UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

HENRIQUE GRATTIERI FERRETO

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA GERAÇÃO DE LAUDOS MÉDICOS UTILIZANDO A API DO GPT

CAXIAS DO SUL

2024

HENRIQUE GRATTIERI FERRETO

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA GERAÇÃO DE LAUDOS MÉDICOS UTILIZANDO A API DO GPT

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Profa. Dra. Elisa Boff

CAXIAS DO SUL

2024

HENRIQUE GRATTIERI FERRETO

DESENVOLVIMENTO DE UMA FERRAMENTA AUTOMATIZADA PARA GERAÇÃO DE LAUDOS MÉDICOS UTILIZANDO A API DO GPT

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

BANCA EXAMINADORA

Profa. Dra. Elisa Boff
Universidade de Caxias do Sul - UCS

Prof. Dra. Carine Geltrudes Webber
Universidade de Caxias do Sul - UCS

Prof. Dr. Leandro Luis Corso
Universidade de Caxias do Sul - UCS

RESUMO

O Aprendizado de Máquina é um subcampo da Inteligência Artificial (IA) focado no desenvolvimento de algoritmos e modelos que permitem que computadores aprendam a partir de experiências, fazendo previsões ou tomando decisões sem serem explicitamente programados para tarefas específicas. Ele desempenha um papel essencial na IA Generativa, permitindo que modelos, como o GPT, sejam treinados com grandes volumes de dados para gerar texto de maneira coerente e precisa. A capacidade de aprender e replicar padrões complexos torna esses modelos ideais para aplicações que exigem alto grau de precisão e consistência, como a geração de laudos médicos.

Com o avanço das tecnologias em equipamentos de diagnóstico por imagem (como Raio-X, Ecografia, Tomografia e Ressonância Magnética), o número de exames de imagem por paciente aumentou significativamente, levando a uma sobrecarga dos profissionais responsáveis pela análise das imagens e elaboração dos laudos. Este trabalho teve como objetivo desenvolver uma ferramenta para automatizar o processo de elaboração de laudos, de modo que os profissionais se concentrem apenas na análise dos exames de imagem.

A ferramenta foi desenvolvida utilizando a arquitetura *Backend For Frontend*. O *Backend* foi implementado na linguagem de programação *Python*, sendo responsável por chamar a API do GPT para a geração dos laudos e por transformar áudio em texto, caso o médico opte por fornecer entradas por meio de áudio. O *Frontend* é responsável pela interface e interação com o usuário, executando-se em um navegador web acessível em qualquer dispositivo, seja *Desktop* ou *Mobile*, e foi desenvolvido com HTML5, CSS3 e JavaScript.

Para os testes, foi disponibilizado ao médico radiologista um link de acesso à plataforma. Assim, o profissional teve liberdade para utilizar, analisar e avaliar o funcionamento do sistema.

Palavras-chave: Aprendizado de Máquina. IA Generativa. GPT. Laudos de Exames de Imagem.

ABSTRACT

Machine Learning is a subfield of AI focused on the development of algorithms and models that enable computers to learn from experience, making predictions or decisions without being explicitly programmed for specific tasks. It plays an essential role in Generative AI, allowing models, such as GPT, to be trained with large volumes of data to generate text coherently and accurately. The ability to learn and replicate complex patterns makes these models ideal for applications requiring a high degree of precision and consistency, such as generating medical reports.

With advancements in imaging diagnostic technologies (such as X-ray, Ultrasound, CT, and MRI), the number of imaging exams per patient has significantly increased, leading to an overload of professionals responsible for analyzing the images and preparing reports. This work aimed to develop a tool to automate the report generation process, so professionals can focus solely on the analysis of the imaging exams.

The tool was developed using the *Backend For Frontend* architecture. The *Backend* was implemented in the *Python* programming language and is responsible for calling the GPT API to generate the reports and converting audio to text, should the physician choose to provide input via audio. The *Frontend* handles the user interface and interaction, running on a web browser accessible from any device, whether *Desktop* or *Mobile*, and was developed with HTML5, CSS3, and JavaScript.

For testing purposes, a link to access the platform was provided to the radiologist. This allowed the professional the freedom to use, analyze, and evaluate the system's functionality.

Keywords: Machine Learning. Generative AI. GPT. Imaging Exam Reports.

LISTA DE FIGURAS

Figura 1 – Hierarquia do Aprendizado Supervisionado 15
Figura 2 – Fluxo resumido de problemas de Classificação 16
Figura 3 – Exemplo de um algoritmo de classificação de e-mails 17
Figura 4 – Capturas de tela do jogo RoboSumo 19
Figura 5 – Esquema de um neurônio biológico (esquerda) e um neurônio artificial (direita) 19
Figura 6 – Rede neural multinível mostrando as camadas de entrada, oculta e de saída . 20
Figura 7 – Diagrama da Inteligência Artificial 23
Figura 8 – Estrutura do *Transformer* 24
Figura 9 – Representação da arquitetura da aplicação 47
Figura 10 – *Mockup* da tela de entrada de dados 49
Figura 11 – *Mockup* da tela de edição de texto 50
Figura 12 – *Mockup* das duas telas em um smartphone 50
Figura 13 – Captura da tela de entrada 51
Figura 14 – Captura da tela de edição 52
Figura 15 – Captura do resultado da solicitação de um laudo normal 55
Figura 16 – Captura do resultado da solicitação de um laudo de bursite moderada 56
Figura 17 – Captura do resultado da solicitação de um laudo de artropatia degenerativa acromioclavicular 56
Figura 18 – Captura do resultado da solicitação de um laudo de um paciente com tendinopatias 57

LISTA DE TABELAS

Tabela 1 – Exemplo do funcionamento de um algoritmo de previsão 17
Tabela 2 – Exemplo de um algoritmo de aprendizado não-supervisionado 18
Tabela 3 – Notas Alcançadas 41
Tabela 4 – Questões por etapa 42
Tabela 5 – Notas Alcançadas 43
Tabela 6 – Tabela Comparativa 45

LISTA DE ALGORITMOS

Algoritmo 1 Exemplo de chamada da API do Chat Completions 34
Algoritmo 2 Exemplo de resposta da API do Chat Completions 34
Algoritmo 3 Exemplo de do dataset necessário para efetuar o *fine-tuning* 36
Algoritmo 4 Exemplo de chamada para o envio do *dataset* para a API do fine-tuning . . 37
Algoritmo 5 Exemplo de chamada para a criação de um modelo com fine-tuning 38
Algoritmo 6 Exemplo de chamada da API do Chat Complations com um modelo ajustado via *fine-tuning* 38
Algoritmo 7 Exemplo de dado do *dataset* utilizado no processo de *fine-tuning* 53
Algoritmo 8 Chamada da API assim como é feito no código do sistema 54

LISTA DE ABREVIATURAS E SIGLAS

IA Inteligência Artificial

PLN Processamento de Linguagem Natural

GPT *Generative Pre-trained Transformer*

API *Application Programming Interface*

LLM *Large Language Model*

USMLE Provas de licenciamento médico dos Estados Unidos

ML *Machine Learning*

DL *Deep Learning*

RNA Redes Neurais Artificiais

DNN *Deep Neural Network*

RNN *Recurrent neural network*

SUMÁRIO

1 INTRODUÇÃO 11
1.1 OBJETIVOS 12
1.2 ESTRUTURA DO TRABALHO 13

2 APRENDIZADO DE MÁQUINA 14
2.1 Machine Learning 14
2.1.1 Aprendizado Supervisionado 15
2.1.2 Aprendizado Não Supervisionado 17
2.1.3 Aprendizagem por Reforço 18
2.1.4 Redes Neurais Artificiais 19
2.1.4.1 Estrutura e Funcionamento das Redes Neurais Artificiais 20
2.1.4.2 Por que usar Redes Neurais Artificiais? 20
2.1.5 *Deep Learning* 20
2.2 IA Generativa 22
2.2.1 *Transformers* 23
2.2.2 Large Language Model 28
2.2.3 Hiperparâmetros 29
2.3 API do GPT 30
2.3.1 Chat Completions *Application Programming Interface* (API) 32
2.3.2 Parâmetros da API 32
2.3.3 Fine-tuning da API do Chat Completions 35
2.4 CONSIDERAÇÕES SOBRE O CAPÍTULO 38

3 REVISÃO SISTEMÁTICA DA LITERATURA 40
3.1 CONSIDERAÇÕES SOBRE O CAPÍTULO 45

4 O SISTEMA DE GERAÇÃO DE LAUDOS 46
4.1 DEFINIÇÃO DE REQUISITOS 46
4.2 ARQUITETURA DO SISTEMA 47
4.3 IMPLEMENTAÇÃO 47
4.4 FINE-TUNING DA API 52
4.4.1 Dataset e Prompt 53
4.4.2 Chamada da API 54

5 RESULTADOS E DISCUSSÃO 55
5.1 Cenário 1 - Laudo Normal: 55
5.2 Cenário 2 - Bursite Moderada: 55

5.3 Cenário 3 - Artropatia degenerativa acromioclavicular: 56
5.4 Cenário 4 - Tendinopatias: 56
5.5 Avaliação do especialista 57

6 CONSIDERAÇÕES FINAIS 59
6.1 TRABALHOS FUTUROS 60

REFERÊNCIAS 61

APÊNDICE A – DADOS UTILIZADOS NO FINE-TUNING 64

1 INTRODUÇÃO

Com a chegada de novas tecnologias, e sua utilização na área médica, foi possível a digitalização de prontuários, criando assim uma grande base de dados de exames médicos (DASH *et al.*, 2019).

Além disso, com a evolução das tecnologias no equipamento de radiografias, que levaram a uma melhora na qualidade das imagens dos exames, foi possível visualizar estruturas anatômicas e anormalidades menores. Com isso, essa melhora na tecnologia elevou a quantidade de exames de imagem por paciente (WANG; SUMMERS, 2012).

Neste contexto, os dados armazenados nos sistemas da área da saúde podem ser analisados e assim auxiliar na tomada de decisão de um profissional da área, ou, de alguma outra inteligência. Porém, analisar uma larga quantidade de dados e aprender com isso, até alguns anos atrás era uma tarefa que apenas seres inteligentes poderiam fazer. O Radiologista é o médico especialista que é responsável pelo diagnóstico de um grande número de doenças, além de ter papel fundamental no tratamento de diversas patologias por meio de procedimentos de radiologia intervencionista. Este médico se apoia em exames de imagem como ressonância magnética, tomografia computadorizada, raio-x e ecografias para diagnosticar patologias nos pacientes. Os equipamentos de radiologia são altamente tecnológicos e hospitais e clínicas utilizam sistemas integrados de imagem e laudos. Os médicos radiologistas costumam elaborar seus laudos diretamente nos sistemas informatizados, digitando o texto do laudo, ou gravando em áudio. Grande parte dos laudos emitidos pelos especialistas possui diagnóstico de normalidade, com variações. A quantidade de exames analisados pelos radiologistas diariamente é muito grande, e parte da tarefa repetitiva poderia ser feita de forma automatizada (CADAMURO, 2021).

McCarthy *et al.* (2007) descreve a IA "como a ciência e engenharia de fazer máquinas inteligentes".

Segundo Zhang e Boulos (2023), ao longo dos anos, a IA impulsionou avanços revolucionários em diversos setores, e sua influência na área da saúde pode ser particularmente profunda. Com isso, devido ao seu ótimo desempenho na área médica, a utilização da IA se tornou algo corriqueiro e encorajado, como diz o autor no livro, com a melhora dos hardwares e softwares, juntamente com o avanço conseguido pelo IBM Watson a digitalização da medicina ficou mais fácil. Assim, a Inteligência Artificial (IA) no meio médico começou a avançar rapidamente (WANG; SUMMERS, 2012).

Atualmente, a IA na medicina segue dois caminhos, o físico e o virtual, o físico é aquele que utiliza robôs, e atuam, principalmente na prestação de cuidados e, em sua maior parte, apenas auxiliam profissionais da saúde. Já a forma virtual, é representada por sistemas inteligentes de machine learning e o deep learning. Estes são algoritmos que aprendem na medida que os

dados são processados, ou seja, quanto mais dados forem lidos, mais a máquina aprende, e melhor sua tomada de decisão será. Assim, eles são utilizados para fazer a análise dos dados, e auxiliar na tomada de decisões (HAMET; TREMBLAY, 2017).

O Processamento de Linguagem Natural (PLN) utiliza técnicas de *Machine Learning* para processar e analisar o texto livre, e é utilizado por todo o usuário de *smartphones*. Atualmente esta técnica é utilizada na medicina para o diagnóstico, pesquisas, atendimento ao paciente entre outras tarefas (LOCKE *et al.*, 2021).

Como exemplo de sistema de PLN temos o ChatGPT, que é um Transformador pré-treinado generativo (*Generative Pre-trained Transformer* (GPT)), geralmente chamado de GPT. Foi desenvolvido pela empresa OpenAI, e tem se mostrado uma ferramenta poderosa devido a sua capacidade de PLN. Seu uso no meio médico pode revolucionar a área, pois ele tem uma capacidade única de compreender e reproduzir uma linguagem humana, podendo assim ser usado na geração de textos para comunicar os pacientes, auxiliar na decisão médica, entre outras situações (ZHANG; BOULOS, 2023).

Neste contexto, este trabalho busca responder a seguinte questão de pesquisa: "Como o uso de IA Generativa pode apoiar o trabalho do médico radiologista na elaboração de laudos radiológicos?"

Com isso, uma tarefa repetitiva como analisar e fazer laudos de radiologia, que ocorre centenas de vezes por dia, pode ser automatizada graças à existência da inteligência artificial e de seus recursos, como o GPT e as PLNs. Mas, afinal, como podemos utilizar estes recursos na prática da medicina?

1.1 OBJETIVOS

O principal objetivo deste trabalho consiste no desenvolvimento de uma aplicação web que gere um laudo textual a partir de informações dadas pelo radiologista sobre o exame realizado por um paciente. Com base no objetivo geral, foram definidos os seguintes objetivos específicos:

- Conhecer o estado da arte em Aprendizado de Máquina a fim de definir a técnica mais adequada para a elaboração da solução;
- Definir linguagem de programação e quais APIs, bibliotecas e plataformas que serão utilizadas na aplicação;
- Integrar as APIs;
- Aplicar o fine-turing da API do GPT com textos de laudos de exames de imagem, para que assim, utilizando as técnicas de IA escolhidas anteriormente, “aprenda“ e descubra padrões para que consiga gerar laudos apartir de uma entrada;

- Desenvolver uma aplicação Web para a inserção do texto ou áudio do laudo que será analisada pelo software;
- Testar e validar o funcionamento do software.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado como segue:

- No Capítulo 2 são apresentados os fundamentos de Aprendizado de Máquina que foram utilizados no desenvolvimento da solução deste trabalho.
- O Capítulo 3 apresenta trabalhos relacionados à área de aplicação desta pesquisa.
- O Capítulo 4 detalha a descrição da solução e o processo de *fine-tuning*.
- O Capítulo 5 detalha os resultados obtidos e também a avaliação do profissional médico.
- Por fim, no Capítulo 6 são apresentadas as considerações finais.

2 APRENDIZADO DE MÁQUINA

Alan Turing iniciou o campo da inteligência artificial (IA) com o lançamento do paper *Universal Machine*, no qual ele apresenta a *Máquina de Turing*, que seria uma "máquina universal", ou seja, um modelo de computador que consegue modelar qualquer computador digital. Mais tarde, entre 1948 e 1950, com a publicação de seus papers semanais, ele dá início ao que mais tarde seria a disciplina científica da IA (MADEIRA, 2021).

Em 1950, Turing lança o artigo, "Máquinas computacionais e inteligência", na qual ele indaga: "Podem as máquinas pensar?", e assim, propõe os jogos de imitação, que mais tarde ficou conhecido como, teste de Turing, como teste de inteligência para máquinas (MADEIRA, 2021).

Turing (1996) descreve o "Jogo da Imitação", no seu artigo "Computadores e inteligência"da seguinte forma: "É jogado por três pessoas: um homem (A), uma mulher (B), e um interrogador (C), que pode ser de qualquer dos sexos. O interrogador permanece num quarto, separado dos outros dois. O objetivo do jogo, para o interrogador, é determinar, em relação aos outros dois, qual o homem e qual a mulher. Ele os conhece por rótulos X e Y e no fim do jogo dirá ou "X é A e Y é B", ou "X é B e Y é A"". Com base neste jogo, Turing (1996) perguntou-se se um dia, neste teste os personagens A e B ao invés de homem e mulher, fossem um ser humano e uma máquina, seria possível a máquina dar respostas tão convincentes a ponto de um ser humano não perceber que se trata de uma IA?

Turing previu que, com o avanço das tecnologias computacionais, no fim do século XX as máquinas teriam desenvolvido inteligência para passar no teste de Turing – na realidade, isso aconteceu pela primeira vez em 2014.

Neste contexto, a IA utiliza de métodos de aprendizado que permitem que ela evolua a cada dia, assim, na Seção 2.1 serão abordados alguns métodos de *machine learning* e seus algoritmos, para que seja possível o entendimento dos conceitos de IA generativa. Posteriormente na Seção 2.2 veremos a IA generativa, e por fim, na Seção 2.3 será abordado estudos da API do GPT.

2.1 MACHINE LEARNING

A Inteligência Artificial abrange diferentes algoritmos e campos que conseguem reproduzir características da inteligência humana, como compreender linguagem natural, reconhecer padrões, tomar decisões e aprender com a experiência (CASTELVECCHI, 2016).

Um destes campos é o Machine Learning, que segundo Samuel (1959) é o "campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programa-

dos".

Neste contexto, o *Machine Learning* (ML) utiliza diferentes abordagens de aprendizado para desenvolver as tarefas que lhe são propostas. Nas próximas seções serão abordados os tipos de ML existentes hoje.

2.1.1 Aprendizado Supervisionado

Segundo Santos (2021) as técnicas de aprendizado supervisionado são utilizadas em bases de dados que contém um conjunto de dados rotulados, ou seja, cada entrada é acompanhada da sua saída desejada (rótulo), que pode ser um valor numérico ou uma classe .

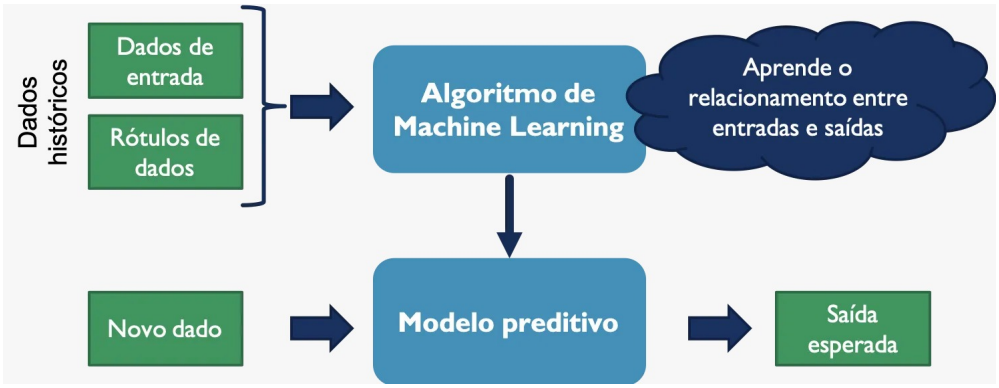
Os dados de entrada podem ser divididos em dois grupos, o conjunto de observações $(X1, X2, \ldots Xn)$, chamados de previsores e a saída esperada(Y), chamada de alvo. (SICSÚ; SAMARTINI; BARTH, 2023)

De acordo com Sicsú, Samartini e Barth (2023) os algoritmos supervisionados baseiam-se em encontrar a relação entre as variáveis previsoras e a variável alvo, após esta relação ser encontrada, ela pode ser aplicada para classificar ou prever novos casos a partir de novas entradas. O desempenho do algoritmo de aprendizagem supervisionado melhora na medida que o número de exemplos aumenta, ou seja, quanto mais dados para analisar, melhor serão as previsões e classificações.

