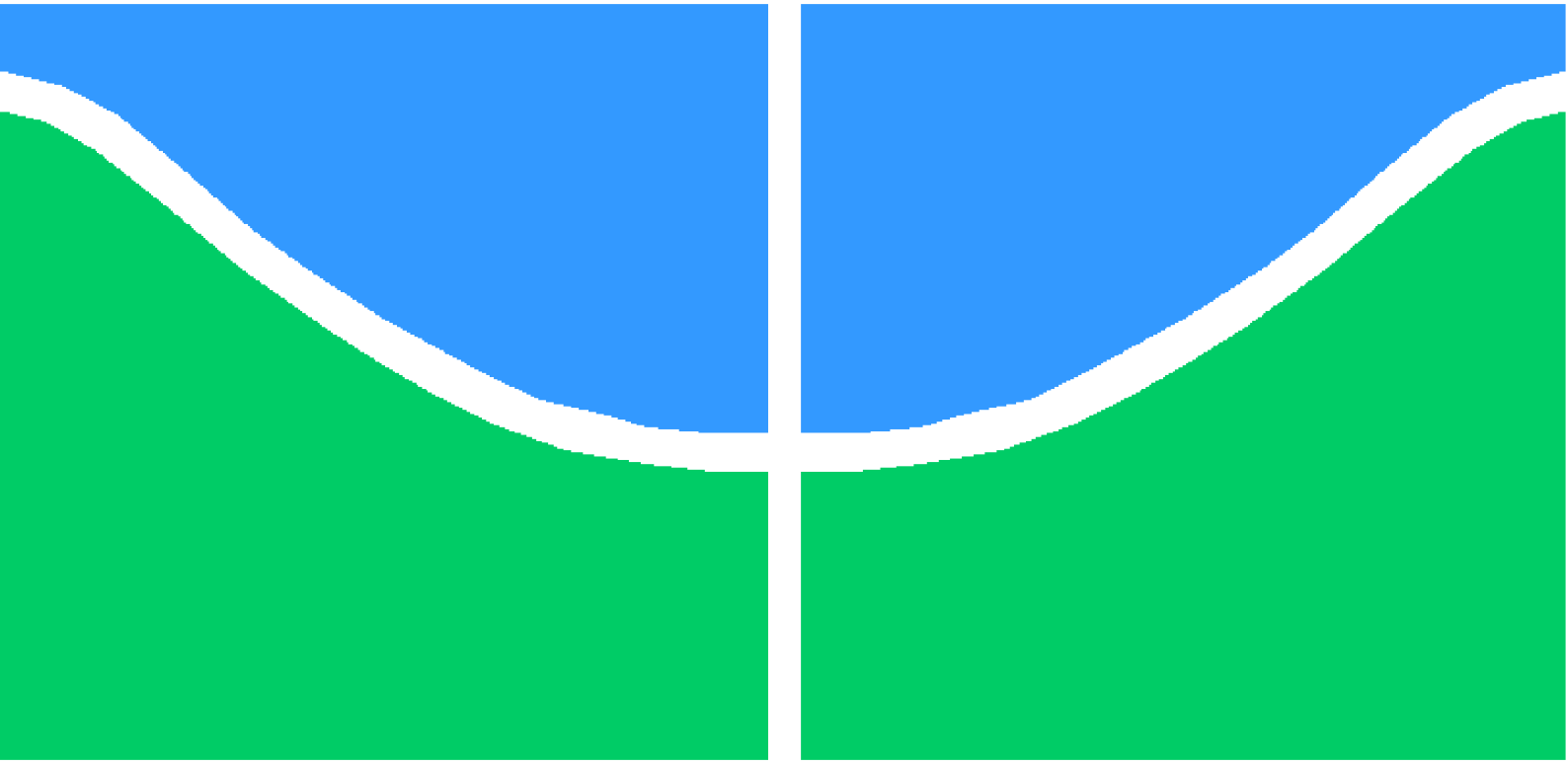


Universidade de Brasília - UnB

Faculdade de Ciências e Tecnologias em Engenharia - FCTE

Engenharia de Software

Alavancando modelo massivo de linguagem para um domínio do esporte por meio de RAG e *fine-tuning*

Autor: Guilherme de Morais Richter

Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF

2025

Guilherme de Morais Richter

Alavancando modelo massivo de linguagem para um domínio do esporte por meio de RAG e *fine-tuning*

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade de Ciências e Tecnologias em Engenharia - FCTE

Orientador: Professor Dr. Fabricio Ataides Braz

Brasília, DF

2025

Guilherme de Morais Richter

Alavancando modelo massivo de linguagem para um domínio do esporte por meio de RAG e *fine-tuning*/ Guilherme de Morais Richter. – Brasília, DF, 2025-
81 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Dr. Fabricio Ataides Braz

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade de Ciências e Tecnologias em Engenharia - FCTE , 2025.

1. LLM. 2. aprimoramento. I. Professor Dr. Fabricio Ataides Braz. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Alavancando modelo massivo de linguagem para um domínio do esporte por meio de RAG e *fine-tuning*

CDU 02:141:005.6

Guilherme de Morais Richter

Alavancando modelo massivo de linguagem para um domínio do esporte por meio de RAG e *fine-tuning*

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Professor Dr. Fabricio Ataides Braz
Orientador

Professor Dr. Nilton Correia da Silva
Convidado 1

Professor Dr. Henrique Marra Taira Menegaz
Convidado 2

Brasília, DF
2025

Este trabalho é dedicado aos meus familiares, e aos que não estão mais aqui, que me ensinaram sobre como o amor é capaz de transformar vidas.

"Confia teus negócios ao Senhor
e teus planos terão bom êxito."
(Bíblia Sagrada, Provérbios 16, 3)

Resumo

Este trabalho investigou o aprimoramento de Modelos Massivos de Linguagem (LLM, do inglês *Large Language Models*) para tarefas específicas de Processamento de Linguagem Natural (PLN) aplicadas ao domínio esportivo. Foram avaliadas técnicas como *Retrieval-Augmented Generation* (RAG) e ajuste fino supervisionado (*fine-tuning*), tendo como base dados estruturados obtidos da plataforma Sofascore relacionados ao futebol. Os resultados mostraram que a combinação dessas técnicas melhora significativamente o desempenho dos modelos, proporcionando respostas mais precisas, contextualizadas e semanticamente coerentes em comparação ao uso isolado dos modelos pré-treinados. Em particular, o ajuste fino com simulação de recuperação semântica (RAFT) demonstrou maior eficiência em tarefas analíticas complexas, como avaliação de desempenho individual, coletivo e resposta a perguntas sobre tabelas hierárquicas.

Palavras-chave: Modelos Massivos de Linguagem, processamento de linguagem natural, Geração Aumentada de Recuperação, ajuste fino, aprendizado supervisionado, futebol, esporte, análise de desempenho.

Abstract

This work investigated the enhancement of Large Language Models (LLMs) for specific Natural Language Processing (NLP) tasks in the sports domain. Techniques such as Retrieval-Augmented Generation (RAG) and supervised fine-tuning were evaluated using structured football data obtained from the Sofascore platform. The results demonstrated that combining these techniques significantly improves model performance, enabling more accurate, contextualized, and semantically coherent responses compared to the isolated use of pre-trained models. In particular, fine-tuning with simulated semantic retrieval (RAFT) proved to be highly effective in complex analytical tasks, such as evaluating individual and team performance, and answering hierarchical table-based questions.

Key-words: Large Language Models, Natural Language Processing, Retrieval-Augmented Generation, fine-tuning, supervised learning, football, sport, performance analysis.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

NLP	EN-US: Natural Language Processing / PT-BR: Processamento de Linguagem Natural
CSV	EN-US: Comma-Separated Values / PT-BR: Valores Separados por Vírgula
LLM	EN-US: Large Language Model / PT-BR: Modelos grandes de linguagem
RAG	EN-US: Retrieval-augmented generation / PT-BR: Geração aumentada por recuperação
IR	EN-US: Information Retrieval / PT-BR: Recuperação de Informação
IA	EN-US: Artificial Intelligence / PT-BR: Inteligência Artificial
ML	EN-US: Machine Learning / PT-BR: Aprendizado de Máquina
HTTP	EN-US: HyperText Transfer Protocol / PT-BR: Protocolo de Transferência de Hipertexto
API	EN-US: Application Programming Interface / PT-BR: Interface de Programação de Aplicações
JSON	EN-US: JavaScript Object Notation / PT-BR: Notação de Objetos JavaScript
QA	EN-US: Question Answering / PT-BR: Resposta a Perguntas
RAM	EN-US: Random Access Memory / PT-BR: Memória de Acesso Aleatório
CPU	EN-US: Central Processing Unit / PT-BR: Unidade Central de Processamento
GPU	EN-US: Graphics Processing Unit / PT-BR: Unidade de Processamento Gráfico
ROUGE	EN-US: Recall-Oriented Understudy for Gisting Evaluation
BLEU	EN-US: Bilingual Evaluation Understudy
METEOR	EN-US: Metric for Evaluation of Translation with Explicit Ordering / PT-BR: Métrica para Avaliação de Tradução com Ordenação Explícita

BERT	EN-US: Bidirectional Encoder Representations from Transformers / PT-BR: Representações Bidirecionais de Codificadores a partir de Transformadores
BLEURT	EN-US: Bilingual Evaluation Understudy for Rewarding Transformers
CoT	EN-US: Chain-of-Thought
RAFT	EN-US: Retrieval Augmented Fine-tuning

Sumário

Parte I

Pré-Texto

1 Introdução

Os modelos LLM (*Large Language Models*) são ferramentas poderosas de inteligência artificial que podem entender e produzir textos em vários idiomas e áreas do conhecimento. Esses modelos são importantes para a sociedade atual, pois podem facilitar e melhorar diversas atividades que dependem da linguagem, como comunicação, educação, informação e entretenimento.

Além de compreender e gerar texto, esses sistemas apresentam versatilidade em diferentes tarefas de processamento de linguagem natural, tais como traduzir conteúdos, resumir documentos, gerar novos textos, responder a perguntas e engajar em diálogos com usuários. Essas aplicações encontram-se em diversos setores, incluindo saúde, finanças, turismo e jornalismo.

Ainda assim, nem sempre um modelo LLM genérico é suficiente ou adequado para um domínio específico. Por essa razão, é comum aprimorar um LLM puro com dados ou conhecimento especializado, adaptando-o às necessidades particulares de cada caso.

O refinamento de modelos LLM para contextos específicos pode trazer benefícios como maior precisão, relevância e qualidade das saídas, além de otimizar o uso de recursos computacionais e mitigar riscos éticos ou sociais, como vieses e disseminação de desinformação.

Dessa forma, os LLMs configuram-se como uma tecnologia inovadora e de grande relevância na sociedade atual, oferecendo múltiplas possibilidades de uso da linguagem em diferentes contextos. Neste trabalho, serão aprofundados os conceitos, características, métodos e exemplos de aplicação de modelos LLM, com enfoque nas técnicas mais recentes e eficazes para adaptar esses modelos a domínios especializados, bem como na validação de suas vantagens, limitações e implicações.

1.1 Objetivos

Este trabalho investiga o impacto de técnicas de aprimoramento de modelos LLM em tarefas de NLP voltadas ao domínio de futebol, por meio de Retrieval-Augmented Generation (RAG) e fine-tuning. Os objetivos específicos são:

1. Integrar RAG a um LLM de base e medir seu desempenho em tarefas de extração e geração de informação no domínio de futebol.

2. Aplicar fine-tuning ao mesmo LLM utilizando um conjunto de textos sobre futebol e reavaliar o sistema RAG.

3. Comparar quantitativa e qualitativamente os resultados obtidos antes e depois do fine-tuning.

4. Discutir vantagens e limitações do uso combinado de RAG e fine-tuning no contexto estudado.

1.2 Estrutura do trabalho

- Pré-texto
  - Capítulo 1, introdução - Este capítulo apresenta o conteúdo inicial da pesquisa que é descrita no projeto, estabelecento o contexto atual da tecnologia utilizada e ressalta sua importância para a sociedade, também descrevendo os objetivos do estudo que serão tratados.
  - Capítulo 2, referencial teórico - Este capítulo descreve os trabalhos correlatos, os conceitos e definições importantes, os quais foram usados como base para o entendimento teórico e prático deste trabalho.
- Texto e Pós-texto
  - Capítulo 3, materiais e métodos - Este capítulo apresenta a metodologia de desenvolvimento do trabalho, onde são abordadas as estratégias e recursos necessários ao desenvolvimento do trabalho. Ele explica a fonte de dados, os métodos de coleta de dados e demais procedimentos e técnicas adotados para alcançar a conclusão dos objetivos.
  - Capítulo 4, resultados - Este capítulo apresenta os resultados alcançados a partir dos métodos adotados no capítulo anterior.
  - Capítulo 5, considerações finais - Este capítulo apresenta as conclusões obtidas a partir dos resultados alcançados neste estudo, destacando a eficácia das técnicas aplicadas e seu impacto nas tarefas de processamento de linguagem natural no domínio esportivo.

2 Referencial teórico

2.1 Trabalhos correlatos

Os *Large Language Models* (LLMs) são cada vez mais usados para diversas tarefas de *Natural Language Processing* (NLP), incluindo geração de conteúdo e como *chatbots*. No entanto, os LLMs ainda enfrentam desafios para acessar e manipular precisamente o conhecimento, especialmente em tarefas específicas de domínio, onde podem produzir respostas alucinadas ou prejudiciais. Além disso, fornecer proveniência para suas decisões e atualizar seu conhecimento do mundo são problemas de pesquisa em aberto.

Uma forma de melhorar o desempenho dos LLMs em tarefas intensivas em conhecimento é integrá-los com sistemas de *Information Retrieval* (IR), que permitem anexar e consultar bases de conhecimento não-paramétricas, como a Wikipédia. Essa abordagem é conhecida como *Retrieval-Augmented Generation* (RAG), que será abordada neste trabalho.

Alguns trabalhos já desenvolveram soluções parecidas com o que será explorada neste trabalho. Em um exemplo deles, o RAG foi utilizado no domínio da medicina, onde os LLMs foram usados para gerar sumários de grandes dados textuais não estruturados. Esse trabalho foi realizado por (MANATHUNGA; ILLANGASEKARA, 2023), que discutiram as vantagens do RAG para facilitar o aprendizado e a avaliação dos estudantes de medicina.

Outro exemplo de trabalho, (LEWIS et al., 2021) utilizaram o modelo BART como base e compararam modelos híbridos, integrados com RAG a modelos somente pré-treinados, chegando à conclusão de que os modelos RAG superaram os modelos puros, gerando linguagem mais específica, diversa e factual.

Em um outro trabalho, (GHODRATNAMA; ZAKERSHAHRAK, 2023) discutiram maneiras de integrar LLMs a sistemas de recuperação de informação, considerando o RAG como uma metodologia bastante promissora, destacando também o potencial de economia de custos computacionais com isso.

Além desses estudos, vale destacar também a pesquisa apresentada por (DODGSON et al., 2023), que se dedicou a explorar maneiras de aprimorar o desempenho dos LLMs por meio de *fine-tuning*, com RAG e *soft-prompting*. O estudo envolveu a avaliação de uma versão original do GPT 3.5, uma versão ajustada e a mesma versão não modificada com acesso a um banco de dados RAG vetorizado. Os resultados revelaram que o modelo ajustado superou o desempenho do GPT 3.5 Turbo, enquanto a abordagem RAG superou ambos.

Tais trabalhos são então utilizados como base de como iniciar o processo de desenvolvimento deste projeto, com sua leitura fornecendo o conhecimento de diversos conceitos, técnicas e ferramentas a serem aqui utilizadas.

