

**UNIVERSIDADE VILA VELHA**

**Dylan Faria Robson**

**PLATAFORMA DE COMPARAÇÃO DE PERFORMANCE DE  
MODELOS NLP**

**VILA VELHA**

**2024**

DYLAN FARIA ROBSON

PLATAFORMA DE COMPARAÇÃO DE PERFORMANCE  
DE MODELOS NLP

**Trabalho de Conclusão de Curso submetido à Universidade Vila Velha, como requisito necessário para obtenção do grau de Bacharel em Sistemas de Informação**

Vila Velha, 10 de dezembro de 2024

UNIVERSIDADE VILA VELHA

DYLAN FARIA ROBSON

Esta Monografia foi julgada adequada para a obtenção do título de Bacharel em Sistemas de Informação, sendo aprovada em sua forma final pela banca examinadora:

---

Orientador: Prof. D.Sc. Jean-Rémi Bourguet  
Universidade Vila Velha - UVV

---

Prof. M.Sc. Erlon Pinheiro  
Universidade Vila Velha - UVV

Vila Velha, 10 de dezembro de 2024



*Dedico esse trabalho aos meus pais que não mediram  
esforços para me dar suporte para chegar até aqui.*



# Agradecimentos

Primeiramente, agradeço aos meus pais, sem os quais eu não poderia chegar tão longe.

Agradeço aos meus colegas de turma que percorreram essa jornada comigo, especialmente Thiago Baiense Peçanha Vieira, Lucas José Dias Gonçalves, Lucas Laranja Alcântara e Guilherme Lima Bernardes.

Meu enorme agradecimento ao professor e orientador Jean-Rémi Bourguet que me apresentou ao tema e me auxiliou no desenvolvimento do presente trabalho.

Agradeço ao meu amigo Marcelo Henrique Pires de Sousa Ramos que me ajudou nos momentos mais difíceis dessa jornada.

Por fim, agradeço a todos que me ajudaram direta ou indiretamente.





*"A educação é a arma mais poderosa  
que você pode usar para mudar o mundo."*  
NELSON MANDELA



# Resumo

*Este trabalho de conclusão de curso aborda o desenvolvimento de uma plataforma destinada à auxiliar na avaliação e extração de métricas de modelos de Processamento de Linguagem Natural, do inglês *Natural Language Processing* (NLP). Com o rápido avanço dos modelos de Inteligência Artificial, especialmente os *Large Language Models* (LLMs), houve um aumento significativo no interesse por parte da comunidade científica e tecnológica em avaliar a eficácia desses modelos. Apesar disso, não existe um padrão de métricas para medir a performance de maneira global, sendo de escolha de cada um como medir o desempenho. Com isso em mente, este trabalho objetiva o desenvolvimento de uma plataforma para extrair métricas de modelos NLP de forma padronizada em atividades específicas.*

**Palavras-chave:** Natural Language Processing. Large Language Models. Avaliação de modelos. Extração de métricas padronizadas.

# Abstract

This Bachelor's thesis addresses the development of a platform aimed at assisting in the evaluation and extraction of metrics for Natural Language Processing (NLP) models. With the rapid advancement of Artificial Intelligence models, especially Large Language Models (LLMs), there has been a significant increase in interest from the scientific and technological communities in evaluating the effectiveness of these models. Despite this, there is no standard set of metrics to measure performance globally, leaving it up to each individual to decide how to measure performance. With this in mind, this work aims to develop a platform to extract standardized metrics from NLP models in specific tasks.