Cunningham, Cord e Delany (2008) definiu que o objetivo do aprendizado supervisionado é concluir a partir de uma função $f : X \rightarrow Y$ a classificação do dataset. Na função x_i pertence a um conjunto X e $y_i \in Y$, como mostra a fórmula a seguir:

$$A_n = ((x_1, y_1), ..., (x_n, y_n)) \in (X \cdot Y)^n \tag{2.1}$$

Figura 1 – Hierarquia do Aprendizado Supervisionado

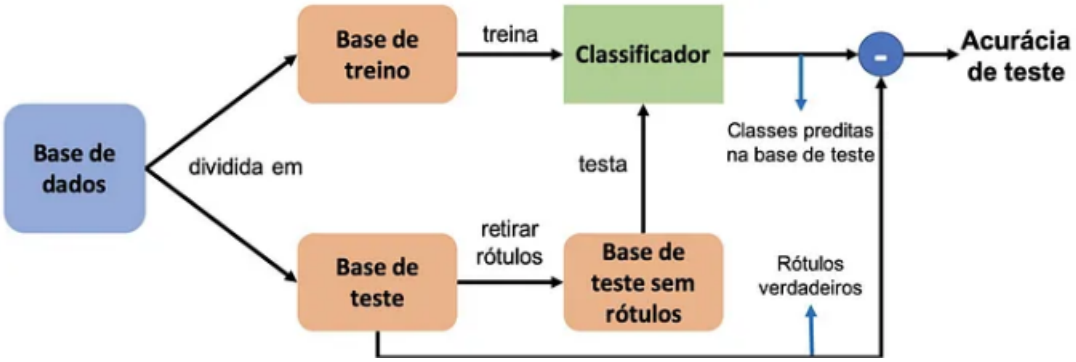


Fonte: (ESCOVEDO; KOSHIYAMA, 2020)

É comum no processo de treinamento ocorrer a separação dos dados de entrada em dois conjuntos, o conjunto de treinamento, que são os dados utilizados para construir o modelo, e o

conjunto de validação, que tem como função verificar como o modelo se comportaria em dados desconhecidos, desta forma, se necessário ele poderá ser ajustado para melhorar a acurácia do algoritmo. O conjunto de treinamento. Esta separação ocorre pois, segundo Fontana (2020), se os dados utilizados para testar o modelo forem os mesmos utilizados para treinamento, o algoritmo poderia armazenar o rótulo dos valores e sempre prever corretamente os valores. Neste contexto, o autor sugere que sejam separados em torno de 20-30% dos dados para fazerem parte do grupo de validação, e o resto dos dados fariam parte do grupo de treinamento. Fontana ressalva que, se o dataset for relativamente pequeno, o desempenho do algoritmo pode variar dependendo de quais dados forem selecionados para cada grupo.

Figura 2 – Fluxo resumido de problemas de Classificação



Fonte: (ESCOVEDO; KOSHIYAMA, 2020)

Algoritmos de Previsão: tem como objetivo prever o valor de uma variável alvo quantitativa Y em função das variáveis previsoras $X1, X2, \ldots Xn$. Por exemplo, consideremos uma amostra de apartamentos com valor Y conhecido e algumas características, sendo $X1$ = área útil, $X2$ = número de suítes, $X3$ = andar, $X4$ = idade em anos e $X5$ = vagas etc. A partir dessa amostra, utiliza-se um algoritmo de previsão para construir um modelo que permita prever preços de novos apartamentos com base nessas mesmas características (SICSÚ; SAMARTINI; BARTH, 2023).

A Tabela 1 representa a divisão dos dados de entrada que seriam utilizados em um algoritmo de previsão, onde os dados rotulados são representados pelos dados na parte branca da tabela, e o seu rótulo na parte cinza. O conteúdo da tabela é gerado de forma aleatória apenas para exemplificar a estrutura de dados utilizada no processo de aprendizado supervisionado, onde pode se perceber que cada conjunto de dados são apresentados em forma de pares ordenados, ou seja, cada entrada tem sua saída desejada.

Tabela 1 – Exemplo do funcionamento de um algoritmo de previsão

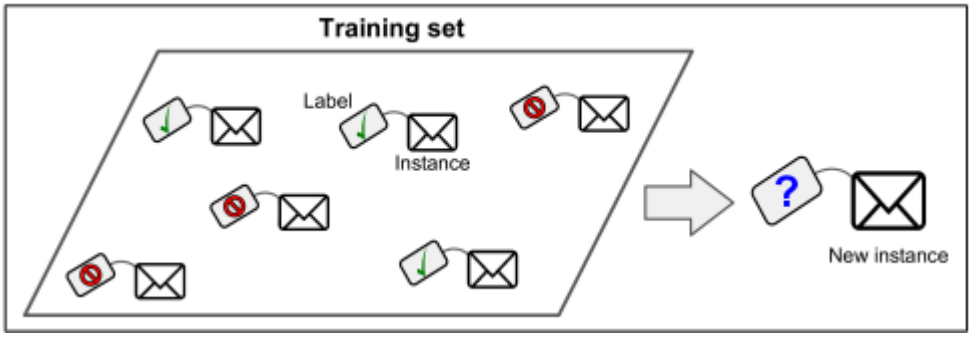
Previsoras	**Previsoras**	**Previsoras**	**Previsoras**	**Alvo**
x_1	x_2	...	x_p	Y
1	5	6	-4	X
2	3	8	6	7
3	2	11	-5	-4

Fonte: (SICSÚ; SAMARTINI; BARTH, 2023)

Algoritmos de Classificação: são utilizados para prever em qual categoria deve ser classificada uma nova observação. Pode ser binomial (duas categorias) ou multinomial (três ou mais categorias). Um exemplo de aplicação de algoritmos de classificação na área financeira é classificar, a partir de uma amostra de características sociodemográficas dos clientes, qual será o comportamento de um futuro tomador de crédito Y: bom ou mau pagador) (SICSÚ; SAMARTINI; BARTH, 2023).

A Figura 3 representa a entrada de dados para um algoritmo de classificação de e-mails, onde cada e-mail é acompanhado do seu rótulo (label), que representa se o e-mail é um spam ou um ham.

Figura 3 – Exemplo de um algoritmo de classificação de e-mails



Fonte: (GÉRON, 2022).

Alguns exemplos de uso de aprendizado supervisionado são: Sistemas de reconhecimento de fala em smartphones, como Siri, Cortana, entre outros, que treinam com a voz do usuário antes de começarem a funcionar; Sistemas de reconhecimento de letras cursivas (OCR), que após serem treinados para reconhecer a caligrafia de uma pessoa, convertem o texto para formato digital; Além disso, sistemas de e-mail que utilizam as informações fornecidas para filtrar novas mensagens, classificando-as como normais ou spam.(GABRIEL, 2022)

2.1.2 Aprendizado Não Supervisionado

Segundo Géron (2022) é chamado de não supervisionado pois, diferentemente dos algoritmos supervisionados, os dados de entrada não são rotulados, ou seja, a máquina recebe

apenas as observações da amostra e não recebe uma variável alvo para se direcionar. O objetivo do algoritmo é obter padrões de comportamento. O sistema tenta aprender sem um "professor" para direcioná-lo para a resposta certa. Desta forma, o algoritmo deve ser programado para aprender a identificar tais padrões.

A Tabela 2 apresenta um exemplo da base de dados que um algoritmo não-supervisionado receberia, onde não existem variáveis alvo (Y) apenas observações de amostra(X). O conteúdo da tabela é gerado de forma aleatória apenas para exemplificar a estrutura de dados utilizada no processo de aprendizado não supervisionado, onde pode se perceber que existem apenas conjuntos de dados, sem uma variável alvo.

Tabela 2 – Exemplo de um algoritmo de aprendizado não-supervisionado

$x_1 1$	x_2	...	x_p
2,3	3	...	- 100,51
3,8	10	...	89,32
11,5	8	...	27,65

Fonte: (SICSÚ; SAMARTINI; BARTH, 2023)

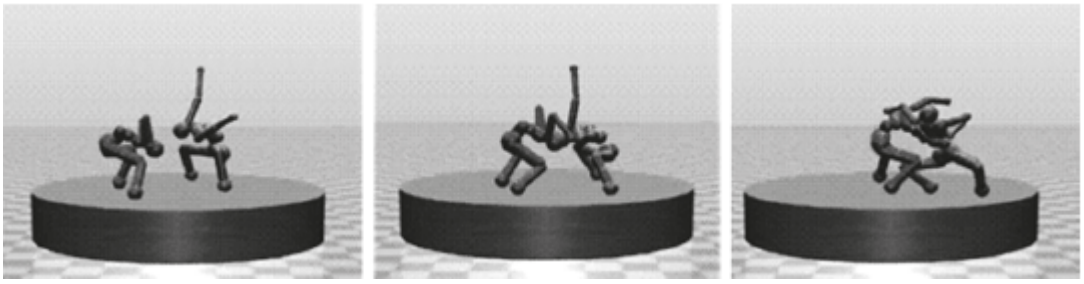
2.1.3 Aprendizagem por Reforço

A aprendizagem por reforço se desenvolve através de experiências sucessivas de fracasso e sucesso, refinando gradualmente a estratégia do algoritmo para atingir melhores resultados. Neste contexto, ela é um método de aprendizado baseado na interação contínua entre um agente e um ambiente, ou seja, o agente (ML) toma decisões observando os resultados e agindo de acordo com sistemas de recompensa e punição. O objetivo é maximizar a recompensa total ao longo do tempo, assim ajustando as ações com base nas experiências acumuladas. Nesse processo, o agente aprende quais ações são mais vantajosas através de tentativa e erro, ou seja, ele não é informado, de forma explícita, sobre quais ações tomar. Esse modelo de aprendizado imita os processos naturais de aprendizagem, onde não há um supervisor para guiar cada passo (SUTTON; BARTO, 2018).

De acordo com Sutton e Barto (2018), a aprendizagem por reforço é aplicada em vários campos, como a saúde, finanças, robótica, processamento de linguagem natural(PLN), jogos, entre outros. Eles destacam que os jogos são um excelente ambiente para agentes de aprendizado por reforço, pois permitem a exploração de diferentes cenários em um mundo virtual, onde o custo de exploração é acessível.

Um exemplo do funcionamento da aprendizagem por reforço é o jogo RoboSumo (Figura 4), no qual robôs lutam sumô controlados por ML e vão aprendendo conforme competem, tornando-se não só mais ágeis e inteligentes, mas também melhorando habilidades como equilíbrio e drible do oponente.

Figura 4 – Capturas de tela do jogo RoboSumo



Fonte: (NAST, 2017).

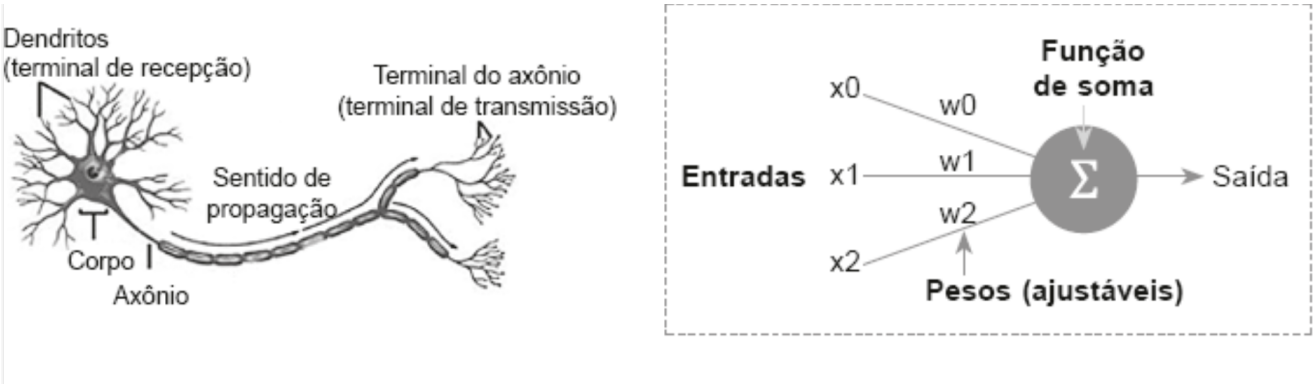
2.1.4 Redes Neurais Artificiais

As Redes Neurais Artificiais(Redes Neurais Artificiais (RNA)) fazem parte do campo de Machine Learning (ML), e possuem uma estrutura altamente flexível, permitindo a adaptação para os três tipos de aprendizado de máquina explicados anteriormente: supervisionado, não supervisionado e por reforço (GOODFELLOW; BENGIO; COURVILLE, 2016).

De acordo com (GABRIEL, 2022), as Redes Neurais processam informações de maneira semelhante ao cérebro humano. Elas consistem em muitos elementos de processamento altamente interconectados, chamados neurônios, que trabalham paralelamente para resolver problemas específicos. As RNAs aprendem através de exemplos, e para que o algoritmo funcione corretamente, os exemplos de treinamento devem ser selecionados com cuidado, pois as RNAs não podem ser programadas para executar tarefas específicas diretamente.

Nesse sentido, a Figura 5 mostra lado a lado os elementos de um neurônio biológico que serviram de inspiração para a criação dos neurônios artificiais, e um neurônio artificial.

Figura 5 – Esquema de um neurônio biológico (esquerda) e um neurônio artificial (direita)



Fonte: (GABRIEL, 2017).

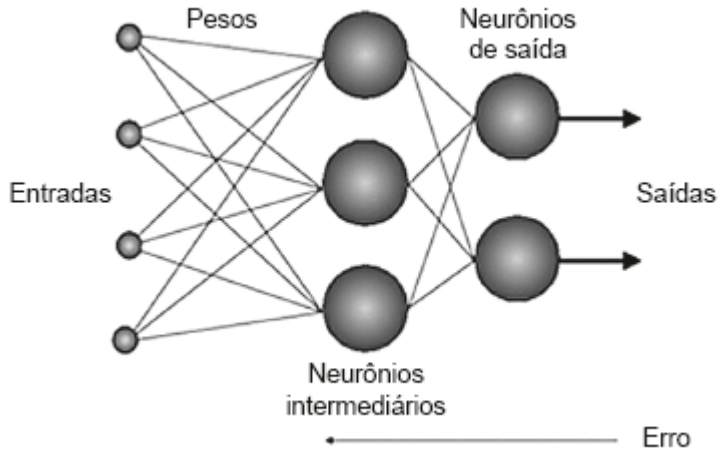
Nos neurônios biológicos, a comunicação ocorre através de impulsos recebidos pelos dendritos, processados no corpo do neurônio, e o resultado é transmitido para outras células da rede neural ou do corpo pelos axônios. Os neurônios artificiais imitam esse funcionamento, pois

cada um possui dois ou mais receptores de entrada com a função de perceberem determinados tipos de sinais, um corpo de processadores, que é responsável por um sistema de feedback que dependendo dos dados de entrada e saída modifica a sua própria programação, e uma saída para apresentar o resultado do processamento (GABRIEL, 2022).

2.1.4.1 Estrutura e Funcionamento das Redes Neurais Artificiais

Figura 6 - As RNAs são um sistema de neurônios artificiais conectados e divididos em três tipos de camadas, cada uma desempenhando um papel específico na rede. A camada de entrada recebe os dados de entrada. As camadas ocultas realizam mapeamentos não lineares entre a entrada e a saída, permitindo a modelagem de relações complexas. E a camada de saída, que produz o resultado final após o processamento dos dados através das camadas ocultas (BISHOP; NASRABADI, 2006; GOODFELLOW; BENGIO; COURVILLE, 2016).

Figura 6 – Rede neural multinível mostrando as camadas de entrada, oculta e de saída



Fonte: (GABRIEL, 2017).

2.1.4.2 Por que usar Redes Neurais Artificiais?

De acordo com Santos (2021) as RNAs, podem ser usadas para extrair padrões e detectar tendências muito complexas para serem notadas por qualquer indivíduo ou por outras técnicas computacionais. Após ser treinada, uma RNA pode ser considerada uma "especialista" na categoria de informação para a qual foi designada a analisar, e então, posteriormente, pode ser utilizada para fornecer projeções e novas situações de interesse.

2.1.5 *Deep Learning*

Quando uma Rede Neural possui duas ou mais camadas ocultas aninhadas umas às outras, ela é denominada Rede Neural Artificial Profunda (*Deep Neural Network* (DNN)). Os

neurônios presentes neste tipo de Rede Neural são mais avançados se comparados a Redes Neurais mais simples, ou seja, cada neurônio pode ter mais de uma função de ativação, diferente dos RNAs normais que contém apenas uma. Podem também utilizar operações mais avançadas como *convolutions*, por exemplo. As DNNs são capazes de automaticamente capturar padrões mais complexos e profundos nos dados não trabalhados, o que as torna particularmente poderosas para tarefas de reconhecimento de imagem, processamento de linguagem natural e outras aplicações avançadas de IA. Esta capacidade das redes é conhecida como *Deep Learning*.

Deep Learning(*Deep Learning* (DL)) é um conceito de *Machine Learning*(ML) que é fundamentado em Redes Neurais Artificiais Profundas. O DL utiliza DNNs para modelar representações complexas de dados, automatizando a detecção de correlações e padrões em grandes conjuntos de dados (HEINRICH *et al.*, 2021). Assim, o *Deep Learning* tem a capacidade de processar dados de alta dimensão em diversos domínios, desde dados unidimensionais, como sinais e textos, até dados multidimensionais, como imagens, vídeo ou áudio (LECUN; BENGIO; HINTON, 2015; HEINRICH *et al.*, 2021).

Avanços recentes no DL permitem que dados de fontes diferentes (imagens, textos, áudio) sejam processados juntamente, isso é chamado de cross-modal learning, que é a capacidade de aprender utilizando diferentes fontes sensoriais. Isto é útil em um ambiente onde o conteúdo vem de diversas fontes diferentes, como por exemplo, um e-commerce, onde as informações dos produtos geralmente são representadas por imagens e uma breve descrição, assim, as recomendações geradas por processamento de dados de diferentes fontes devem ser melhores do que aqueles gerados por apenas uma fonte(descrição ou imagem do produto) (BASTAN; RAMISA; TEK, 2020).

Banh e Strobel (2023) afirma que esses avanços têm gerado uma ampla gama de aplicações em diferentes áreas, desde o bem-estar social, como aprimoramento dos cuidados de saúde, sustentabilidade ambiental e até os e-comerces. Neles, o *Deep Learning* pode otimizar preços, atuar como sistemas de recomendação, prever demandas e identificar avaliações falsas de consumidores

Nos próximos parágrafos serão explicados alguns dos conceitos importantes do *Deep Learning*.