2.2 Levantamento de ferramentas

2.2.1 Python

Python é uma linguagem de programação de alto nível, de uso geral e gratuita, que é muito popular (CHOLLET, 2017) entre os desenvolvedores de software, especialmente nas áreas de inteligência artificial (IA) e aprendizado de máquina (ML).

O Python é utilizado para aplicações de IA por vários motivos, como a simplicidade: é uma linguagem fácil de aprender (RASCHKA; PATTERSON; NOLET, 2020), com uma sintaxe clara e objetiva, que permite que o desenvolvedor foque na solução de problemas, em vez de perder tempo com detalhes técnicos.

Além disso, possui uma grande quantidade de bibliotecas e frameworks que facilitam a implementação de algoritmos complexos e o processamento de grandes volumes de dados, que são essenciais para IA e ML. Algumas das bibliotecas mais usadas são TensorFlow, PyTorch, scikit-learn e pandas.

Também tem uma comunidade ativa e crescente, que contribui com o desenvolvimento e a manutenção da linguagem, além de oferecer suporte e recursos para os iniciantes e os profissionais da área. Existem vários fóruns, blogs, cursos e livros sobre Python e IA, que ajudam a disseminar o conhecimento e a resolver dúvidas.

Por último, a versatilidade: o Python pode ser aplicado em diversas áreas e fins, como desenvolvimento web, análise de dados, automação de tarefas, computação gráfica, desenvolvimento de jogos, criptomoedas e muito mais. Isso permite que o desenvolvedor tenha mais flexibilidade e criatividade para criar soluções inovadoras e personalizadas.

2.2.2 Colab

O Colab é uma plataforma online que possibilita a utilização da capacidade computacional do Google para a elaboração de projetos de inteligência artificial, especialmente aqueles que envolvem grandes modelos de linguagem (LLM). Optar pelo uso do Colab na construção de LLM oferece diversos benefícios (CARNEIRO et al., 2018), incluindo a gratuidade, facilidade de uso, colaboração eficiente e fomento à inovação.

Essa ferramenta disponibiliza uma versão gratuita que permite a utilização das GPUs e TPUs do Google sem qualquer custo, sendo especialmente útil para treinar e testar LLM que demandam consideráveis recursos de hardware. O Colab emprega notebooks

interativos, simplificando a escrita e execução de código em Python, além de viabilizar a integração com bibliotecas e frameworks renomados para LLM, tais como Hugging Face, Transformers e Diffusers.

Adicionalmente, a plataforma permite o compartilhamento e edição colaborativa de notebooks, promovendo o trabalho em equipe e a troca de ideias e experiências. E também oferece a funcionalidade de salvar e acessar notebooks no Google Drive, GitHub ou em outros serviços de armazenamento em nuvem, o que garante ao usuário um backup automático com facilidade de recuperação em caso de perda.

2.2.3 Hugging Face

A plataforma Hugging Face se destaca como um ambiente líder para a aplicação e aprimoramento de LLMs. Esta plataforma oferece diversas vantagens para aqueles que buscam utilizar e realizar ajustes finos em modelos de LLM.

Dentro desse contexto, existe a biblioteca de código aberto conhecida como Transformers (WOLF et al., 2020), que abrange diversos modelos de LLM pré-treinados em vários idiomas, englobando uma variedade de tarefas de NLP, como tradução, resumo, classificação e geração de texto. Além disso, a plataforma disponibiliza uma interface intuitiva chamada Pipeline, simplificando o uso desses modelos sem a necessidade de extenso código ou configuração. Neste projeto, a Pipeline do Hugging Face é utilizada para instanciar o modelo Llama, que é disponibilizado de maneira gratuita pela Meta AI para estudantes. A partir disso, é possível utilizar o modelo junto ao Autotrain, que é uma ferramenta que permite treiná-lo por meio de *fine-tuning*.

A utilização das bibliotecas e recursos do Hugging Face para empregar e aprimorar modelos de LLM representa uma maneira eficiente de explorar o estado-da-arte em NLP de forma descomplicada, rápida e transparente.

2.3 Levantamento de bibliotecas

2.3.1 Requests

A biblioteca Requests do Python é uma ferramenta amplamente utilizada para requisições HTTP e interação com APIs de forma descomplicada. Com uma sintaxe clara e intuitiva, ela permite operações como GET, POST, PUT, DELETE, facilitando o envio de parâmetros, cabeçalhos, cookies, arquivos e dados JSON. Sua capacidade de lidar com erros, redirecionamentos, timeouts e encodings a torna robusta, sendo por isso utilizada para a captação do dado utilizado como referência neste trabalho.

Além do acesso simplificado a conteúdo, status e metadados da resposta, a biblioteca oferece recursos como criação de sessões, autenticação, configuração de proxies e

gestão de certificados, ampliando sua versatilidade. Valiosa para extração de dados de APIs, a Requests simplifica a interação, eliminando a necessidade de lidar com detalhes de baixo nível da comunicação HTTP.

No contexto deste projeto, a biblioteca Requests é utilizada para interceptar as requisições get da API do Sofascore, que ambas retornam respostas em formato JSON.

2.3.2 Pandas

O Pandas, uma biblioteca Python amplamente reconhecida, é essencial para manipulação, análise e visualização de diversos tipos de dados, como tabelas, séries temporais e textos. Ela facilita a leitura e escrita em formatos como CSV e JSON, proporcionando integração em diferentes ambientes.

Suas estruturas de dados, como DataFrame e Series, oferecem organização eficiente e acesso simplificado aos dados, enquanto suas operações matemáticas, estatísticas e de texto simplificam ações como filtrar, agrupar, ordenar e transformar dados. O Pandas também suporta a exploração visual de dados, incluindo gráficos e histogramas. Além disso, desempenha um papel crucial na extração de dados brutos em formato JSON, convertendo-os para formato tabular. Destaca-se não apenas no tratamento, mas também na limpeza, padronização, enriquecimento e validação de dados, contribuindo para a qualidade e confiabilidade dos mesmos.

Em vista disso, essa biblioteca é utilizada neste projeto para a transformação dos dados de JSON para formato CSV, em sendo suas features usadas para estruturar os dados em tabelas e colunas divididas por temas, além de utilizá-la para criar o dataset de fine-tuning no formato esperado pelos modelos LLM utilizados.

2.3.3 Langchain

O Langchain é um framework para desenvolver aplicações usando LLMs. Ele permite que as aplicações sejam contextuais, capazes de raciocinar e gerar linguagem natural. Uma das aplicações possíveis com o Langchain é o Retrieval-Augmented Generation, explorado por este projeto, que consiste em responder perguntas usando informações recuperadas de uma fonte externa, como um documento, um site ou um banco de dados.

As bibliotecas do Langchain facilitam (GHODRATNAMA; ZAKERSHAHRAK, 2023) a integração de um LLM com um componente de recuperação, como o ChromaDB ou o FAISS, utilizado neste projeto. Elas também permitem configurar os parâmetros de entrada e saída do LLM, como o prompt, a temperatura, o comprimento máximo e o formato da resposta.

Utilizar o Langchain e suas bibliotecas para realizar Retrieval QA em modelos de LLM tem como vantagens aproveitar o poder dos LLMs para gerar respostas precisas,

coerentes e relevantes para as perguntas dos usuários. Não somente, com ele é possível personalizar o LLM para diferentes tarefas e domínios, usando técnicas como fine-tuning, prompt engineering ou few-shot learning. E principalmente, simplificar o desenvolvimento e o gerenciamento de aplicações de Retrieval QA, usando uma plataforma unificada e integrada com diversas ferramentas e serviços.

2.3.4 ChromaDB

O ChromaDB é uma ferramenta de banco de dados vetorial open-source desenvolvida para atender as demandas de aplicações modernas em inteligência artificial, com foco especial em tarefas de Retrieval Augmented Generation (RAG) com modelos LLM. Essa ferramenta permite o armazenamento, indexação e recuperação eficiente de embeddings de alta dimensionalidade, facilitando a extração de informações relevantes a partir de grandes volumes de dados. Para realizar a comparação entre vetores, o ChromaDB utiliza métricas como a similaridade de cosseno e a distância Euclidiana, o que garante a precisão na identificação dos elementos mais similares.

Projetado para oferecer escalabilidade e integração simplificada com pipelines de aprendizado de máquina, o ChromaDB disponibiliza uma API intuitiva para operações de inserção, consulta e atualização dos dados vetoriais. Sua arquitetura robusta permite manipular conjuntos de dados que ultrapassam a memória RAM disponível, mantendo um desempenho consistente, o que o torna uma escolha vantajosa para sistemas de Retrieval QA em ambientes com LLM (JOHNSON; DOUZE; JéGOU, 2017).

2.4 Modelos estudados e conceitos relevantes

2.4.1 Estado da arte – A Inteligência Artificial aplicada ao Processamento de Linguagem Natural

O campo da Inteligência Artificial (IA) (CHOLLET, 2017) abrange diversas aplicações que envolvem o processamento de dados e a tomada de decisões. Algumas dessas aplicações são o reconhecimento facial, que utiliza algoritmos de IA para analisar e reconhecer características faciais e permitir o acesso a dispositivos e serviços, como smartphones, e o carro autônomo, que representa uma inovação na indústria de transporte, baseada em algoritmos de IA capazes de controlar o veículo sem intervenção humana.

A IA também está presente em diversas atividades cotidianas, como a recomendação de conteúdo em plataformas de streaming, que utiliza algoritmos de IA para analisar as preferências, o histórico e os dados de comportamento do usuário e gerar recomendações personalizadas, auxiliando o usuário a encontrar conteúdo mais adequado ao seu

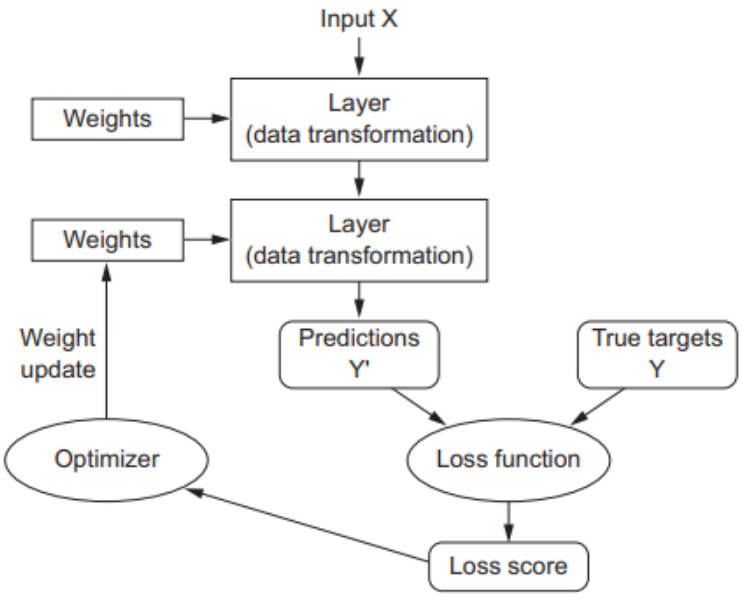
perfil. A IA também demonstrou competência em tarefas orientadas por dados, como a análise de sentimentos, a detecção de fraudes, entre outras.

No entanto, historicamente, a IA apresentava dificuldades em relação à compreensão de contexto, à resposta a perguntas abertas e à geração de respostas humanas em diálogos. Avanços recentes em modelos de IA baseados em linguagem resultaram em uma mudança paradigmática. Esses modelos, conhecidos como *Large Language Models* (LLMs) (ZHAO et al., 2023), possuem a habilidade de processar a linguagem de forma similar aos seres humanos.

Entretanto, hierarquicamente, para chegar aos LLM, a IA, que é um termo mais geral, é dividida em subcampos: o primeiro deles é o *Machine Learning* (ML), subcampo da Inteligência Artificial que possibilita que modelos aprendam padrões a partir de dados, sem necessidade de instruções explícitas. A partir do ML, há mais uma vertente, que é o *Deep Learning* (CHOLLET, 2017). Enquanto o machine learning tradicional pode depender de algoritmos específicos para realizar tarefas, o deep learning busca simular o funcionamento do cérebro humano ao empregar redes neurais artificiais compostas por camadas interconectadas de neurônios virtuais.

Nesse contexto, as redes neurais profundas possuem múltiplas camadas, permitindo a extração de características complexas e abstratas a partir dos dados de entrada. Durante o treinamento, o sistema ajusta automaticamente os pesos das conexões entre os neurônios para otimizar o desempenho na tarefa desejada, sendo essa abordagem visualizada na figura 1.

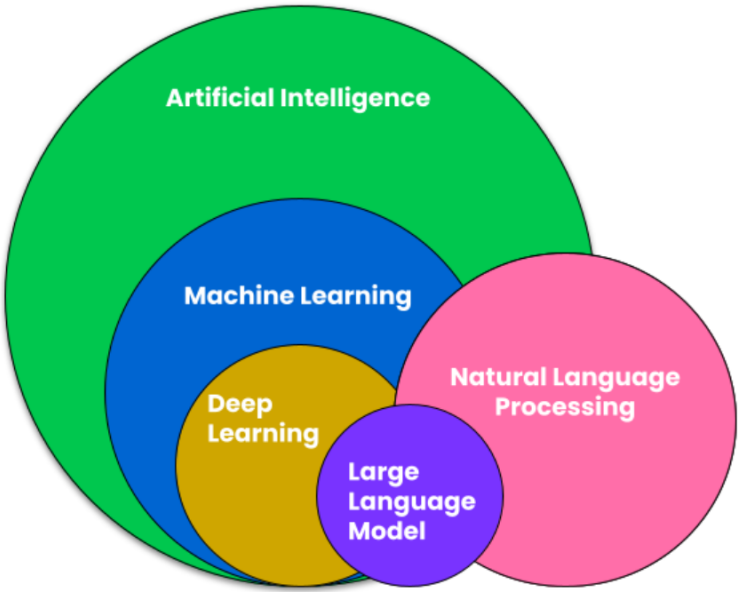
Figura 1 – Processo de treinamento de uma rede neural.



Fonte: (CHOLLET, 2017)

Não sendo parte integrante da IA, mas utilizando dela, há o *Natural Language Processing* (NLP), que faz uso de técnicas de aprendizado de máquina, entre outras, para compreender e processar a linguagem humana por meio de sistemas computacionais. Os LLMs, empregam técnicas de *Deep Learning* para realizar diversas tarefas relacionadas ao *Natural Language Processing*, tais como classificação de texto, sumarização, geração de texto, entre outras. A relação entre as vertentes pode ser ilustrada pela Figura 2.

Figura 2 – Relação entre os conceitos de IA e NLP.



Fonte: Autor

Chamados de "grandes" devido à exigência de uma grande quantidade de dados de treinamento e recursos para operarem eficientemente, os LLMs se destacam por sua eficiência no processamento e análise de dados linguísticos. Esses modelos estabeleceram novos padrões em diversas tarefas de NLP, superando seus predecessores e abrindo novas possibilidades no campo da IA. O termo "modelos" refere-se à capacidade de aprender padrões complexos por meio de dados. No caso específico dos LLMs, esses dados consistem em textos provenientes da internet.

