula uma consulta em português, esta é traduzida para o inglês para processamento. Após a geração da resposta, ocorre a tradução inversa, garantindo uma comunicação fluida com o usuário em sua língua nativa.

6 RESULTADOS

Nesta seção, apresentaremos os resultados obtidos dos modelos propostos. Cada modelo foi avaliado com base em sua precisão, robustez e capacidade de generalização. Vamos discutir o desempenho de cada modelo e os desafios identificados e oferecer insights sobre as potenciais melhorias e direções futuras.

6.1 RANDOM FOREST E TF-IDF

Na análise do modelo de classificação baseado no algoritmo Random Forest, um ponto de crítica fundamental foi observado, questionando a adequação do modelo para implementações em contextos práticos: a presença de overfitting. Embora os resultados iniciais indiquem uma performance notável, com uma acurácia de 96,34% para o conjunto de teste e 97,04% para o conjunto de treinamento, conforme Figura 28, a curva ROC, que se aproxima do valor unitário, aponta para um cenário de sobreajuste evidente, conforme ilustrado na Figura 29. Adicionalmente, a matriz de confusão, a despeito de seu desempenho quase impecável, revela inconsistências ao ser aplicada em testes práticos, como pode ser visualizado na Figura 30.

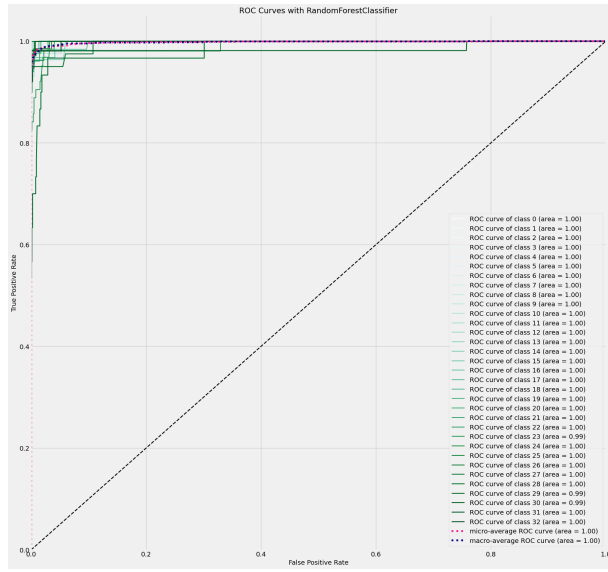
O overfitting, onde um modelo se adapta excessivamente aos dados de treinamento, é uma preocupação recorrente em aprendizado de máquina. Na análise

Figura 28 – Desempenho - RF e TF-IDF.

RandomForest						
	precision	recall	f1-score	support	pred	AUC
0	1.000000	0.962963	0.981132	27.0	26.0	0.998701
1	1.000000	0.888889	0.941176	36.0	32.0	0.995313
2	1.000000	0.937500	0.967742	32.0	30.0	0.985795
3	1.000000	1.000000	1.000000	27.0	27.0	1.000000
4	1.000000	1.000000	1.000000	28.0	28.0	1.000000
5	1.000000	0.966667	0.983051	30.0	29.0	0.999932
6	1.000000	1.000000	1.000000	33.0	33.0	1.000000
7	0.965517	1.000000	0.982456	28.0	29.0	1.000000
8	1.000000	0.971429	0.985507	35.0	34.0	1.000000
9	1.000000	1.000000	1.000000	25.0	25.0	1.000000
10	0.972222	0.945946	0.958904	37.0	36.0	0.999223
11	1.000000	1.000000	1.000000	33.0	33.0	1.000000
12	1.000000	1.000000	1.000000	29.0	29.0	1.000000
13	0.911765	0.968750	0.939394	32.0	34.0	0.995914
14	1.000000	0.968750	0.984127	32.0	31.0	0.994669
15	1.000000	1.000000	1.000000	28.0	28.0	1.000000
16	1.000000	0.972973	0.986301	37.0	36.0	0.997419
17	1.000000	1.000000	1.000000	30.0	30.0	1.000000
18	1.000000	1.000000	1.000000	24.0	24.0	1.000000
19	0.673469	0.970588	0.795181	34.0	49.0	0.999127
20	1.000000	1.000000	1.000000	35.0	35.0	1.000000
21	0.956522	0.758621	0.846154	29.0	23.0	0.989483
22	0.862069	0.892857	0.877193	28.0	29.0	0.993769
23	1.000000	0.937500	0.967742	32.0	30.0	1.000000
24	0.969697	0.969697	0.969697	33.0	33.0	0.999845
25	1.000000	0.975000	0.987342	40.0	39.0	0.997528
26	1.000000	1.000000	1.000000	19.0	19.0	1.000000
27	0.931034	0.964286	0.947368	28.0	29.0	0.998747
28	1.000000	1.000000	1.000000	30.0	30.0	1.000000
29	1.000000	1.000000	1.000000	29.0	29.0	1.000000
30	1.000000	1.000000	1.000000	28.0	28.0	1.000000
31	1.000000	0.967742	0.983607	31.0	30.0	0.999276
32	0.911765	0.968750	0.939394	32.0	34.0	0.997446
avg / total	0.973495	0.968348	0.969372	1011.0	1011.0	0.997739

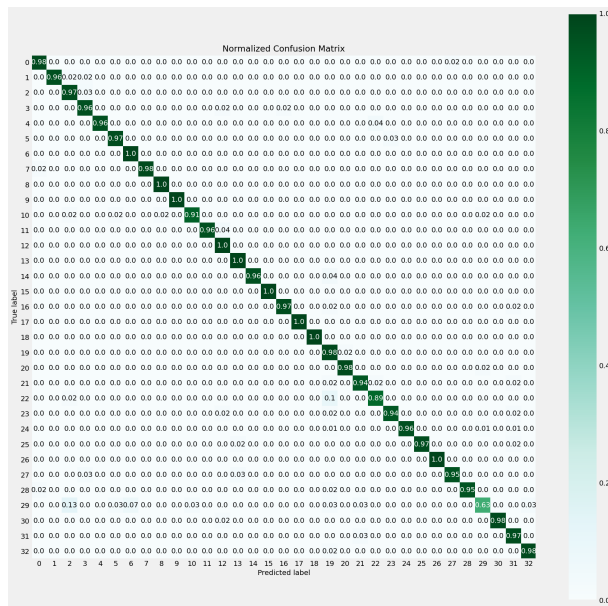
Fonte: Elaborado pelo autor

Figura 29 – Curva ROC - RF e TF-IDF.



Fonte: Elaborado pelo autor

Figura 30 – Matriz de confusão - RF e TF-IDF.