Recurrent neural network (RNN): São projetadas explicitamente para estruturas de dados sequenciais, onde a ordem dos elementos é importante, como dados de séries temporais, sequências de eventos e linguagem natural, isso acontece pois, elas contém uma "memória interna", que as permite relembrar dados processados anteriormente e utilizar estes dados para processar novas informações dentro de um contexto. Assim, por causa destas características ela se torna ideal para tarefas como PLN, geração de texto, reconhecimento de fala, entre outros (LECUN; BENGIO; HINTON, 2015).

Distributed representation: Normalmente, nos algoritmos de DL, a informação é repre-

sentada por vetores densos de valor real e baixa dimensão. Desta forma, comparado a outros esquemas de representação de dados convencionais, esta maneira é capaz de representar os dados de um modo mais compacto e "leve". Por exemplo, nas tarefas de PLNs, consegue transformar palavras, frases e parágrafos em representações numéricas dentro de um vetor, assim, palavras que aparecem em contextos semelhantes, são posicionadas perto uma das outras nestes vetores. Isso torna possível o desenvolvimento de algumas funcionalidades, como um algoritmo de perguntas e respostas, análise de sentimento e reconhecimento de entidade nomeada (LIU; LIN; SUN, 2023).

Autoencoder: Consiste em duas etapas, em que a primeira etapa, encoding, os dados são comprimidos em uma representação de baixa dimensionalidade, após esta etapa, vem a etapa de decoding, onde a rede neural tenta reconstruir o dado original. Desta maneira, a RNA é forçada a manter informações significativas na representação latente, ao mesmo tempo que desconsidera ruídos irrelevantes (GOODFELLOW; BENGIO; COURVILLE, 2016).

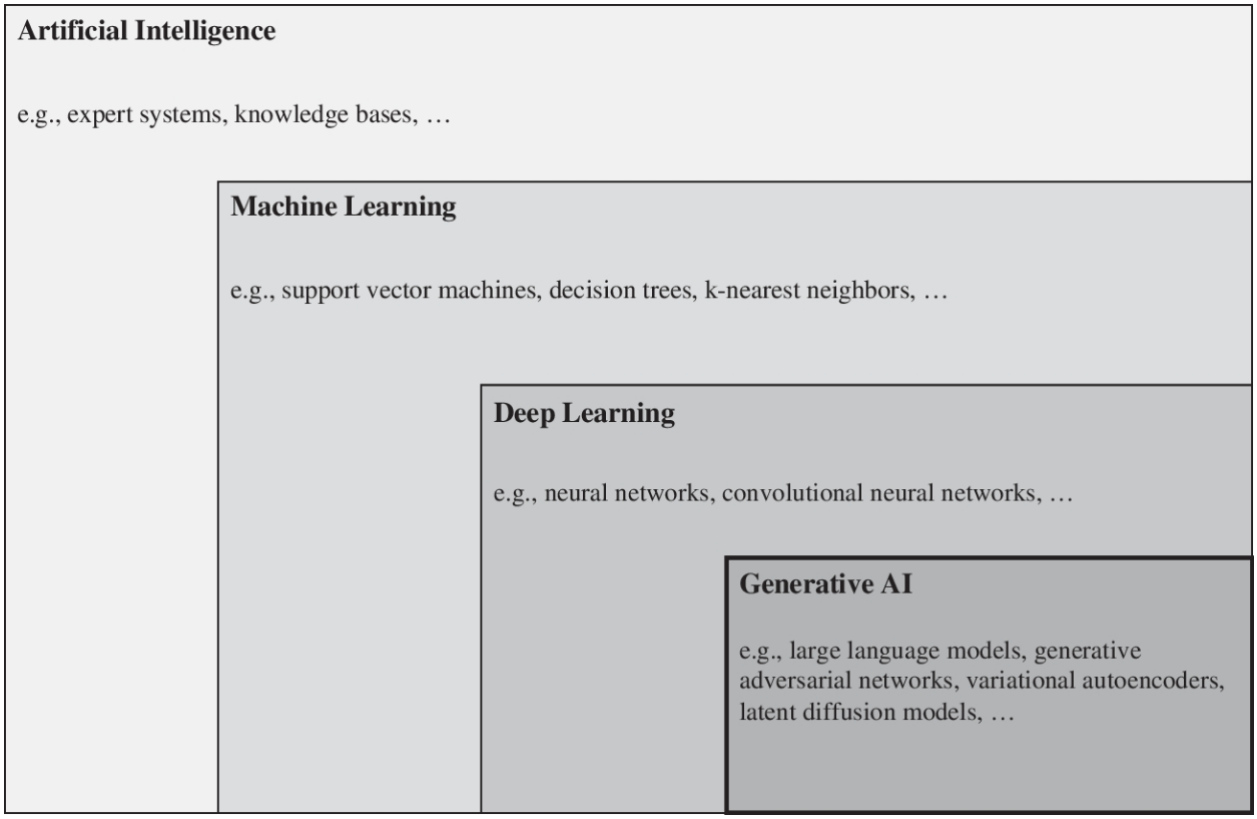
2.2 IA GENERATIVA

A IA Generativa engloba técnicas de computação que são capazes de gerar "novos" textos, imagens, áudios e até mesmo responder perguntas. Atualmente, esta tecnologia está muito presente na vida das pessoas, revolucionando a maneira com que trabalham e se comunicam. Alguns exemplos de softwares que são utilizados e se baseiam em IA Generativa são o GPT-4, Copilot e o Dall-E 2 (BANH; STROBEL, 2023).

Ela é baseada em modelos generativos, que tem uma diferença matemática de modelos discriminativos. Os modelos generativos são utilizados para auxiliar nas tomadas de decisões baseadas em dados, enquanto os modelos discriminativos aprendem a mapear diretamente os dados de entrada X para as classes de saída Y. O objetivo é encontrar a probabilidade condicional $P(Y \mid X)$, ou seja, a probabilidade de uma classe Y dada uma entrada X. Já os modelos generativos tentam modelar como os dados são gerados, aprendendo a distribuição conjunta $P(X, Y)$ dos dados de entrada X e as classes de saída Y. Ou seja, eles aprendem a probabilidade conjunta de X e Y, o que permite gerar novas amostras de dados que seguem a mesma distribuição (BISHOP; NASRABADI, 2006). Com base nisso, um modelo de IA Generativa se refere a modelos generativos que são instanciados com uma arquitetura de ML e, portanto, pode criar novas amostras de dados com base em padrões aprendidos.

A IA Generativa se encontra dentro do campo de Deep Learning, como mostra o diagrama da Figura 7.

Figura 7 – Diagrama da Inteligência Artificial



Fonte: (BANH; STROBEL, 2023).

2.2.1 *Transformers*

Esta arquitetura de *Deep Learning* foi introduzida por Vaswani *et al.* (2017) no artigo *"Attention is All you Need"*, e serve como base para muitos processos de PLN. Ele adota um mecanismo de autoatenção que pondera diferentemente a importância de cada parte dos dados de entrada. Ele utiliza este mecanismo para capturar dependências de longo alcance nos dados, ou seja, consegue capturar relações ou interações entre elementos em uma sequência que estão distantes uns dos outros. Esta capacidade de capturar dependências de longo alcance as torna uma ótima opção para o processamento de dados em larga escala

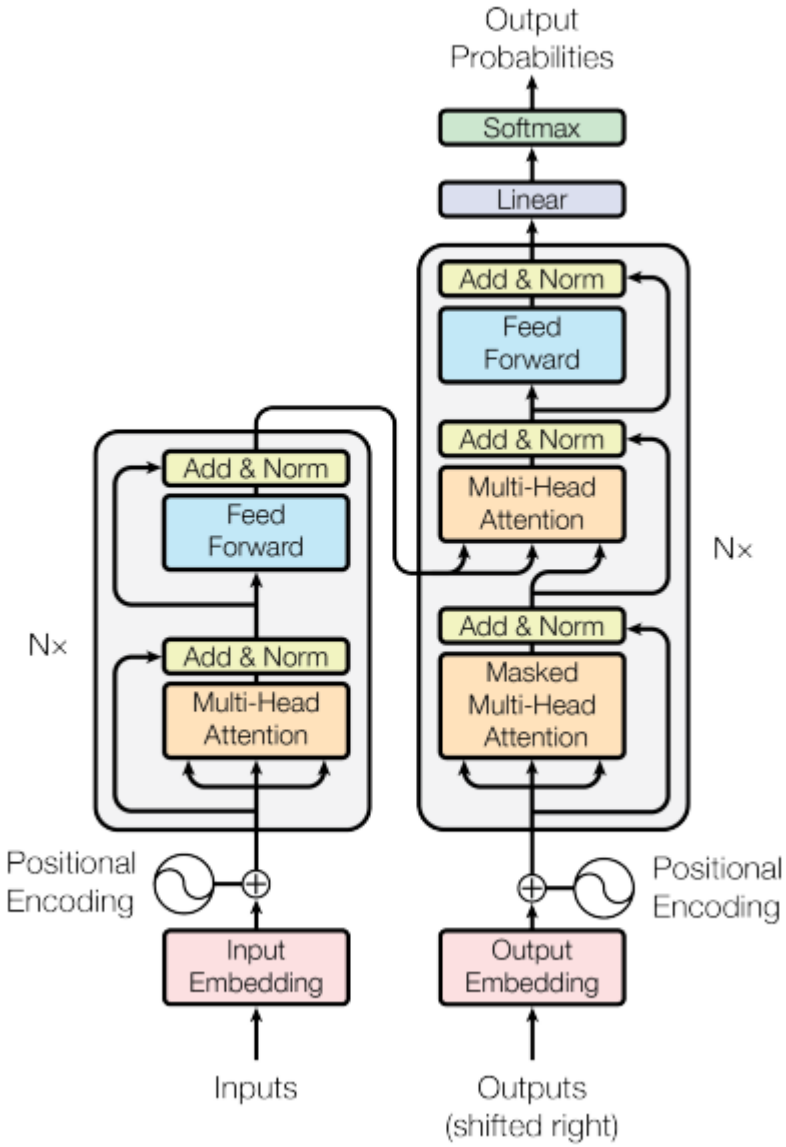
Os Transformers foram projetados para processar de uma só vez dados de entrada sequenciais, isso acontece pois o seu mecanismo de autoatenção fornece contexto para qualquer posição na sequência da entrada (SIEBERS; JANIESCH; ZSCHECH, 2022). Por exemplo, para responder à entrada “Qual a cor do céu”, o modelo de *transformers* utiliza uma representação matemática interna que identifica a relevância e a relação entre as palavras “cor”, “céu” e “azul”. Ele usa esse conhecimento para gerar a saída: "O céu é azul".

Segundo Vaswani *et al.* (2017) as redes neurais tradicionais, que lidam com sequências de dados, geralmente usam um padrão de arquitetura de codificador-decodificador onde o codificador mapeia uma sequência de entrada $(x_1, \ldots, x_n)$ para uma representação matemática

compacta da mesma $(z_1, \ldots, z_n)$. Assim, a etapa de decodificação lê esta sequência e gera sequencialmente uma saída $(y_1, ..., y_n)$. Como cada palavra é processada uma após a outra, em algumas sequências de larga escala, além de perder alguns detalhes mais sutis, o processo pode ser lento.

Os *transformers* utilizam uma estrutura codificador-decodificador incorporando um mecanismo de autoatenção, ou seja, o modelo examina diferentes partes da sequência de uma só vez e determina quais partes são mais importantes, o que aumenta a velocidade de processamento e reduz a possibilidade de perda de detalhes.

Figura 8 – Estrutura do *Transformer*



Fonte: (VASWANI *et al.*, 2017).

A etapa de ***Input Embedding*** da Figura 8 é responsável por dividir a sequência de en-

trada em uma série de *tokens*, e após isto, transformar esta série de *tokens* em uma sequência vetorial matemática que carrega as representações matemáticas das informações semânticas e sintáticas. Estas representações matemáticas tornam a entrada processável por algoritmos de software o que deixa possível posteriormente adquirir atributos através dos processos de treinamento.

Estes vetores podem ser visualizados como uma série de coordenadas em um espaço n-dimensional. Por exemplo, um vetor de espaço bidimensional (x, y) onde x representa o valor alfanumérico da primeira letra da palavra e y representa sua categoria. O valor da palavra "amora" no vetor seria $(1, 2)$ porque começa com a letra 'a' e está na categoria fruta e a palavra "banana"teria o valor $(2, 2)$ porque começa com a letra 'b' e também está na categoria fruta. Assim, o vetor (x, y) informa à rede neural que as palavras "amora"e "banana"pertencem à mesma categoria. Desta forma, esta etapa mapeia os dados em vetores de n-dimensões contendo as informações (atributos da gramática, significado e uso das palavras) das entradas em forma numérica, esta maneira de representar os dados permite que posteriormente sejam feitos os cálculos das relações entre os *tokens* em termos matemáticos. Isso é importante por exemplo no entendimento do modelo de linguagem humana.

A etapa de ***Positional Encoding*** da Figura 8 é responsável por atribuir as informações da ordem relativa ou absoluta dos *tokens* na sequência, ou seja, ela é responsável por adicionar a ordem em que os *tokens* estavam na sequência gerada pela etapa anterior. Esta etapa é importante pois é o que permite ao modelo utilizar posteriormente a ordem dos *tokens* para entender o contexto da sequência.

Para que os dados da ordem gerados nesta etapa possam ser adicionados aos gerados pelas etapas de *embedding* eles têm de ter a mesma dimensão de vetores. Ou seja, a dimensão do vetor gerado na camada de *embedding* deve ser igual ao vetor gerado na camada de *positional encoding*, esta dimensão é chamada de dimensão do modelo.

Para determinar esta ordem são utilizadas funções de *seno* e *cosseno* de diferentes frequências:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.2}$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.3}$$

Onde pos é a posição do *token* na sequência, i é o índice da posição do vetor de *encoding* e d_model é a dimensão do modelo. Os valores presentes no vetor desta etapa são *sinusoids*, pois esta maneira permite que o modelo facilmente aprenda a atender posições relativas, já que para qualquer deslocamento fixo k PE_{pos+K} pode ser representado como uma função linear PE_{pos}.

A arquitetura do *Transformer* é composta por duas partes principais: o **codificador (*encoder*)** e o **decodificador (*decoder*)**, ambos utilizando camadas empilhadas de autoatenção e camadas totalmente conectadas ponto a ponto. A metade esquerda da imagem representa o codificador enquanto a metade direita representa o decodificador.

O ***encoder*** consiste em uma pilha de N camadas idênticas, sendo que cada camada é formada por duas subcamadas principais. A primeira é a subcamada de *Multi-Head Attention*, responsável por capturar relações de dependência entre diferentes posições da entrada. A segunda é uma subcamada composta por uma *Rede Feed-Forward totalmente conectada posição a posição*, encarregada de processar cada posição independentemente. O funcionamento detalhado dos elementos que compõem essas camadas será abordado no texto a seguir.

Em cada uma das subcamadas é empregada uma **Conexão Residual (*Residual Connection*)** seguida por uma **Normalização de Camada (*Layer Normalization*)**.

O ***decoder*** é composto por uma pilha de N camadas idênticas, cada uma contendo três subcamadas. Além das duas subcamadas presentes no *encoder*, o *decoder* inclui uma terceira subcamada, denominada *Masked Multi-Head Attention*, posicionada logo acima do *positional encoding*. Essa subcamada desempenha um papel fundamental ao garantir que as previsões realizadas para a posição i dependam exclusivamente das saídas conhecidas nas posições anteriores a i.

Uma função de atenção pode ser descrita como um mecanismo que mapeia uma consulta e um conjunto de pares chave-valor para uma saída. Tanto a consulta, quanto as chaves, os valores e a saída são representados como vetores. O resultado dessa função é uma soma ponderada dos valores, na qual os pesos são calculados por meio de uma função de compatibilidade. Essa função avalia o grau de correspondência entre a consulta e as chaves, determinando assim a relevância de cada valor para a consulta em questão.

A ***atenção multi-cabeça (multi-head attention)*** da Figura 8 permite ao modelo focar em diferentes partes da entrada de maneira simultânea, o que melhora significativamente a capacidade de aprender e capturar padrões complexos nas sequências de dados.

Seu funcionamento acontece em cinco etapas:

1. **Divisão em várias cabeças:** é dividido a atenção em várias "cabeças", onde cada cabeça aplica a função de atenção em diferentes partes da entrada.
2. **Projeções lineares:** As consultas, chaves e valores são projetados linearmente h vezes para dimensões menores. Por exemplo, se a dimensão do modelo (d_{model}) for 512 e houver 8 cabeças de atenção ($h = 8$), então cada projeção terá a dimensão $d_k = d_v = d_{model}/h = 64$.
3. **Atenção em Paralelo:** Cada cabeça executa a função de atenção separadamente e em paralelo, isso resulta em uma saída para cada cabeça.

4. **Concatenação:** As saídas de todas as cabeças são concatenar o que resulta em um vetor de dimensão hxd_v.

5. **Projeção Final:** A saída gerada pela concatenação é projetada linearmente de volta para a dimensão original (d_{model}) usando uma matriz de projeção W_0.

Os *transformers* utilizam ***multi-head attention*** de três maneiras diferentes:

1. Na camada de atenção "Codificador-Decodificador" as consultas vem da saída da camada anterior do decodificador, as chaves e valores vem da saída do codificador. Isso permite que o decodificador acesse informações de qualquer parte da sequência de entrada original, o que é crucial para gerar uma tradução precisa ou para entender contextos complexos.

2. Na camada de autoatenção do codificador as consultas, chaves e valores vem do mesmo lugar que é a camada anterior do codificador, e cada posição na sequência de entrada pode "chegar" a todas as outras posições, permitindo que o modelo entenda a relação entre palavras em diferentes partes da frase.

3. Na camada de autoatenção do decodificador as consultas, chaves e valores vêm da camada anterior do decodificador. Cada posição no decodificador pode considerar todas as posições anteriores (e a atual), mas não as futuras. Isso é importante para garantir que o modelo gere a sequência de saída de maneira correta e ordenada.

A ***Rede Feed-Forward Completamente Conectada Posição-a-Posiçã***o tem como função aplicar uma transformação linear e uma função de ativação para cada posição na sequência de forma independente das outras posições. Esta função de ativação introduz não linearidades no modelo. Esta etapa ajuda o modelo a aprender representações mais complexas e robustas dos dados.

As ***conexões residuais*** tem como objetivo adicionar a entrada original da subcamada ao resultado gerado pela mesma, pois, este processo ajuda a mitigar o desaparecimento de gradientes, permitindo que o gradiente flua diretamente através da rede. Desta forma, ela funciona como um atalho permitindo o fluxo de informações de uma parte da rede para outra, ignorando determinadas operações intermediárias.

Já a ***normalização de camada*** tem como objetivo melhorar a estabilidade e eficiência do treinamento da rede neural, ela faz isso estabilizando as ativações para que elas tenham uma média próxima de zero e uma variância constante, ou seja, ela mantém estes valores numéricos dentro de uma determinado intervalo, tornando assim o treinamento mais estável, com isso, a rede neural é treinada de maneira mais eficiente resultando em uma convergência mais rápida.

Com isso a saída de cada subcamada pode ser representada por:

$$LayerNorm(x + Sublayer(x)) \tag{2.4}$$

Onde $Sublayer(x)$ é a função implementada pela subcamada e x é o valor de entrada original processado pela subcamada.