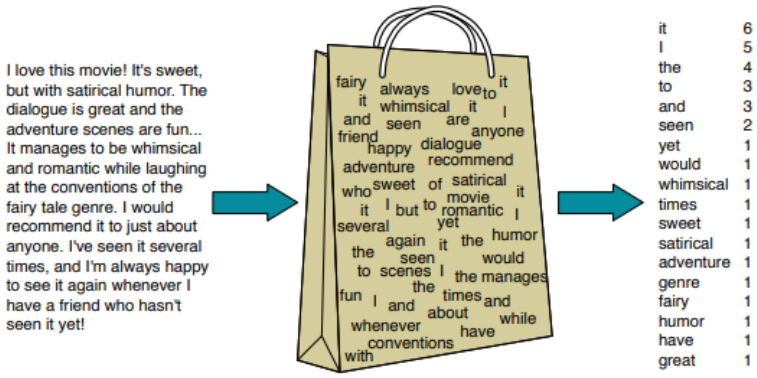
A série GPT da OpenAI se destaca como uma das famílias de LLMs mais conceituadas, notabilizando-se pela popularidade de sua versão gratuita, o GPT 3.5. Embora existam outros geradores de linguagem no mercado, novos modelos são constantemente introduzidos. Um exemplo é o Llama 3.1, desenvolvido pela Meta e pela Microsoft, que é um modelo de linguagem de última geração, disponível gratuitamente para uso comercial e de pesquisa, escolhido para ser utilizado neste projeto. Os LLMs apresentam a capacidade de realizar uma ampla gama de tarefas, abrangendo desde a análise de sentimentos, identificação de temas, tradução de texto ou fala, até mesmo a geração de código.

2.4.2 NLP e o pré-processamento de texto

O Processamento de Linguagem Natural (NLP) (JURAFSKY; MARTIN, 2022) introduz técnicas vitais para transformar dados de texto em uma forma compreensível por máquinas, especialmente para o objetivo deste trabalho, que são LLMs. O processo abrange diversas etapas de pré-processamento textual, como tokenização, remoção de palavras irrelevantes e lematização, cuja ordem pode variar dependendo da tarefa específica.

A tokenização segmenta o texto em pequenos pedaços, como palavras, que aqui são chamadas de tokens, incluindo pontuações. A remoção de palavras irrelevantes, conhecida como *stop word removal*, elimina termos frequentes que não acrescentam muito significado ao texto, enquanto a lematização reduz palavras ao que no português é conhecido como radical, eliminando derivações, mantendo apenas o significado constante.

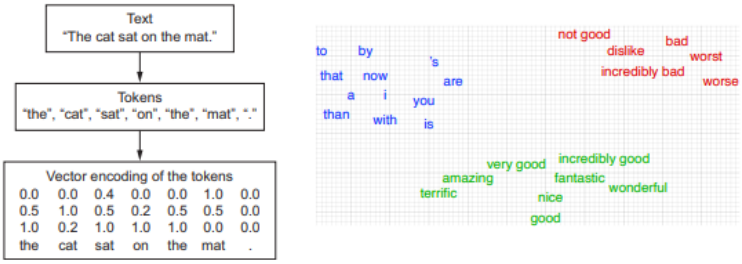
Figura 3 – Exemplo de tokenização e *bag of words*.



Fonte: (JURAFSKY; MARTIN, 2022)

A representação subsequente do texto, que facilita a compreensão por computadores, é realizada por meio de técnicas como *bag of words* e *word embeddings*. Apesar de o bag of words converter o texto em uma matriz de contagem de palavras, suas limitações estão na dificuldade de capturar significado e contexto. Nesse contexto, os *word embeddings* surgem como uma solução, superando essas limitações ao atribuir representações numéricas que refletem relações semânticas entre as palavras.

Figura 4 – Exemplo de tokenização e conversão para *embeddings*.



Fonte: (CHOLLET, 2017)

2.4.3 Pré-treinamento: Como os LLMs são treinados

Os LLMs são treinados usando a técnica de pré-treinamento generativo, que envolve fornecer ao modelo um conjunto de dados de tokens de texto e treiná-lo para prever esses tokens. Duas abordagens comuns de pré-treinamento generativo são a previsão da próxima palavra (*next word prediction*) e a modelagem de linguagem mascarada (*masked language modeling*) (JURAFSKY; MARTIN, 2022).

A previsão da próxima palavra é uma técnica de aprendizado supervisionado, onde o modelo é treinado para prever a próxima palavra em uma frase, capturando as dependências entre palavras no contexto mais amplo. Durante o treinamento, o modelo é apresentado a pares de exemplos de entrada e saída. Além disso, o modelo pode ser treinado com uma grande quantidade de dados de texto semelhantes para melhorar sua capacidade de previsão.

Outra abordagem é a modelagem de linguagem mascarada, que treina o modelo para prever uma palavra mascarada em uma frase ocultada seletivamente. Isso é realizado durante o treinamento com textos originais e textos mascarados como entrada, e o objetivo é prever corretamente a palavra ausente. Essas técnicas capacitam o modelo a aprender representações contextuais de palavras.

2.4.4 Transformers e mecanismos de atenção

Os *transformers* são uma parte crucial do pré-treinamento e aprimoram as técnicas existentes. O artigo *Attention Is All You Need* (VASWANI et al., 2023) foi fundamental na mudança do paradigma do modelamento de linguagem. A arquitetura do *transformer* destaca a importância de relacionamentos de longo alcance entre palavras para gerar texto preciso e coerente, composta por quatro componentes essenciais: pré-processamento, codificação posicional, codificadores e decodificadores.

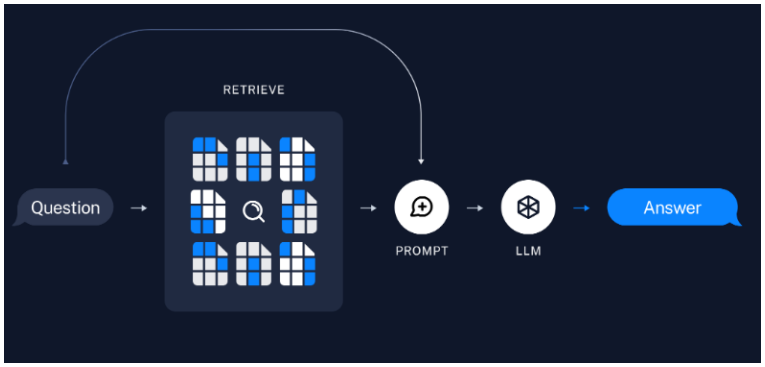
O pré-processamento converte o texto em números, incorpora referências de posição e, em seguida, os codificadores e decodificadores trabalham em conjunto para prever palavras subsequentes, contribuindo para a conclusão da sentença de entrada. O *transformer* trata o texto através de etapas como a divisão em *tokens*, representação numérica por meio de *embeddings* de palavras e codificação posicional. Os codificadores, com mecanismo de atenção e rede neural, dirigem a atenção para palavras específicas e seus relacionamentos, enquanto os *transformers*, ao contrário de modelos sequenciais tradicionais, processam várias partes do texto simultaneamente, acelerando o entendimento e a geração de texto.

2.4.5 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) (LEWIS et al., 2021) é uma técnica avançada de inteligência artificial que combina recuperação de informação com geração de texto, permitindo que modelos de IA busquem informações relevantes de uma fonte de conhecimento e as incorporem no texto gerado. Essa técnica é importante para melhorar um modelo LLM para um contexto específico, pois garante que o modelo tenha acesso às informações mais atuais, confiáveis e pertinentes ao domínio da consulta do usuário. Além disso, RAG pode reduzir a necessidade de treinar continuamente o modelo em novos dados e atualizar seus parâmetros à medida que as circunstâncias evoluem.

RAG utiliza uma base de dados vetorial (*vector database*) para armazenar e recuperar as informações de uma fonte de conhecimento, como uma coleção de documentos ou uma enciclopédia. Uma base de dados vetorial é um tipo de banco de dados que usa vetores de alta dimensão para representar e indexar os dados, facilitando a busca por similaridade semântica. Essa abordagem permite que RAG encontre as informações mais relevantes para a consulta do usuário, mesmo que não haja uma correspondência exata entre as palavras. Essa esteira pode ser visualizada na figura 5.

Figura 5 – Esquema de funcionamento de um RAG.



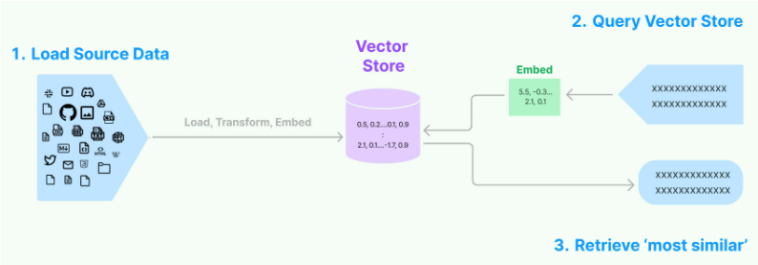
Fonte: (LANGCHAIN, 2023)

RAG pode ser mais eficiente que um *fine-tuning* para adaptar um modelo LLM a um contexto, pois evita o problema de esquecimento catastrófico, que ocorre quando o modelo perde a capacidade de generalizar para tarefas anteriores depois de ser treinado em um novo domínio. Além disso, RAG pode reduzir os custos computacionais e financeiros de executar modelos LLM em um ambiente empresarial, pois não requer o armazenamento e o processamento de grandes quantidades de dados rotulados.

2.4.6 Vector Database

Uma base de dados vetorial (*vector database*), também conhecido como *vector store*, é um tipo de banco de dados que usa vetores de alta dimensão para representar e indexar os dados, facilitando a busca por similaridade semântica.

Figura 6 – Como um *vector store* armazena as informações buscadas por um RAG.



Fonte: (LANGCHAIN, 2023)

Uma base de dados vetorial é importante para ser utilizada em um RAG para melhorar um modelo LLM para um contexto, pois permite que o modelo encontre as informações mais relevantes para a consulta do usuário, mesmo que não haja uma correspondência exata entre as palavras, conforme é possível observar na figura 6. Por exemplo, se o usuário perguntar sobre "o maior rio do mundo", uma base de dados vetorial pode recuperar o documento que fala sobre o "rio Amazonas", mesmo que não contenha a palavra "maior" no texto. Isso torna o modelo LLM mais preciso e capaz de gerar textos mais informativos e coerentes.

2.4.7 LLaMA 3.1

O modelo **LLaMA 3.1 (Large Language Model Meta AI)** é parte da terceira geração de modelos de linguagem desenvolvidos pela Meta AI (TOUVRON et al., 2024). Esta versão representa um avanço significativo em relação ao LLaMA 2, trazendo melhorias em arquitetura, desempenho, estabilidade e escalabilidade para aplicações em linguagem natural.

A série LLaMA 3 foi lançada inicialmente em abril de 2024 com modelos de 8B e 70B parâmetros, ambos treinados com conjuntos de dados de alta qualidade contendo mais de 15 trilhões de tokens. Em julho de 2024, a versão **3.1** foi publicada com uma base refinada e novos pesos treinados, corrigindo limitações de estabilidade e trazendo ganhos de performance expressivos. Essa versão tornou-se amplamente adotada na comunidade de código aberto, destacando-se tanto por sua qualidade quanto pela sua flexibilidade de uso.

O LLaMA 3.1 apresenta um **context window estendido de 8K tokens** — uma melhoria importante em relação aos 2K tokens do LLaMA 2. Essa ampliação permite que o modelo compreenda instruções mais longas, mantenha coerência ao longo de contextos extensos e seja mais adequado para tarefas complexas como geração aumentada por recuperação (RAG), análise contextual e respostas condicionadas a grandes volumes de texto.

Além disso, o LLaMA 3.1 introduz melhorias internas como:

- Arquitetura otimizada com melhores mecanismos de atenção e normalização;
- Inicialização de parâmetros aprimorada, aumentando a estabilidade do fine-tuning;
- Curadoria mais rigorosa do corpus de treinamento, com foco em diversidade e qualidade;
- Compatibilidade direta com ferramentas de open-source como Hugging Face, Ollama e frameworks como Unsloth.

Outro aspecto central da proposta do LLaMA 3.1 é a sua **licença permissiva e gratuita para uso acadêmico e comercial**, incentivando pesquisas reprodutíveis e aplicações práticas em diversas áreas. O modelo está disponível nos formatos base e chat, podendo ser integrado em fluxos de inferência customizados com ou sem ajuste fino (*fine-tuning*).

Considerando sua robustez, capacidade contextual e ampla adoção, o LLaMA 3.1 foi a escolha natural para este projeto. Ele proporciona uma base poderosa para o treinamento supervisionado no domínio futebolístico, equilibrando performance, custo computacional e acessibilidade.

2.4.8 Por que realizar Fine-Tuning de LLMs

Nem todos precisam treinar um LLM do zero, pois modelos pré-treinados por *big techs* podem ser ajustados para tarefas específicas. O *fine-tuning* (DODGSON et al., 2023) é uma abordagem eficaz para superar desafios associados à escala e complexidade desses modelos. LLMs, como o Llama 3.1 e o GPT, são pré-treinados em enormes quantidades de texto, sendo capazes de realizar uma ampla variedade de tarefas de Processamento de Linguagem Natural (NLP). No entanto, quando há a necessidade de aplicá-los em contextos ou domínios especializados, como a geração de resumos médicos ou jurídicos, o fine-tuning permite que o modelo seja ajustado para responder de forma mais precisa a essas demandas específicas (TOUVRON et al., 2023).

Esse processo consiste em utilizar um modelo que já foi treinado em um grande conjunto de dados gerais e adaptá-lo a um novo conjunto de dados mais específico. Dessa forma, o modelo consegue aproveitar o conhecimento adquirido no treinamento inicial e aprender os padrões específicos da nova tarefa (FACE, n.d.). Construir LLMs do zero requer poderosos computadores e infraestrutura especializada, resultando em custos computacionais significativos. O treinamento eficiente é crucial devido ao tempo extenso envolvido, podendo chegar a semanas ou meses. Além disso, a qualidade dos dados de treinamento é vital para que o modelo aprenda as nuances da linguagem.

O ajuste fino enfrenta esses desafios ao adaptar modelos pré-treinados para tarefas específicas, aproveitando a estrutura geral da linguagem aprendida durante o pré-

treinamento. Essa abordagem é mais eficiente em termos de recursos, pois pode ser realizada com uma única CPU e GPU, exigindo menos dados e tempo em comparação com o pré-treinamento completo. O Simple Fine-Tuning (SFT), uma biblioteca oferecida pela Hugging Face, automatiza tarefas como o carregamento de dados e a definição de hiperparâmetros, facilitando o processo para desenvolvedores com menos experiência (FACE, n.d.).

O *fine-tuning* não requer o retreinamento completo do modelo desde o início, sendo eficiente em termos de tempo e recursos computacionais. Ele ajusta os parâmetros do modelo de maneira incremental, tornando-o mais adequado para tarefas específicas sem sacrificar sua capacidade de generalização.

2.4.9 Como avaliar se o modelo foi realmente melhorado?

A geração de texto é uma tarefa desafiadora que envolve produzir textos coerentes, fluentes e informativos a partir de dados estruturados ou não estruturados. Um exemplo de aplicação da geração de texto é a resposta a perguntas, que consiste em gerar uma resposta natural para uma pergunta formulada por um usuário. Porém, para entender se essa resposta é realmente aceitável como correta, é necessário avaliar.

E para avaliar a qualidade das respostas geradas por este sistema, existem algumas métricas criadas pela academia, das quais algumas serão aqui neste trabalho utilizadas: ROUGE, BLEU, BLEURT e METEOR. Essas métricas no geral utilizam o conceito de n-gramas.

Um n-grama é uma sequência de n palavras consecutivas em um texto. Por exemplo, se n = 2, temos um bigrama, que é um par de palavras. Se n = 3, temos um trigrama, que é um trio de palavras. Os n-gramas são usados para modelar a linguagem e analisar o conteúdo dos textos. Eles são usados nas métricas mencionadas, para medir a sobreposição de n-gramas entre a resposta do sistema e a referência humana.