Fonte: Elaborado pelo autor

Figura 31 – Resultado final - RF e TF-IDF.

```

The asked question is : Which database is not best for mobile apps?
The best category of dataset is Firebase
*****
The asked question is : Which database is best for data storage?
The best category of dataset is Cloud_Storage
*****
The asked question is : Which database is best for geospatial data?
The best category of dataset is PostgreSQL
*****
The asked question is : Which database is best for analysis?
The best category of dataset is MySQL
*****
The asked question is : Which database is best for high availability and fault toleran
The best category of dataset is MySQL
*****
The asked question is : Which database is best for document-oriented data?
The best category of dataset is PostgreSQL
*****
The asked question is : Which database is best for graph data?
The best category of dataset is Neo4j
*****
The asked question is : Which database is best for unstructured data?
The best category of dataset is MongoDB

```

Fonte: Elaborado pelo autor

com o modelo Random Forest, percebemos que, apesar de uma elevada performance nos dados de treinamento, o modelo pode não ter uma capacidade adequada de generalização em cenários práticos, especialmente ao enfrentar dados não vistos anteriormente. Esse comportamento pode ser atribuído a um ajuste muito preciso dos hiperparâmetros e a um pré-processamento de texto extremamente detalhado. As deficiências tornam-se evidentes em aplicações reais, com previsões potencialmente imprecisas, conforme exemplificado na Figura 31. Além disso pelo uso do TF-IDF não é considerado o contexto da frase, caso o usuário questione "não quero utilizar a Azure", para o modelo ele considerará a resposta como Azure, por conta da frequência em relação a outras palavras.

6.2 ÁRVORE DE DECISÃO E TF-IDF

Na recente análise do modelo baseado em árvore de decisão, emergem preocupações cruciais sobre sua aplicabilidade prática, mesmo diante de métricas impressionantes. A estratégia de usar quatro submodelos distintos para categorização pode se revelar problemática. Uma das maiores vulnerabilidades é a dependência de palavras-chave para determinar o modelo a ser aplicado. Se a segregação inicial falhar, o sistema pode recorrer a um submodelo inadequado, resultando em respostas incorretas.

Este problema é exacerbado pelo desequilíbrio observado nos dados. Com uma distribuição desigual de categorias entre os submodelos, o risco de previsões distorcidas aumenta. Além disso, a estrutura atual do código assume um cenário estático, tanto para os dados quanto para os modelos. No entanto, o mundo da tecnologia é

Figura 32 – Métricas do modelo AWS.

```

Accuracy for AWS Model: 0.98
Classification Report for AWS Model:

```

	precision	recall	f1-score	support
Amazon_Aurora	1.00	1.00	1.00	18
DocumentDB	1.00	1.00	1.00	16
DynamoDB	1.00	1.00	1.00	8
ElastiCache	1.00	1.00	1.00	12
Keyspace	1.00	1.00	1.00	4
Neptune	1.00	1.00	1.00	2
QLDB	1.00	1.00	1.00	14
RDS	0.89	1.00	0.94	16
Redshift	1.00	0.75	0.86	8
Timestream	1.00	1.00	1.00	4
accuracy			0.98	102
macro avg	0.99	0.97	0.98	102
weighted avg	0.98	0.98	0.98	102

Fonte: Elaborado pelo autor

dinâmico, bancos de dados e serviços estão em constante evolução, tornando imperativo o monitoramento e atualização contínua para manter a relevância e precisão dos modelos.

Mesmo com métricas louváveis, como precisão, recall e f1-score, não podemos ignorar o aparente overfitting, evidenciado pelas curvas ROC. Tal comportamento sugere que o modelo pode não estar adequadamente generalizado ou pode estar operando com um conjunto de dados insuficiente.

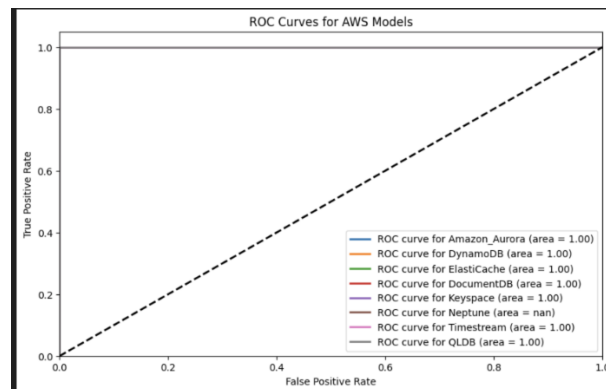
Os submodelos da AWS, GCP, Azure e Outros apresentaram resultados que merecem um exame detalhado. As métricas do submodelo AWS podem ser visualizadas nas Figuras 32, 33 e 34. O desempenho do submodelo GCP é delineado nas Figuras 35, 36 e 37. As métricas relacionadas ao submodelo Azure estão detalhadas nas Figuras 38, 39 e 40. Por fim, as métricas do submodelo Outros são apresentadas nas Figuras 41, 42 e 43.

Em síntese, mesmo com métricas indicando um desempenho notável, os desafios inerentes ao modelo suscitam dúvidas sobre sua eficácia em situações reais. Adicionalmente, os testes conduzidos revelaram incongruências entre as perguntas feitas e as respostas esperadas.

6.3 RANDOM FOREST, KNN E REGRESSÃO LOGÍSTICA COM BERT

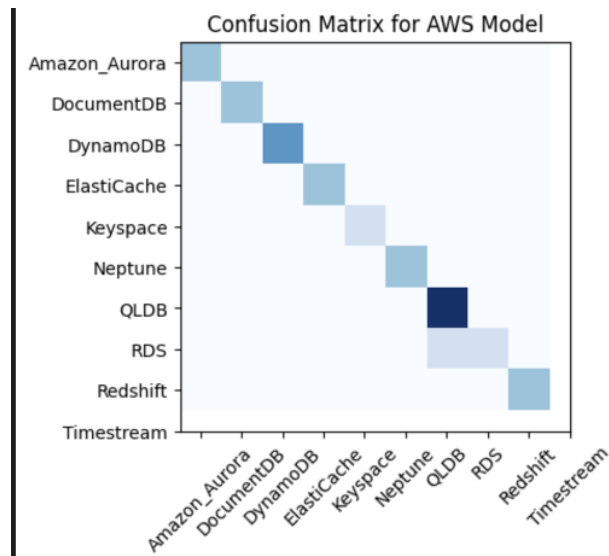
Os modelos foram avaliados com base em várias métricas, cujos detalhes específicos estão apresentados na Figura 44. Em nossa análise, o modelo Random Forest, kNN, Regressão Logística e o modelo combinado de kNN e Random Forest foram submetidos a testes para determinar sua eficácia na categorização de perguntas.

Figura 33 – Curva ROC do modelo AWS.



Fonte: Elaborado pelo autor

Figura 34 – Matriz de confusão do modelo AWS.