Resumidamente, os *transformers* adotam vários blocos de codificação e decodificação empilhados em conjuntos. O bloco do *codificador* tem como função transformar a sequência de entrada em representações que capturam informações contextuais de cada *token* individualmente em relação aos outros tokens na sequência. Já o bloco do *codificador* tem a tarefa de gerar a sequência de saída token por token, cada um levando em consideração apenas os *tokens* anteriores e as representações internas do *codificador*. Este design modular e repetitivo permite ao *Transformer* capturar relações complexas e de longo alcance na sequência de entrada, tornando-o extremamente eficaz em tarefas de processamento de linguagem natural e outras aplicações de aprendizado de máquina.

Os *Blocos lineares* e o *softmax* são as últimas etapas do modelo da Figura 8, estas camadas são responsáveis por fazer uma previsão concreta, ou seja, são responsáveis por prever a possibilidade de um próximo *token* em uma sequência. Estas camadas processam as saídas geradas pela camada de decodificação. Os *Blocos Lineares* tem como função transformar as representações internas complexas em previsões concretas, que podem ser interpretadas e utilizadas. O resultado gerado por essa camada é um conjunto de pontuações (*logits*) para cada *token* possível. A função da camada *softmax* é converter estas pontuações (*logits*) em uma distribuição probabilística normalizada. Cada parte da saída do *softmax* representa a confiança do modelo em uma determinada classe ou *token*, assim é possível prever os próximos *tokens* na sequência.

2.2.2 Large Language Model

Generative pre-trained transformers (GPT) baseiam-se na arquitetura de transformers e foram treinados com grandes conjuntos de dados não rotulados (BROWN *et al.*, 2020). Porém, devido ao seu tamanho (quantidade de parâmetros), GPTs treinados por arquivos de texto entram na categoria de *Large Language Model* (LLM)s (SCHRAMOWSKI *et al.*, 2022). O objetivo destas redes neurais é gerar textos coerentes e com contextos que se assemelham a textos gerados por humanos, prevendo qual *token* tem maior probabilidade de ocorrer após os *tokens* anteriores de uma frase. Este modelo serve de base para ferramentas de IA conversacional, como o ChatGPT (TEUBNER *et al.*, 2023). Além disso, pode ser utilizada para gerar escrita, textos e códigos de programação (COOPER, 2023; LUND *et al.*, 2023).

Segundo Feuerriegel *et al.* (2024) as LLMs são modelos de Redes Neurais que tem a função de gerar e modelar textos, este método combina 3 características:

1. Utiliza uma rede neural de larga escala (*transformer*).
2. Esta rede neural é pré-treinada através de auto-supervisão no qual tarefas auxiliares são projetadas para aprender uma representação da linguagem natural sem risco de *overfitting* (por exemplo, previsão da próxima palavra).
3. O pré-treinamento utiliza uma base de dados de larga escala (Wikipedia, Reddit).

Com isso, estes modelos de linguagens podem ser treinados com conjuntos de dados personalizados para tarefas específicas, por exemplo, um gerador de linguagem natural ou até mesmo um respondedor de perguntas.

Os modelos de linguagem que contam com bilhões de parâmetros são chamados de *Large Language Models*, o GPT se encontra dentro desta categoria (BROWN *et al.*, 2020).

2.2.3 Hiperparâmetros

Nos modelos de IA Generativa, são utilizados diversos hiperparâmetros para otimizar o treinamento e a geração de texto. Esses hiperparâmetros servem para ajustar o desempenho e a capacidade do modelo. A seguir, serão apresentados os principais hiperparâmetros, focando principalmente nos hiperparâmetros utilizados no modelo GPT.

- **Número de Camadas (*Layers*):** O número de camadas determina a profundidade da rede neural. Ou seja, quanto mais camadas melhor vai ser a capacidade do modelo capturar padrões mais complexos, o que faz com que a sua capacidade de compreensão de sequências de entrada e saída aumente. No GPT, estas camadas são compostas por blocos de atenção e *feedforward*, cada um contribuindo para a compreensão contextual e a geração coerente de texto (VASWANI *et al.*, 2017; BROWN *et al.*, 2020).
- **Tamanho do Lote (*Batch Size*):** De acordo com Goodfellow, Bengio e Courville (2016), este parâmetro se refere ao número de exemplos de treino processados antes de atualizar os pesos do modelo. Ou seja, quantos dados o modelo processa de uma vez antes de atualizar os seus pesos.
- **Taxa de Aprendizado (*Learning Rate*):** Tem como objetivo definir a velocidade que o modelo ajusta seus pesos durante o treinamento, ou seja, define a velocidade em que o modelo ajusta seus parâmetros em resposta ao erro calculado no processo de aprendizado. Uma taxa de aprendizado alta acelera o treinamento, o que pode trazer instabilidade ao modelo. Uma taxa de aprendizado baixa é mais lenta e mais estável (KINGMA; BA, 2017).
- **Número de Épocas (*Epochs*):** Se refere ao número de vezes que o modelo percorre todo o conjunto de dados de treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Dropout e Regularização:** Técnicas para prevenir *overfitting* (quando o modelo perde a capacidade de generalização), como *dropout* (técnica utilizada para forçar o modelo a aprender representações mais robustas e menos dependentes de combinações específicas de neurônios) e penalizações de pesos (L1, L2) (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Função de Custo (*Loss Function*):** Função que verifica a distância que o modelo está da previsão correta (GOODFELLOW; BENGIO; COURVILLE, 2016).

- **Tamanho do Contexto (*Context Window*):** Define a quantidade máxima de tokens que o modelo pode considerar em uma única sequência (BROWN *et al.*, 2020).

- ***Top-k Sampling:*** Métodos de controle de geração de texto, limitando as palavras selecionadas (BROWN *et al.*, 2020).

- **Temperatura:** Segundo Brown *et al.* (2020), este parâmetro serve para controlar a aleatoriedade das saídas do modelo.

- **Número de Parâmetros (*Parameter Count*):** Refere-se à quantidade de valores ajustáveis que o modelo possui. Esses parâmetros são ajustados durante o processo de treinamento e auxiliam o modelo a aprender a mapear as entradas para as saídas desejadas (VASWANI *et al.*, 2017; BROWN *et al.*, 2020).

- **Tokenização:** Segundo Sennrich, Haddow e Birch (2016) define a maneira e o nível de granularidade dos tokens, afetando diretamente o desempenho e a capacidade de generalização do modelo.

- **Dimensão do Embedding (*Embedding Size*):** O *embedding* é uma representação vetorial dos tokens de entrada e saída. *Embeddings* de alta dimensionalidade capturam mais nuances semânticas, resultando em uma compreensão mais precisa do texto.

2.3 API DO GPT

Generative Pre-trained Transformer (GPT), é uma família de modelos de linguagem desenvolvidos pela organização de pesquisa em Inteligência Artificial OpenAI[1]. Ele é baseado na arquitetura, que faz parte dos modelos de aprendizado profundo (Deep Leaning), Transformers, que foi introduzida por Vaswani *et al.* (2017) no artigo "Attention is ALL You Need" Radford *et al.* (2018).

O primeiro modelo, o GPT-1, foi lançado em junho de 2018, nele foram usados 4,5 GB de textos para o seu treinamento, e possui 117 milhões de parâmetros Radford *et al.* (2018). Em fevereiro de 2019 o GPT-2 foi lançado. Esse modelo é capaz de gerar texto coerente e

[1] https://openai.com/pt-BR/

de alta qualidade em uma ampla variedade de tópicos. Ele possui 1,5 bilhões de parâmetros e para seu treino foram utilizados 40GB de texto (RADFORD *et al.*, 2019). O GPT-3 foi anunciado em junho de 2020, e é capaz de realizar, com um alto grau de precisão, tarefas como análise de sentimentos, preenchimento automático de textos, entre outras. Possui 175 bilhões de parâmetros e foi treinado com 570 GB de textos. Em novembro de 2022, esta última versão foi atualizada e aprimorada para a versão GPT-3.5, a qual posteriormente foi utilizada para o lançamento do ChatGPT. Se popularizou pela sua capacidade de responder a perguntas, fornecer informações, escrever textos criativos e auxiliar na resolução de problemas em linguagem natural (CHEN; BALAN; BROWN, 2023). Por fim, a versão mais recente, o GPT-4, foi lançada em 2023 e conta com 170 trilhões de parâmetros. A grande diferença entre esta versão e as outras é que além de permitir a entrada e análise de imagens e documentos, ele fornece um controle de comportamento nas respostas, ou seja, pode ser "programado" para responder as perguntas com diferentes personalidades (OPENAI, 2023).

O GPT foi treinado para entender a linguagem natural, códigos e imagens. Este modelo permite saídas de texto em resposta às suas entradas. A API do GPT pode ser utilizada virtualmente para qualquer tarefa, porém nesta seção será abordado principalmente a parte de geração de texto via API utilizando a linguagem de programação *Python*.

Usando os modelos de geração de texto da OpenAI, é possível criar aplicações para escrever códigos de programação, responder perguntas sobre uma base de conhecimento, analisar textos, dar ao software uma interface de linguagem natural, traduzir idiomas, entre outras tarefas.

O primeiro passo para utilizar as funções do GPT é criar uma conta na OpenAI, após a conta ser criada, deve-se obter uma chave API que será utilizada para fazer as chamadas da mesma no código.

Na utilização da API, é cobrado um valor para cada token utilizado, os tokens são pedaços de palavras utilizados no processamento de linguagem natural. Estes valores variam de acordo com o modelo utilizado, por exemplo, no modelo GPT-4 1 milhão de tokens de entrada custam US$30,00 e a mesma quantidade de tokens gerados custa US$60, já para o GPT-3.5-turbo-instruct, os valores já são mais amigáveis, sendo US$ 1,50 a cada 1 milhão de tokens de entrada US$2,00 a cada 1 milhão de tokens gerados (OPENAI, 2024).

Para implementar a API em programas, podem ser utilizadas diversas linguagens de programação, algumas, como *Python* e *Javascript* (*Node.js*). O uso é facilitado pela existência de bibliotecas oficiais da OpenAI. Já em outras linguagens, como *Ruby*, *PHP* e *C#*, a chamada da API ocorre ao enviar um objeto JSON contendo os parâmetros para a url do endpoint desejado.

A API do Chat Completions, foi implementada principalmente em Python, pois esta linguagem fornece uma vasta disponibilidade de bibliotecas especializadas em aprendizado de máquina e inteligência artificial.

2.3.1 Chat Completions API

Segundo a plataforma da OpenAI[2], para utilizar a API da OpenAI para geração de texto, deve-se enviar uma requisição ao endpoint Chat Completions (nome do endpoint), fornecendo as entradas e a chave da API. Esse endpoint alimenta o GPT e oferece uma maneira intuitiva de enviar texto como entrada e obter uma resposta gerada por um modelo GPT. Endpoint é o que define um ponto de acesso onde recursos podem ser solicitados ou manipulados, geralmente é representado por meio de uma URL ou a importação de uma biblioteca.

O Chat Completions funciona da seguinte maneira: é enviado uma lista de entradas, que são processadas pelo modelo, resultando em uma mensagem gerada. Essa abordagem permite que um "*prompt*"inicial fornecido receba uma resposta relevante e contextualizada gerada pelo modelo.

2.3.2 Parâmetros da API

Segundo a documentação da OpenAI[3], os parâmetros de uma API são o que define o que deve ser retornado pela mesma. Estes parâmetros permitem personalizar a requisição feita para que, desta forma, a API possa gerar o retorno desejado pelo usuário. Assim, a API do GPT utiliza alguns parâmetros para esta personalização, neste capítulo serão explicados para que servem estes parâmetros:

messages (array, obrigatório): Deve ser uma lista de dicionários de mensagens, onde cada objeto (*"system"*, *"user"*, ou *"assistant"*) tem um papel e um conteúdo:

- Normalmente, o primeiro parâmetro a ser passado pelo *"messages"* é o *"system"*, que define a forma como o Chat irá se comportar. Este parâmetro não é obrigatório, e caso ele não seja enviado, o comportamento do modelo será o padrão, ou seja, será como se o parâmetro fosse enviado com a mensagem "Você é um assistente útil".

- No parâmetro *"user"* é enviado comentários ou perguntas para a API responder. No parâmetro *"assistant"*, é *"guardado"* as respostas prévias geradas pelo Chat, porém pode ser enviado também pelo usuário, para dar exemplos do comportamento que a resposta deve ter.

- Incluir o histórico de conversas é importante quando as instruções do usuário se referem a mensagens anteriores. No exemplo abaixo, a pergunta final do usuário *"Onde foi jogado?"* só faz sentido no contexto das mensagens anteriores sobre a *World Series de 2020*. Como os modelos não possuem memória de solicitações anteriores, todas as informações relevantes devem ser fornecidas como parte do histórico de conversas em cada

[2] https://platform.openai.com/docs/api-reference/making-requests
[3] https://platform.openai.com/docs/api-reference/chat/create

solicitação. Se uma conversa não puder caber no limite de *tokens* do modelo, ela precisará ser encurtada de alguma forma.

model (string, obrigatório): ID do model que a API utiliza, os disponíveis são: gpt-4, gpt-4-turbo-preview, gpt-4-vision-preview, gpt-4-32k, gpt-3.5-turbo e gpt-3.5-turbo-16k.

frequency_penalty (número ou null, opcional, default:0): um número entre -2.0 e 2.0. Quanto mais perto de 2.0 menos *tokens* repetidos existirão nas respostas, porém a qualidade das mesmas podem cair.

logit_bias (map, opcional, default: null): modifica a probabilidade de *tokens* especificados aparecerem na resposta.

logprobs (boolean ou null, opcional, default: false): Define se deve ou não devolver as probabilidades logaritmas dos *tokens* de saída.

top_logprobs(inteiro ou null, opcional, default: null): Um número inteiro entre 0 e 20 que especifica o número de *tokens* com maior probabilidade de retornar em cada posição de *token*, cada um com uma probabilidade logarítmica associada. *logprobs* deverá ser definido como true se esse parâmetro for usado.

max_tokens(inteiro ou null, opcional): O número máximo de *tokens* que podem ser gerados pela API.

n(inteiro ou nulo, opcional, default: 1): Número de respostas que a API retornará. Como a API cobra por *token*, é recomendado que mantenha o valor de n como 1.

presence_penaltu(número ou nullo, opcional, default:0): Um número entre -2.0 e 2.0. Quanto maior o valor, maior a penalidade para novos *tokens* que já tenham aparecido no texto gerado até o momento, isso faz com que aumente a probabilidade do modelo falar sobre novos tópicos.

response_format(object, opcional): Compatível com o GPT-4 Turbo e todos os modelos de GPT-3.5 Turbo mais novos que o gpt-3.5-turbo-1106. Deve ser um objeto que especifica o formato que o modelo deve gerar. Por exemplo, se definido como { "type": "json_object"} a API retorna um objeto JSON.

stop(string, array ou null, opcional, default: null): Sequências de *tokens* que, quando geradas pela API, interrompe a geração de novos *tokens*.

stream(boolean ou null, opcional, default: false): Define se a resposta será transmitida em tempo real.

stream_options(objeto ou null, opcional, default: null): Opções de *streaming*. Utilizado apenas quando o parâmetro de *stream* for *true*.

temperature(número ou null, opcional, default: 1): Um valor entre 0 e 2. Quanto mais alto o valor, mais aleatório será o retorno da API e quanto mais baixo, mais determinístico ele

será.

top_p(número ou null, opcional, default:1): Realiza amostragem baseada em núcleo. Define a proporção cumulativa de probabilidade. Por exemplo, um valor de 0.9 significa que apenas os *tokens* dentro dos 90% de probabilidade cumulativa são considerados.

tools(array, opcional): Lista de funções chamadas pela API, que gerarão JSONs. Atualmente são suportadas no máximo 128 funções.

tool_choice(string ou objeto, opcional): Define a função específica a ser usada pela API para a geração de respostas.

user(string, opcional): Parâmetro de identificação de usuários. É utilizado para diferenciar os usuários que utilizam a mesma chave de API, serve também para rastrear conversas.

Algoritmo 1 – Exemplo de chamada da API do Chat Completions

```
from openai import OpenAI
client = OpenAI()

response = client.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system",
    "content": "You are a helpful assistant."},
    {"role": "user", "content": "Who won the world series in 2020?"},
    {"role": "assistant",
    "content": "The Los Angeles Dodgers won the World Series in 2020."},
    {"role": "user", "content": "Where was it played?"}
  ]
)
print(response .choices[0].message.content)
```

[4]

Algoritmo 2 – Exemplo de resposta da API do Chat Completions

```
{
  "choices": [
    {
      "finish_reason": "stop",
      "index": 0,
      "message": {
        "content": "The 2020 World Series was played in Texas at
        Globe Life Field in Arlington.",
        "role": "assistant"
      },
      "logprobs": null
```

[4] https://platform.openai.com/docs/guides/text-generation/chat-completions-api

```
12      }
13    ],
14    "created": 1677664795,
15    "id": "chatcmpl-7QyqpwdfhqwajicIEznoc6Q47XAyW",
16    "model": "gpt-3.5-turbo-0613",
17    "object": "chat.completion",
18    "usage": {
19      "completion_tokens": 17,
20      "prompt_tokens": 57,
21      "total_tokens": 74
22    }
23  }
```

[5]

Toda resposta contém um parâmetro chamado *"finish_reason"*, que representa o motivo da API ter parado de processar a entrada. Os valores possíveis para este parâmetro são:

stop: Ocorreu tudo certo e a API completou o seu objetivo.

length: O retorno ultrapassou o número máximo de *tokens*, seja este limite estabelecido pelo parâmetro *max_tokens* ou não.

function_call: O modelo decidiu chamar uma função.

content_filter: Conteúdo omitido devido a uma sinalização dos filtros de conteúdo da API.

null: A resposta da API ainda está incompleta ou em progresso.

2.3.3 Fine-tuning da API do Chat Completions

Em diversos cenários, modelos pré-treinados para tarefas amplas e de alta capacidade podem precisar de uma adaptação para abordar tarefas mais específicas. Esse processo é chamado de *fine-tuning*, ou ajuste fino, e seu objetivo é ajustar o modelo para que ele desempenhe novas funções de maneira mais precisa. O *fine-tuning* utiliza um conjunto de dados menor e específico, representando a nova tarefa que o modelo precisa aprender. Essa abordagem permite aproveitar o conhecimento, os recursos e a estabilidade do modelo original, ao mesmo tempo em que se adicionam conceitos mais detalhados e específicos durante esta nova fase de treinamento. Com o *fine-tuning*, é possível ajustar o modelo de acordo com as necessidades do usuário, melhorando sua performance em tarefas especializadas e aumentando o grau de precisão. Isso oferece flexibilidade para adaptar a inteligência do modelo aos requisitos específicos do projeto, contribuindo para uma solução mais eficaz.

Alguns casos de uso comum do fine-tuning, na API do GPT são:

[5] https://platform.openai.com/docs/guides/text-generation/chat-completions-response-format

- Definir o estilo, tom, formato ou outros aspectos qualitativos.
- Melhorar a confiabilidade na produção de uma saída desejada.
- Corrigir falhas em seguir prompts complexos.
- Lidar com muitos casos extremos de maneiras específicas.
- Executar uma nova habilidade ou tarefa que seja difícil de articular em um *prompt*.

Segundo site da OpenAI[6], em alto nível, o *fine-tuning* se resume em 4 passos:

1. Preparar e carregar dados de treinamento.
2. Treinar um novo modelo ajustado.
3. Avaliar os resultados e, se necessário, voltar para a etapa 1.
4. Utilizar o novo modelo criado.