**Keywords:** Natural Language Processing. Large Language Models. Model evaluation. Extraction of standardized metrics.

# Lista de ilustrações

Figura 1 – Diagrama de Venn. . . . .	36
Figura 2 – Comparação entre neurônio biológico e neurônio artificial. . . . .	37
Figura 3 – Ilustração de uma DNN. . . . .	38
Figura 4 – Processo de tokenização com transformers (simplificada). . . . .	39
Figura 5 – Demonstração da vetorização em cenários diferentes. . . . .	40
Figura 6 – Processo de retirada de <i>stop words</i> . . . . .	40
Figura 7 – Grid Search. . . . .	42
Figura 8 – Dualidade de NLU e NLG. . . . .	45
Figura 9 – Componentes de análise NLU. . . . .	46
Figura 10 – Análise morfológica da palavra “desqualificado”. . . . .	47
Figura 11 – Análise lexical. . . . .	47
Figura 12 – Componentes de análise NLG. . . . .	50
Figura 13 – Uso de NLP em Machine Translation. . . . .	52
Figura 14 – O processo de perguntas e respostas. . . . .	53
Figura 15 – Reconhecimento de entidade nomeada com spaCy. . . . .	55
Figura 16 – Processo de análise de sentimentos de reviews de produto. . . . .	56
Figura 17 – Exemplo de código de pré-processamento. . . . .	61
Figura 18 – Fluxograma de funcionamento geral da plataforma. . . . .	63
Figura 19 – Visão geral da plataforma. . . . .	64
Figura 20 – Tensor. . . . .	68
Figura 21 – Exemplo do uso da função “main”. . . . .	79
Figura 22 – Uso da função “main” após a adição de modelos customizados. . . . .	79
Figura 23 – Uso da GUI na plataforma. . . . .	80
Figura 24 – Documentação da função <code>criar_modelo</code> . . . . .	81
Figura 25 – Exemplo de refatoração de código. . . . .	81
Figura 26 – GUI na versão final. . . . .	82
Figura 27 – Fluxo padrão - 1. . . . .	83
Figura 28 – Fluxo padrão - 2. . . . .	84
Figura 29 – Fluxo padrão - final. . . . .	84
Figura 30 – Fluxo customizado - 1. . . . .	85
Figura 31 – Fluxo customizado - 2. . . . .	86
Figura 32 – Fluxo customizado - 3. . . . .	86
Figura 33 – Fluxo customizado - 4. . . . .	87
Figura 34 – Fluxo customizado - 5. . . . .	87
Figura 35 – Fluxo customizado - 6. . . . .	88
Figura 36 – Fluxo customizado - final. . . . .	88

Figura 37 – Fluxograma da plataforma. . . . . 89

# Lista de tabelas

Tabela 1 – Exemplo de métricas da plataforma. . . . .	62
Tabela 2 – Arquitetura dos 4 modelos. . . . .	72
Tabela 3 – Métricas NER. . . . .	90
Tabela 4 – Métricas Análise de Sentimentos. . . . .	91
Tabela 5 – Métricas de Perguntas e Respostas. . . . .	91
Tabela 6 – Métricas de todas as atividades. . . . .	92

# Lista de códigos

Código 1 – Código Python responsável por treinar e avaliar o modelo DistilBERT 76



# Lista de abreviaturas e siglas

## Lista de Siglas

ANN Artificial Neural Networks

API Application Programming Interface

BERT Bidirectional Encoder Representations from Transformers

BO Bayesian Optimization

CLUE Chinese Language Understanding Evaluation

CPU Central Processing Unit

DNN Deep Neural Networks

DP Deep Learning

GLUE General Language Understanding Evaluation

GPU Graphics Processing Unit

GPT Generative Pre-Training

GUI Graphical User Interface

IA Inteligência Artificial

IMDB Internet Movie Database

LLM Large Language Models

ML Machine Learning

MLM Masked Language Model

NER Named Entity Recognition

NLP Natural Language Processing

NLG Natural Language Generation

NLU Natural Language Understanding

NSP Next Sentence Prediction

Q&A Questions and Answers

RAM Random Access Memory

REPL Read-Evaluate-Print Loop

SQUAD Stanford Question Answering Dataset

SST2 Stanford Sentiment Treebank

VRAM Video Random Access Memory

# Sumário

1	INTRODUÇÃO	27
1.1	Justificativa	28
1.2	Objetivo	28
1.2.1	Objetivo Geral	28
1.2.2	Objetivos Específicos	28
1.3	Metodologia	29
1.4	Organização	29
2	TRABALHOS RELACIONADOS	31
2.1	GLUE	31
2.2	SuperGLUE	31
2.3	DataCLUE	32
2.4	Dynabench	32
2.5	Vega v2	33
3	GESTÃO DOS CONHECIMENTOS	35
3.1	<i>Machine Learning</i>	35
3.2	<i>Deep Learning</i>	36
3.3	Natural Language Processing	38
3.3.1	Criação e aprimoramento de um Modelo NLP	38
3.3.1.1	Escolha de Dataset Adequado	38
3.3.1.2	Pré-Processamento	39
3.3.1.3	Treinamento do Modelo	41
3.3.1.4	Avaliação	43
3.3.1.5	<i>Fine-Tuning</i>	44
3.3.2	<i>Natural Language Understanding e Natural Language Generation</i>	45
3.3.2.1	Principais componentes de NLU	45
3.3.2.1.1	Fonologia	46
3.3.2.1.2	Morfologia	46
3.3.2.1.3	Lexical	47
3.3.2.1.4	Sintaxe	48
3.3.2.1.5	Semântica	48
3.3.2.1.6	Discurso	48
3.3.2.1.7	Pragmática	49
3.3.2.1.8	Resumo dos Componentes de NLU	49
3.3.2.2	Principais componentes de NLG	50