Neste texto, são explicadas como cada uma dessas métricas funciona e porque elas são adequadas para medir o desempenho das respostas de um modelo:

- ROUGE (LIN, 2004) é uma métrica baseada na sobreposição de n-gramas entre a resposta do sistema e a referência humana. Quanto mais n-gramas em comum, maior é o *score*. ROUGE pode medir diferentes níveis de n-gramas, como unigramas, bigramas, trigramas, etc. Também pode considerar variações como stemming, sinônimos e subsequências comuns. ROUGE é útil para medir a precisão e a relevância do conteúdo gerado. Por exemplo, se a referência é “O Brasil venceu a Argentina por 3 a 0” e a resposta do sistema é “O Brasil ganhou da Argentina por três a zero”, o ROUGE-1 (unigrama) seria 1.0, pois todos os unigramas são iguais ou equivalentes.

Já o ROUGE-2 (bigrama) seria 0.8, pois há 4 bigramas em comum de um total de 5.

- BLEU (SAADANY; ORăSAN, 2021) é outra métrica baseada na sobreposição de n-gramas, mas com uma diferença importante: ela penaliza as respostas que são muito longas ou muito curtas em relação à referência. Para isso, ela usa um fator de correção chamado brevidade, que é calculado pela razão entre o comprimento da resposta e o comprimento da referência. BLEU também combina diferentes níveis de n-gramas usando uma média geométrica ponderada. BLEU é mais adequada para medir a fluência e a concisão do texto gerado. Por exemplo, se a referência é "O Brasil venceu a Argentina por 3 a 0" e a resposta do sistema é "O Brasil derrotou a Argentina com três gols e não sofreu nenhum", o BLEU-4 (até 4-gramas) seria 0.41, pois há pouca sobreposição de n-gramas e a resposta é mais longa que a referência.

- BLEURT (SELLAM; DAS; PARIKH, 2020) é uma métrica criada com base no BERT, um modelo de linguagem pré-treinado em grandes quantidades de texto. BLEURT usa o BERT para extrair representações semânticas das respostas e das referências, e depois aplica uma camada de regressão para calcular um *score* que indica o quanto elas são similares. BLEURT é treinada em dados de avaliação humana, o que permite capturar nuances semânticas que as métricas baseadas em n-gramas podem ignorar. BLEURT é uma métrica robusta e geral, que pode se adaptar a diferentes domínios e tarefas de geração de texto. Por exemplo, se a referência é "O Brasil venceu a Argentina por 3 a 0" e a resposta do sistema é "A seleção brasileira de futebol derrotou a rival argentina por três gols de diferença", o BLEURT seria 0.86, pois há uma alta similaridade semântica entre as duas frases.

- METEOR (SAADANY; ORăSAN, 2021) é uma métrica que combina a precisão e o *recall* dos unigramas alinhados entre a resposta e a referência, usando uma média harmônica ponderada. Além disso, ela usa um fator de penalização que leva em conta o número e a ordem dos unigramas alinhados. METEOR também usa recursos como stemming, sinônimos e correspondência exata para encontrar os unigramas alinhados. METEOR é uma métrica que busca uma alta correlação com as avaliações humanas, tanto no nível de sentença quanto no nível de corpus. Por exemplo, se a referência é "O Brasil venceu a Argentina por 3 a 0" e a resposta do sistema é "Por 3 a 0, o Brasil venceu a Argentina", o METEOR seria 0.94, pois há um alto *recall* e uma baixa penalização pela inversão da ordem.

Essas são quatro métricas que podem ser usadas para avaliar se o modelo LLM que será melhorado por RAG e fine-tuning teve seu desempenho realmente melhorado, comparando os *scores* obtidos antes e depois das melhorias. No entanto, é importante notar que nenhuma métrica é perfeita ou definitiva, e que cada uma tem suas vantagens

e limitações. Por isso, é recomendável usar mais de uma métrica, e sempre verificar os resultados com avaliações humanas, sempre que possível.

2.4.10 Avaliação de Desempenho Utilizando GPT-4o

Avaliar a eficácia de um modelo após o *fine-tuning* é uma etapa crucial. Embora métricas automáticas como ROUGE, BLEU e BLEURT sejam úteis para comparar n-gramas, elas frequentemente não capturam nuances semânticas e a naturalidade do texto. Por isso, avaliadores baseados em LLMs, como o GPT-4o, têm sido adotados para complementar essas métricas.

Os modelos LLM, como é o caso do GPT-4o, podem comparar diretamente uma saída gerada com a referência, julgando coerência, fluidez e precisão contextual (LIU et al., 2023). No entanto, é importante estar atento às limitações que são intrínsecas a esse tipo de uso do modelo: a sensibilidade à formulação do prompt e a falta de transparência em seus critérios internos, o que pode afetar reprodutibilidade e introduzir vieses.

Portanto, uma boa estratégia é utilizar uma avaliação híbrida, combinando métricas automáticas para análises em larga escala com avaliações pontuais via modelos como GPT-4o, a fim de obter uma visão equilibrada do desempenho do modelo após o ajuste fino.

Parte II

Texto e Pós Texto

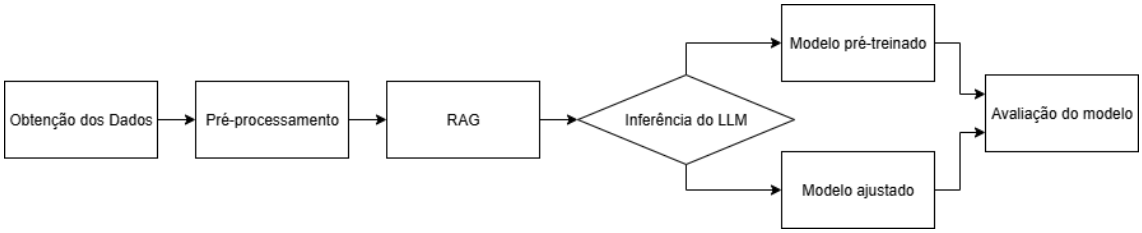
3 Materiais e métodos

3.1 Metodologia do trabalho

A metodologia deste trabalho consiste em aplicar técnicas de aprendizado profundo para processar e gerar textos em linguagem natural. O objetivo é obter um modelo capaz de responder a perguntas complexas e específicas sobre um determinado domínio. Para isso, serão realizados os seguintes passos, ilustrados na Figura 7:

- **Obtenção dos dados**: Os dados utilizados para treinar e avaliar o modelo serão coletados e selecionados de acordo com o tema e o nível de dificuldade das perguntas que se deseja responder.
- **Pré-processamento** dos dados obtidos: Os dados coletados serão pré-processados para adequá-los ao formato e ao tamanho esperados pelo modelo.
- **Uso do modelo LLM**: O modelo utilizado neste trabalho é um *Large Language Model* (LLM), que é um tipo de modelo de aprendizado profundo que usa uma arquitetura *transformer* para processar e entender a linguagem natural.
  - Melhoria do modelo por **RAG**: Para melhorar a qualidade e a confiabilidade das respostas geradas pelo modelo LLM sobre o domínio selecionado, será utilizada a técnica *Retrieval-Augmented Generation* (RAG).
  - Melhoria do modelo por ***fine-tuning***: Além do RAG, outra técnica utilizada para aprimorar o modelo LLM será o *fine-tuning*.
- **Avaliação do modelo**: Para avaliar o modelo LLM melhorado pelas técnicas de RAG e *fine-tuning*, serão utilizadas métricas próprias de desempenho e avaliação relevantes para o modelo LLM experimentado, validando se os passos anteriores trouxeram melhorias para o modelo na solução do problema.

Figura 7 – Metodologia do trabalho.



Fonte: Autor

3.2 Obtenção dos dados de futebol por meio do Sofascore

O domínio de aplicação definido é o futebol — esporte de grande relevância cultural e econômica —, cujo objetivo é gerar textos analíticos sobre partidas, estatísticas e avaliações de jogadores por meio de técnicas de aprimoramento de LLM (Retrieval-Augmented Generation e fine-tuning). Para tanto, é necessário dispor de uma base de dados rica, atualizada e estruturada especificamente para esse esporte.

O Sofascore é uma plataforma online que oferece cobertura completa de competições futebolísticas em nível global. Por meio de sua API REST, fornecem-se, em formato JSON, informações como:

- Resultados de partidas: placares, tempo de jogo e fases de competições;
- Escalações e substituições: lista de jogadores em campo, alterações táticas e momentos de entrada/saída;
- Eventos de jogo: gols, assistências, cartões e demais acontecimentos cronometrados;
- Métricas de desempenho individual: número de passes, finalizações, desarmes, distância percorrida e estatísticas avançadas;
- Avaliações de jogadores: notas e scores agregados por especialistas e algoritmos.

A escolha pelo Sofascore baseou-se nos seguintes critérios:

- Disponibilidade de dados estruturados via API, sem necessidade de scraping de HTML, simplificando a coleta e reduzindo riscos de inconformidade com os termos de uso.
- Atualização em tempo real ou quase real, garantindo que os modelos de linguagem sejam treinados e avaliados com informações recentes.
- Cobertura de ligas profissionais nacionais e internacionais, conferindo diversidade e profundidade ao conjunto de textos jornalísticos e estatísticos.

A extração dos dados foi realizada por meio da interceptação das requisições HTTP efetuadas pela interface web do Sofascore, utilizando as ferramentas de desenvolvedor do navegador. Identificaram-se os endpoints relevantes e as chamadas foram reproduzidas diretamente em scripts automatizados, permitindo o download eficiente e replicável de grandes volumes de JSON.

Assim, foram criados scripts que são responsáveis por interagir com a API do Sofascore para coletar dados relacionados a **estatísticas de jogadores**, **estatísticas de equipes** e **tabelas de classificação** de torneios de futebol.

3.2.1 Obtenção de dados de jogadores

O script `get_players` realiza a obtenção de dados de jogadores. Ele utiliza as bibliotecas pandas e requests para obter detalhes sobre jogadores de equipes específicas. Com base nas partidas existentes (confronto entre times, chamados aqui de *events*), ele itera sobre essas partidas e coleta dados dos jogadores usando a API do Sofascore, por meio do endpoint ***lineups***.

Como é possível notar no JSON a seguir, o dado retornado é um dicionário com informações do jogador e suas estatísticas durante aquela partida.


```
{
  "confirmed": true,
  "home": {
    "players": [
        {
        "player": {
            "name": "Gabriel Barbosa",
            "...": "..."
        }
        "position": "M",
        "...": "..."
        "statistics": {
            "totalPass": 16,
            "accuratePass": 15,
            "totalLongBalls": 1,
            "accurateLongBalls": 1,
            "goals": 2,
            "...": "..."
        }
        },
        "...": "..."
    }
}
```


A obtenção desses dados ao iterar por diversos *events* (representam as partidas) e seus respectivos *lineups* (representam os jogadores presentes na partida) permite a listagem de uma grande quantidade de dados estatísticos de diversos jogadores durante diversos eventos.

3.2.2 Obtenção de dados de equipes

O script `get_teams` realiza a obtenção de dados de equipes. Ele foca em obter estatísticas específicas da equipes durante as partidas. Também com base nas partidas existentes, ele itera sobre essas partidas e coleta dados dos jogadores usando a API do Sofascore, por meio do endpoint ***events***.

No JSON a seguir, nota-se que os valores estatísticos da partida são retornados como home, cujo valor representa o time da casa, e away, cujo valor representa o time visitante durante aquela partida.

```
{
  "statistics": [
    {
        "...": "...",
        "groups": [
        {
            "groupName": "Match overview",
            "statisticsItems": [
            {
                "name": "Ball possession",
                "home": "48%",
                "away": "52%",
                "...": "..."
            },
            {
                "name": "Expected goals",
                "home": "2.87",
                "away": "0.34",
                "...": "..."
            },
        }
    }
}
```

A obtenção desses dados ao iterar por diversos *events* (representam as partidas) e seus respectivos *lineups* (representam os jogadores presentes na partida) permite a listagem de uma grande quantidade de dados estatísticos desses jogadores durante esses eventos.

3.2.3 Obtenção de dados de tabelas de classificação

O script get_tables realiza a obtenção de dados de tabelas de classificação. Ele também utiliza as bibliotecas requests e pandas para obter dados de tabelas de classificação de campeonatos. Ele percorre os torneios e as suas temporadas, obtendo informações das tabelas usando a API do Sofascore.

No JSON a seguir, percebe-se que o dado retornado contém a identificação da equipe presente naquele torneio, seguido dos dados relativo à sua campanha no presente momento.


```
{
  "standings": [
    {
        "tournament": {
            "name": "Brasileirão Betano",
            "...": "..."
        }
        "rows": [
            {
            "team": {
                "name": "Palmeiras",
                "...": "..."
                },
            "...": "..."
            "position": 1,
            "matches": 38,
            "wins": 20,
            "scoresFor": 64,
            "scoresAgainst": 33,
            "id": 834894,
            "losses": 8,
            "draws": 10,
            "points": 70,
            "scoreDiffFormatted": "+31"
            },
            "...": "..."
        ]
    }
}
```


A inclusão das tabelas de classificação no conjunto de dados é interessante pela necessidade de explorar estruturas relacionais complexas próprias do futebol: cada tabela

reúne atributos como posição, pontos, saldo de gols, número de vitórias, empates e derrotas, cuja interpretação conjunta exige a compreensão de comparações e hierarquias entre equipes.

Além disso, a conversão desse formato tabular em narrativas claras — por exemplo, "a equipe X soma 70 pontos em 38 jogos, com saldo de +31" — requer um mapeamento preciso entre cabeçalhos e valores, de modo a evitar qualquer perda ou distorção de informação.

Essa abordagem, ao recriar em um cenário real o desafio de gerar descrições fiéis a dados tabulares, é uma área de interesse bastante relevante dentro do contexto de modelos generativos.

Isso porque os modelos de linguagem pré-treinados em texto natural apresentam dificuldades para interpretar dados semi-estruturados em formato de tabela, pois seu pré-treinamento não contempla noções explícitas de relacionamentos entre células e hierarquias de campos (ZHANG et al., 2024).

Dessa forma, a adoção das tabelas de classificação não apenas enriquece o dataset com informações essenciais ao domínio do futebol, mas também recria, em um cenário real, o desafio de gerar descrições fiéis a dados tabulares — essa questão também buscou ser resolvida pelo TableLlama, ao qual este trabalho se inspira, como descrito nos Trabalhos Correlatos.

3.3 Pré-processamento

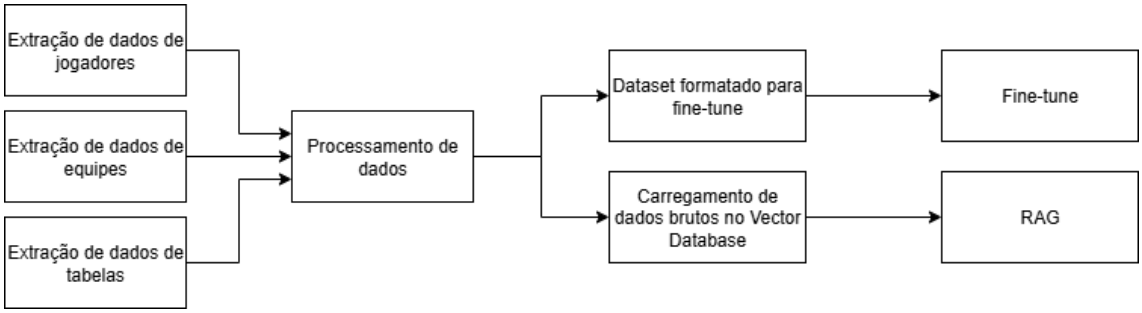
Após a extração dos dados ter sido feita, é o momento então de pré-processar os dados para que eles estejam completamente prontos para serem ingeridos como entrada de dado do modelo LLM.

Nesta fase, o dado extraído deve ser transformado para satisfazer o *input* necessário para tanto o ser introduzido na Vector Database e ser utilizado como fonte de recuperação de informação para o RAG, quanto também para ser dado de treinamento do *fine-tuning* do modelo.