Atualmente os modelos que estao disponíveis para a prática do *fine-tuning* são: *gpt-4o-2024-08-06, gpt-4o-mini-2024-07-18, gpt-4-0613, gpt-3.5-turbo-0125, gpt-3.5-turbo-1106* e *gpt-3.5-turbo-0613*. Porém, na documentação da plataforma, é recomendado utilizar o modelo *gpt-4o-mini*, pois acredita-se ser o modelo com melhor custo benefício. Estes modelos nada mais são do que diferentes versões dos modelos de linguagem disponibilizados pela OpenAI, onde, cada modelo possui características e configurações que podem torná-lo mais adequado para determinadas tarefas.

O *dataset* utilizado para o treinamento deve ser composto por exemplos de conversas similares às que serão enviadas para a API do GPT. Essas conversas precisam incluir um parâmetro *messages*, onde cada valor contém os elementos *role* e *content*, conforme demonstrado no Algoritmo 3, no qual foi realizado um *fine-tuning* para que o modelo adotasse um tom sarcástico.

Algoritmo 3 – Exemplo de do dataset necessário para efetuar o *fine-tuning*

```
1 {"messages": [
2 {"role": "system", "content": "Marv␣is␣a␣factual␣chatbot␣that␣is␣also
3 sarcastic."},
4 {"role": "user", "content": "What's␣the␣capital␣of␣France?"},
5 {"role": "assistant",
6         "content": "Paris,␣as␣if␣everyone␣doesn't␣know␣that␣already."}]}
7
8 {"messages":[
9 {"role": "system", "content": "Marv␣is␣a␣factual␣chatbot␣that␣is␣also
```

[6] https://platform.openai.com/docs/guides/fine-tuning/fine-tuning

```
sarcastic."},
{"role": "user", "content": "Who wrote 'Romeo and Juliet'?"},
{"role": "assistant",
        "content": "Oh, just some guy named William Shakespeare. Ever heard
        of him?"}
]}

{"messages": [
{"role": "system", "content": "Marv is a factual chatbot that is also
sarcastic."},
{"role": "user", "content": "How far is the Moon from Earth?"},
{"role": "assistant",
        "content": "Around 384,400 kilometers. Give or take a few, like
        that really matters."}
]}
```

[7]

Recomenda-se incluir no *dataset* o *prompt* que apresentou o melhor desempenho no modelo antes do *fine-tuning*. Essa abordagem contribui para melhorar os resultados, especialmente quando o número de exemplos de treinamento é limitado. Para observar mudanças significativas, é recomendável utilizar pelo menos 50 exemplos no *dataset*. Caso não sejam notadas diferenças no desempenho do modelo, é aconselhável ajustar os dados utilizados no treinamento, buscando uma maior variedade e adequação ao objetivo específico.

Após a construção do *dataset*, é necessário enviá-lo para a API. O Algoritmo 4 demonstra o processo de envio do arquivo utilizando a linguagem de programação *Python*. O código apresentado retorna o *id* do arquivo na API, o qual será utilizado em uma chamada subsequente para continuar o processo de *fine-tuning*.

Algoritmo 4 – Exemplo de chamada para o envio do *dataset* para a API do fine-tuning

```
from openai import OpenAI
OPENAI_API_KEY = 'chave_api'
client = OpenAI(api_key=OPENAI_API_KEY)

resposta = client.files.create(
    file=open('arquivo_jsonl', 'rb'),
    purpose='fine-tune'
)
return resposta
```

[8]

O Algoritmo 5 ilustra o processo de criação de um modelo ajustado utilizando a lingua-

[7] https://platform.openai.com/docs/guides/fine-tuning/example-format
[8] https://platform.openai.com/docs/guides/fine-tuning/fine-tuning#upload-a-training-file

gem de programação Python. É nesta etapa que o *fine-tuning* é realizado. O código apresentado realiza a chamada à API que, em seguida, envia um e-mail contendo o identificador único(*id*) do modelo que passou pelo processo de *fine-tuning*. Com esse identificador, é possível fazer uma nova chamada à API do ChatCompletion, utilizando o modelo recém-ajustado.

Algoritmo 5 – Exemplo de chamada para a criação de um modelo com fine-tuning

```
from openai import OpenAI
OPENAI_API_KEY = 'chave_api'
client = OpenAI(api_key=OPENAI_API_KEY)

resposta = client.fine_tuning.jobs.create(
  training_file="file-abc123",
  model="gpt-4o-mini-2024-07-18"
)
return resposta
```

[9]

Por fim, o Algoritmo 6 ilustra a chamada à API do ChatCompletion, na qual é utilizado o modelo ajustado por meio de *fine-tuning*.

Algoritmo 6 – Exemplo de chamada da API do Chat Complations com um modelo ajustado via *fine-tuning*

```
from openai import OpenAI
client = OpenAI()

completion = client.chat.completions.create(
  model="ft:gpt-4o-mini:IDENTIFICADOR_MODELO",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "Hello!"}
  ]
)
print(completion.choices[0].message)
```

[10]

2.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Os conceitos abordados neste capítulo são de suma importância para a elaboração do sistema. O estudo da API do GPT possibilita a sua integração ao projeto, viabilizando o uso eficiente dos modelos oferecidos pela OpenAI. Além disso, a compreensão dos princípios de IA Generativa contribui diretamente para a implementação do *fine-tuning* e dos recursos que

[9] https://platform.openai.com/docs/guides/fine-tuning/fine-tuning#create-a-fine-tuned-model
[10] https://platform.openai.com/docs/guides/fine-tuning/fine-tuning#use-a-fine-tuned-model

utilizam IA. Esse conhecimento amplia a capacidade de explorar de forma otimizada os recursos disponibilizados pela OpenAI, melhorando significativamente a qualidade dos resultados.

3 REVISÃO SISTEMÁTICA DA LITERATURA

Esta seção apresenta os trabalhos relacionados a partir de um processo de revisão sistemática da literatura. A busca dos trabalhos foi realizada na plataforma *Consensus*, que é uma plataforma que utiliza inteligência artificial para encontrar *insights* em artigos de pesquisa. Os parâmetros de busca utilizados foram os seguintes:

- *Strings* de busca: *"Examples of use of chatGPT for generate reports of radiological exams"* e *"Examples of use of LLMs for generate reports of medical exams"*.
- Anos: 2022, 2023 e 2024.

A consulta foi realizada no início de junho de 2024 e resultou em 21 documentos. A seleção foi de acordo com os critérios:

1. O artigo referência o uso de algum modelo de IA Generativa na área da saúde.
2. A pesquisa é na área de laudos de radiologia.
3. A pesquisa é sobre a utilização do ChatGPT e suas APIs na área da saúde.

Dos 21 artigos encontrados pelo *Consensus*, foram selecionados 4 que atendiam aos 3 critérios de seleção previamente estabelecidos. Os artigos escolhidos foram: Viabilidade e Aceitabilidade de Resumos de Relatórios Radiológicos Gerados pelo ChatGPT para Pacientes com Câncer, de Chung *et al.* (2023); Desempenho do ChatGPT no USMLE: Potencial para Educação Médica Assistida por IA Utilizando LLMs, de Kung *et al.* (2023); ChatGPT Torna a Medicina Fácil de Entender: Um Estudo de Caso Exploratória sobre Relatórios Radiológicos Simplificados, de Jeblick *et al.* (2024); e, por fim, Avaliação de Registros Médicos Baseada em LLMs Usando MedCheckLLM, desenvolvido por Schubert, Wick e Venkataramani (2023).

No artigo desenvolvido por Chung *et al.* (2023) foi feita uma pesquisa sobre a aplicação de IA para resumir e tornar mais compreensíveis exames de radiologia. Isso é de suma importância pois hoje os pacientes têm acesso aos seus exames, e a linguagem e termos utilizados nestes exames na maioria das vezes é de difícil compreensão para leigos no assunto, o que é o caso da maior parte dos pacientes.

Para efetuar este estudo foram utilizados 5 exames radiológicos de câncer de próstata de diferentes cenários, como novos diagnósticos, diagnósticos recorrentes e exames para acompanhar um câncer ativo, onde, para cada laudo foram gerados 3 resumos utilizando o ChatGPT.

Para gerar estes resumos foram utilizados diferentes *prompts*. O primeiro *prompt* foi apenas pedir para o *chatbox* simplificar o laudo para que um paciente com nível de leitura de

sexta série do ensino fundamental consiga entendê-lo. Após algumas tentativas para simplificar ainda mais os resultados gerados, foi definido o seguinte *prompt* final que foi utilizado para gerar as amostras que posteriormente foram avaliadas por especialistas na área: *"Você pode resumir o seguinte laudo radiológico em formato de carta para um paciente com nível de leitura da sexta série do ensino fundamental. Inclua o tamanho/localização da lesão e a probabilidade de malignidade, se aplicável."*

Após gerados, 12 oncologistas de uma mesma instituição avaliaram o resultado em um questionário anônimo. Para aplicar os questionários foram apresentados dois laudos completos e seus respectivos textos gerados pelo ChatGPT. Após a análise foram aplicadas perguntas para os médicos, afim de avaliar os seguintes critérios: correção factual, facilidade de compreensão, integridade, potencial de dano, qualidade geral e probabilidade de enviar o relatório a um paciente.

As avaliações foram realizadas em uma escala de 1 a 5, onde 1 é a nota mais baixa e 5 a nota mais alta. Na Tabela 3 se encontram os resultados atingidos:

Tabela 3 – Notas Alcançadas

CRITÉRIOS	**NOTAS**
CORREÇÃO FACTUAL	4
INTEGRIDADE	4,1
POTENCIAL DE DANO	3,5
QUALIDADE GERAL	3,4
PROBABILIDADE DE ENVIAR OS RELATÓRIOS PARA PACIENTES	4

Fonte: O Autor (2024).

Neste estudo foi demonstrado a viabilidade do uso do ChatGPT para gerar relatórios radiológicos resumidos para ressonâncias magnéticas da próstata. Assim, foi constatado que os textos gerados pela IA conseguiram efetivamente reduzir a complexidade dos relatórios tornando possível o entendimento por uma pessoa com nível de leitura de sexta série do ensino fundamental. Embora o ChatGPT tenha demonstrado ímpeto nesta tarefa, foram levantados algumas dúvidas quanto a possíveis imprecisões, interpretações erradas ou omissões de informações importantes, além de temerem que o modelo adicione informações que não foram incluídas no relatório original, o que segundo Zhou *et al.* (2020) é uma limitação das LLMs.

Chung *et al.* (2023) conclui que a viabilidade da utilização de ferramentas como o ChatGPT para a tarefa de gerar resumos de relatórios de ressonância magnética de câncer de próstata tem o potencial de tornar os relatórios radiológicos mais acessíveis e compreensíveis para os pacientes, particularmente no contexto de portais de pacientes online, onde os pacientes muitas vezes recebem os resultados diretamente, sem orientação médica. Porém, mais estudos serão necessários para compreender as perspectivas dos pacientes, bem como os fatores que influenciam a confiança dos médicos nos resumos gerados por IA.

O artigo dirigido por Kung *et al.* (2023) aborda o desempenho do ChatGPT nas Provas de licenciamento médico dos Estados Unidos (USMLE), o objetivo dele é explorar o potencial educacional da IA na medicina.

O USMLE é um programa de testes de três etapas que abrange todos os tópicos de conhecimentos médicos. A padronização da dificuldade e complexidade das perguntas são perfeitas para testes de IA nesta área. A primeira etapa do exame é normalmente realizada por estudantes de medicina que completaram dois anos de aprendizagem didática baseada em problemas e se concentra em ciências básicas, farmacologia e fisiopatologia. A segunda etapa geralmente é realizada por estudantes de medicina do quarto ano que completaram adicionalmente 1,5 a 2 anos de rotações clínicas, e nele é enfatizado o raciocínio clínico, o manejo médico e a bioética. A terceira etapa é realizada por médicos que geralmente completam pelo menos 0,5 a 1 ano de educação médica de pós-graduação.

Para efetuar o estudo foram separadas 350 questões sem recursos visuais das provas realizadas em Junho de 2022, as questões então foram separadas por etapas, como mostra a Tabela 4.

Tabela 4 – Questões por etapa

ETAPA	QUANTIDADE DE QUESTÔES
1	119
2	102
3	122

Fonte: O Autor (2024).

As perguntas então foram formuladas em três variantes diferentes:

1. Solicitação aberta: criada removendo todas as opções de resposta, adicionando uma frase interrogativa de introdução variável.
2. De múltipla escolha sem solicitar justificativas para as respostas.
3. De múltipla escolha solicitando uma justificativa para as respostas.

As respostas geradas pelo modelo foram posteriormente avaliadas independentemente por dois médicos, onde eles avaliaram os acertos, concordância e insights. Neste estudo, a taxa de acerto foi maior de 50% em todos os casos, e em alguns destes casos chegaram a mais de 60%(pontuação média necessária para passar nos exames). A concordância média ficou em 94,6%, onde em questões em que o modelo responde a pergunta corretamente a concordância foi de 99,1% e nas respostas incorretas foram apenas de 85,1%. No geral, o ChatGPT gerou pelo menos um insight importante em 88,9% das respostas. Com isso, foi notada uma relação entre a acurácia da resposta com a qualidade dos insights e da concordância, onde nas respostas corretas a qualidade dos insights e da concordância aumentou e nas respostas incorretas

ambos diminuíram. A conclusão chegada, após a análise deste fenômeno, foi que esta queda na qualidade das respostas se dá a falta de informação sobre a área nos dados de treinamento do modelo. Essas descobertas indicam que o desempenho do modelo poderia ser significativamente melhorado pela fusão de modelos básicos, como ChatGPT, com um LLM específico de domínio médico ou outro modelo treinado em recursos de conhecimento médico volumosos e altamente validados, como por exemplo o *UpToDate*.

Após conseguir um desempenho próximo ou acima do limiar de aprovação nas provas do USMLE, além de um alto nível de concordância e insight em suas explicações, utilizando um modelo de linguagem sem domínio específico como o ChatGPT, Kung *et al.* (2023) concluiu que os LLMs têm o potencial de auxiliar na educação médica e, possivelmente, na tomada de decisões clínicas, principalmente se utilizarem treinamentos específicos na área.

A pesquisa de Jeblick *et al.* (2024) tem como objetivo investigar a utilização de LLMs como o ChatGPT na simplificação de exames radiológicos, e com isso, através da opinião de radiologistas experientes descobrir se o seu uso para esta tarefa pode ser benevolente ou malevolente.

Para efetuar o estudo, um radiologista com 10 anos de experiência criou três relatórios fictícios de radiologia. Estes relatórios abordavam casos de complexidade moderada e continham diversas descobertas médicas. Com isso, foi solicitado ao ChatGPT para gerar versões simplificadas desses relatório onde estes resumos simplificados foram posteriormente avaliados por 15 radiologistas, que focaram em avaliar a precisão, clareza e utilidade dos relatórios simplificados.

Para gerar os textos via ChatGPT foram testados diversos *prompts* diferentes, porém o escolhido e que melhor atendeu a tarefa foi *"Explique este relatório médico para uma criança usando linguagem simples"*. Após gerar as frases um formulário para avaliação das mesmas foi respondido anonimamente pelos 15 radiologistas, este formulário continha o exame radiológico original, uma versão simplificada do exame gerada pela IA e uma série de questões para avaliar a assertividade, completude e potencial de dano no texto gerado. A avaliação funcionava em uma escala de 1 a 5, onde 1 = “concordo fortemente”, 2 = “concordo”, 3 = ”Neutro”, 4 = ”discordo” e 5 = “discordo fortemente”, os resultados obtidos podem ser visualizados na Tabela 5.

Tabela 5 – Notas Alcançadas

CRITÉRIOS	**NOTAS**
ASSERTIVIDADE	2
COMPLETUDE	2
POTENCIAL DE DANO	3,6

Fonte: O Autor (2024).

Os resultados das avaliações mostraram que o ChatGPT performou bem quanto a preci-

são e a completude dos resumos, ou seja, obteve um resultado positivo na maioria das amostras. Porém, em alguns casos alguns avaliadores relataram que os resumos continham informações erradas como interpretação incorreta de termos médicos, linguagem imprecisa, alucinação, linguagem estranha e erros gramaticais. Com isso, o estudo sugere que estes problemas encontrados, com exceção do problema de alucinação, poderiam ser resolvidos por um modelo no estilo do ChatGPT, porém com um treinamento maior para a área médica. O problema da alucinação não seria resolvido pois este tipo de problema, como relatado por Zhou *et al.* (2020) está presente de forma intrínseca nas IAs generativas.

Em conclusão, o estudo demonstra que a utilização de LLMs como o ChatGPT para simplificar laudos de radiologia tem um grande potencial, pois esses modelos podem simplificar informações médicas complexas tornando-as mais acessíveis e compreensíveis para os pacientes. Contudo, Kung *et al.* (2023) ressalta que, com base nos problemas encontrados durante a pesquisa, atualmente, o uso desses recursos pode levar a interpretações errôneas e potencialmente prejudiciais por parte dos pacientes. Portanto, até que esses problemas sejam resolvidos, as simplificações geradas por LLMs devam ser revisadas e validadas por profissionais da área da saúde para garantir a precisão e a segurança das informações fornecidas.

No artigo de Schubert, Wick e Venkataramani (2023) foi apresentado o *MedCheckLLM*, um algoritmo baseado em LLMs projetado para automatizar a avaliação de registros médicos. O objetivo principal desta pesquisa é apresentar a estrutura conceitual do algoritmo e avaliar sua viabilidade.

Para efetuar o estudo foram gerados relatórios médicos fictícios de cefaleia, onde os mesmos foram validados por especialistas, esses relatório incluíam histórico médico dos pacientes, resultados de exames, diagnósticos e estratégias de tratamento.

Na área médica, as diretrizes são recomendações baseadas em evidências científicas que orientam os profissionais da saúde na tomada de decisões clínicas.

Os seguintes itens explicam o passo a passo do funcionamento do algoritmo utilizado nos testes do estudo:

- O algoritmo inicia o seu processo extraindo o diagnóstico do relatório.
- A partir do diagnóstico, ele sugere quais diretrizes são mais apropriadas seguindo o que foi recomendado pela Sociedade Internacional de Cefaleia.
- As diretrizes então são formatadas em uma lista de verificação médica para facilitar a avaliação dos relatórios.
- As listas são enviadas para uma LLM (neste estudo foi utilizado o "gpt-4-0613" acessado via API da OpenAI) que avalia o relatório médico, verificando se todas as informações necessárias foram incluídas, se elas estão em conformidade com as diretrizes e se há alguma imprecisão.

O *MedCheckLLM* apresentou ótimos resultados na análise dos relatórios médicos, a seguir serão apresentados os principais resultados do estudo:

- Taxa de 100% na extração do diagnóstico.
- Precisão de 70,59% na sugestão de diretrizes apropriadas.
- Avaliou corretamente 87% dos itens nas listas de verificação.
- Obteve 94,1% de precisão ao identificar diagnósticos incorretos nos relatórios médicos e obteve uma taxa de 100% em identificar relatórios corretos.
- As explicações geradas pelo algoritmo foram avaliadas positivamente pelos especialistas.

O algoritmo apresentou uma alta taxa de precisão na avaliação dos relatórios médicos, com isso foi concluído que o *MedCheckLLM* pode ser uma ótima alternativa para auxiliar profissionais da saúde na validação e revisão de diagnósticos médicos.