3.3.2.2.1	Comunicador e Geração . . . . .	50
3.3.2.2.2	Componentes e Níveis de Representação . . . . .	50
3.3.2.2.3	Aplicação ou Comunicador . . . . .	51
3.3.2.2.4	Resumo dos Componentes de NLG . . . . .	51
3.3.3	Principais Atividades NLP . . . . .	51
3.3.3.1	<i>Machine Translation</i> . . . . .	51
3.3.3.2	Perguntas e Respostas . . . . .	51
3.3.3.3	Resumo de Textos . . . . .	53
3.3.3.4	Reconhecimento de Entidades Nomeadas . . . . .	54
3.3.3.5	Análise de Sentimentos . . . . .	55
3.3.4	Principais Modelos NLP . . . . .	57
3.3.4.1	GPT . . . . .	57
3.3.4.2	BERT . . . . .	57
3.3.4.3	Llama 2 . . . . .	58
3.4	<b><i>Large Language Models</i></b> . . . . .	<b>59</b>
4	<b>PROJETO</b> . . . . .	<b>61</b>
5	<b>DESENVOLVIMENTO</b> . . . . .	<b>65</b>
5.1	<b>Hardware e Sistema Operacional</b> . . . . .	<b>65</b>
5.2	<b>Tecnologias</b> . . . . .	<b>65</b>
5.2.1	Python . . . . .	65
5.2.2	<i>Hugging Face</i> . . . . .	66
5.2.2.1	Transformers . . . . .	66
5.2.2.2	<i>Datasets</i> . . . . .	66
5.2.2.3	Evaluate . . . . .	67
5.2.3	PyTorch . . . . .	68
5.2.4	Scikit-learn . . . . .	68
5.2.5	Pandas . . . . .	69
5.2.6	Anaconda . . . . .	69
5.2.7	Jupyter Notebook . . . . .	69
5.2.8	PySimpleGUI . . . . .	70
5.3	<b>Modelos Utilizados</b> . . . . .	<b>70</b>
5.3.1	BERT . . . . .	71
5.3.2	ALBERT . . . . .	71
5.3.3	DistilBERT . . . . .	71
5.3.4	GPT2 . . . . .	72
5.4	<b>Atividades Contempladas</b> . . . . .	<b>72</b>
5.5	<b>Datasets</b> . . . . .	<b>73</b>
5.5.1	CoNLL2003 . . . . .	73

5.5.2	SST2 . . . . .	74
5.5.3	IMDB . . . . .	74
5.5.4	SQUaD . . . . .	74
<b>5.6</b>	<b>Métricas utilizadas . . . . .</b>	<b>75</b>
<b>5.7</b>	<b>Etapas de Desenvolvimento . . . . .</b>	<b>75</b>
5.7.1	Avaliação Individual . . . . .	76
5.7.2	Agregação e Primeira Versão . . . . .	78
5.7.3	Modularidade . . . . .	79
5.7.4	Criação de uma GUI . . . . .	79
5.7.5	Documentação . . . . .	80
5.7.6	Melhorias Adicionais . . . . .	80
5.7.6.1	Versão Final . . . . .	82
<b>6</b>	<b>RESULTADOS . . . . .</b>	<b>83</b>
<b>6.1</b>	<b>Fluxo da Plataforma . . . . .</b>	<b>83</b>
6.1.1	Fluxo de Modelos Padrões . . . . .	83
6.1.2	Fluxo de Modelos Customizados . . . . .	84
6.1.3	Fluxograma . . . . .	89
<b>6.2</b>	<b>Métrica com Resultados . . . . .</b>	<b>90</b>
6.2.1	Reconhecimento de Entidades Nomeadas . . . . .	90
6.2.2	Análise de Sentimentos . . . . .	90
6.2.3	Perguntas e Respostas . . . . .	91
6.2.4	Todas as Atividades e Modelos . . . . .	91
<b>7</b>	<b>CONCLUSÃO . . . . .</b>	<b>93</b>
<b>7.1</b>	<b>Considerações Finais . . . . .</b>	<b>93</b>
<b>7.2</b>	<b>Trabalhos Futuros . . . . .</b>	<b>94</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>95</b>