Quanto à necessidade de transformação para RAG, os dados de saída dos scripts da fase de obtenção de dados são convertidos em arquivos de texto. A partir disso, nenhuma transformação adicional é necessária, já que a próxima etapa do RAG já é a conversão do conteúdo desses arquivos em *embeddings*, e após isso, o respectivo armazenamento desses *embeddings* na *Vector Database*, para ser utilizado pelo modelo.

Porém, para o processo de fine-tuning é necessário transformar os dados no formato que o modelo espera para treinamento. A figura 8 ilustra o *pipeline* de processamento necessário para cada tarefa.

Figura 8 – Pipeline de processamento de dados.



Fonte: Autor

3.3.1 Formato de *prompt* utilizado

Foi adotado o prompt do **Alpaca**, um estilo de *prompt* estruturado utilizado para ajustar modelos de linguagem com base em instruções. Esse formato foi popularizado pelo projeto Stanford Alpaca, que demonstrou como é possível ajustar modelos de linguagem de código aberto (caso do Llama 3.1 8B) com instruções geradas de forma sintética a um custo relativamente baixo, mantendo boa qualidade de respostas (TAORI et al., 2023).

O formato Alpaca segue uma estrutura com três campos principais: `instruction`, `input` (opcional) e `output`. O campo `instruction` define claramente a tarefa que o modelo deve realizar; o campo `input` fornece um contexto adicional se necessário; e o campo `output` representa a resposta esperada do modelo. Em código, o prompt se organiza da maneira a seguir.

```
alpaca_prompt = """Below is an instruction that describes a task, paired with
↪  an input that provides further context. Write a response that appropriately
↪  completes the request.

### Instruction:
{}

### Input:
{}

### Response:
{}"""
```

Dessa maneira, o prompt é montado com base nos dados obtidos de equipes, jogadores e tabelas de campeonatos. Para cada tarefa, são selecionadas colunas que contêm as informações relevantes para estarem presentes no *input*. A instrução (`instruction`) define o tipo de análise ou resposta que se espera do modelo, de acordo com a tarefa desejada, enquanto o campo `output` representa a resposta-alvo esperada.

Com base nessa estrutura, os prompts são gerados de forma automática a partir de regras específicas associadas à tarefa. Na Tabela 1, apresenta-se um exemplo de dado de treinamento construído no formato Alpaca. A instrução define uma tarefa de classificação do desempenho do time, o input contém dados objetivos da partida (estatísticas coletadas da API), e o output é a resposta-alvo, que neste caso é uma das três categorias possíveis: "bom", "mediano" ou "ruim".

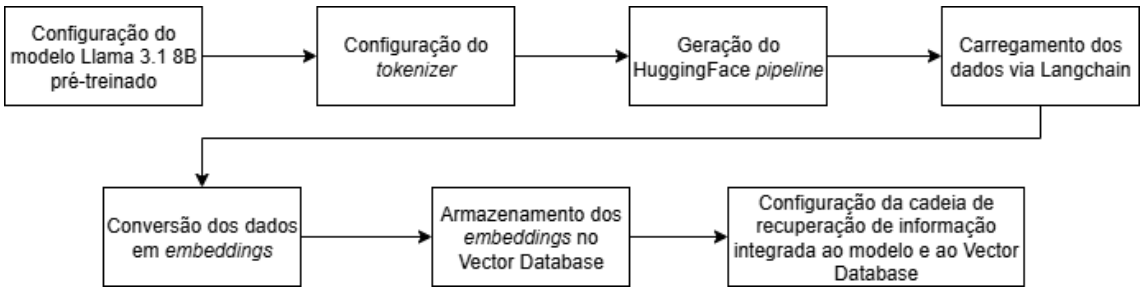
Tabela 1 – Exemplo de prompt no formato Alpaca, para a tarefa de classificação de desempenho do time

Instruction	**Input**	**Output**
Com base nos dados apresentados, como você avalia o desempenho do time Manchester City na partida Manchester City vs Wolverhampton pelo campeonato Premier League, na temporada 2025? Sua resposta deve ser apenas uma dessas palavras: bom, mediano ou ruim.	Manchester City teve 45% de posse, 10 chutes (1 no gol), 80% de precisão de passe, 7 escanteios e 10 faltas na partida contra o Wolverhampton, na temporada 2025.	ruim

3.4 Adicionando RAG ao Llama 3.1 8B

Após o pré-processamento dos dados, foi feito o desenvolvimento do RAG, seguindo a Figura 9:

Figura 9 – *Pipeline* do modelo com RAG.



Fonte: Autor

O desenvolvimento começa instalando as diversas bibliotecas necessárias, incluindo Transformers da Hugging Face, para acelerar operações e manipular dados. Em seguida, define-se o modelo a ser usado, que foi o Llama 3.1 de 8 bilhões de parâmetros e configura o ambiente para ser usado na GPU T4 do Colab. Nisso, são utilizadas técnicas de quantização para carregar o LLM na GPU.

Em seguida, o *tokenizer* da Transformers é associado ao modelo e cria uma lista de tokens de parada, que são usados para indicar o fim de uma geração de texto, onde também são definidos os critérios personalizados para interromper a geração de texto com base em determinados tokens de parada.

Então é instanciado um *pipeline* de geração de texto da Transformers para criar respostas a partir de consultas específicas, controlando fatores como a temperatura (aleatoriedade) e penalidades de repetição, e esse pipeline é integrado ao da biblioteca LangChain, permitindo uma abordagem mais flexível e modular para processamento de texto.

A partir disso passa a ser utilizado o LangChain para carregar os arquivos de texto. Sobre esses dados. Foi utilizado o modelo de embeddings da Hugging Face para representar os documentos e os armazena em uma Vector Store usando o ChromaDB, para operações de similaridade nos conjuntos de dados.

Por fim, é instanciado do LangChain uma *ConversationalRetrievalChain*, integrando o modelo de linguagem e a Vector Store. Essa cadeia (*chain*) é capaz de recuperar informações relevantes em resposta a perguntas específicas. A partir de agora, é possível realizar uma consulta à cadeia de recuperação com uma pergunta específica sobre o assunto dos dados salvos no Vector Store. Os exemplos das Tabelas 2 e 3 abaixo demonstram como o sistema responde a consultas complexas usando o formato Alpaca. A Tabela 2 mostra o modelo sem acesso aos dados, e a Tabela 3 mostra o mesmo modelo com RAG ativado, com o valor de *input* sendo preenchido.

Tabela 2 – Modelo pré-treinado (sem recuperação)

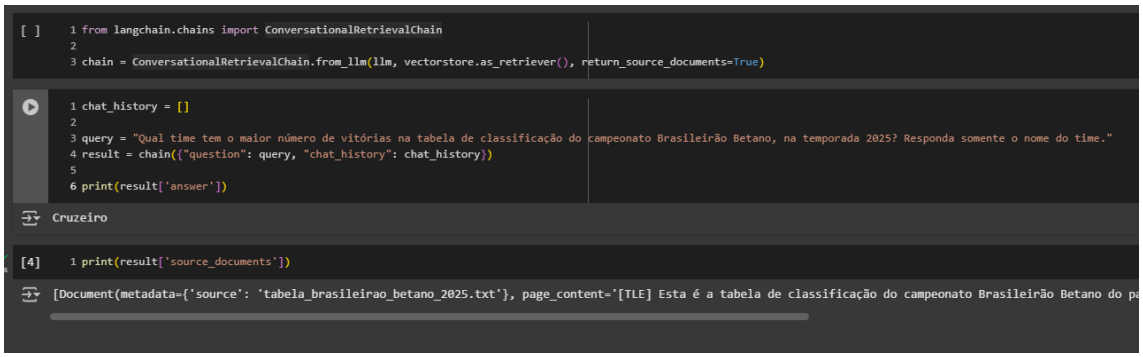
Instruction	Qual time tem o maior número de vitórias na tabela de classificação do campeonato Brasileirão Betano, na temporada 2025? Responda somente o nome do time.
Input	
Output	Flamengo é um dos principais times do Brasil, e costuma ter um bom desempenho. Possivelmente foi um dos melhores da temporada, mas não tenho certeza da posição exata.

Tabela 3 – Modelo com recuperação de informação (RAG)

Instruction	Qual time tem o maior número de vitórias na tabela de classificação do campeonato Brasileirão Betano, na temporada 2025? Responda somente o nome do time.
Input	Retornado da consulta RAG: [TLE] Esta é a tabela de classificação do campeonato Brasileirão Betano do país Brazil na temporada 2025 [TAB] | Posição | Team | Pontos | Jogos | Vitórias | Empates | Derrotas | Gols_Marcados | Gols_Sofridos | Saldo_de_Gols | | 1 | Cruzeiro | 33 | 15 | 10 | 3 | 2 | 27 | 9 | 18 | | 2 | Flamengo | 30 | 14 | 9 | 3 | 2 | 27 | 5 | 22 | | 3 | Bragantino | 27 | 15 | 8 | 3 | 4 | 18 | 15 | 3 | ...
Output	Fluminense

Como é possível observar, o modelo que não possui acesso à recuperação de informação não retornou uma resposta correta. Isso ocorre principalmente por esse dado não ter sido usado para treino do modelo original, principalmente por se tratar de um dado mais recente, de 2025. Na Figura 10, é possível verificar que a cadeia de recuperação do Langchain também disponibiliza a fonte (*source*) de onde o modelo retirou a informação.

Figura 10 – Inferência a partir do modelo com recuperação.



Fonte: Autor

3.5 Preparação de Dados para o Fine-Tuning

A preparação de dados para o fine-tuning de um modelo de linguagem de grande porte (LLM) é uma etapa fundamental, pois garante que o modelo seja ajustado com exemplos representativos e balanceados, de acordo com o comportamento desejado. Neste projeto, foram consideradas três tarefas distintas, todas relacionadas à análise de desempenho no futebol, como já descrito anteriomente. Cada uma delas recebeu um tratamento específico de pré-processamento, estruturação e balanceamento de classes:

- **Avaliação de Desempenho de Jogadores:** o modelo aprende a classificar o desempenho individual de um jogador em uma partida como **bom**, **mediano** ou **ruim**,

com base em suas estatísticas. Os dados foram agregados a partir das múltiplas requisições ao Sofascore API, e rotulados com base no campo de nota técnica (*rating*). Após o agrupamento, o conjunto foi balanceado de forma estratificada, considerando a menor classe.

- **Avaliação de Desempenho de Equipes:** nesta tarefa, o modelo avalia o desempenho coletivo de um time a partir de métricas agregadas como posse de bola, número de chutes, escanteios, entre outros. O dataset passou por uma etapa de reescrita textual dos campos *input* e *question*, de forma a gerar instruções consistentes e padronizadas. Também foi aplicado balanceamento entre as categorias de resposta (**bom**, **mediano** ou **ruim**).

- **Question Answering sobre Tabelas:** baseada em um conjunto com perguntas e respostas derivadas de dados tabulares dos campeonatos, essa tarefa foca em raciocínio contextual sobre tabelas hierárquicas. O dataset passou por limpeza de colunas, normalização de nomes de campeonatos e balanceamento entre os diferentes tipos de tarefa (***task_type***).

Para todas as tarefas, foi utilizado o formato de prompt Alpaca, com os campos *instruction*, *input*, *question* e *response*. Com o objetivo de enriquecer o contexto de entrada apresentado ao modelo, cada instância do campo `input` foi expandida por meio da composição com dois outros contextos similares, selecionados aleatoriamente. O resultado dessa simulação de recuperação semântica foi armazenado no campo `rag_input`, que serviu como entrada final utilizada durante o processo de treinamento. Essa estratégia caracteriza a aplicação da abordagem RAFT (*Retrieval-Augmented Fine-Tuning*), cuja implementação será detalhada a seguir.

Ao final do processo, os três datasets foram harmonizados com a mesma estrutura e concatenados após nova amostragem balanceada, gerando um único conjunto unificado para *fine-tuning*. Esse conjunto foi então dividido em treino e teste (80/20) e posteriormente publicado na plataforma Hugging Face Hub.

A seguir também, são apresentadas em detalhes as etapas de preparação e transformação de dados específicas para cada uma das três tarefas, com foco nas estratégias adotadas de balanceamento, geração de entradas e construção de perguntas e respostas.

3.5.1 Enriquecimento das entradas com RAFT

Com o objetivo de aproximar o modelo das condições reais de uso em sistemas baseados em recuperação de informações, foi adotada neste trabalho a abordagem *Retrieval-Augmented Fine-Tuning* (RAFT). Essa técnica, conforme proposta por (ZHANG; MOOSAVI; HOULSBY, 2024), combina a recuperação de contexto com o *fine-tuning* supervi-

sionado, de modo que o modelo aprenda a responder com base em múltiplas evidências — algumas relevantes e outras não — reforçando sua capacidade de selecionar corretamente as informações úteis para gerar uma resposta adequada.

Embora o foco principal do fine-tuning seja adaptar o modelo ao domínio do futebol — com tarefas como avaliação de desempenho de jogadores, equipes e interpretação de tabelas — também foi relevante aprimorar sua habilidade de localizar e utilizar corretamente informações contextuais. Essa habilidade é especialmente crítica na tarefa de *Question Answering* sobre tabelas, onde a resposta pode depender de dados espalhados por diferentes colunas ou registros.

Para implementar o RAFT neste trabalho, foi utilizado um método que simula um ambiente de recuperação semântica. Para cada exemplo do dataset, dois *inputs* adicionais foram selecionados aleatoriamente, além do *input* principal. Os três blocos de texto foram então embaralhados, separados por delimitadores visuais por hífens e precedidos da frase "Retornado da consulta RAG:". A entrada final gerada foi armazenada na coluna *rag_input*, que substituiu o campo de *input* tradicional no processo de treinamento.

Essa simulação de consulta em base vetorial foi aplicada uniformemente aos três subconjuntos de dados (jogadores, equipes e tabelas), garantindo consistência na estrutura das entradas. A Figura 11 ilustra conceitualmente o funcionamento da abordagem RAFT.

Figura 11 – Funcionamento do RAFT: uso de contexto recuperado para enriquecimento do fine-tuning.

Question: The Oberoi family is part of a hotel company that has a head office in what city?

context: [The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through The Oberoi Group]...[It is located in city center of Jakarta, near Mega Kuningan, adjacent to the sister JW Marriott Hotel. It is operated by The Ritz-Carlton Hotel Company. The complex has two towers that comprises a hotel and the Airlangga Apartment respectively]...[The Oberoi Group is a hotel company with its head office in Delhi.]

Instruction: Given the question, context and answer above, provide a logical reasoning for that answer. Please use the format of: ##Reason: {reason} ##Answer: {answer}.

CoT Answer: ##Reason: The document ##begin_quote## The Oberoi family is an Indian family that is famous for its involvement in hotels, namely through The Oberoi Group. ##end_quote## establishes that the Oberoi family is involved in the Oberoi group, and the document ##begin_quote## The Oberoi Group is a hotel company with its head office in Delhi. ##end_quote## establishes the head office of The Oberoi Group. Therefore, the Oberoi family is part of a hotel company whose head office is in Delhi. ##Answer: Delhi

Figure 3: RAFT prompt to help LLM evaluate its own generated reasoning and answers, contrasting them with the correct reasoning and answers. The LLM is prompted to identify errors in its reasoning and extract key insights for improvement. This figure specifically represents the 'GenerateExplanation' step in the RAFT algorithm (Section 3).

Fonte: Adaptado de (ZHANG; MOOSAVI; HOULSBY, 2024)

Com isso, o modelo foi exposto a entradas mais realistas e densas, desenvolvendo não apenas conhecimento sobre o domínio-alvo, mas também a capacidade de identifi-

car, entre múltiplas fontes possíveis, qual informação é mais relevante para responder corretamente à pergunta proposta.