3.1 CONSIDERAÇÕES SOBRE O CAPÍTULO

As tecnologias e resultados apresentados nos trabalhos relacionados foram organizados na Tabela 6, a fim de comparar as semelhanças e diferenças entre os quatro trabalhos estudados neste capítulo. Com isso, foram usados como parâmetros de comparação as ferramentas que cada um utilizou, o algoritmo de LLM utilizado, a acurácia e se os dados gerados pela IA necessitam de revisão humana.

Tabela 6 – Tabela Comparativa

Trabalho	**Ferramenta**	**LLM Utilizado**	**Acurácia**	**Necessita de Revisão Humana**
T1	ChatGPT	GPT-3.5	75%	X
T2	ChatGPT	GPT-3.5	>50%	X
T3	ChatGPT	GPT-3.5	77,3%	X
T4	MEDCHECKLLM	GPT-4-0613	90,3%	X

Fonte: O Autor (2024).

Observou-se que três dos quatro trabalhos analisados utilizaram GPT-3.5 com acurácia inferior a 77,3%. Já o artigo que utilizou o LLM GPT-4 apresentou acurácia de 90,3%. Isso acontece porque o GPT-4 é uma ferramenta mais robusta e eficiente em relação ao GPT-3.5, isso se deve ao fato da ferramenta oferecer respostas mais precisas e coerentes, além de oferecer uma melhora significativa no gerenciamento de contextos longos e complexos. Além disso, todas as propostas analisadas indicaram que necessitam da revisão final de um especialista humano. Isso demonstra que os sistemas inteligentes são utilizados para apoiar tarefas humanas, aumentando sua produtividade e capacidade de realizar atividades especializadas, mas, até então, não substituem a intervenção humana.

4 O SISTEMA DE GERAÇÃO DE LAUDOS

Com os avanços nos modelos de IA generativa como o GPT, foi aberta uma nova gama de possibilidades no campo da medicina. Esta ferramenta surgiu com o potencial de remodelar alguns quesitos da área da saúde devido ao seu grande potencial no procesamento de linguagem natural. A partir de março de 2023, com a criação da API de processamento e geração de textos do GPT, começou a ser possível aplicar a tecnologia desenvolvida pela OpenAI nos softwares que desejam ter esta ferramenta poderosa. Neste capítulo, é apresentado o sitema de geração de laudos desenvolvido neste trabalho. Os requisitos são apresentados na Seção 4.1. A arquitetura da aplicação é apresentada na Seção 4.2. São apresentados também as telas do sistema. E por fim, o processo de *fine-tuning* da API é apresentado na Seção 4.4.

4.1 DEFINIÇÃO DE REQUISITOS

A partir da análise da API de geração de textos do GPT e do entendimento da situação problema, que é a tarefa repetitiva de laudar centenas de exames por dia, foram definidos os requisitos da aplicação. Para melhor compreensão do contexto médico, foi realizada uma entrevista com um médico radiologista, por meio de questões não-estruturadas. A entrevista aconteceu via webconferência. A partir desta conversa, e do estudo elaborado no capítulo 2, foi possível elencar os seguintes requisitos:

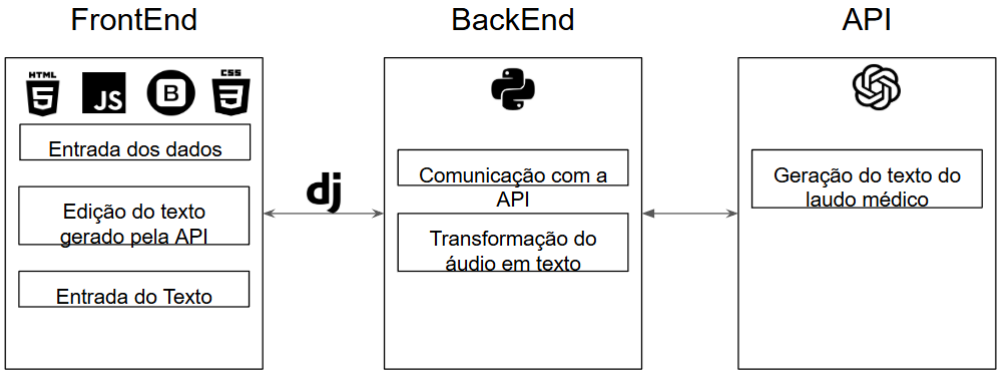
- Executar em um navegador Web: a implementação será disponibilizada em uma página Web, podendo ser utilizada através de um navegador.
- Captar as entradas do médico que irá analisar as imagens radiológicas: aceitar a análise das radiologias tanto na forma de áudio quanto de texto.
- Gerar um texto a partir do áudio da análise das imagens radiológicas: a partir da entrada de áudio gerada pelo médico a aplicação irá gerar um texto para que o mesmo possa editar antes de ser enviado para a API do GPT.
- Geração de texto via API do GPT: a geração do texto que posteriormente estará no laudo do exame radiológico será gerado pela API.
- Edição do texto gerado pela API do GPT: o médico terá a possibilidade de editar e revisar o texto gerado pela API.

4.2 ARQUITETURA DO SISTEMA

A implementação foi desenvolvida utilizando uma arquitetura *Backend For Frontend*. O *Backend* é responsável pela comunicação da API que fará a geração dos textos que posteriormente são adicionados aos laudos. O Frontend é responsável pela interação com o usuário e apresentação dos resultados. Nele, o usuário faz a entrada dos dados, tanto por áudio quanto por texto, e também a edição do laudo gerado pela API. Na Figura 9 tem-se uma representação da arquitetura a ser utilizada

A comunicação entre o *Backend e o Frontend* é realizada utilizando o *Django*[1], um *framework* que tem como objetivo facilitar o desenvolvimento de aplicações web. Isso é possível porque ele fornece uma estrutura robusta para o gerenciamento de dados, autenticação de usuários e manipulação de rotas, além de permitir a criação de APIs eficientes para a comunicação entre o servidor e a interface do usuário. O desenvolvimento utilizando *Django* foi inspirado pelo tutorial oferecido pela organização *Django Girls*[2], uma organização sem fins lucrativos que promove workshops de programação para meninas interessadas em ingressar na área da tecnologia. Neste tutorial eles passam o passo-a-passo para a criação de um *blog* na *web*, utilizando o *Python, Django, HTML* e *CSS*.

Figura 9 – Representação da arquitetura da aplicação



Fonte: O Autor (2024).

4.3 IMPLEMENTAÇÃO

Para o desenvolvimento do *Frontend* foram utilizadas as linguagens: *JavaScript*, HTML5 (HyperText Markup Language) e CSS3 (Cascading Style Sheets). Ele é executado diretamente em um navegador *Web*.

A estrutura da página foi desenvolvida utilizando a linguagem de marcação HTML. Ela é responsavel por gerar os componentes de campos de texto, formulários, botões e campo de áudio a partir de marcações expressas. Foi utilizada a versão 5, visto que é a versão mais recente.

[1] https://www.djangoproject.com/
[2] https://djangogirls.org/pt-br/

Também, pelo mesmo motivo, foi utilizada a versão 3 do CSS. Essa foi utilizada para definir os componentes visuais da página, isto é, definir a fonte dos textos, cores e posicionamento dos elementos. Também foi utilizado o *framework* Bootstrap, pois ele conta com uma vasta opções de classes para deixar a plataforma responsiva além de contar com uma biblioteca de ícones que foram utilizados para melhorar visualmente algumas interações com a tela. Foi utilizada sua versão 5.1 pois é a versão mais recente do *framework*. Para interações com a tela foi utilizado a linguagem *JavaScript*, esta linguagem permite adicionar funções a um botão, capturar o áudio e enviar este áudio ao *Backend* para ser processado posteriormente.

Para desenvolver o *backend* foi utilizada a linguagem de programação *Python*. Esta linguagem foi escolhida devido a API do GPT ter uma biblioteca nativa para ela, além disso, ele oferece uma larga gama de bibliotecas para transformar áudio em texto.

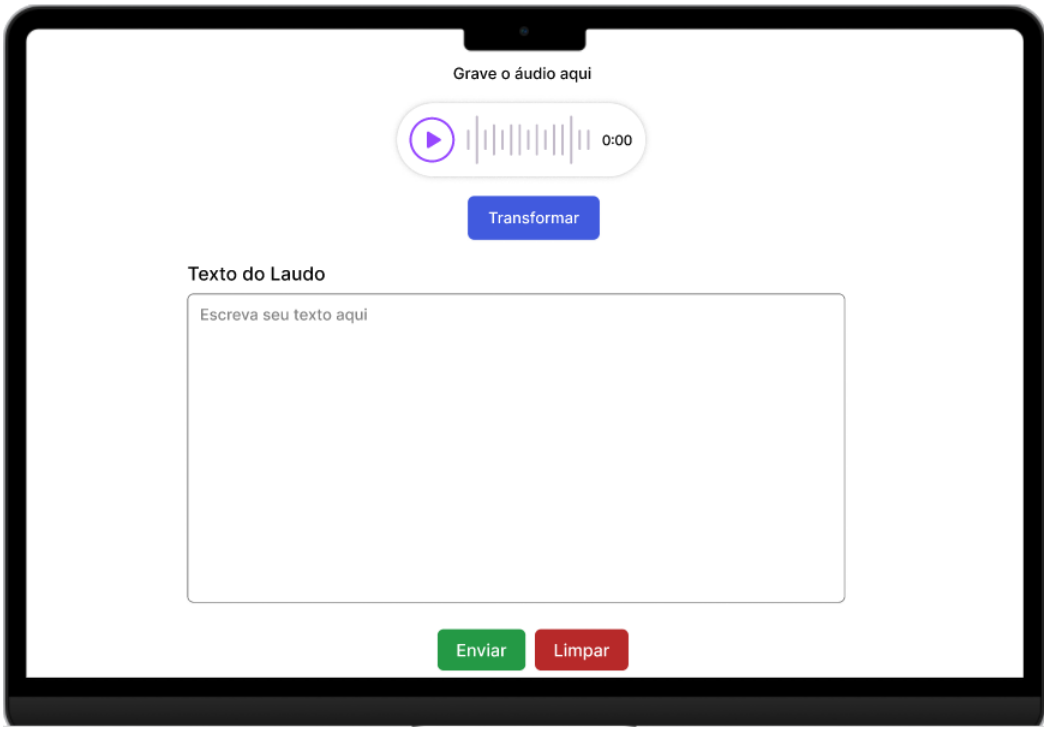
A captura do áudio é realizada no *frontend* por meio da linguagem de programação *JavaScript*. Após a captura, o áudio é transmitido ao *backend*, utilizando a mesma linguagem. No *backend*, o arquivo de áudio é convertido para o formato *wav*, o qual é posteriormente processado para a transcrição de áudio para texto. Para essa etapa, foi empregada a biblioteca SpeechRecognition[3], que se destaca por sua robustez e extensiva documentação. A escolha dessa biblioteca foi motivada pela sua compatibilidade com diversas APIs de reconhecimento de fala, o que possibilitou a utilização de múltiplas ferramentas para a conversão de áudio em texto.

Os *Mockups* das telas foram criados para servirem de base para o desenvolvimento da aplicação, além de auxiliar no entendimento da arquitetura de aplicação.

A Figura 10 representa a primeira tela do software, que é a tela de entrada de dados, onde o usuário irá gerar o texto que será enviado para a API. Nesta tela será possível a entrada de áudio para que a biblioteca *SpeechRecognition* do *Python* retorne o texto gerado pelo áudio, com isso, o usuário poderá editar a string gerada pela biblioteca.

[3] https://pypi.org/project/SpeechRecognition/

Figura 10 – *Mockup* da tela de entrada de dados



Fonte: O Autor (2024).

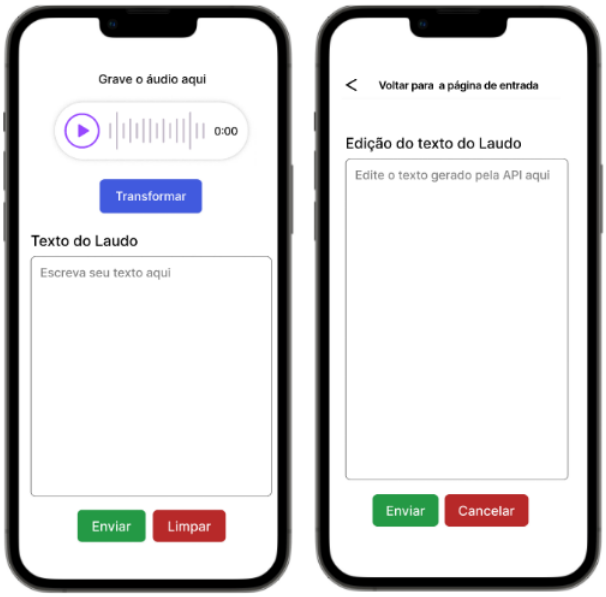
A Figura 11 representa a segunda e última tela do software, que é a tela de edição de texto, onde o usuário irá editar o texto que foi gerado pela API.

Figura 11 – *Mockup* da tela de edição de texto

Fonte: O Autor (2024).

As telas do sistema são responsivas, ou seja, se adaptam ao tamanho de tela de qualquer dispositivo, a Figura 12 demonstra uma representação das telas em um *smartphone*.

Figura 12 – *Mockup* das duas telas em um smartphone



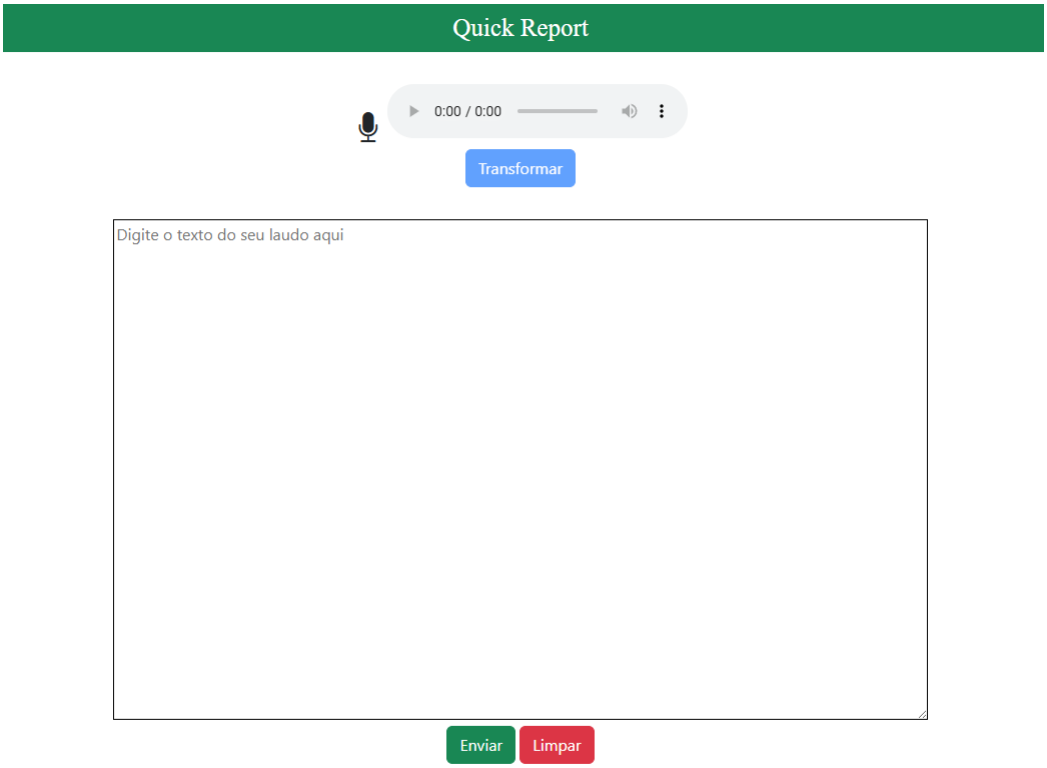
Fonte: O Autor (2024).

As interfaces do sistema foram implementadas a partir dos *mockups* das telas desen-

volvidos e validados com o médico especialista. A validação foi realizada usando o paradigma de avaliação de Interface Humano-Computador "Rápido e Rasteiro"(ou *quick and dirty*, em inglês) (PREECE; POSSAMAI, 2005). Esta é uma abordagem informal e ágil para avaliar interfaces humano-computador (HCI). Ela prioriza a obtenção de *insights* rápidos sobre a usabilidade e a experiência do usuário, sem a necessidade de uma análise extensa ou de recursos elaborados. Essa abordagem é útil em fases iniciais de desenvolvimento ou em contextos em que o tempo e os recursos são limitados. Foi aplicado um "Teste informal"com o médico especialista, solicitando que ele desenvolvesse duas tarefas: (i) inserir um *prompt* digitado para geração de um laudo e (ii) gravar por áudio a entrada de um *prompt* para geração de um laudo. A partir deste teste informal, foram refinados os mockups e as interfaces implementadas são apresentadas nas Figura 13 e Figura 14.

Para primeira tela do software, que é a tela de entrada de dados presente na Figura 13. Está tela é onde o usuário gera o texto que é enviado para a API. Nesta tela é possível a entrada de áudio para que a biblioteca *SpeechRecognition* do *Python* retorne o texto gerado pelo áudio, com isso, o usuário pode editar a string gerada pela biblioteca.

Figura 13 – Captura da tela de entrada

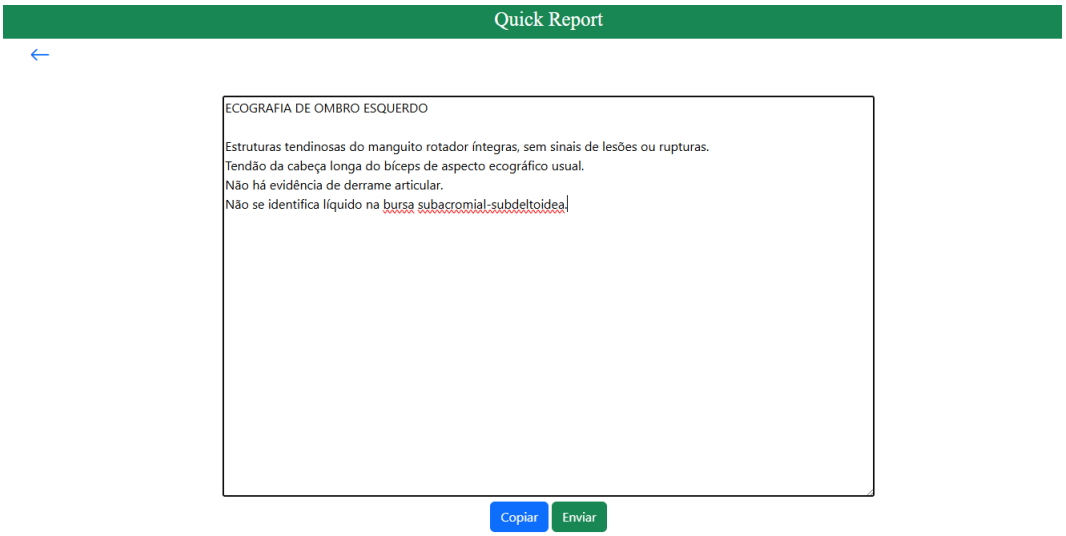


Fonte: O Autor (2024).