Fonte: Elaborado pelo autor

Figura 35 – Métricas do modelo GCP.

```

Accuracy for GCP Model: 0.96
Classification Report for GCP Model:

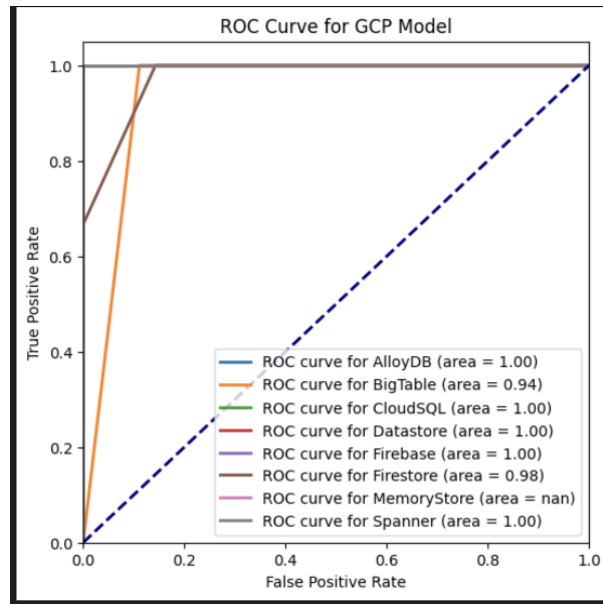
```

	precision	recall	f1-score	support
AlloyDB	1.00	1.00	1.00	4
BigTable	0.80	1.00	0.89	8
CloudSQL	1.00	1.00	1.00	2
Datastore	1.00	1.00	1.00	2
Firebase	1.00	1.00	1.00	6
Firestore	1.00	0.90	0.95	20
MemoryStore	1.00	1.00	1.00	2
Spanner	1.00	1.00	1.00	6
accuracy			0.96	50
macro avg	0.97	0.99	0.98	50
weighted avg	0.97	0.96	0.96	50

Models for GCP category saved successfully.

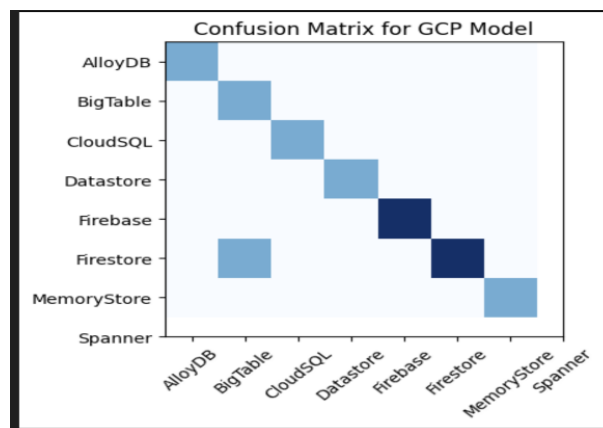
Fonte: Elaborado pelo autor

Figura 36 – Curva ROC do modelo GCP.



Fonte: Elaborado pelo autor

Figura 37 – Matriz de confusão do modelo GCP.



Fonte: Elaborado pelo autor

Figura 38 – Métricas do modelo Azure.

```

Accuracy for Azure Model: 0.93
Classification Report for Azure Model:
              precision    recall  f1-score   support

 Azure_Cache      0.88      1.00      0.93      14
 Azure_Database   1.00      1.00      1.00       6
 Azure_Time_Series_Insights  1.00      0.83      0.91      12
 CosmosDB         0.92      0.92      0.92      26

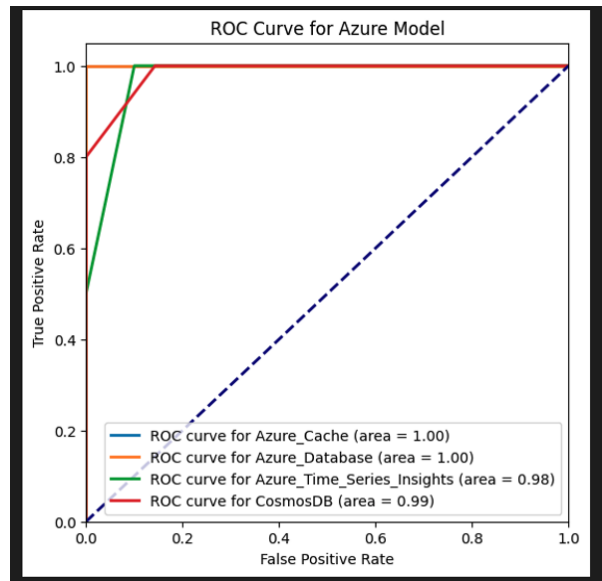
 accuracy              0.93      58
 macro avg            0.95      0.94      0.94      58
 weighted avg         0.94      0.93      0.93      58

Models for Azure category saved successfully.

```

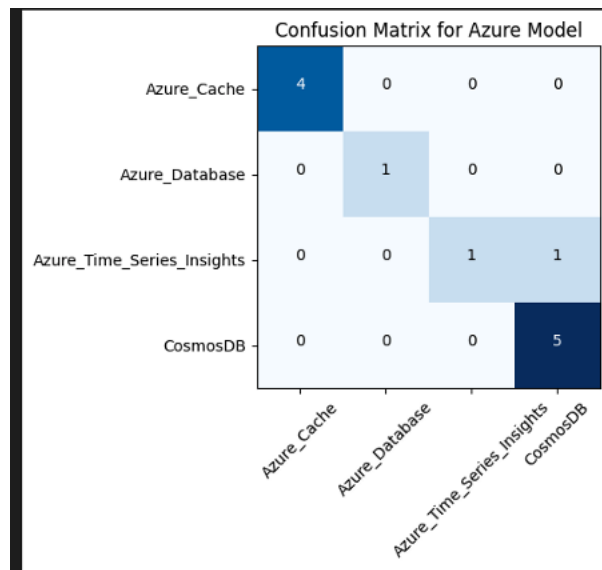
Fonte: Elaborado pelo autor

Figura 39 – Curva ROC do modelo Azure.



Fonte: Elaborado pelo autor

Figura 40 – Matriz de confusão do modelo Azure.



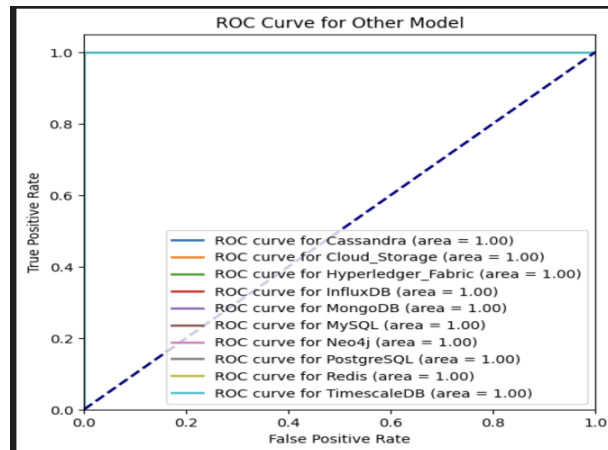
Fonte: Elaborado pelo autor

Figura 41 – Métricas do modelo Outros.