3.5.2 Avaliação de Desempenho de Jogadores

A primeira tarefa considerada no processo de fine-tuning tem como objetivo ensinar o modelo a classificar o desempenho individual de jogadores com base em suas estatísticas em partidas específicas. Para isso, foi construído um conjunto de dados gerados previamente pelas requisições à API do Sofascore, contendo métricas de performance por jogador.

Todos os dados extraídos relativos aos jogadores foram carregados com o `pandas` e concatenados em um único *dataframe*. A partir da coluna *rating*, foi criada uma nova variável categórica chamada *ratingLabel*, utilizada como rótulo para treinamento. Essa classificação não foi definida arbitrariamente: foi baseada na própria convenção visual utilizada pela plataforma Sofascore, que destaca os ratings dos jogadores com cores distintas. Em geral, notas abaixo de 6,5 são destacadas em tons avermelhados (desempenho ruim), entre 6,5 e 6,9 em amarelo ou laranja (mediano), e acima de 7,0 em verde (bom). A Tabela 4 apresenta o critério adotado.

Tabela 4 – Critérios de classificação do desempenho dos jogadores

Faixa de Nota	Rótulo Atribuído
Inferior a 6,5	ruim
Entre 6,5 e 6,9	mediano
Igual ou superior a 7,0	bom

A correspondência entre faixas de nota e rótulos foi baseada nas cores que acompanham o *rating* exibido na interface pública do Sofascore, conforme ilustrado na Figura 12.

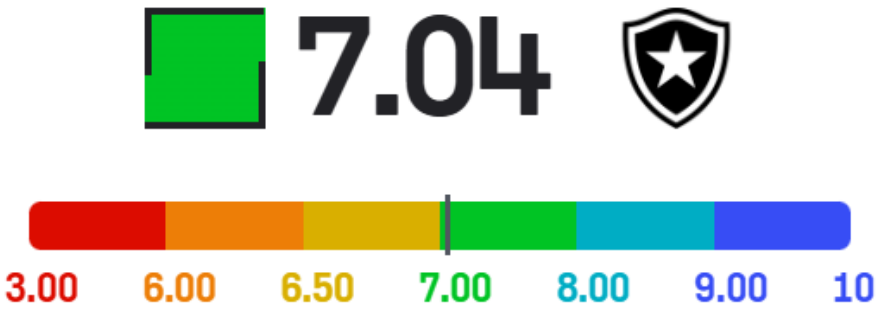


Figura 12 – Escala de cores associada às notas atribuídas pelo Sofascore.

Fonte: Sofascore.

Apesar dessa sinalização visual pública, o algoritmo interno utilizado pelo Sofascore para calcular a nota de performance (*rating*) não é divulgado em sua totalidade. Sabe-se que ele leva em consideração múltiplas estatísticas (como gols, passes, cortes, desarmes, entre outros), mas os pesos atribuídos e fórmulas exatas permanecem desconhecidos. Isso torna inviável a replicação exata desse cálculo de forma determinística. Assim, uma alternativa viável é treinar um modelo de linguagem que, exposto às estatísticas principais do jogador, aprenda a abstrair implicitamente o padrão de classificação contido nas notas já atribuídas pela plataforma.

Em seguida, o conjunto de dados foi balanceado com base na menor classe presente, por meio de amostragem estratificada. Isso assegurou que o número de exemplos para cada rótulo fosse uniforme, evitando viés no aprendizado do modelo.

Para estruturar os dados no formato de entrada do modelo, foi gerado um novo `DataFrame` contendo os seguintes campos:

- **instruction**: pergunta explícita ao modelo solicitando a classificação (bom, mediano ou ruim);
- **input**: sequência textual com o resumo estatístico do jogador na partida;
- **response**: rótulo correto da amostra, conforme definido anteriormente.

O campo *input* foi montado dinamicamente a partir das colunas do conjunto original, utilizando informações como gols marcados, assistências, chutes a gol, passes precisos, desarmes, cortes e interceptações. A *instruction* reforça a identificação do jogador e da partida, solicitando ao modelo uma resposta categórica.

Para simular um cenário de recuperação semântica e enriquecer as entradas com múltiplos contextos, foi gerado para cada amostra, um novo campo *rag_input* contendo o *input* original embaralhado com outros dois contextos estatísticos de jogadores diferentes. Esse resultado foi utilizado como entrada final no processo de fine-tuning, conforme descrito anteriormente sobre RAFT.

Essa tarefa foi especialmente importante para treinar o modelo a reconhecer padrões individuais de desempenho e compará-los implicitamente a expectativas associadas a diferentes posições em campo, o que favorece sua capacidade de generalização para partidas variadas.

A tabela 5 ilustra um registro do *dataset* gerado para realização de fine-tune sobre esta tarefa.

Tabela 5 – Exemplo de entrada final utilizada no fine-tuning para a tarefa de avaliação de desempenho de jogadores

Instruction	Considerando os dados apresentados, avalie o desempenho do jogador Renzo Saravia na partida Atlético Mineiro vs Internacional do campeonato Brasileirão Betano. Sua resposta deve ser apenas uma dessas palavras: bom, mediano ou ruim.
Input	Retornado da consulta RAG: Na partida Atlético Mineiro vs Internacional do campeonato Brasileirão Betano, o jogador Renzo Saravia, atuando como defender, marcou 0 gols, deu 0 assistências, realizou 0 chutes a gol, fez 30 passes precisos, criou 0 chances, fez 2 desarmes, 1 cortes defensivos, fez 0 interceptações, realizou 0 defesas, apresentou um índice de 0.0 gols evitados, e venceu 0 duelos aéreos. Na partida Getafe vs Osasuna do campeonato La Liga, o jogador Mauro Arambarri, atuando como midfield, marcou 0 gols, deu 0 assistências, realizou 0 chutes a gol, fez 9 passes precisos, criou 1 chances, fez 1 desarmes, 2 cortes defensivos, fez 0 interceptações, realizou 0 defesas, apresentou um índice de 0.0 gols evitados, e venceu 0 duelos aéreos. Na partida Rayo Vallecano vs Real Betis do campeonato La Liga, o jogador Aitor Ruibal, atuando como defender, marcou 0 gols, deu 0 assistências, realizou 0 chutes a gol, fez 13 passes precisos, criou 0 chances, fez 1 desarmes, 3 cortes defensivos, fez 1 interceptações, realizou 0 defesas, apresentou um índice de 0.0 gols evitados, e venceu 0 duelos aéreos.
Response	ruim

3.5.3 Avaliação de Desempenho de Equipes

A segunda tarefa proposta visa treinar o modelo a classificar o desempenho coletivo de uma equipe em uma partida, com base em um conjunto de métricas agregadas. Diferente da tarefa anterior, que focava em estatísticas individuais, aqui o objetivo é avaliar a atuação do time como um todo, o que exige ao modelo um entendimento contextual de performance coletiva.

Os dados foram obtidos a partir de requisições à API do Sofascore para cada evento finalizado. Para cada equipe participante de uma partida, foi extraída uma nota técnica global atribuída pela plataforma, além de estatísticas como posse de bola, total de chutes, chutes no gol, precisão de passe, escanteios e faltas.

Essa nota técnica reflete a atuação geral da equipe durante a partida, sendo também calculada de forma interna pela plataforma a partir de múltiplos fatores combinados. Assim como no caso dos jogadores, a nota aparece acompanhada de uma sinalização visual em cores na interface do Sofascore, sugerindo o mesmo mapeamento interpretativo entre

faixa de nota e qualidade de desempenho.

Com base nesse padrão visual, os rótulos utilizados para classificação da equipe também foram definidos conforme mostra a Tabela 6.

Tabela 6 – Critérios de classificação do desempenho de equipes com base na nota técnica do Sofascore

Faixa de Nota	Rótulo Atribuído
Inferior a 6,5	ruim
Entre 6,5 e 6,9	mediano
Igual ou superior a 7,0	bom

Esse mesmo padrão de cores já havia sido apresentado na Figura 12, e aqui também se aplica à nota das equipes. Apesar de a nota técnica não ser um número reproduzível de forma transparente, ela representa uma síntese confiável da atuação do time, podendo ser utilizada como referência supervisionada para ensinar o modelo a reconhecer diferentes níveis de desempenho coletivo.

Após carregamento e concatenação dos dados, foi gerada uma variável categórica `ratingLabel` com base na nota técnica da equipe, utilizada como rótulo para o fine-tuning. O conjunto foi então balanceado de forma estratificada entre as três categorias de resposta.

Os dados foram organizados no formato de prompt Alpaca, com os seguintes campos:

- **instruction**: pergunta ao modelo solicitando a classificação com base na atuação do time;
- **input**: resumo textual das estatísticas da equipe em uma partida específica;
- **response**: rótulo extraído da nota técnica de desempenho, como bom, mediano ou ruim.

Como nas demais tarefas, foi aplicado o enriquecimento por recuperação (RAFT), por meio da criação do campo `rag_input`, que mistura a entrada principal com dados contextuais de outras partidas, forçando o modelo a localizar e interpretar corretamente as estatísticas do time de interesse.

A Tabela 7 apresenta um exemplo real de entrada utilizada no fine-tuning.

Tabela 7 – Exemplo de entrada final utilizada no fine-tuning para a tarefa de avaliação de desempenho de equipes

Instruction	Com base nos dados apresentados, como você avalia o desempenho do time Lazio na partida Lazio vs Empoli pelo campeonato Serie A, na temporada 2025? Sua resposta deve ser apenas uma dessas palavras: bom, mediano ou ruim.
Input	Retornado da consulta RAG: Na partida Flamengo vs Red Bull Bragantino pela temporada 2023, o time Flamengo teve 53.0% de posse de boa, deu 16 chutes, sendo 4 no gol, a precisão de passe foi de 95.4%, obteve 8 escanteios e sofreu 10 faltas. Na partida Barcelona vs Athletic Club pela temporada 2018, o time Barcelona teve 62.0% de posse de boa, deu 12 chutes, sendo 7 no gol, a precisão de passe foi de 91.2%, obteve 4 escanteios e sofreu 11 faltas. Na partida Lazio vs Empoli pela temporada 42415, o time Lazio teve 54.0% de posse de boa, deu 18 chutes, sendo 4 no gol, a precisão de passe foi de 94.3%, obteve 10 escanteios e sofreu 17 faltas.
Response	mediano

3.5.4 Resposta a Perguntas sobre Tabelas Hierárquicas (Hierarchical Table QA)

A terceira tarefa de fine-tuning envolve o raciocínio sobre tabelas de classificação de campeonatos de futebol, tarefa conhecida como *Hierarchical Table QA*. O objetivo foi treinar o modelo para interpretar tabelas estruturadas contendo dados sobre desempenho de times em diferentes ligas e temporadas, e responder a perguntas baseadas nessas informações. Cada entrada inclui uma instrução, uma tabela serializada como texto, uma pergunta e a resposta esperada. O modelo deve inferir, calcular ou recuperar diretamente a informação solicitada.

As tabelas foram obtidas via API pública do Sofascore, com dados completos de dezenas de temporadas em múltiplos campeonatos, como Premier League, Ligue 1, La Liga, Brasileirão, entre outros. A entrada do modelo inclui os metadados da tabela (país, campeonato e temporada) no marcador `[TLE]`, seguidos pela tabela em si serializada com `[TAB]`.

Essa formatação com prefixos `[TLE]` (*table title*) e `[TAB]` (*table body*) foi inspirada no trabalho TableLlama (ZHANG et al., 2024), que demonstrou resultados promissores ao adaptar modelos de linguagem para tarefas tabulares por meio de tokenizações explícitas de contexto e conteúdo. A separação clara entre contexto e tabela permite que o modelo compreenda com mais precisão os diferentes blocos de informação e aprimore seu raciocínio

sobre estruturas hierárquicas.

```
"""
[TLE] Esta é a tabela de classificação do campeonato Serie A do país Itália na
↪  temporada 2023. \n
[TAB] | Posição | Team | Pontos | Jogos | Vitorias | Empates | Derrotas |
↪  Gols_Marcados | Gols_Sofridos | Saldo_de_Gols | \n
| 1 | Napoli | 90 | 38 | 28 | 6 | 4 | 77 | 28 | 49 |
| 2 | Lazio | 74 | 38 | 22 | 8 | 8 | 60 | 30 | 30 |
...
"""
```

O modelo foi treinado com múltiplas formas lógicas de raciocínio tabular. A Tabela 8 descreve os tipos utilizados, com exemplos ilustrativos.

Tabela 8 – Tipos de perguntas geradas para Hierarchical Table QA

Tipo de Pergunta	Descrição e Exemplo
Cell Selection	Recuperação direta de um valor da tabela. *Exemplo: Qual é o saldo de gols do time Napoli?*
Superlative	Seleção do time com maior valor para uma métrica. *Exemplo: Qual time tem o maior número de vitórias?*
Difference	Diferença entre duas equipes em um critério. *Exemplo: Qual a diferença de pontos entre Napoli e Lazio?*
Aggregation	Soma de valores de um subconjunto de linhas. *Exemplo: Quantos pontos somam os 4 primeiros times?*
Average	Cálculo de média entre valores por jogo. *Exemplo: Qual time teve a maior média de gols marcados por jogo?*
Distance to Threshold	Distância de um time a marcos críticos (ex: classificação ou rebaixamento). *Exemplo: Quantos pontos faltam para o time Lazio alcançar o primeiro colocado?*

A Tabela 9 mostra um exemplo real do dataset gerado com base na temporada 2025 da *Liga Profesional de Fútbol*, contendo todos os campos presentes no formato final.

Tabela 9 – Exemplo de entrada final utilizada no fine-tuning para a tarefa de Hierarchical Table QA

Instruction	Esta é uma tarefa de respostas a perguntas sobre uma tabela de classificação de um campeonato de futebol (Hierarchical Table QA). Com base nos dados apresentados, realize a leitura da tabela relativa ao campeonato citado e responda à pergunta, dizendo somente o valor perguntado.
Input	Retornado da consulta RAG: [TLE] Esta é a tabela de classificação do campeonato Liga Profesional de Fútbol do país Argentina na temporada 2025 \n [TAB] | Posição | Team | Pontos | Jogos | Vitórias | Empates | Derrotas | Gols_Marcados | Gols_Sofridos | Saldo_de_Gols | \n | 1 | Racing Club | 41 | 19 | 13 | 2 | 4 | 30 | 16 | 14 | \n | 2 | River Plate | 39 | 19 | 11 | 6 | 2 | 34 | 13 | 21 | \n | 3 | Lanús | 35 | 19 | 10 | 5 | 4 | 28 | 23 | 5 | \n . . . [TLE] Esta é a tabela de classificação do campeonato Liga Profesional de Fútbol do país Argentina na temporada 57478 \n [TAB] | Posição | Team | Pontos | Jogos | Vitórias | Empates | Derrotas | Gols_Marcados | Gols_Sofridos | Saldo_de_Gols | \n | 1 | Vélez Sarsfield | 51 | 27 | 14 | 9 | 4 | 38 | 16 | 22 | \n | 2 | Talleres | 48 | 27 | 13 | 9 | 5 | 34 | 27 | 7 | \n . . .
Question	Qual time tem o maior número de pontos na tabela de classificação do campeonato Liga Profesional de Fútbol 2025, na temporada 2025? Responda somente o nome do time.
Response	Racing Club

3.6 Treinamento Supervisionado com Fine-Tuning Multitarefa

Com a preparação e balanceamento dos dados finalizados para cada uma das três tarefas descritas — avaliação de jogadores, avaliação de times e QA sobre tabelas hierárquicas —, iniciou-se a etapa de treinamento supervisionado do modelo com ajuste fino (fine-tuning) multitarefa.