# 1 Introdução

Uma linguagem consiste em um conjunto finito de frases construídas a partir de um alfabeto também finito. De forma análoga, o *Natural Language Processing* (NLP) é uma coleção de técnicas computacionais finitas e uma das áreas de Inteligência Artificial (IA) que objetivam representar a linguagem humana para o entendimento por máquinas. Nesse contexto, IA é um segmento da ciência da computação que tem como objetivo simular a Inteligência humana (Chowdhary, 2020).

Para avaliar se um algoritmo pode, de fato, ser considerado inteligente, Turing (1950) propôs o *Jogo da Imitação*, também conhecido como *Teste de Turing*. O teste consiste em um homem e uma máquina que são colocados em quartos distintos e uma terceira pessoa, o examinador, que faz perguntas aos dois. Sem contato direto ou conhecimento prévio de quem é a máquina e quem é o ser humano, se o examinador não conseguir distinguir entre os dois, o algoritmo é considerado inteligente (Turing, 1950).

Nesse mesmo âmbito, avanços significativos em NLP começaram no final de 1940, em uma época que o próprio termo referido não existia formalmente. Naquela época, já existiam trabalhos focados em tradução automática de idiomas, conhecidos como *Machine Translation*. Os principais representantes linguísticos da época eram russo e inglês (Khurana et al., 2023). Em 1966 houve o primeiro Inverno de IA, momento no qual se teve grandes cortes de investimento em pesquisas pelos governos, principalmente, dos Estados Unidos e Reino Unido. As duas principais causas foram as observações feitas pela ALPAC - 1966 e Lighthill - 1973, que mostravam uma perspectiva fraca de evolução nas pesquisas (Toosi et al., 2021).

Após essa quebra de expectativa, nos anos seguintes houveram altos e baixos, com investimentos do Japão e EUA, em meados de 1980. Porém, um segundo Inverno de IA começou com a crise econômica nos EUA em 1987 (Haenlein; Kaplan, 2019). Na década de 90, houveram avanços significativos com o surgimento da sexta geração de computadores como o Deep Blue que conseguiu vencer o campeão mundial de xadrez em seu próprio jogo (Campbell; Hoane; Hsu, 2002). Com essa perspectiva, a chegada do novo milênio resultou no desenvolvimento de algoritmos mais complexos e robustos com usos nos mais variados cenários, como educação, Indústria 4.0 e jogos.

Nessa mesma tendência, em 2018, a OpenAI, empresa originalmente sem fins lucrativos e focada em IA, fez sua primeira publicação sobre o *Generative Pre-trained Transformer* (GPT) (Radford et al., 2018). GPT é um modelo de *Machine Learning* baseado em rede neural pré-treinada para responder perguntas de usuários humanos. Em 2020 a empresa anunciou o desenvolvimento de seu primeiro modelo comercial, o GPT-3,

um modelo de linguagem treinado com um grande conjunto de dados providos da internet, que ficou disponível para o público em 2022.

## 1.1 Justificativa

Após a rápida popularização do modelo GPT-3, chamado *ChatGPT*, modelos de IA, especialmente de *Large Language Models* (LLM), estão sendo desenvolvidos de forma acelerada. Essa revolução teve grande impacto e atraiu a atenção das comunidades de NLP (Qin et al., 2023). No entanto, surgiram questões sobre como medir a eficácia desses modelos e identificar quais são mais eficientes para determinadas tarefas.

Medir a eficiência de processos e ferramentas é essencial para garantir seu sucesso e melhor entendimento. Por isso, a utilização de dados quantitativos para ajudar na classificação de modelos de IA, como *precision*, *accuracy*, *recall* e *F1-score*<sup>1</sup> são extremamente relevantes. Esses dados possibilitam análises comparativas, identificação de pontos fortes e fracos e melhoria em hiperparâmetros, para citar algumas das utilidades.

Nesse sentido, diversos estudos comparativos têm surgido ao longo dos últimos anos com o objetivo de definir quais são os melhores modelos de NLP. Apesar do objetivo comum, cada estudo estabelece uma *pipeline* para testes, o que resulta em métricas similares, mas obtidas a partir de procedimentos diferentes e não intercambiáveis (Zhou et al., 2021).