Accuracy for 'Other' Model: 0.98

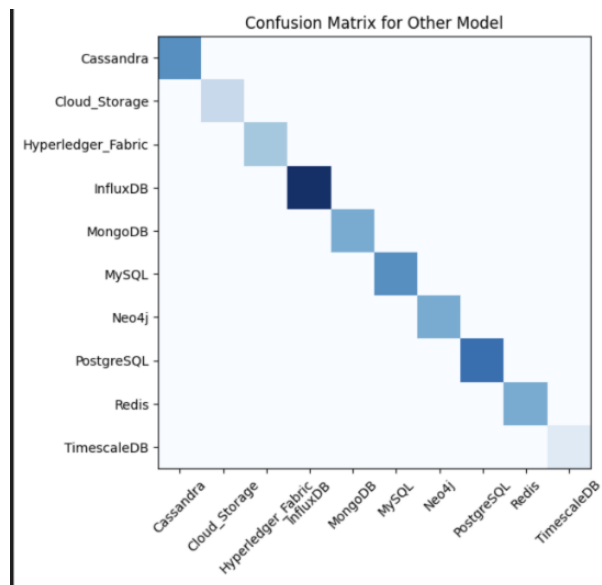
Fonte: Elaborado pelo autor

Figura 42 – Curva ROC do modelo Outros.



Fonte: Elaborado pelo autor

Figura 43 – Matriz de confusão do modelo Outros.



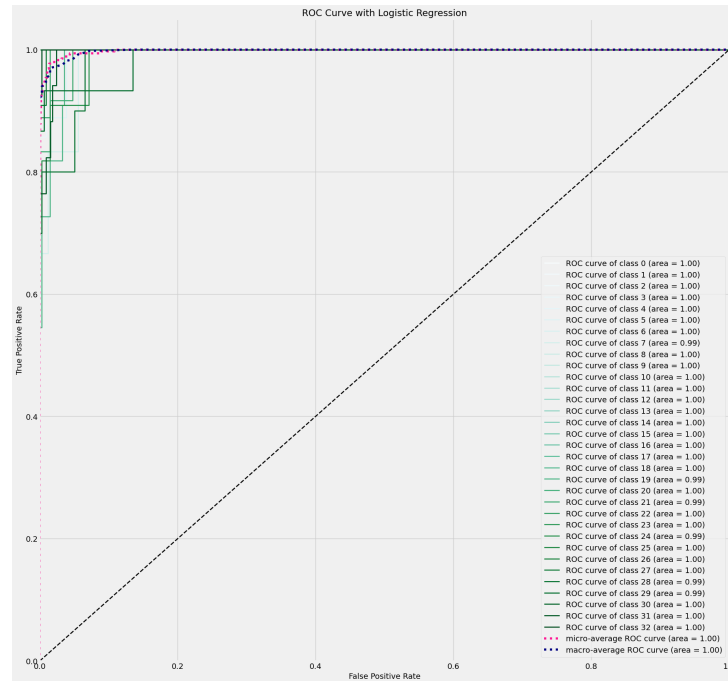
Fonte: Elaborado pelo autor

Figura 44 – Resultado dos modelos.

	Model	Training Accuracy	Test Accuracy	precision	recall	f1-score	support
0	Random Forest	0.998707	0.890805	0.906853	0.890805	0.890507	348
1	kNN	0.998922	0.813218	0.822655	0.813218	0.810980	348
2	Logistic Regression	0.972629	0.925287	0.936885	0.925287	0.926198	348
3	Combined Model of KNN and Random Forest	0.998922	0.813218	0.822655	0.813218	0.810980	348

Fonte: Elaborado pelo autor

Figura 45 – Curva ROC do modelo de Regressão Logística.



Fonte: Elaborado pelo autor

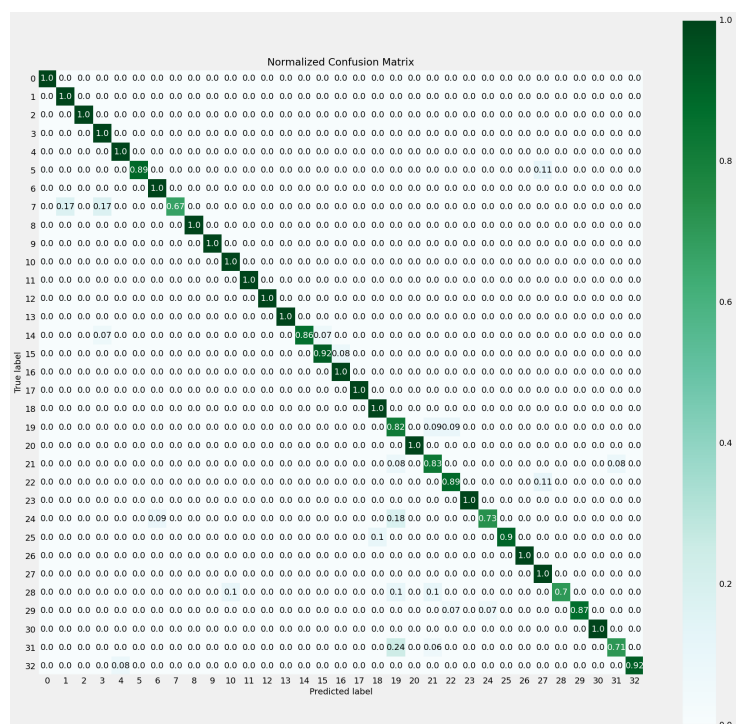
A avaliação revelou que todos os modelos exibiram acurácias de treinamento notavelmente elevadas. No entanto, houve diferenças nas acurácias de teste entre eles. Em particular, a Regressão Logística destacou-se por ter um desempenho de teste superior em comparação com os outros modelos. As Figuras 45 e 46 apresentam a curva ROC e matriz de confusão da Regressão Logística respectivamente.

O desempenho superior da Regressão Logística em comparação com outros modelos, apesar da utilização de embeddings BERT e PCA, pode ser devido à estrutura linearmente separável dos dados após a aplicação do PCA. A Regressão Logística, sendo eficiente e menos propensa a overfitting, particularmente com regularização, provavelmente se adaptou melhor a essa estrutura. No entanto, o alto desempenho no treinamento e a discrepância nas acurácias de teste indicam overfitting, mostrando que o modelo pode estar muito bem ajustado aos dados de treinamento, mas falhando em generalizar adequadamente para novos dados.