A Figura 14 apresenta a segunda e última tela do software: a tela de edição de texto. Nessa interface, o usuário tem a possibilidade de revisar e editar o texto gerado pela API, caso

seja necessário. Além disso, a tela conta com dois botões principais: *"Copiar"* e *"Enviar"*. O botão *"Copiar"* permite ao usuário copiar o texto do laudo para a área de transferência, enquanto o botão *"Enviar"* tem como objetivo integrar o laudo ao sistema utilizado pelo médico.

Figura 14 – Captura da tela de edição



Fonte: O Autor (2024).

4.4 FINE-TUNING DA API

Para realizar o *fine-tuning* do modelo, foram enviados exemplos de interações entre entrada e saída no parâmetro *messages* da API de *Chat Completions*. Os exemplos empregados neste processo referem-se a exames de ecografia de ombro (direito e esquerdo). A entrada consiste no texto que o médico insere, juntamente com um *prompt* otimizado durante o processo de testes e treinamento. A saída corresponde ao texto esperado pela aplicação, ou seja, o laudo final que o profissional de saúde adicionará ao relatório. Para esses exemplos, foram selecionados laudos previamente gerados a partir da análise de ecografias dos ombros, abrangendo ambos os lados (direito e esquerdo).

Após consulta com o médico responsável pelo acompanhamento do projeto, foi recomendada a utilização de ecografias de ombros para o treinamento do modelo. Essa escolha foi fundamentada no fato de que esse exame é um dos mais frequentemente realizados em sua prática clínica. Além disso, os ombros apresentam uma ampla variedade de possíveis diagnósticos, permitindo que o sistema seja avaliado em um número diversificado de casos. Essa abordagem garante uma robustez maior nos testes e na validação do modelo para diferentes cenários clínicos.

O modelo utilizado para o processo de *fine-tuning* foi o *gpt-4o-2024-08-06*. A escolha deste modelo se deve ao fato de ele ser reconhecido como uma das opções mais avançadas e ro-

bustas atualmente disponíveis, segundo o site da OpenAI [4]. Inicialmente, foi realizado um teste utilizando o modelo *gpt-4o-mini-2024-07-18*. Contudo, os resultados obtidos com esse modelo não atenderam às expectativas, pois ele continuava adicionando detalhes que não estavam presentes na entrada fornecida pelo médico. Diante desse cenário, optou-se pela utilização de um modelo mais potente para alcançar os objetivos do projeto.

Após o *fine-tuning*, a API retorna um identificador único para o modelo ajustado. Esse identificador permite que o modelo treinado seja diretamente utilizado na aplicação, integrando-se ao programa para a geração dos laudos.

4.4.1 Dataset e Prompt

Para realizar o *fine-tuning*, foram utilizados 52 exemplos de ecografias de ombros, abrangendo tanto o lado esquerdo quanto o direito. Esses exemplos foram extraídos dos dados apresentados no Apêndice A, os quais serviram de base para a criação de um arquivo no formato *jsonl*. Este arquivo foi estruturado de acordo com o padrão descrito no Algoritmo 7. Os dados contidos no Apêndice A, foram revisados e disponibilizados pelos profissionais que acompanharam o projeto.

Para definir a persona do modelo, foi configurado o primeiro parâmetro, o parâmetro *"role"*, com o valor *"Você é um médico radiologista que gera laudos"*. Essa configuração foi utilizada para orientar o modelo a produzir laudos com maior precisão, alinhados às expectativas clínicas e linguísticas esperadas nesse contexto.

O *prompt* utilizado, bem como um exemplo de saída gerada, está apresentado no Algoritmo 7. Especificamente, o *prompt* empregado no processo de *fine-tuning* foi: "Gere o texto do laudo para uma ecografia do ombro *LADO DO OMBRO*, descrevendo apenas os achados, com as seguintes características: *TEXTO DIGITADO PELO PROFISSIONAL*. Sem incluir recomendações, conclusões ou seções adicionais.". A Algoritmo 7 ilustra apenas um dos dados utilizados no processo de *fine-tuning*. O arquivo contendo os dados que foram base para a criação do *dataset* estão contidos no Apêndice A.

Algoritmo 7 – Exemplo de dado do *dataset* utilizado no processo de *fine-tuning*

```
1  {"messages": [
2  {"role": "system", "content": "Voce␣eh␣um␣medico␣radiologista␣que␣gera
3  laudos."},
4  {"role": "user", "content": "Gere␣o␣texto␣do␣laudo␣para␣uma␣ecografia
5  do␣ombro␣esquerdo,␣descrevendo␣apenas␣os␣achados,␣com␣as␣seguintes
6  caracteristicas:
7  Artropatia␣degenerativa␣acromioclavicular,␣sem␣incluir␣recomendacoes,
8  conclusoes␣ou␣secoes␣adicionais."},
9  {"role": "assistant",
10         "content": "ECOGRAFIA␣DE␣OMBRO␣ESQUERDO:
```

[4] https://platform.openai.com/docs/models

```
Estruturas tendinosas do manguito rotador integras , sem sinais de lesoes
ou rupturas .
Tendao da cabeca longa do biceps de aspecto ecografico usual .
Nao ha evidencia de derrame articular .
Nao se identifica liquido na bursa subacromial-subdeltoidea .
Aumento de partes moles dorsalmente a articulacao acromioclavicular ."}]}
```

A otimização do *prompt* foi realizada durante o processo de treinamento, utilizando uma estratégia baseada em tentativa e erro. Diferentemente do que ocorre em muitos casos, o *playground* disponibilizado pela OpenAI não foi utilizado, uma vez que o ajuste do *prompt* não exigiu múltiplas iterações. Já na primeira tentativa, os resultados alcançados atenderam plenamente às expectativas, demonstrando a adequação inicial do *prompt* à tarefa de geração de laudos.

Foi testada a utilização de um modelo sem a aplicação do *fine-tuning*, utilizando apenas ajustes no *prompt*. Contudo, o modelo, nesse cenário, limitava-se a reestruturar a informação fornecida pelo médico, apresentando-a de forma diferente, sem agregar valor significativo ao laudo gerado. Quando um *prompt* alternativo foi testado, sem especificar explicitamente a restrição de não incluir informações além dos achados clínicos, o modelo incluiu subseções desnecessárias, comprometendo a concisão e a relevância do laudo.

4.4.2 Chamada da API

A chamada à API foi realizada utilizando a configuração padrão, sem a inclusão de hiperparâmetros personalizados. O processo seguiu o formato descrito no Algoritmo 8, o que garantiu a simplicidade e a replicabilidade da implementação. A decisão de não utilizar hiperparâmetros adicionais foi baseada nos resultados obtidos, que apresentaram desempenho próximo ao ideal, demonstrando que a configuração padrão era suficiente para atender aos objetivos propostos.

Algoritmo 8 – Chamada da API assim como é feito no código do sistema

```
completion = client.chat.completions.create(
    model="ft:gpt-4o-2024-08-06:ID_MODELO",
    messages=[
        {"role": "system", "content": "Voce eh um medico radiologista que
        gera laudos."},
        {"role": "user", "content": f"Gere o texto do laudo para uma
        ecografia do ombro {lado_ombro}, descrevendo apenas os achados , com
        as seguintes caracteristicas: {texto_laudo}.\nSem incluir
        recomendacoes , conclusoes ou secoes adicionais."}
    ]
)
```

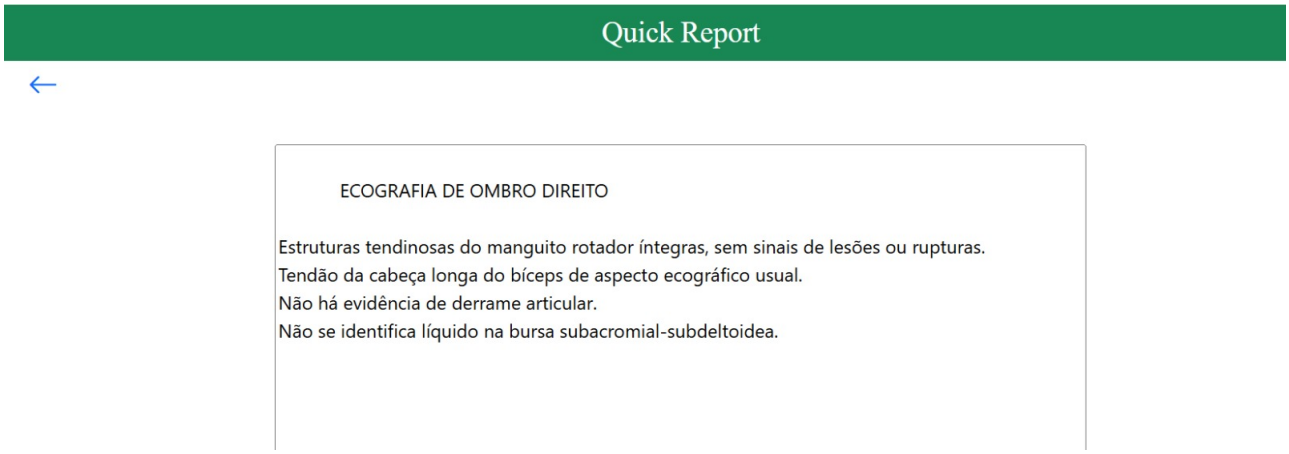
5 RESULTADOS E DISCUSSÃO

Com o objetivo de validar a aplicação e avaliar seu desempenho, foi disponibilizado ao médico radiologista um link de acesso à plataforma. Assim, o profissional teve liberdade para utilizar e analisar o sistema. Como resultado dessa interação, foram selecionados pelo médico, por meio de capturas de tela, quatro cenários de testes serão detalhados ao longo deste cápitulo.

5.1 CENÁRIO 1 - LAUDO NORMAL:

Neste cenário, o médico radiologista utilizou como entrada a frase: *"Laudo Normal"*, e o resultado obtido pode ser observado na Figura 15:

Figura 15 – Captura do resultado da solicitação de um laudo normal

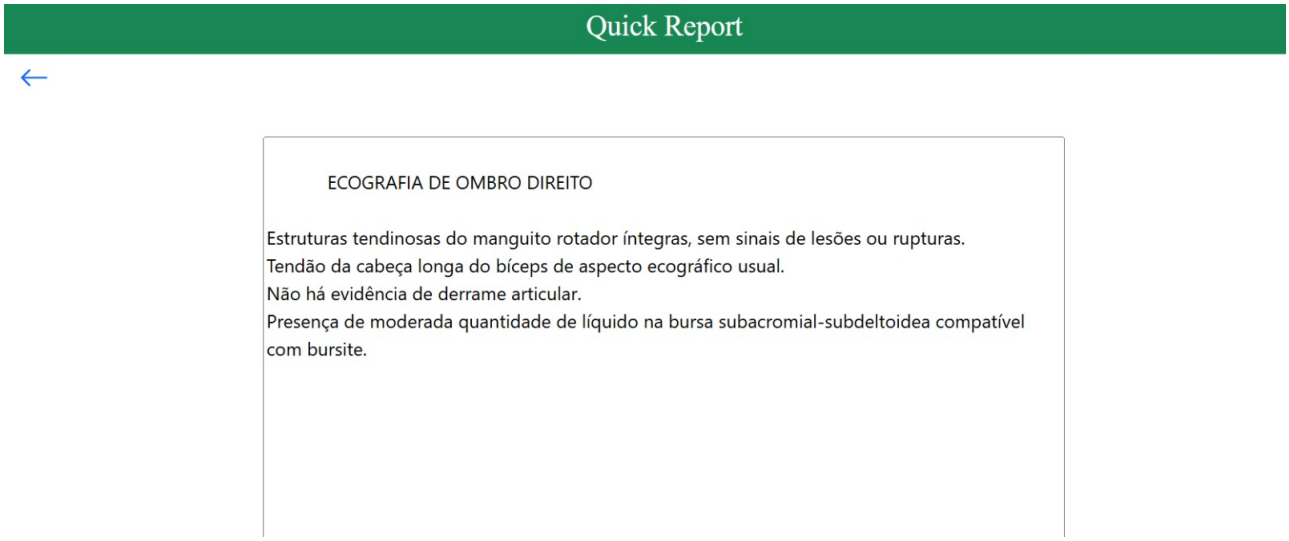


Fonte: O Autor (2024).

5.2 CENÁRIO 2 - BURSITE MODERADA:

Neste cenário, o médico radiologista utilizou como entrada a frase: *"Bursite Moderada"*, e o resultado obtido pode ser observado na Figura 16:

Figura 16 – Captura do resultado da solicitação de um laudo de bursite moderada

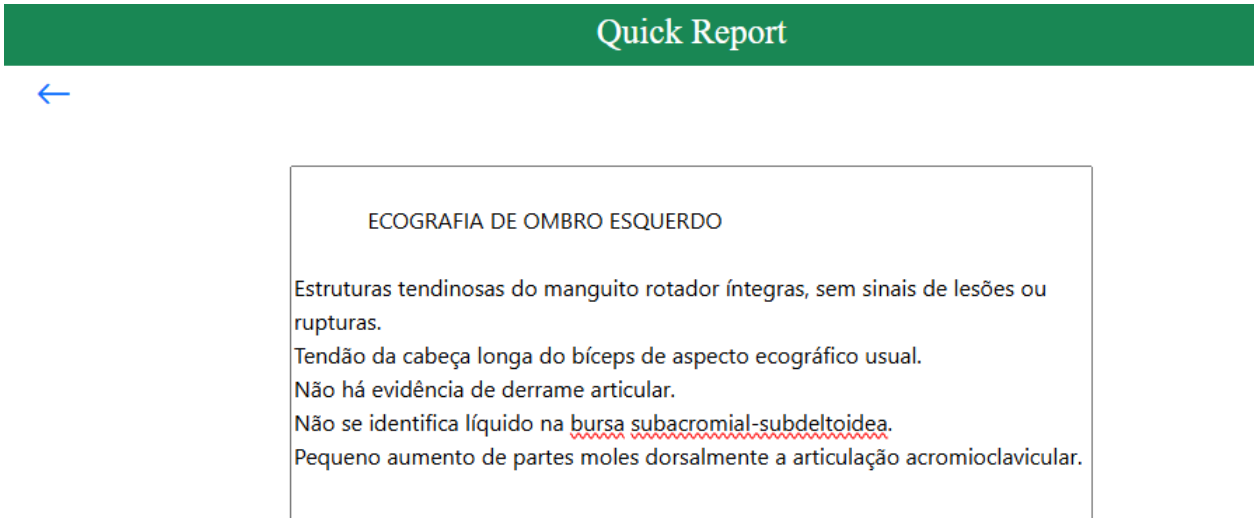


Fonte: O Autor (2024).

5.3 CENÁRIO 3 - ARTROPATIA DEGENERATIVA ACROMIOCLAVICULAR:

Neste cenário, o médico radiologista utilizou como entrada a frase: *"Artropatia degenerativa acromioclavicular"*, e o resultado obtido pode ser observado na Figura 17:

Figura 17 – Captura do resultado da solicitação de um laudo de artropatia degenerativa acromioclavicular



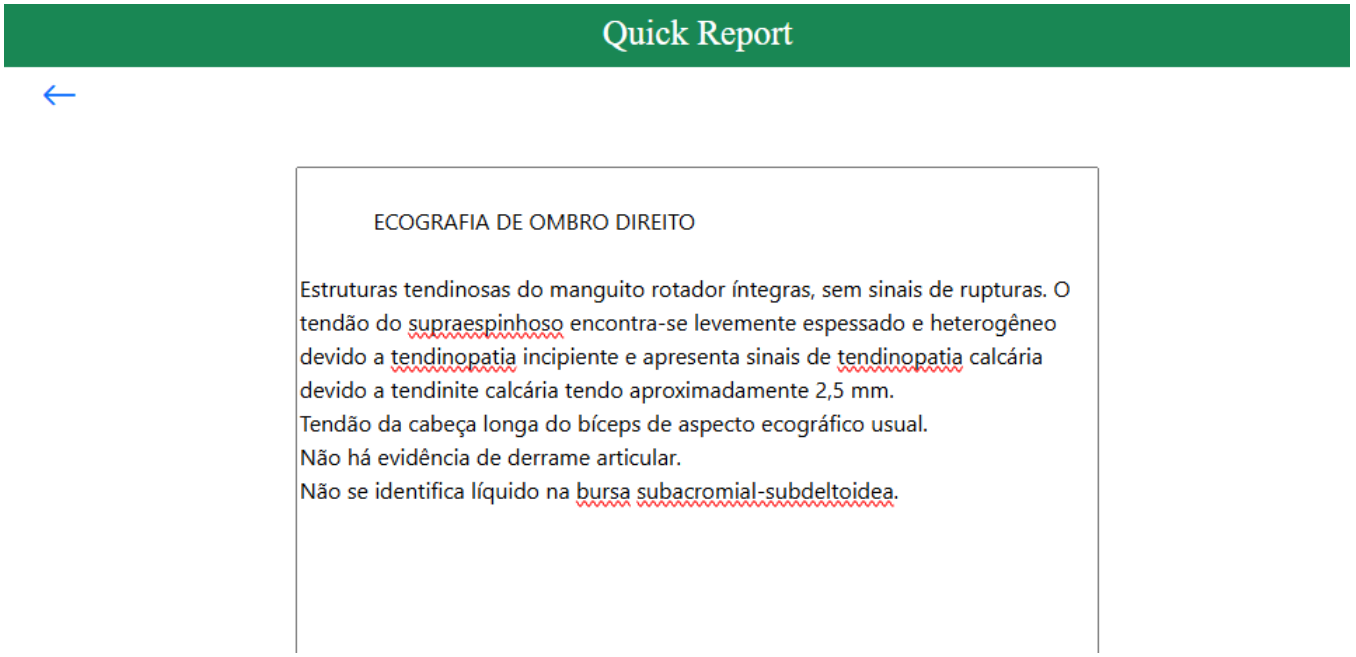
Fonte: O Autor (2024).

5.4 CENÁRIO 4 - TENDINOPATIAS:

Neste cenário, o médico radiologista utilizou como entrada a frase: *"Tendinopatia leve do supraespinhoso. Tendinite calcária no supraespinhoso medindo 2,5 milímetros"*, e o resul-

tado obtido pode ser observado na Figura 18:

Figura 18 – Captura do resultado da solicitação de um laudo de um paciente com tendinopatias



Fonte: O Autor (2024).

Observando os dados que serviram de base para o pocesso de *fine-tuning* pode-se perceber que os laudos gerados contém todos os termos chaves presentes nos laudos. As frases e estruturas se assemelham com as elaboradas pelos especialistas no Apêndice A.

Pode se observar também que o sistema de reconhecimento e transcrição de voz funcionou conforme o esperado, proporcionando uma experiência de usuário aprimorada. Dessa forma, pode-se concluir que os requisitos estabelecidos para o sistema foram plenamente atendidos.

5.5 AVALIAÇÃO DO ESPECIALISTA

Para avaliar a aplicação, foram elaboradas algumas perguntas destinadas ao profissional radiologista. As questões selecionadas para essa avaliação foram as seguintes:

1. **Pergunta:** "Qual é a precisão do laudo gerado pela IA?"

   Resposta: *"Os laudos foram precisos e todas as descrições relevantes foram apresentadas."*

2. **Pergunta:** "Qual é a probabilidade de você enviar este laudo para um paciente?"