## 1.2 Objetivo

### 1.2.1 Objetivo Geral

O objetivo geral do presente trabalho é desenvolver uma plataforma para auxiliar na comparação entre modelos de NLP, através da extração de métricas de forma padronizada em atividades selecionadas. Para isso, o usuário utilizará um modelo já treinado, um código de pré-processamento criado por ele e a possibilidade de especificar o *dataset* utilizado no teste ou usar o padrão. Como resultado serão retornadas as métricas de forma padronizada para que o teste possa ser repetido com outros modelos e pré-processamentos, mas mantendo o mesmo *dataset*.

### 1.2.2 Objetivos Específicos

Considerando o desenvolvimento do trabalho e o objetivo geral apresentado, destacam-se os seguintes objetivos específicos:

- Pesquisar sobre NLP, seus principais conceitos, atividades e aplicações;

---

<sup>1</sup> Conformer explicado em 3.3.1.4.



- Definir as métricas e metodologias adequadas para a avaliação de performance de modelos de NLP;
- Desenvolver e modelar uma plataforma capaz de realizar as avaliações necessárias para identificar os parâmetros outrora estabelecidos de cada modelo;
- Analisar os resultados obtidos.

## 1.3 Metodologia

Para este trabalho foi escolhida a metodologia exploratória, uma vez que ela apresenta maior flexibilidade, permitindo a busca pelo conhecimento e conduzindo-o de forma a explicar satisfatoriamente conceitos anteriormente dúbios (Raupp; Beuren, 2006). Nessa esfera, é comum fazer o levantamento de pesquisas e outras fontes bibliográficas para “estimular a compreensão” (Selltiz et al., 1975). É também disseminado a pesquisa com indivíduos com experiência no assunto, porém, devido à natureza exploratória desse estudo, seguirá-se-a com fontes de teor teórico.

Após a coleta de informações e consolidação dos conhecimentos necessários e relevantes, será desenvolvido uma plataforma para comparação de modelos NLP.

## 1.4 Organização

O presente trabalho está organizado em 7 capítulos. O capítulo 2 se destinará aos trabalhos relacionados com a pesquisa de fontes bibliográficas para referencial teórico de publicações com objetivos similares a este. Já o capítulo 3 apresentará a gestão de conhecimentos, base teórica fundamental para o desenvolvimento do trabalho proposto. Ademais, no capítulo 4 será feita uma proposta de solução. O capítulo 5 demonstrará as tecnologias e etapas envolvidas no desenvolvimento do trabalho. Além disso, o capítulo 6 focará nos resultados obtidos com imagens, um fluxograma e tabelas que demonstram a utilização da plataforma. Por fim, o capítulo 7 será a conclusão do trabalho.

## 2 Trabalhos relacionados

Este capítulo examina trabalhos relacionados cujo objetivo é possibilitar a comparação padronizada de modelos de NLP e áreas correlatas.

### 2.1 GLUE

Wang et al. (2019b) apresentam o GLUE (*General Language Understanding Evaluation*), um benchmark para avaliação de modelos de *Natural Language Understanding* (NLU). A proposta inclui uma plataforma online para avaliação de modelos, exigindo apenas que o modelo consiga processar sentenças de forma individual e em pares. O GLUE disponibiliza diversos *datasets* para testes e inclui nove tarefas de entendimento de sentenças em inglês, como CoLA, que avalia a aceitabilidade gramatical de uma sentença, e SST-2, que verifica a compreensão de sentimentos. Os resultados são apresentados em uma escala de 0 a 100, com métricas padronizadas distintas, como a Correlação de Matthews para CoLA e a acurácia para SST-2.

### 2.2 SuperGLUE

Com o sucesso do GLUE, Wang et al. (2019a) construiu um novo benchmark, o SuperGLUE, uma versão atualizada com atividades mais desafiadoras que o seu antecessor. Após pouco mais de um ano do lançamento do GLUE, diversos modelos superaram o desempenho humano em diferentes tarefas. Com isso em mente, foi proposto o SuperGLUE que oferece testes mais rigorosos, sendo oito tarefas, duas do GLUE, um conjunto de datasets padrão e uma ferramenta para análise de resultados. Ademais, o novo benchmark apresenta melhorias no formato das tarefas, não se limitando apenas a classificação de sentenças e pares de sentença.