Para este experimento, foi utilizado o modelo **LLaMA 3.1 8B** disponibilizado pela **Unsloth**, um framework otimizado para o treinamento eficiente de grandes modelos de linguagem (*LLMs*) como LLaMA, Mistral, Phi e Gemma. O Unsloth oferece suporte nativo a quantização em 4 bits, LoRA de alto desempenho, checkpointing otimizado e integração com bibliotecas da Hugging Face (`transformers`, `datasets` e `trl`) (TEAM, 2024b; TEAM, 2024a; TEAM, 2024c).

O modelo usado neste trabalho foi o `Meta-Llama-3.1-8B-bnb-4bit`, cuja nomenclatura indica duas características importantes:

- **Quantização em 4 bits**: permite reduzir drasticamente o consumo de memória e acelerar o treinamento, o que viabiliza o uso do modelo mesmo em GPUs com recursos limitados, como a Nvidia T4, disponível gratuitamente no Google Colab.

- **bnb-4bit**: refere-se ao uso da biblioteca *BitsAndBytes* (DETTMERS et al., 2022), que implementa operadores otimizados para quantização e inferência eficiente de modelos com parâmetros em baixa precisão.

Além disso, o modelo `Meta-Llama-3.1-8B` representa uma evolução do LLaMA original da Meta AI, com 8 bilhões de parâmetros treinados com 15 trilhões de tokens, demonstrando forte capacidade de generalização em tarefas de linguagem (AI, 2024).

O processo de fine-tuning seguiu a documentação oficial da Unsloth, que fornece tutoriais otimizados para LLaMA 3.1 (TEAM, 2024b). Os detalhes técnicos desse treinamento são apresentados nas próximas subseções.

3.6.1 Balanceamento dos Dados e Construção do Dataset Final

Como cada tarefa possuía uma quantidade desigual de exemplos, foi necessário aplicar um balanceamento por amostragem estratificada para garantir uma distribuição equitativa no treinamento multitarefa. A menor subtarefa, avaliação de desempenho de times, continha 3.834 exemplos. Assim, as demais foram amostradas aleatoriamente para igualar esse número, assegurando que nenhuma tarefa dominasse o aprendizado.

Após o balanceamento, os datasets de jogadores, times e tabelas foram padronizados para conter os campos `instruction`, `input` (com contexto simulado via RAG), `question` e `response`. Em seguida, os três conjuntos foram unificados e embaralhados, resultando em um *dataset* com 11.502 exemplos, conforme resumido na Tabela 10.

Tabela 10 – Resumo do balanceamento e unificação dos datasets

Critério	Valor
Quantidade usada por tarefa	3.834 exemplos
Total de exemplos no dataset final	11.502 exemplos
Campos incluídos	instruction, input, question, response

Além do balanceamento entre tarefas, também foi aplicado um controle interno sobre o dataset de Hierarchical Table QA. Essa tarefa incluía diferentes tipos lógicos de perguntas tabulares, como seleções de célula, cálculos de diferença, médias e distâncias para zonas da tabela. Para garantir equilíbrio entre essas categorias, foram selecionados 429 exemplos para cada uma, conforme mostra a Tabela 11.

Tabela 11 – Distribuição balanceada por tipo de pergunta no dataset de Hierarchical Table QA

Tipo de Pergunta	**Total de Exemplos**
Cell Selection	429
Superlative	429
Difference	429
Aggregation	429
Average	429
Distance to First	429
Distance to Last	429
Distance to Qualification	429
Distance to Relegation	429

Também foi considerada a diversidade de campeonatos representados nas tabelas de classificação. A Tabela 12 mostra uma boa distribuição entre ligas europeias e sul-americanas, o que reforça a capacidade do modelo de generalizar padrões tabulares em diferentes contextos competitivos.

Tabela 12 – Distribuição de exemplos por campeonato no dataset de Hierarchical Table QA

Campeonato	**Total de Exemplos**
Ligue 1	498
Premier League	459
Brasileirão Betano	436
Serie A	435
Bundesliga	428
La Liga	422
Liga Profesional de Fútbol	413
Eredivisie	390
Liga Portugal	380

Por fim, o dataset balanceado foi dividido em subconjuntos de treino (80%) e teste (20%), e publicado no Hugging Face Hub[1] para reprodutibilidade.

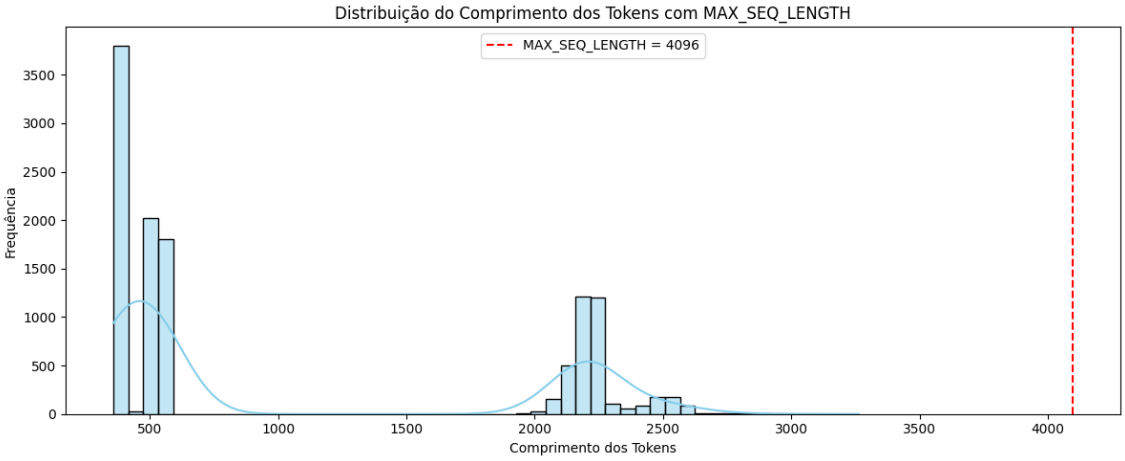
3.6.2 Tokenização e análise de comprimento

O processo de tokenização foi realizado utilizando o tokenizer embutido no próprio modelo unsloth/meta-Llama-3.1-8B-bnb-4bit. O comprimento dos tokens gerados foi analisado para garantir que os dados estivessem dentro do limite de sequência, que aqui foi determinado como 4096, que está dentro do limite de contexto do modelo selecionado, que é de 8192.

[1] <https://huggingface.co/datasets/guilhermemoraisr/football-qa-analysis-raft-alpaca>

A Figura 13 mostra a distribuição do comprometo dos *tokens* e destaca que todos os registros do dataset de treinamento se enquadram dentro do limite determinado.

Figura 13 – Distribuição do comprimento de *tokens*.



Fonte: Autor

3.6.3 Adaptação com LoRA: Eficiência no Fine-Tuning de Grandes Modelos

Durante o treinamento supervisionado, optou-se pelo uso da técnica **LoRA (Low-Rank Adaptation of Large Language Models)** (HU et al., 2021), um método eficiente e amplamente adotado para adaptar grandes modelos de linguagem (*LLMs*) sem a necessidade de atualizar todos os seus bilhões de parâmetros. Em vez disso, LoRA congela os pesos do modelo original e introduz matrizes de baixo ranque nos caminhos de projeção linear (como `q_proj`, `k_proj`, `v_proj`, etc.) dentro dos blocos de atenção e feed-forward. Essas matrizes são treináveis e representam a "adaptação" que o modelo aprende ao longo do fine-tuning.

Do ponto de vista matemático, LoRA decompõe a matriz de pesos original W em dois fatores treináveis de baixa dimensão, $A \in R^{d\times r}$ e $B \in R^{r\times k}$, onde r é o **rank** da decomposição. Essa parametrização reduz drasticamente o número de parâmetros ajustáveis, o que economiza memória, acelera o treinamento e ainda permite que o modelo mantenha seu desempenho original ao ser adaptado para uma nova tarefa (HU et al., 2021).

A Tabela 13 apresenta os principais hiperparâmetros utilizados para a configuração do LoRA neste experimento, com base nas recomendações do framework **Unsloth** (TEAM, 2024c; TEAM, 2024a; TEAM, 2024b):

Tabela 13 – Hiperparâmetros utilizados na adaptação com LoRA

Parâmetro	Valor e Justificativa
`r`	16 — define o rank da decomposição; valor intermediário que equilibra capacidade e custo computacional.
`lora_alpha`	16 — fator de escala para controlar a contribuição dos adaptadores LoRA na saída do modelo.
`lora_dropout`	0 — ausência de regularização, conforme recomendado para maximizar desempenho com Unsloth.
`bias`	`"none"` — elimina o uso de viés adicional nas projeções, economizando memória e tempo.
`use_gradient_checkpointing`	`"unsloth"` — ativa mecanismo proprietário para redução de consumo de memória, viabilizando contextos de até 4096 tokens em GPUs limitadas.

Essa abordagem é especialmente vantajosa em cenários com restrições de hardware, como ambientes de desenvolvimento gratuitos no Google Colab, permitindo treinar grandes modelos como o LLaMA 3.1 8B de forma leve e eficiente. Além disso, como apenas os adaptadores LoRA são salvos ao final do processo, os arquivos gerados são compactos e facilmente reutilizáveis, simplificando a publicação no Hugging Face Hub ou o carregamento para inferência futura.

3.6.4 Treinamento com `SFTTrainer` e Análise da Perda

Com o modelo LLaMA 3.1 8B carregado via Unsloth com quantização 4 bits e adaptado com LoRA, foi iniciado o processo de fine-tuning supervisionado com auxílio do módulo `SFTTrainer` da biblioteca `trl`. Esse componente é amplamente utilizado para ajuste supervisionado de modelos de linguagem e possui integração otimizada com a arquitetura da Hugging Face (TEAM, 2024b).

O corpus de treinamento foi preparado conforme o template Alpaca, contendo os campos `instruction`, `input` e `response`, concatenados em um único campo textual com adição do token de fim de sequência (`EOS_TOKEN`). A Tabela 14 resume os principais hiperparâmetros utilizados durante o processo de fine-tuning:

Tabela 14 – Hiperparâmetros utilizados no treinamento supervisionado

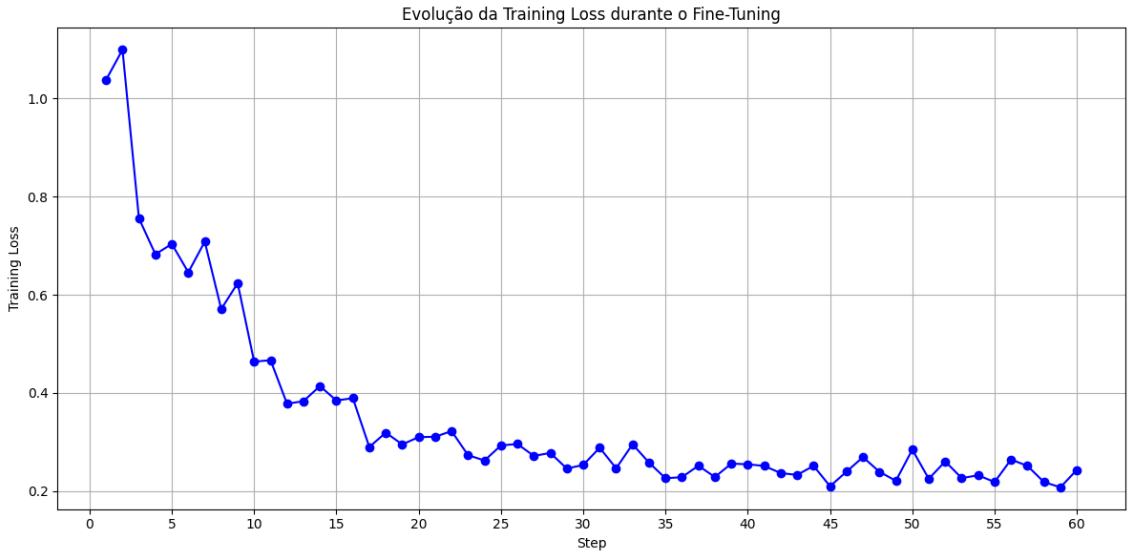
Parâmetro	Valor
`model_name`	`unsloth/Meta-Llama-3.1-8B-bnb-4bit`
`trainable_parameters`	41,943,040 (aprox. 0.52% do total)
`per_device_train_batch_size`	2
`gradient_accumulation_steps`	4
`max_steps`	60
`learning_rate`	2×10^{-4}
`warmup_steps`	5
`weight_decay`	0.01
`lr_scheduler_type`	linear
`optim`	`adamw_8bit`
`max_seq_length`	4096
`packing`	False
`random_seed`	3407

Diversas combinações de hiperparâmetros foram testadas durante os experimentos preliminares. No entanto, a configuração apresentada na Tabela 14 mostrou-se ideal, atingindo os níveis desejados de desempenho segundo a métrica de perda (*loss*) sugerida pela própria equipe do Unsloth.

3.6.4.1 Evolução da Perda e Critério de Parada

Durante o treinamento, a perda foi monitorada a cada etapa. A Figura 14 mostra a curva resultante, composta por 60 passos ao longo de uma única época.

Figura 14 – Curva de perda ao longo das etapas de treinamento supervisionado.



Fonte: Autor

A perda apresentou redução rápida nos primeiros passos, saindo de valores próxi-

mos a 1.0 até estabilizar entre 0.2 e 0.25 nas iterações finais. Esse comportamento segue exatamente a orientação oficial do Unsloth: o objetivo é minimizar a perda ao máximo, com um bom resultado sendo qualquer valor próximo de 0.2 — sem ultrapassar esse limite inferior, o que poderia indicar overfitting ou aprendizagem incorreta (TEAM, 2024c).

4 Resultados

4.1 Avaliação do Modelo Fine-tuned

Após a finalização do treinamento supervisionado do modelo LLaMA 3.1 8B com LoRA 4bit, conforme descrito anteriormente, foi realizada uma avaliação sistemática para medir o desempenho do modelo ajustado frente ao modelo base e ao modelo GPT-4o da OpenAI. Esta comparação objetiva quantificar os ganhos obtidos com o processo de fine-tuning supervisionado, considerando as tarefas específicas do domínio futebolístico propostas no dataset multitarefa.

4.1.1 Metodologia de Avaliação

A avaliação foi realizada sobre o subconjunto da divisão de teste do dataset publicado no Hugging Face Hub. Cada exemplo foi estruturado com base no template Alpaca, contendo um `instruction`, `input` e uma `question`. As respostas geradas por três modelos distintos — o modelo fine-tuned, o modelo base (sem ajuste) e o modelo GPT-4o — foram comparadas à resposta esperada utilizando métricas automáticas amplamente adotadas na literatura de NLP:

- **BLEU**: mede a precisão de n-gramas entre a resposta gerada e a referência;
- **ROUGE-1**, **ROUGE-2**, **ROUGE-L**: avaliam o recall de n-gramas e a similaridade estrutural;
- **BERTScore (P, R, F1)**: avalia a similaridade semântica entre textos via embeddings do modelo BERT;
- **METEOR**: métrica que considera sinônimos, stemming e alinhamento lexical;
- **GPT-4 Score**: avaliação qualitativa contextualizada, atribuída pelo modelo GPT-4o, em uma escala de 0 a 100.