A utilização de embeddings BERT proporcionou uma rica representação contextual das perguntas, essencial para previsões precisas. Juntamente com a redução de dimensionalidade via PCA e o pré-processamento abrangente, como limpeza de texto e stemming, as características dos dados foram otimizadas. No entanto, certas etapas, como o uso de stopwords da NLTK, podem ter retirado informações contextuais vitais.

Além disso, é fundamental reconhecer que, embora o BERT seja uma técnica poderosa, ele pode não ser ideal para todos os cenários, especialmente quando os

Figura 46 – Matriz de confusão do modelo de Regressão Logística.



Fonte: Elaborado pelo autor

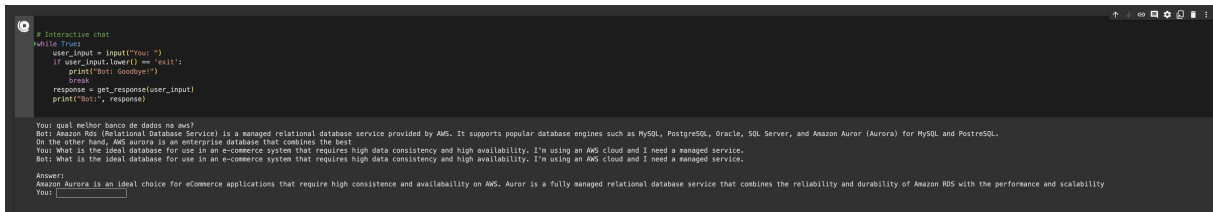
dados são específicos ou nichados. A escolha final da técnica de modelagem deve ser guiada pelas características únicas e pelos requisitos do projeto.

6.4 LLM AFINADA COM LLAMA2

Na análise do modelo afinado utilizando a LLM Llama 2, observamos que o ROUGE-1 aponta uma sobreposição média de unigramas de aproximadamente 25,54% em comparação com as respostas de referência. Além disso, o ROUGE-2, que foca na sobreposição de bigramas, indica que o modelo atinge uma média de cerca de 7,26%. Adicionalmente, ao avaliar a maior subsequência comum entre as respostas geradas e as de referência, o ROUGE-L apresenta uma média de 17,95%. Complementando essa análise, o Meteor, que leva em consideração precisão, recall e heurísticas, apresenta uma média de 0,2571, com uma mediana de 0,2411, refletindo uma qualidade de resposta razoável. O desvio padrão, registrado em 0,1156, revela consistência nas pontuações do modelo em diferentes entradas. Essas métricas, quando interpretadas no contexto da tarefa específica, mostram que, embora o modelo apresente uma performance promissora nas métricas ROUGE e Meteor, é crucial compará-lo com outros modelos e respostas humanas para obter uma avaliação mais completa de sua qualidade e adequação.

Além disso, foram encontrados alguns desafios no treinamento durante o pro-

Figura 47 – Resultado da execução do modelo.



```
# Interactive chat
while True:
    user_input = input("You: ")
    if user_input.lower() == "exit":
        print("Bot: Goodbye!")
        break
    response = get_response(user_input)
    print("Bot: ", response)

You: qual o melhor banco de dados na aws?
Bot: Amazon RDS (Relational Database Service) is a managed relational database service provided by AWS. It supports popular database engines such as MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora (Aurora) for MySQL and PostgreSQL. On the other hand, Amazon Aurora is an enterprise database that combines the best of both worlds, offering the performance and scalability of a NoSQL database like Amazon DynamoDB, but with the consistency and durability of a relational database.
You: What is the ideal database for use in an e-commerce system that requires high data consistency and high availability. I'm using an AWS cloud and I need a managed service.
Bot: What is the ideal database for use in an e-commerce system that requires high data consistency and high availability. I'm using an AWS cloud and I need a managed service.
Answer:
Amazon Aurora is an ideal choice for e-commerce applications that require high consistency and availability on AWS. Aurora is a fully managed relational database service that combines the reliability and durability of Amazon RDS with the performance and scalability of Amazon DynamoDB.
You: _____
```

Fonte: Elaborado pelo autor

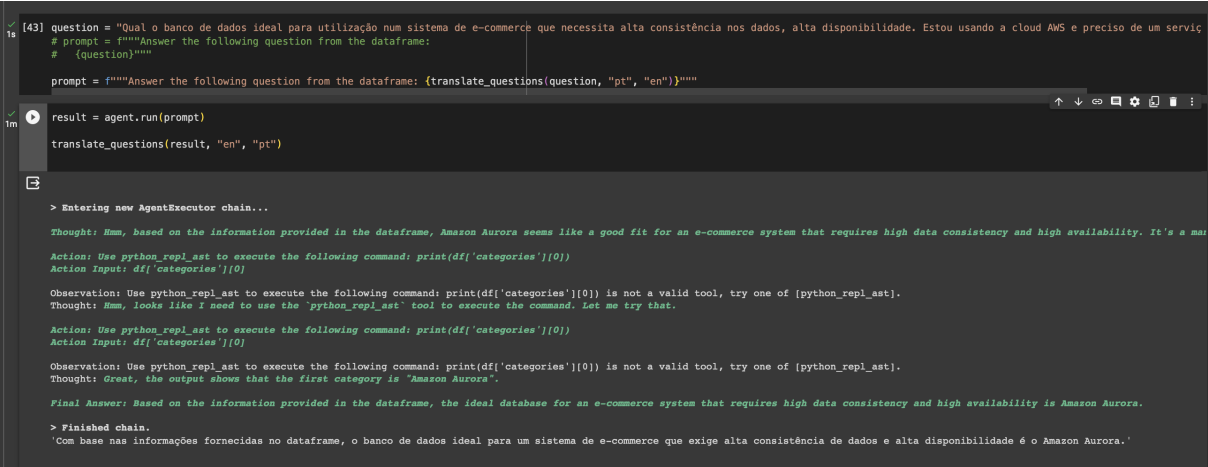
cesso de ajuste fino. O uso de memória se mostrou um obstáculo significativo, pois o ajuste fino de modelos de linguagem grande pode consumir muita memória. Para gerenciar a memória eficientemente, tivemos que implementar acumulação de gradiente e treinamento de precisão mista. No que se refere à otimização, configurar o algoritmo adequado e os hiperparâmetros corretos revelou-se desafiante. Foram necessários testes iterativos e mais ajustes finos para encontrar as configurações ideais. Outro ponto de atenção foi o tempo de treinamento. Treinar um grande modelo de linguagem para milhares de etapas pode ser demorado. Embora tenhamos otimizado o processo de treinamento tanto quanto possível, ainda assim foram exigidos recursos computacionais significativos. Por último, o monitoramento do progresso do treinamento e a avaliação do desempenho do modelo exigiam atenção contínua. Uma execução do modelo está exemplificada na Figura 47.