Além disso, o SuperGLUE possibilita a resposta de tarefas e resolução de correferência, novas estimativas para o desempenho humano em tarefas e ferramentas modulares. Dentre as tarefas disponibilizadas, BoolQ apresenta um texto curto e uma pergunta de sim ou não sobre ele e COPA que, dado uma premissa, precisa determinar a causa ou efeito entre duas opções. Quanto as métricas, dependendo da tarefa, há uma métrica padronizada distinta em escala de 0 a 100. Para facilitar o uso do SuperGLUE, foi desenvolvido o jiant, um software modular construído com Pytorch e AllenNLP e o pacote de transformers. jiant suporta a avaliação de modelos customizados e métodos de treinamento para as tarefas do benchmark. Outrossim, há o suporte para modelos pre treinados como OpenAI GPT e BERT, além de múltiplos-estágios e aprendizagem multi-tarefas.

## 2.3 DataCLUE

Xu et al. (2021) desenvolveu o DataCLUE - um benchmark *Data-centric AI* para modelos de NLP baseado no CLUE - Chinese Language Understanding Evaluation (Xu et al., 2020). Em outras palavras, seu objetivo é possibilitar e ajudar na avaliação e melhora da qualidade de datasets. Com isso em mente, a plataforma disponibiliza uma série de recursos para modificar e aprimorar um dataset como o *Random Flip* - substituição de labels reais do dataset de forma aleatória e *Hard Flip* - utiliza um modelo para prever os dados e então definir os cinco labels mais prováveis por similaridade, depois substitui o label real por um similar de forma aleatória. Essas técnicas têm como objetivo dificultar o dataset para possibilitar uma avaliação de desempenho do modelo em cenários reais e desafiadores. As atividades propostas têm as características *Representative*, *Challenging*, *Resource Friendly* e *Academic Friendly*.

O DataCLUE também apresenta integração com o DataCLUE Toolkit que permite ler, gerar resultados e validá-los de forma simples e sem muito código. A principal métrica utilizada para avaliação dos modelos foi o macro f1 score.

## 2.4 Dynabench

Kiela et al. (2021) demonstra com o o Dynabench, uma nova visão e proposta para avaliação de modelos NLP. O Dynabench é uma plataforma *open-source* e web para a criação dinâmica de dataset e o benchmark de modelos. Essa proposta foi feita tendo em vista que a utilização de métricas de forma isolada pode tendenciar uma conclusão, como por exemplo, de que modelos estão com desempenho superior ao de humanos, porém ao colocar essa tese em teste é possível verificar lacunas em conhecimentos linguísticos básicos.

Com isso em mente, diferente de benchmarks convencionais, ele propõe teste em múltiplas rodadas, tendo um modelo de referência, previamente treinado, e utilizando-se da interação com humanos no processo de avaliação. O processo é feito em ciclos de rodadas e em atividades diferentes, na qual um ou mais modelos de referência são colocados para interagir com seres humanos que objetivam encontrar situações nas quais os modelos tenham dificuldade, incerteza ou mesmo errem e identificar o erro e o porquê de sua ocorrência.

Estas situações são verificadas por outros humanos para serem confirmadas. É a partir desse processo, chamado de *Dynamic Adversarial Data Collection* que o Dynabench propõem a coleta de dados para treino e teste em um ciclo virtuoso para aprimorar os modelos. Com isso em mente, a utilização do “pior cenário” é feito de forma proposital, dado o fato de que esse cenário é muito mais interessante para análises de desempenho

em situações de interação real com humanos, além de proporcionar um cenário ideal para identificar e preencher as lacunas dos modelos.

Foram escolhidas inicialmente 4 atividades para serem avaliadas - *Natural language inference*, *Question answering*, *Sentiment analysis* e *Hate speech detection*, todas feitas em rodadas. Para a métrica utilizada para medir desempenho em cada atividade, foi escolhida o vMER - validated Model Error Rate ou taxa de erro de modelo validada em português. o vMER consiste da divisão do total de erros validados por humanos pelo total de exemplos.

## 2.5 Vega v2

Durante o estudo de Zhong et al. (2022), foi desenvolvido o modelo pré-treinado Vega v2 com 6 bilhões de parâmetros e arquitetura *transformers*. Ademais, para maior eficiência no pré-treino do modelo, foi utilizado as técnicas de *Masked Language Modeling* e *Self-Evolution-Learning*. Este último é um diferencial com relação aos outros modelos, permitindo que ele faça o mascaramento de forma não aleatória a partir da identificação de tokens que não foram aprendidos de forma satisfatória. Ademais, foram apresentadas 3 formas para fazer o aperfeiçoamento do modelo treinado (*fine-tuning*): *Transductive Fine-tuning*, *Prompt-Tuning* e *Adversarial Fine-Tuning*.