4.1.2 Resultados Quantitativos

A Tabela 15 apresenta as pontuações obtidas pelos três modelos em cada métrica:

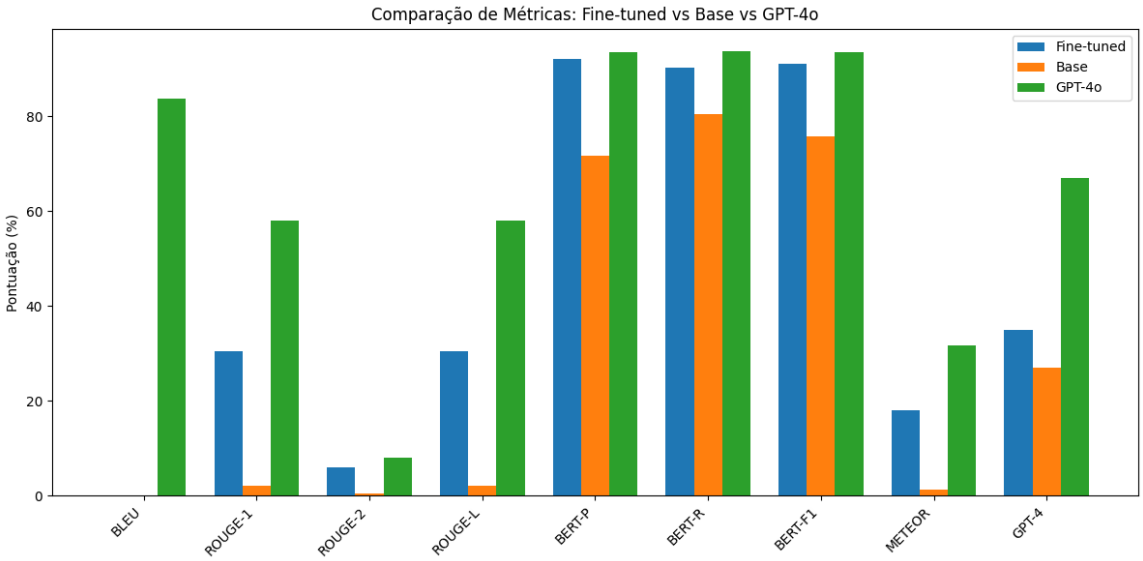
Tabela 15 – Comparação das métricas entre os modelos Fine-tuned, Base e GPT-4o

Métrica	Fine-tuned	Base	GPT-4o
BLEU	0.00	0.04	83.47
ROUGE-1	25.50	1.49	54.67
ROUGE-2	2.00	0.19	4.67
ROUGE-L	25.50	1.49	54.67
BERTScore-P	91.98	71.16	93.18
BERTScore-R	90.98	80.37	93.39
BERTScore-F1	91.30	75.42	93.19
METEOR	13.71	1.36	28.77
GPT-4 Score	35.30	15.20	65.00

Os resultados demonstram ganhos substanciais do modelo fine-tuned em relação ao modelo base, especialmente nas métricas de similaridade semântica (BERTScore) e na pontuação qualitativa atribuída pelo GPT-4. Isso evidencia que o fine-tuning proporcionou ao modelo uma capacidade de geração mais precisa, contextualizada e aderente às tarefas propostas.

A Figura 15 apresenta uma visualização gráfica das métricas, reforçando a superioridade do modelo fine-tuned sobre o modelo base e aproximando-se do desempenho de um modelo de ponta como o GPT-4o.

Figura 15 – Comparação de Métricas: Fine-tuned vs Base vs GPT-4o



Fonte: Autor

4.1.3 Análise Qualitativa de Exemplos

Durante a avaliação, foram analisados manualmente exemplos individuais para comparar as respostas dos modelos. Em muitos casos, o modelo fine-tuned se aproximou

da resposta correta de maneira precisa, concisa e adequada ao contexto da tarefa. Sua performance revelou domínio sobre o padrão de resposta esperado no dataset multitarefa, com capacidade de resumir corretamente dados tabulares, realizar avaliações subjetivas com base em estatísticas e identificar padrões em inputs estruturados.

Por outro lado, o modelo base (sem qualquer ajuste supervisionado) frequentemente apresentou respostas verbosas, genéricas ou incorretas. Isso evidencia sua limitação em compreender as nuances do domínio futebolístico — como termos técnicos, indicadores estatísticos e estilos de perguntas. Em geral, o modelo base pareceu deslocado frente às instruções, demonstrando falta de adaptação ao tipo de linguagem e tarefa esperada, gerando frequentemente explicações desnecessárias ou incorretas.

Para fins de comparação adicional, utilizou-se o modelo GPT-4o como referência de desempenho de modelos de última geração. As respostas por ele geradas seguiram exatamente o mesmo prompt aplicado aos demais modelos — ou seja, um template padronizado no formato Alpaca, com campos de `instruction`, `input` e `question`. O objetivo não foi compará-lo em profundidade, mas incluí-lo como um limite superior de qualidade, a fim de observar quão distante o modelo fine-tuned se encontra de um sistema considerado estado da arte.

Além disso, explorou-se uma técnica qualitativa na qual o próprio modelo GPT-4o atua como avaliador. Ele recebe tanto a *resposta esperada* quanto a *resposta gerada* e retorna uma nota objetiva de 0 a 100. Essa prática aborda uma avaliação contextual mais alinhada ao raciocínio humano, principalmente em domínios subjetivos ou com múltiplas possibilidades de resposta correta.

A Tabela 16 apresenta exemplos reais, extraídos da avaliação qualitativa, comparando diretamente as saídas dos três modelos. Esses casos ilustram os padrões observados durante a análise.

Tabela 16 – Exemplos reais comparando as saídas dos modelos

Resposta Esperada	**Fine-tuned**	**Base**	**GPT-4o**
mediano	mediano	Internacional: boa posse, muitos chutes e escanteios. [...]	mediano
mediano	mediano	Brentford: 50% de posse, 4 gols, 5 faltas, 0% passe. [...]	bom
1. FC Köln	FC Bayern München	Bayern, Augsburg, Stuttgart, Leverkusen... (muitos times listados)	1. FC Köln

5 Considerações finais

Este trabalho investigou o uso integrado de técnicas avançadas de aprendizado profundo, especificamente Retrieval-Augmented Generation (RAG) e fine-tuning supervisionado, para melhorar o desempenho de modelos de linguagem de grande porte (LLMs) em tarefas relacionadas à análise futebolística. A metodologia proposta envolveu um pipeline estruturado, desde a obtenção e pré-processamento de dados específicos de futebol, extraídos da plataforma Sofascore, até a avaliação rigorosa do modelo ajustado LLaMA 3.1 8B.

A utilização dos dados do Sofascore foi fundamental para garantir que o modelo tivesse acesso a informações precisas, atualizadas e relevantes sobre jogadores, equipes e tabelas de classificação. Esse acesso possibilitou o treinamento especializado do modelo em tarefas específicas, como classificação de desempenho individual e coletivo, além da interpretação avançada de dados tabulares complexos. A abordagem de enriquecimento dos dados via RAFT (Retrieval-Augmented Fine-Tuning) mostrou-se particularmente eficiente, permitindo ao modelo identificar contextos relevantes em meio a múltiplas informações e responder de maneira assertiva às consultas propostas.

O fine-tuning multitarefa com LoRA em quantização de 4 bits demonstrou alta eficiência computacional, permitindo o ajuste de um modelo robusto mesmo em ambientes com restrições de hardware. Essa estratégia não apenas viabilizou o uso de recursos computacionais limitados, como também produziu resultados superiores em relação ao modelo base, destacando o potencial prático desta abordagem.

Os resultados quantitativos e qualitativos da avaliação comprovaram que o modelo ajustado superou significativamente o desempenho do modelo base em todas as métricas automáticas analisadas, aproximando-se, em certos casos, do desempenho de modelos estado-da-arte, como o GPT-4o. Destaca-se particularmente a melhoria acentuada na capacidade do modelo em responder perguntas baseadas em dados tabulares hierárquicos, uma tarefa reconhecidamente complexa para LLMs pré-treinados.

No entanto, é importante ressaltar algumas limitações observadas. O sucesso das técnicas aplicadas depende diretamente da qualidade e da abrangência dos dados utilizados, o que sugere a necessidade constante de atualização e expansão do dataset para manter o desempenho ótimo do modelo. Além disso, embora a quantização e o uso de LoRA tenham reduzido significativamente a demanda computacional, o treinamento de modelos de grande escala ainda exige recursos substanciais, o que pode restringir sua acessibilidade.

Em resumo, este estudo contribui significativamente para o avanço da aplicação

prática de modelos de linguagem em domínios especializados, demonstrando o valor da integração entre recuperação semântica, fine-tuning supervisionado e estratégias eficientes de treinamento. Esses avanços abrem caminhos promissores tanto para pesquisa acadêmica quanto para aplicações comerciais no campo esportivo e além.

5.1 Trabalhos futuros

Várias direções promissoras podem ser exploradas para expandir a compreensão e a aplicabilidade dos modelos de linguagem de grande porte em domínios especializados. Primeiramente, é interessante a implementação e avaliação de uma gama mais ampla de modelos, incluindo aqueles com tamanhos de parâmetro maiores ou arquiteturas diferentes, para uma comparação mais abrangente de desempenhos.

Além disso, um estudo mais detalhado sobre o impacto de diferentes fontes de dados e a inclusão de dados multimodais—como a combinação de texto com imagens ou vídeos—poderia melhorar a capacidade do modelo de entender e gerar respostas mais ricas e contextuais. Explorar técnicas avançadas de aumento e pré-processamento de dados também pode aprimorar a robustez e a capacidade de generalização do modelo.

Outro aspecto interessante seria aplicar os modelos ajustados em outros contextos específicos, como previsão de resultados de partidas, análise de transferências de jogadores ou detecção de padrões em estratégias de equipes. Além disso, a integração desses modelos em sistemas mais amplos, como plataformas interativas de análise ou ferramentas de apoio à decisão para gestão esportiva, poderia oferecer novas perspectivas sobre sua aplicabilidade prática.

Por fim, investigar métodos para reduzir os custos computacionais, como destilação de modelos ou técnicas de *fine-tuning* eficientes, tornaria esses modelos avançados mais acessíveis para implantação em ambientes com recursos limitados.

Essas direções não apenas fortaleceriam o campo do processamento de linguagem natural em domínios especializados, mas também contribuiriam para o avanço da inteligência artificial em aplicações práticas e relevantes.

Referências

AI, M. *Introducing LLaMA 3*. 2024. Acesso em: jul. 2025. Disponível em: <https://ai.meta.com/blog/meta-llama-3/>. Citado na página 62.

CARNEIRO, T. et al. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access*, v. 6, p. 61677–61685, 2018. Citado na página 26.

CHOLLET, F. *Deep Learning with Python*. Manning Publications Company, 2017. ISBN 9781617294433. Disponível em: <https://books.google.de/books?id=Yo3CAQAACAAJ>. Citado 4 vezes nas páginas 26, 29, 30 e 32.

DETTMERS, T. et al. *BitsAndBytes: 8-bit Optimizers and Quantization for Transformers*. 2022. Acesso em: jul. 2025. Disponível em: <https://github.com/TimDettmers/bitsandbytes>. Citado na página 62.

DODGSON, J. et al. Establishing performance baselines in fine-tuning, retrieval-augmented generation and soft-prompting for non-specialist llm users. 2023. Citado 2 vezes nas páginas 25 e 36.

FACE, H. *SFTTrainer - Hugging Face*. [S.l.], n.d. Acesso em: 14 de outubro de 2024. Disponível em: <https://huggingface.co/docs/trl/main/sft_trainer>. Citado 2 vezes nas páginas 36 e 37.

GHODRATNAMA, S.; ZAKERSHAHRAK, M. Adapting llms for efficient, personalized information retrieval: Methods and implications. 2023. Citado 2 vezes nas páginas 25 e 28.

HU, E. J. et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. Citado na página 64.

JOHNSON, J.; DOUZE, M.; JéGOU, H. Billion-scale similarity search with gpus. 2017. Citado na página 29.

JURAFSKY, D.; MARTIN, J. H. *Speech and Language Processing. 3rd*. [S.l.]: Disponível em: <https://web. stanford. edu/̃jurafsky/slp3/> Acesso em: 05/12/2023., 2022. Citado 2 vezes nas páginas 32 e 33.

LANGCHAIN. *Python Langchain Documentation*. [S.l.]: Disponível em: <https://python.langchain.com/docs>. Acesso em: 05/12/2023., 2023. Citado 2 vezes nas páginas 34 e 35.

LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. 2021. Citado 2 vezes nas páginas 25 e 34.

LIN, C.-Y. ROUGE: A package for automatic evaluation of summaries. In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, 2004. p. 74–81. Disponível em: <https://aclanthology.org/W04-1013>. Citado na página 37.

LIU, Y. et al. G-eval: Nlg evaluation using gpt-4 with better human alignment. *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, p. 8776–8788, 2023. Disponível em: <https://aclanthology.org/2023.emnlp-main.543>. Citado na página 39.

MANATHUNGA, S. S.; ILLANGASEKARA, Y. A. Retrieval augmented generation and representative vector summarization for large unstructured textual data in medical education. 2023. Citado na página 25.

RASCHKA, S.; PATTERSON, J.; NOLET, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. 2020. Citado na página 26.

SAADANY, H.; ORăSAN, C. Bleu, meteor, bertscore: Evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. In: *Proceedings of the Translation and Interpreting Technology Online Conference TRITON 2021*. INCOMA Ltd. Shoumen, BULGARIA, 2021. (TRITON 2021). Disponível em: <http://dx.doi.org/10.26615/978-954-452-071-7_006>. Citado na página 38.

SELLAM, T.; DAS, D.; PARIKH, A. P. *BLEURT: Learning Robust Metrics for Text Generation*. 2020. Citado na página 38.

TAORI, R. et al. *Stanford Alpaca: An Instruction-Following LLaMA Model*. 2023. <https://crfm.stanford.edu/2023/03/13/alpaca.html>. Acesso em julho de 2025. Citado na página 49.

TEAM, U. *Fine-tuning LLMs Guide*. 2024. Acesso em: jul. 2025. Disponível em: <https://docs.unsloth.ai/get-started/fine-tuning-llms-guide>. Citado 2 vezes nas páginas 61 e 64.

TEAM, U. *How to Finetune LLaMA 3 and Use in Ollama*. 2024. Acesso em: jul. 2025. Disponível em: <https://docs.unsloth.ai/basics/tutorials-how-to-fine-tune-and-run-llms/tutorial-how-to-finetune-llama-3-and-use-in-ollama>. Citado 4 vezes nas páginas 61, 62, 64 e 65.

TEAM, U. *LoRA Hyperparameters Guide*. 2024. Acesso em: jul. 2025. Disponível em: <https://docs.unsloth.ai/get-started/fine-tuning-llms-guide/lora-hyperparameters-guide>. Citado 3 vezes nas páginas 61, 64 e 67.

TOUVRON, H. et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2404.00213*, 2023. Disponível em: <https://arxiv.org/abs/2404.00213>. Citado na página 36.

TOUVRON, H. et al. *LLaMA 3 Technical Report*. 2024. <https://ai.meta.com/llama/>. Accessed: July 2025. Citado na página 35.

VASWANI, A. et al. Attention is all you need. 2023. Citado na página 33.

WOLF, T. et al. Huggingface's transformers: State-of-the-art natural language processing. 2020. Citado na página 27.

ZHANG, T. et al. TableLlama: Towards open large generalist models for tables. In: DUH, K.; GOMEZ, H.; BETHARD, S. (Ed.). *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Mexico City, Mexico: Association for Computational Linguistics, 2024. p. 6024–6044. Disponível em: <https://aclanthology.org/2024.naacl-long.335/>. Citado 2 vezes nas páginas 48 e 59.

ZHANG, Y.; MOOSAVI, N. S.; HOULSBY, N. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131*, 2024. Disponível em: <https://arxiv.org/abs/2403.10131>. Citado 2 vezes nas páginas 53 e 54.

ZHAO, W. X. et al. A survey of large language models. 2023. Citado na página 30.

Apêndices

APÊNDICE A – Materiais de suporte

A seguir estão os *links* dos materiais de suporte que possuem os *notebooks* e artefatos do trabalho realizado:

- Repositório com notebooks no GitHub
- Dataset de treinamento do fine-tuning no Hugging Face
- Modelo Llama 3.1 ajustado por fine-tuning no Hugging Face