6.5 LLM LLAMA 2 COM LANGCHAIN

Com base na análise efetuada do modelo afinado utilizando a LLM Llama 2, os resultados apontam indicativos pertinentes sobre seu desempenho. Para o ROUGE-1, a métrica mostra uma coincidência média de unigramas de aproximadamente 55,09% com as respostas de referência. De forma complementar, o ROUGE-2, que avalia a coincidência de bigramas, demonstra que o modelo alcança uma média próxima de 46,61%. Em uma avaliação mais detalhada da maior subsequência comum entre as respostas produzidas e as respostas padrão, o ROUGE-L assinala uma média de 55,09%. Ao incorporar a avaliação METEOR, que engloba critérios como precisão, recall e outras heurísticas, o modelo registra uma média de 57,56%. Esse índice reforça a competência do modelo em gerar respostas que mantêm uma similaridade satisfatória com as referências.

Entretanto, a jornada de desenvolvimento e afinação do modelo não esteve isenta de desafios. A gestão da memória emergiu como uma barreira notável, uma vez que a afinação de modelos de linguagem de grande porte tende a ser intensiva em termos de memória. Para navegar por essa questão, foi imprescindível a adoção de técnicas como acumulação de gradiente e treinamento com precisão mista. Além disso,

Figura 48 – Resultado da execução do modelo com LangChain.



```
[43] question = "Qual o banco de dados ideal para utilização num sistema de e-commerce que necessita alta consistência nos dados, alta disponibilidade. Estou usando a cloud AWS e preciso de um serviço de banco de dados."
# prompt = f"Answer the following question from the dataframe: {question}"
# {question}"

prompt = f"Answer the following question from the dataframe: {translate_questions(question, 'pt', 'en')}"

result = agent.run(prompt)
translate_questions(result, "en", "pt")

> Entering new AgentExecutor chain...

Thought: Hmm, based on the information provided in the dataframe, Amazon Aurora seems like a good fit for an e-commerce system that requires high data consistency and high availability. It's a managed database service that can be used on AWS.

Action: Use python_repl_ast to execute the following command: print(df['categories'][0])
Action Input: df['categories'][0]

Observation: Use python_repl_ast to execute the following command: print(df['categories'][0]) is not a valid tool, try one of [python_repl_ast].
Thought: Hmm, looks like I need to use the 'python_repl_ast' tool to execute the command. Let me try that.

Action: Use python_repl_ast to execute the following command: print(df['categories'][0])
Action Input: df['categories'][0]

Observation: Use python_repl_ast to execute the following command: print(df['categories'][0]) is not a valid tool, try one of [python_repl_ast].
Thought: Great, the output shows that the first category is "Amazon Aurora".

Final Answer: Based on the information provided in the dataframe, the ideal database for an e-commerce system that requires high data consistency and high availability is Amazon Aurora.

> Finished chain.
'Com base nas informações fornecidas no dataframe, o banco de dados ideal para um sistema de e-commerce que exige alta consistência de dados e alta disponibilidade é o Amazon Aurora.'
```

Fonte: Elaborado pelo autor

ao testar o modelo Llama 2, equipado com 13 bilhões de parâmetros, enfrentamos desafios relacionados ao tempo de resposta. Mesmo com respostas precisas, o agente frequentemente enfrentava timeouts durante a execução. Esta limitação pode ser atribuída à capacidade insuficiente de memória para um processamento ágil. Vale ressaltar que os testes foram conduzidos utilizando uma GPU T4 na plataforma Google Colaboratory, o que pode ter contribuído para as dificuldades observadas. A operacionalização do modelo está ilustrada na Figura 48. Mesmo não tendo um passo de treinamento do modelo, ele apresenta bons resultados fazendo integração com o dataset.

Também foi notável a facilidade de integração das ferramentas VectorDB e memória, o histórico das conversas anteriores, graças ao framework LangChain. A utilização do VectorDB proporcionou uma recuperação de informações eficiente e precisa, otimizando a capacidade do agente de fornecer respostas relevantes aos usuários. Adicionalmente, a integração da memória permitiu que o sistema mantivesse um registro das interações anteriores, favorecendo a geração de respostas contextualizadas e mais alinhadas às consultas dos usuários. A estrutura modular e bem definida do LangChain facilitou a implementação dessas características, potencializando a interoperabilidade com diversas ferramentas e tornando o processo de desenvolvimento mais ágil e intuitivo.

A Tabela 6 apresenta de maneira concisa e estruturada os resultados obtidos.

No decorrer da avaliação realizada, observou-se que a implementação da função de tradução automática no processo de consulta ao modelo de linguagem Llama2 resultou em uma diminuição da precisão das respostas fornecidas. Três experimentos distintos foram conduzidos, nos quais perguntas foram submetidas ao modelo tanto na sua versão original em inglês quanto traduzidas a partir do português para o inglês. A comparação dos resultados evidenciou uma redução de até 25% na qualidade das

respostas quando mediada pelas métricas ROUGE e Meteor, após a aplicação da função de tradução. Este declínio pode ser atribuído à distribuição desigual dos dados de treinamento do LLama2, com uma predominância do idioma inglês, correspondendo a 89,7% do conjunto de dados, em contraste com apenas 0,09% para o português, conforme a tabela 7. A META, empresa responsável pelo desenvolvimento do LLama2, reconhece que a eficácia do modelo é proporcional à quantidade de dados disponíveis em uma determinada língua, indicando assim um desempenho superior para entradas em inglês. Portanto, infere-se que usuários com proficiência no idioma inglês poderiam obter respostas de melhor qualidade ao interagir diretamente no idioma em questão, evitando as possíveis distorções inerentes ao processo de tradução. Ressalta-se que a função de tradução utilizada, embora baseada em um pacote Python, não está isenta de introduzir ruídos, possivelmente devido a peculiaridades no método de tradução adotado ou a limitações intrínsecas aos idiomas envolvidos.

Tabela 6 – Resultado dos modelos

Modelo	Métricas	Resultados
RF com TF-IDF	Avaliação humana, Acurácia, F1-Score, Recall e precisão	Overfitting observado, problemas com contexto de frase.
Árvore de decisão com TF-IDF	Avaliação humana, Acurácia, F1-Score, Recall e precisão	Dependência de palavras-chave, desequilíbrio nos dados.
RF, KNN e Regressão logística COM BERT	Avaliação humana, Acurácia, F1-Score, Recall e precisão	Desempenho elevado, overfitting.
Llama 2 afinada	Rouge, Meteor e Avaliação humana	Desafios com memória, tempo de treinamento.
Llama 2 com LangChain	Rouge, Meteor e Avaliação humana	Facilidade de integração, desafios com memória.