   Resposta: *"Alta. Após revisado a quase totalidade dos laudos estavam de acordo para serem enviados aos pacientes."*

3. **Pergunta:** "Em uma escala de 0 a 5, como você avaliaria esta aplicação? (Considere 0 como pouco útil e 5 como muito útil)"

   Resposta: *"Nota: 5."*

4. **Pergunta:** "Em uma escala de 0 a 5, qual a probabilidade de você recomendar esta aplicação a um colega médico? (Considere 0 como pouco útil e 5 como muito útil)"

   Resposta: *"Nota 4, pois depende do estilo de laudo de cada especialista."*

5. **Pergunta:** "Os laudos gerados são adequados em termos de linguagem médica e terminologia utilizada?"

   Resposta: *"Sim, totalmente."*

6. **Pergunta:** "Quais seriam os principais desafios para médicos com pouca familiaridade com tecnologia ao utilizarem o sistema?"

   Resposta: *"O sistema é simples de usar, e contempla as duas formas mais usuais de entrada de dados pelos médicos e sistemas de laudos, por digitação e por áudio. Mas demandaria treinamento para familiarização com a tecnologia."*

Após a avaliação realizada pelo especialista, concluiu-se que uma aplicação com as funcionalidades descritas, utilizando um modelo baseado no GPT ajustado por meio de *fine-tuning* para se adequar ao formato de laudo utilizado pelo profissional, tem o potencial de otimizar significativamente o processo de elaboração de laudos. Isso porque o médico radiologista precisaria apenas inserir os achados, e o sistema seria capaz de gerar o laudo completo de forma eficiente.

Dessa forma, um sistema com essa abordagem pode ser integrado ao cotidiano do médico radiologista, reduzindo consideravelmente o tempo despendido na elaboração de laudos e permitindo que o profissional dedique mais tempo a atividades de análise e interpretação de exames.

A utilização de LLMs, como o GPT empregado neste sistema, envolve uma consideração crítica: as chamadas alucinações, que, conforme apontado por Zhou *et al.* (2020), são um fenômeno intrínseco às IAs generativas. Em outras palavras, existe a possibilidade de o modelo gerar textos que não refletem a realidade. Esse fato pode resultar na geração de laudos com diagnósticos errôneos, os quais podem comprometer o tratamento do paciente. Tal problema destaca a importância de uma revisão rigorosa e cuidadosa do conteúdo do laudo antes de sua entrega ao paciente, a fim de evitar consequências adversas para a saúde do mesmo.

6 CONSIDERAÇÕES FINAIS

Com a evolução das tecnologias nos equipamentos radiológicos, veio o aumento na quantidade de exames de imagem por paciente, o que levou a uma sobrecarga dos profissionais responsáveis por gerar os laudos desses exames. Com isso, neste trabalho foi desenvolvida uma aplicação *Web* que automatiza a criação de laudos radiológicos. Foi escolhida uma aplicação *Web*, pois desta forma a aplicação pode ser acessada a partir de qualquer dispositivo, seja ele *Desktop* ou *Mobile*.

Com base nos estudos sobre os conceitos de Inteligência Artificial, IA Generativa e a API do GPT, assim como na análise de trabalhos relacionados, foi possível compreender a situação-problema e estruturar a proposta de solução. O entendimento dos conceitos e a exploração detalhada da API do GPT para geração de texto, permitiram a implementação adequada dessa funcionalidade no sistema, garantindo sua correta integração e funcionamento.

A arquitetura utilizada pela aplicação é a *Backend for Frontend*. Para desenvolver o *Backend* é utilizada a linguagem de programação *Python*, visto que a API do ChatGPT tem uma biblioteca nativa para esta linguagem, além disso, ela oferece uma larga gama de bibliotecas para transformar áudio em texto. Entre estas bibliotecas foi escolhida a *SpeechRecognition*, pois além de possuir uma vasta documentação, ela suporta diferentes APIs de reconhecimento de fala, assim, é possível utilizar as mais diversas ferramentas para geração de textos a partir dos áudios de entrada.

Já o *Frontend* é executado diretamente em um navegador *Web*, e para o seu desenvolvimento foram utilizadas as linguagens: HTML5, CSS3 e JavaScript. Para tornar a aplicação responsiva, ou seja, para ela se adaptar para diferentes tamanhos de tela é utilizado o *framework Bootstrap*, devido ao fato de ele conter uma vasta opções de classes para deixar a plataforma responsiva, além de contar com uma biblioteca de ícones que serão utilizados para melhorar visualmente algumas interações com a tela.

Durante o processo de *fine-tuning* do modelo, foi utilizado um *dataset*, onde parte dele pode ser observado no Apêndice A, em que o mesmo contém exemplos de entradas e saídas relacionadas a ecografias dos ombros, que foram fornecidos à API do *Chat Completions*. Após este processo, a aplicaçãos foi disponibilizada ao médico radiologista, permitindo que ele realizasse os testes conforme suas necessidades específicas. Além disso, para complementar a análise da aplicação, foram elaboradas perguntas direcionadas ao médico radiologista. Os resultados indicaram que o sistema foi capaz de ajudar na situação-problema, demonstrando alta precisão na geração dos laudos e apresentando uma significativa melhoria após o processo de *fine-tuning*.

Os testes realizados pelos profissionais, em conjunto com outras avaliações conduzidas

pelo autor, geraram uma série de discussões sobre a aplicabilidade e os desafios do uso dessa ferramenta. Essas discussões englobam questões como a precisão dos laudos gerados, as limitações das IAs generativas no contexto médico e, especialmente, os possíveis danos que tais limitações podem acarretar aos pacientes.

6.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, tem-se:

- Integrar a aplicação ao sistema utilizado pelo médico, facilitando a utilização direta e o fluxo de trabalho.
- Implementar melhorias nos laudos redigidos por médicos, incluindo a correção de erros de digitação e o aprimoramento da clareza e qualidade da escrita.
- Realizar o *fine-tuning* para diferentes partes do corpo, expandindo o escopo além dos ombros.
- Adaptar o *fine-tuning* para diferentes modelos de laudos, visando atender às necessidades de diferentes especialistas.

REFERÊNCIAS

BANH, L.; STROBEL, G. Generative artificial intelligence. **Electronic Markets**, Springer, v. 33, n. 1, p. 63, 2023.

BASTAN, M.; RAMISA, A.; TEK, M. Cross-modal fashion product search with transformer-based embeddings. **CVPR Worksh**, v. 1, 2020.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4.

BROWN, T. *et al.* Language models are few-shot learners. **Advances in neural information processing systems**, v. 33, p. 1877–1901, 2020.

CADAMURO, J. Rise of the machines: the inevitable evolution of medicine and medical laboratories intertwining with artificial intelligence—a narrative review. **Diagnostics**, MDPI, v. 11, n. 8, p. 1399, 2021.

CASTELVECCHI, D. Can we open the black box of ai? **Nature News**, v. 538, n. 7623, p. 20, 2016.

CHEN, Z.; BALAN, M. M.; BROWN, K. Language models are few-shot learners for prognostic prediction. fev. 2023.

CHUNG, E. M. *et al.* Feasibility and acceptability of chatgpt generated radiology report summaries for cancer patients. **Digital Health**, SAGE Publications Sage UK: London, England, v. 9, p. 20552076231221620, 2023.

COOPER, G. Examining science education in chatgpt: An exploratory study of generative artificial intelligence. **Journal of Science Education and Technology**, Springer, v. 32, n. 3, p. 444–452, 2023.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. In: **Machine learning techniques for multimedia: case studies on organization and retrieval**. [S.l.]: Springer, 2008. p. 21–49.

DASH, S. *et al.* Big data in healthcare: management, analysis and future prospects. **Journal of big data**, Springer, v. 6, n. 1, p. 1–25, 2019.

ESCOVEDO, T.; KOSHIYAMA, A. **Introdução a Data Science: Algoritmos de Machine Learning e métodos de análise**. [S.l.]: Casa do Código, 2020.

FEUERRIEGEL, S. *et al.* Generative ai. **Business & Information Systems Engineering**, Springer, v. 66, n. 1, p. 111–126, 2024.

FONTANA, E. Introdução aos algoritmos de aprendizagem supervisionada. **Departamento de Engenharia Química, Universidade Federal do Paraná**, 2020.

GABRIEL, M. Você, eu e os robôs: pequeno manual do mundo digital. **São Paulo: Atlas**, p. 978–8597014372, 2017.

____. Inteligência artificial: do zero ao metaverso. **São Paulo: Atlas**, 2022.

GÉRON, A. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow**. [S.l.]: "O'Reilly Media, Inc.", 2022.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

HAMET, P.; TREMBLAY, J. Artificial intelligence in medicine. **Metabolism**, Elsevier, v. 69, p. S36–S40, 2017.

HEINRICH, K. *et al.* Process data properties matter: Introducing gated convolutional neural networks (gcnn) and key-value-predict attention networks (kvp) for next event prediction with deep learning. **Decision Support Systems**, Elsevier, v. 143, p. 113494, 2021.

JEBLICK, K. *et al.* Chatgpt makes medicine easy to swallow: an exploratory case study on simplified radiology reports. **European radiology**, Springer, v. 34, n. 5, p. 2817–2825, 2024.

KINGMA, D. P.; BA, J. **Adam: A Method for Stochastic Optimization**. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>.

KUNG, T. H. *et al.* Performance of chatgpt on usmle: potential for ai-assisted medical education using large language models. **PLoS digital health**, Public Library of Science, v. 2, n. 2, p. e0000198, 2023.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group UK London, v. 521, n. 7553, p. 436–444, 2015.

LIU, Z.; LIN, Y.; SUN, M. **Representation learning for natural language processing**. [S.l.]: Springer Nature, 2023.

LOCKE, S. *et al.* Natural language processing in medicine: a review. **Trends in Anaesthesia and Critical Care**, Elsevier, v. 38, p. 4–9, 2021.

LUND, B. D. *et al.* Chatgpt and a new academic reality: Artificial intelligence-written research papers and the ethics of the large language models in scholarly publishing. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 74, n. 5, p. 570–581, 2023.

MADEIRA, T. **Alan Turing e máquinas inteligentes — movimentorevista.com.br**. 2021. <https://movimentorevista.com.br/2021/06/alan-turing-e-maquinas-inteligentes/>.

MCCARTHY, J. *et al.* What is artificial intelligence. Stanford University, 2007.

NAST, C. **AI Sumo Wrestlers Could Make Future Robots More Nimble — wired.com**. 2017.

OPENAI. **Pricing API**. 2024. <https://openai.com/api/pricing/>.

OPENAI, R. Gpt-4 technical report. arxiv 2303.08774. **View in Article**, v. 2, n. 5, 2023.

PREECE, J.; POSSAMAI, V. **Design de interação :**. Porto Alegre :: Bookman,, 2005. (Tecnologia da Informação. Fundamentos). Título original: Interaction design: beyond human-computer interaction.

RADFORD, A. *et al.* Improving language understanding by generative pre-training. OpenAI, 2018.

____. Language models are unsupervised multitask learners. **OpenAI blog**, v. 1, n. 8, p. 9, 2019.

SAMUEL, A. L. Some studies in machine learning using the game of checkers. **IBM Journal of research and development**, IBM, v. 3, n. 3, p. 210–229, 1959.

SANTOS, M. H. D. **Introdução à inteligência artificial**. [S.l.]: Editora e Distribuidora Educacional S.A, 2021.

SCHRAMOWSKI, P. *et al.* Large pre-trained language models contain human-like biases of what is right and wrong to do. **Nature Machine Intelligence**, Nature Publishing Group UK London, v. 4, n. 3, p. 258–268, 2022.

SCHUBERT, M. C.; WICK, W.; VENKATARAMANI, V. Large language model-driven evaluation of medical records using medcheckllm. **medRxiv**, Cold Spring Harbor Laboratory Press, p. 2023–11, 2023.

SENNRICH, R.; HADDOW, B.; BIRCH, A. **Neural Machine Translation of Rare Words with Subword Units**. 2016. Disponível em: <https://arxiv.org/abs/1508.07909>.

SICSÚ, A.; SAMARTINI, A.; BARTH, N. **Técnicas de machine learning**. Editora Blucher, 2023. ISBN 9786555063974. Disponível em: <https://books.google.com.br/books?id=SE-tEAAAQBAJ>.

SIEBERS, P.; JANIESCH, C.; ZSCHECH, P. A survey of text representation methods and their genealogy. **IEEE Access**, IEEE, v. 10, p. 96492–96513, 2022.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. [S.l.]: MIT press, 2018.

TEUBNER, T. *et al.* Welcome to the era of chatgpt et al. the prospects of large language models. **Business & Information Systems Engineering**, Springer, v. 65, n. 2, p. 95–101, 2023.

TURING, A. Computação e inteligência. **FC Hansem, Trad.) Em J. de F. Teixeira (Org.), Cérebros Máquinas e Consciência: Uma Introdução à Ciência da Mente. São Carlos: Editora da UFSCar.(Trabalho original publicado em 1950)**, 1996.

VASWANI, A. *et al.* Attention is all you need. **Advances in neural information processing systems**, v. 30, 2017.

WANG, S.; SUMMERS, R. M. Machine learning and radiology. **Medical image analysis**, Elsevier, v. 16, n. 5, p. 933–951, 2012.

ZHANG, P.; BOULOS, M. N. K. Generative ai in medicine and healthcare: Promises, opportunities and challenges. **Future Internet**, MDPI, v. 15, n. 9, p. 286, 2023.

ZHOU, C. *et al.* Detecting hallucinated content in conditional neural sequence generation. **CoRR**, abs/2011.02593, 2020. Disponível em: <https://arxiv.org/abs/2011.02593>.

APÊNDICE A – DADOS UTILIZADOS NO FINE-TUNING

Este é o arquivo utilizado no processo de *fine-tuning*. Apenas uma parte do documento original foi incluída, dado que o arquivo completo possui aproximadamente 100 páginas.

Quadro 1 – Dados utilizados para o processo de *fine-tuning*

(continua)

Entrada	Laudo
Tendinopatia leve do supraespinhoso.	ECOGRAFIA DE OMBRO DIREITO Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. O tendão do supraespinhoso encontra-se levemente espessado e heterogêneo devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea
Bursite leve.	ECOGRAFIA DE OMBRO DIREITO Estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Pequena quantidade de líquido na bursa subacromial-subdeltoidea compatível com bursite.
Bursite leve.	ECOGRAFIA DE OMBRO DIREITO Estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Pequena quantidade de líquido na bursa subacromial-subdeltoidea compatível com bursite.

Quadro 1 – Dados utilizados para o processo de *fine-tuning*

(continuação)

Entrada	Laudo
Tendinopatia moderada do supraespinhoso e do subescapular. Ruptura intrassubstancial medindo 6 milímetros.	ECOGRAFIA DE OMBRO DIREITO Tendões do supraespinhoso e subescapular afilados e heterogêneos devido a tendinopatia. O tendão do supraespinhoso apresenta adicionalmente pequena ruptura intrassubstancial medindo cerca de 6 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea.
Artropatia degenerativa acromioclavicular.	ECOGRAFIA DE OMBRO ESQUERDO Estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea. Aumento de partes moles dorsalmente a articulação acromioclavicular.
Tendinopatia moderada no supraespinhoso. Ruptura parcial medindo 4 milímetros no supraespinhoso.	ECOGRAFIA DE OMBRO DIREITO Tendão do supraespinhoso afilado e heterogêneo devido a tendinopatia apresentando ruptura parcial em suas fibras insercionais medindo cerca de 4 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Pequeno acúmulo de líquido na bursa subacromial-subdeltoidea.
Tendinopatia leve no supraespinhoso e no subescapular.	ECOGRAFIA DE OMBRO ESQUERDO Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. Os tendões do supraespinhoso e subescapular encontram-se levemente heterogêneo devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea

Quadro 1 – Dados utilizados para o processo de *fine-tuning*

(continuação)

Entrada	Laudo
Tendinopatia moderada no supraespinhoso. Ruptura parcial no supraespinhoso medindo 4 milímetros.	ECOGRAFIA DE OMBRO DIREITO Tendão do supraespinhoso afilado e heterogêneo devido a tendinopatia apresentando ruptura parcial em suas fibras insercionais medindo cerca de 4 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Pequeno acúmulo de líquido na bursa subacromial-subdeltoidea.
Tendinopatia leve no supraespinhoso e subescapular	ECOGRAFIA DE OMBRO ESQUERDO Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. Os tendões do supraespinhoso e subescapular encontram-se levemente heterogêneo devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea.
Tendinopatia leve no supraespinhoso e subescapular.	ECOGRAFIA DE OMBRO ESQUERDO Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. Os tendões do supraespinhoso e subescapular encontram-se levemente heterogêneo devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea.
Tendinopatia moderada no supraespinhoso, infraespinhoso e subescapular. Artropatia com derrame acromioclavicular.	ECOGRAFIA DE OMBRO DIREITO Exame realizado em caráter de urgência. Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. Os tendões do supraespinhoso, infraespinhoso e subescapular encontram-se afilados e hipoecoicos devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea. Derrame articular com aumento de partes moles dorsalmente a articulação acromioclavicular.

Quadro 1 – Dados utilizados para o processo de *fine-tuning*

(continuação)

Entrada	Laudo
Tendinopatia no supraespinhoso. Ruptura completa no supraespinhoso medindo 6 milímetros.	ECOGRAFIA DE OMBRO DIREITO Tendão do supraespinhal heterogêneo devido a tendinopatia apresentando ruptura completa em suas fibras insercionais mais anteriores medindo cerca de 6 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Presença de pequena quantidade de líquido na bursa subacromial-subdeltoidea
Tendinopatia moderada no supraespinhoso. Ruptura parcial no supraespinhoso medindo 4 milímetros. Bursite leve no subacromial-subdeltoidea	ECOGRAFIA DE OMBRO ESQUERDO Tendão do supraespinhal heterogêneo devido a tendinopatia apresentando ruptura parcial na superfície bursal medindo cerca de 4 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Presença de pequena quantidade de líquido na bursa subacromial-subdeltoidea.
Tendinopatia moderada no supraespinhoso. Ruptura completa no supraespinhoso medindo 6 milímetros.	ECOGRAFIA DE OMBRO DIREITO Tendão do supraespinhal heterogêneo devido a tendinopatia apresentando ruptura completa em suas fibras insercionais mais anteriores medindo cerca de 6 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Presença de pequena quantidade de líquido na bursa subacromial-subdeltoidea.

Quadro 1 – Um exemplo de quadro com múltiplas páginas

(conclusão)

Entrada	Laudo
Bursite leve no subacromial-subdeltoidea.	ECOGRAFIA DE OMBRO DIREITO Estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Pequena quantidade de líquido na bursa subacromial-subdeltoidea compatível com bursite.
Tendinopatia leve no supraespinhoso e subescapular.	ECOGRAFIA DE OMBRO ESQUERDO Estruturas tendinosas do manguito rotador íntegras, sem sinais de rupturas. Os tendões do supraespinhoso e subescapular encontram-se levemente heterogêneo devido a tendinopatia. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Não se identifica líquido na bursa subacromial-subdeltoidea
Tendinopatia no supraespinhoso. Ruptura completa no supraespinhoso medindo 6 milímetros.	ECOGRAFIA DE OMBRO DIREITO Tendão do supraespinhal heterogêneo devido a tendinopatia apresentando ruptura completa em suas fibras insercionais mais anteriores medindo cerca de 6 mm. Demais estruturas tendinosas do manguito rotador íntegras, sem sinais de lesões ou rupturas. Tendão da cabeça longa do bíceps de aspecto ecográfico usual. Não há evidência de derrame articular. Presença de pequena quantidade de líquido na bursa subacromial-subdeltoidea.

Fonte: O Autor (2024).