Para avaliar o modelo, foi escolhido o benchmark SuperGLUE por ser um dos mais conhecidos e consistentes disponíveis. Os resultados foram muito promissores, demonstrando uma incrível capacidade em diferentes tarefas, de forma a superar o *record score* de outros modelos, como o PaLM 540B e ST-MoE-32B no dia 8 de outubro de 2022. Por fim, a utilização do SuperGLUE se mostrou especialmente relevante nesse caso, ao proporcionar uma plataforma para avaliação do modelo de forma padronizada e eficiente, permitindo assim a comparação de modelos.

## 3 Gestão dos conhecimentos

Neste capítulo serão descritos os conceitos, fundamentais, necessários para o entendimento do presente trabalho desenvolvido.

### 3.1 *Machine Learning*

A ideia de uma máquina que pode pensar por conta própria volta no tempo até a Grécia antiga. Nesse contexto, apesar das primeiras IAs serem capazes de resolver problemas matemáticos complexos, tudo que elas podiam fazer era baseado em regras matemáticas, não sendo capazes de resolver problemas reais complexos do dia a dia. Para contornar essa situação, a ideia de uma máquina que pode aprender por conta própria parecia a solução mais eficiente (Hao; Zhang; Ma, 2016).

A partir dessa ideia surgiu o *Machine Learning* (ML), um conjunto de técnicas que procuraram prover a habilidade de melhoria de desempenho em um sistema, sem a necessidade de programar explicitamente todas as regras, com base em observações e compreensão de padrões (Bishop, 2006). Estes por sua vez estão presentes em formato de dados, a partir dos quais um sistema aprende de forma autônoma (Zhou, 2021). Nesse cenário, segue abaixo lista com os 3 processos principais de ML e exemplo de suas aplicações:

- **Aprendizado Supervisionado** - Esse processo requer dados que tenham uma classificação feita por humanos ou valor alvo (Janiesch; Zschech; Heinrich, 2021). É comumente utilizado no processo de classificação de imagens, como em lojas para categorizar visitantes como “observadores” e “compradores”.
- **Aprendizado não Supervisionado** - Utiliza dados não classificados que devem ter seus padrões identificados de forma autônoma. A aplicação mais comum é em clusterização, que junta dados em grupos por similaridade.
- **Aprendizado por Reforço** - Baseado em tentativa e erro, um modelo recebe recompensas quando acerta e punições quando erra. Uma de suas aplicações é no desenvolvimento de IA para jogos (Taulli; Oni, 2019).

O treinamento de um modelo de ML permite a estruturação de novos conhecimentos a partir dos dados disponibilizados. Esse conhecimento é então assimilado para que possa ser utilizados em tarefas, imitando a capacidade dos seres vivos de forma a superar desafios outrora impossíveis de serem concluídos (Chowdhary, 2020). Com essa visão de solução de problemas, ML pode ser utilizado em tarefas como agendamento de manutenção preventiva,

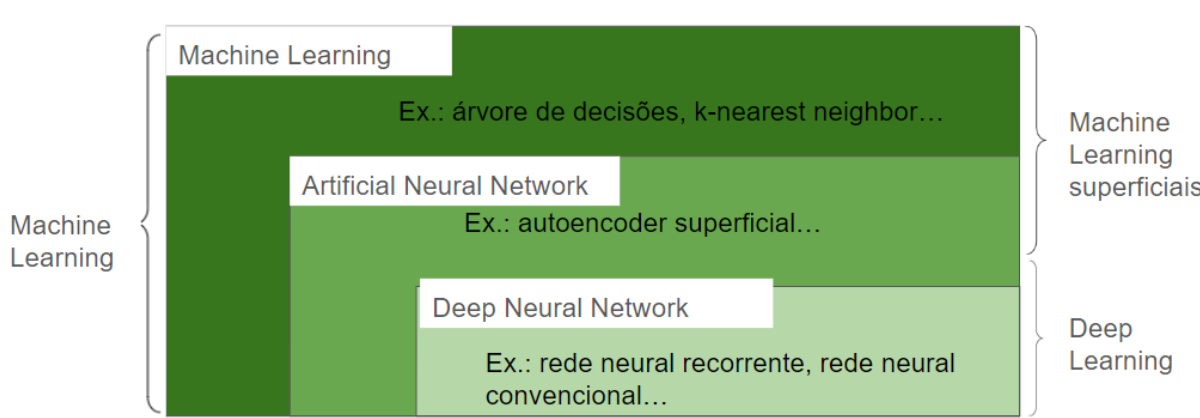
recrutamento de empregados, experiências customizadas de usuários, dentre outros (Taulli; Oni, 2019).