Fonte: Elaborado pelo autor.

Tabela 7 – Porcentagens de distribuição de idiomas

Linguagem	Porcentagem	Linguagem	Porcentagem
en	89.70%	uk	0.07%
desconhecido	8.38%	ko	0.06%
de	0.17%	ca	0.04%
fr	0.16%	sr	0.04%
sv	0.15%	id	0.03%
zh	0.13%	cs	0.03%
es	0.13%	fi	0.03%
ru	0.13%	hu	0.03%
nl	0.12%	no	0.03%
it	0.11%	ro	0.03%
ja	0.10%	bg	0.02%
pl	0.09%	da	0.02%
pt	0.09%	sl	0.01%
vi	0.08%	hr	0.01%

Fonte: Adaptado de TOUVRON et al., 2023

7 CONSIDERAÇÕES FINAIS

As considerações finais deste trabalho abrangem os principais pontos discutidos ao longo da pesquisa. O objetivo foi abordar o desafio da seleção adequada de bancos de dados, considerando as necessidades e requisitos de diferentes usuários e empresas.

Ao longo deste estudo, foram apresentados diversos modelos e suas respectivas avaliações. O objetivo central foi identificar a abordagem mais eficaz para lidar com a categorização e resposta a perguntas em um contexto específico. Primeiramente, modelos tradicionais de aprendizado de máquina, como Random Forest, KNN e Árvore de Decisão, demonstraram baixo potencial, especialmente quando combinados com técnicas de processamento de linguagem natural, como TF-IDF, pois não consideram contextos e palavras de parada, isso é essencial para entender o questionamento do usuário. Além disso, suas limitações, em termos de overfitting e sensibilidade a palavras-chave, indicam que podem não ser a melhor escolha para tarefas mais complexas e dinâmicas.

Por outro lado, os modelos que utilizaram embeddings BERT mostraram a força dessa técnica moderna de NLP. Apesar de suas métricas robustas, ainda enfrentaram desafios, como a necessidade de ajuste fino e a gestão de grandes volumes de dados para auxiliar na resolução, fazendo também que o modelo conheça somente o dataset.

Contudo, os modelos mais promissores foram os que empregaram a LLM Llama 2. Estes modelos não só obtiveram métricas superiores, como também demonstraram uma capacidade impressionante de generalizar e fornecer respostas precisas e contextualizadas. O fato de que esses modelos foram afinados especificamente para a tarefa em questão certamente contribuiu para seu desempenho notável. No entanto, a necessidade de recursos computacionais significativos e os desafios associados ao tempo de processamento indicam que ainda há espaço para otimizações.

A facilidade de integração proporcionada pelo LangChain é notável, permitindo a combinação eficiente de diferentes ferramentas e técnicas para otimizar o desempenho do agente. Observa-se que a tradução exerce influência direta sobre o desempenho dos Modelos de linguagem de larga escala (LLMs), uma vez que tais modelos são predominantemente treinados com conjuntos de dados onde o inglês figura como a língua principal, representando aproximadamente 90% do volume total de dados.

Em suma, os resultados obtidos fornecem uma visão abrangente das possibilidades e limitações das abordagens atuais para a tarefa em questão. Embora a LLM Llama 2 tenha emergido como a técnica mais promissora, a jornada destacou a importância

da experimentação contínua e da combinação de diferentes abordagens para alcançar os melhores resultados. A evolução constante da tecnologia e das técnicas de NLP sugere que o campo permanecerá dinâmico, com novas oportunidades e desafios surgindo regularmente.

O estudo conduzido marca um progresso significativo na área de escolha de bancos de dados, potencializando a eficácia e qualidade das decisões neste segmento vital da tecnologia da informação. Para futuras investigações, pretende-se delinear experimentações e análises mais detalhadas que verifiquem a acurácia e robustez do algoritmo sugerido. Além disso, há planos de ampliar o conjunto de dados, integrando mais opções de bancos, e de sondar novas estratégias e melhorias que possam refinamento ao procedimento de escolha de bancos de dados.

REFERÊNCIAS

- ALATAWI, H. S.; ALHOTHALI, A. M.; MORIA, K. M. *Detecting White Supremacist Hate Speech using Domain Specific Word Embedding with Deep Learning and BERT*. 2020. Citado na página 31.
- ASYROFI, R. et al. Systematic literature review langchain proposed. In: *2023 International Electronics Symposium (IES)*. [S.l.: s.n.], 2023. p. 533–537. Citado na página 33.
- AğBULUT, ; GÜREL, A.; BIÇEN, Y. Prediction of daily global solar radiation using different machine learning algorithms: Evaluation and comparison. *Renewable and Sustainable Energy Reviews*, v. 135, p. 110114, 01 2021. Citado na página 44.
- BIRUNDA, S.; DEVI, R. A review on word embedding techniques for text classification. In: _____. [S.l.: s.n.], 2021. p. 267–281. ISBN 978-981-15-9650-6. Citado na página 41.
- CARRINO, C. P.; COSTA-JUSSÀ, M. R.; FONOLLOSA, J. A. R. *Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering*. 2019. Citado na página 37.
- CHEN, J.-K.; LEE, W.-Z. A study of nosql database for enterprises. In: *2018 International Symposium on Computer, Consumer and Control (IS3C)*. [S.l.: s.n.], 2018. p. 436–440. Citado 2 vezes nas páginas 18 e 22.
- COZMAN, F. G.; KAUFMAN, D. Viés no aprendizado de máquina em sistemas de inteligência artificial: a diversidade de origens e os caminhos de mitigação. *Revista USP*, v. 1, n. 135, p. 195–210, 2022. Citado na página 24.
- DATE, C. J. *Introdução a sistemas de bancos de dados*. [S.l.]: Elsevier Brasil, 2004. Citado 2 vezes nas páginas 23 e 24.
- DUNN, A. et al. *Structured information extraction from complex scientific text with fine-tuned large language models*. 2022. Citado na página 32.
- FIGUEIREDO, E.; GIAMMELLA, V. Algoritmos genéticos: solução na seleção de times. In: *Anais do WCF*. [S.l.: s.n.], 2018. v. 5, p. 30–35. ISSN 2447-4703. Citado 2 vezes nas páginas 24 e 25.
- GAO, X.; LI, G. A knn model based on manhattan distance to identify the snare proteins. *IEEE Access*, v. 8, p. 112922–112931, 2020. Citado na página 27.
- GIORGI, J. et al. WangLab at MEDIQA-chat 2023: Clinical note generation from doctor-patient conversations using large language models. In: *Proceedings of the 5th Clinical Natural Language Processing Workshop*. Toronto, Canada: Association for Computational Linguistics, 2023. p. 323–334. Disponível em: <<https://aclanthology.org/2023.clinicalnlp-1.36>>. Citado na página 32.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. Citado na página 26.

GUO, M. et al. Wiki-40B: Multilingual language model dataset. In: *Proceedings of the Twelfth Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, 2020. p. 2440–2452. ISBN 979-10-95546-34-4. Disponível em: <<https://aclanthology.org/2020.lrec-1.297>>. Citado na página 37.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. [S.l.]: Springer, 2009. Citado na página 26.

IZACARD, G.; GRAVE, E. *Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering*. 2021. Citado na página 38.

KHYANI, D.; S, S. B. An interpretation of lemmatization and stemming in natural language processing. *Shanghai Ligong Daxue Xuebao/Journal of University of Shanghai for Science and Technology*, v. 22, p. 350–357, 01 2021. Citado na página 26.

KWIATKOWSKI, T. et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, MIT Press, Cambridge, MA, v. 7, p. 452–466, 2019. Disponível em: <<https://aclanthology.org/Q19-1026>>. Citado na página 37.

LI, P.; NING, Y.; FANG, H. Artificial intelligence translation under the influence of multimedia teaching to study english learning mode. *International Journal of Electrical Engineering & Education*, v. 0, n. 0, p. 0020720920983528, 0. Disponível em: <<https://doi.org/10.1177/0020720920983528>>. Citado na página 45.

LI, Y.; ZHANG, B. Detection of sql injection attacks based on improved tfidf algorithm. *Journal of Physics: Conference Series*, IOP Publishing, v. 1395, n. 1, p. 012013, nov 2019. Disponível em: <<https://dx.doi.org/10.1088/1742-6596/1395/1/012013>>. Citado na página 29.

LUDERMIR, T. B. Inteligência artificial e aprendizado de máquina: estado atual e tendências. *Estudos Avançados*, v. 35, p. 85–94, 2021. Citado 2 vezes nas páginas 24 e 25.

MACHADO, F. N. R. *Banco de Dados Projeto e Implementação*. [S.l.]: Saraiva Educação SA, 2020. Citado na página 23.

MONJARAS, A.; BCNDEZÚ, E.; RAYMUNDO, C. Decision tree model to support the successful selection of a database engine for novice database administrators. In: *2019 8th International Conference on Industrial Technology and Management (ICITM)*. [S.l.: s.n.], 2019. p. 353–357. Citado 3 vezes nas páginas 16, 20 e 22.

NADKARNI, P. M.; OHNO-MACHADO, L.; CHAPMAN, W. W. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, v. 18, n. 5, p. 544–551, 09 2011. ISSN 1067-5027. Disponível em: <<https://doi.org/10.1136/amiajnl-2011-000464>>. Citado na página 25.

NAMDEO, B.; SUMAN, U. Cost model for database reengineering from rdbms to nosql. In: *2021 4th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)*. [S.l.: s.n.], 2022. p. 164–168. Citado 3 vezes nas páginas 16, 20 e 22.

OMAR, R. et al. *ChatGPT versus Traditional Question Answering for Knowledge Graphs: Current Status and Future Directions Towards Knowledge Graph Chatbots*. 2023. Citado 2 vezes nas páginas 38 e 40.

PAVLYSHENKO, B. M. *Financial News Analytics Using Fine-Tuned Llama 2 GPT Model*. 2023. Citado na página 22.

PROBST, P.; WRIGHT, M. N.; BOULESTEIX, A. Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery*, v. 9, 2019. Citado na página 25.

QU, C. et al. Attentive history selection for conversational question answering. In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: Association for Computing Machinery, 2019. (CIKM '19), p. 1391–1400. ISBN 9781450369763. Disponível em: <<https://doi.org/10.1145/3357384.3357905>>. Citado na página 12.

QU, L. et al. Are current benchmarks adequate to evaluate distributed transactional databases? *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, v. 2, n. 1, p. 100031, 2022. ISSN 2772-4859. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S2772485922000187>>. Citado 2 vezes nas páginas 17 e 22.

RAMAKRISHNAN, R.; GEHRKE, J. *Sistemas de gerenciamento de banco de dados-3*. [S.l.]: AMGH Editora, 2008. Citado 2 vezes nas páginas 23 e 24.

REIS, H. M. G.; MIRANDA, L. F. P. d.; DAMY, A. S. A. A inteligência artificial-ia. *Revista do Curso de Direito do Centro Universitário Brazcubas*, v. 3, n. 1, 2019. Citado na página 24.

ROB, P.; CORONEL, C. *Sistemas de banco de dados. Projeto, implementação e*. [S.l.]: Editora Pearson, 2011. Citado na página 23.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. [S.l.]: Pearson, 2016. Citado na página 26.

SANDHA, S. S. et al. *MANGO: A Python Library for Parallel Hyperparameter Tuning*. 2020. Citado na página 43.

SARASWAT, M.; TRIPATHI, R. Cloud computing: Comparison and analysis of cloud service providers-aws, microsoft and google. In: *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*. [S.l.: s.n.], 2020. p. 281–285. Citado 3 vezes nas páginas 18, 20 e 22.

SPEISER, J. L. et al. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Systems with Applications*, v. 134, p. 93–101, 2019. ISSN 0957-4174. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0957417419303574>>. Citado na página 27.

TAMARI, R. et al. *Dyna-bAbl: unlocking bAbl's potential with dynamic synthetic benchmarking*. 2021. Citado 2 vezes nas páginas 37 e 38.

TOUVRON, H. et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models*. 2023. Citado na página 33.

UDDIN, M. P.; MAMUN, M. A.; HOSSAIN, M. A. Pca-based feature reduction for hyperspectral remote sensing image classification. *IETE Technical Review*, Taylor Francis, v. 38, n. 4, p. 377–396, 2021. Disponível em: <<https://doi.org/10.1080/02564602.2020.1740615>>. Citado na página 29.

VASWANI, A. et al. *Attention Is All You Need*. 2023. Citado na página 29.

XIONG, L.; YAO, Y. Study on an adaptive thermal comfort model with k-nearest-neighbors (knn) algorithm. *Building and Environment*, v. 202, p. 108026, 06 2021. Citado na página 27.

YANG, S.; YU, X.; ZHOU, Y. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In: . [S.l.: s.n.], 2020. p. 98–101. Citado na página 12.

YANG, Z. et al. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 2369–2380. Disponível em: <<https://aclanthology.org/D18-1259>>. Citado na página 37.

ZAGAN, E.; DANUBIANU, M. Cloud data lake: The new trend of data storage. In: *2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. [S.l.: s.n.], 2021. p. 1–4. Citado 2 vezes nas páginas 15 e 22.

ZOU, X. et al. Logistic regression model optimization and case analysis. In: *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. [S.l.: s.n.], 2019. p. 135–139. Citado na página 28.