Ademais, outro processo importante em ML é o *fine-tunning* que consiste no treino de um modelo pré-treinado. Neste sentido, é possível aperfeiçoar o modelo para atividades específicas, adquirindo novos conhecimentos na área desejada e mantendo seu entendimento geral, útil em atividades gerais. Dessa forma, um modelo *fine-tuned* pode ser utilizado em atividades como criação de propagandas de forma personalizadas e pesquisa de mercado e marketing (Wu et al., 2022).

## 3.2 Deep Learning

Primeiramente, é importante estabelecer as diferenças entre ML e *Deep Learning* (DL), já que esses conceitos são comumente confundidos (Taulli; Oni, 2019). Enquanto ML descreve de forma genérica a capacidade de um sistema em aprender com base em observações, sem ser diretamente programado (Bishop, 2006), o DL é uma subárea de ML (Taulli; Oni, 2019), parte da família de *Artificial Neural Networks* (ANN) e contido em *Deep Neural Networks* (DNN). O diagrama de Venn na Figura 1 representa esses conjuntos.

Figura 1 – Diagrama de Venn.

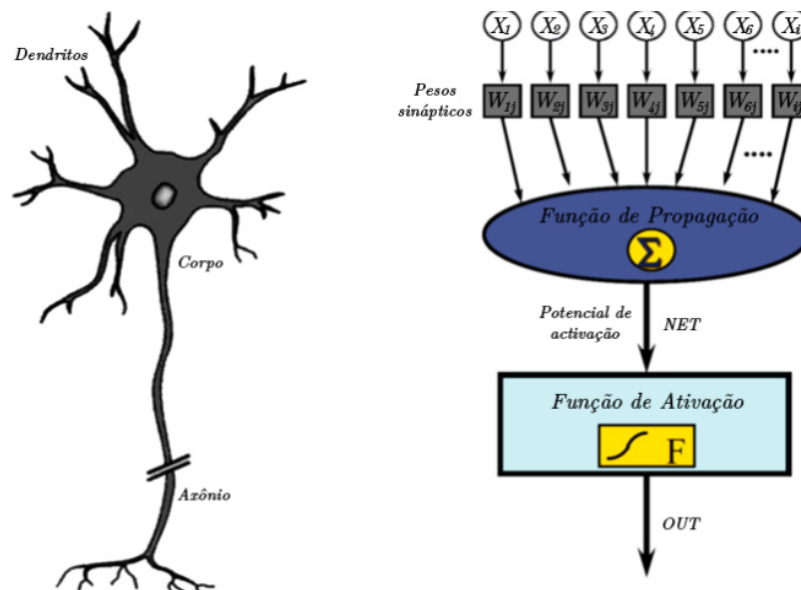


Fonte: Janiesch, Zschech and Heinrich (2021) (Adaptado pelo Autor).

A ANN é uma estrutura flexível, o que permite que seja usado com os 3 tipos de aprendizado em ML, inspirado por princípios do sistema biológico, consistindo de representações matemáticas dos processos conduzidos em unidades chamadas de neurônios artificiais. De forma similar as sinapses do cérebro, os neurônios artificiais transmitem sinais aos seus vizinhos, que podem ser fortalecidos ou enfraquecidos em decorrência de um peso que é continuamente ajustado. Outrossim, o processamento de sinais só ocorre se um limiar for excedido, conforme uma função de ativação. Os neurônios, por sua vez, são organizados em redes com diferentes camadas, podendo conter zero ou mais camadas escondidas que ficam responsáveis por compreender um mapeamento não linear de *input* (entrada) e

*output* (saída). Hiperparâmetros como número de camadas e camadas escondidas, taxa de aprendizagem e função de ativação não podem ser aprendida pelo modelo e devem ser especificados de forma manual ou por uma rotina de otimização (Janiesch; Zschech; Heinrich, 2021). A Figura 2 ilustra um neurônio biológico na esquerda e um neurônio artificial na direita.

Figura 2 – Comparação entre neurônio biológico e neurônio artificial.

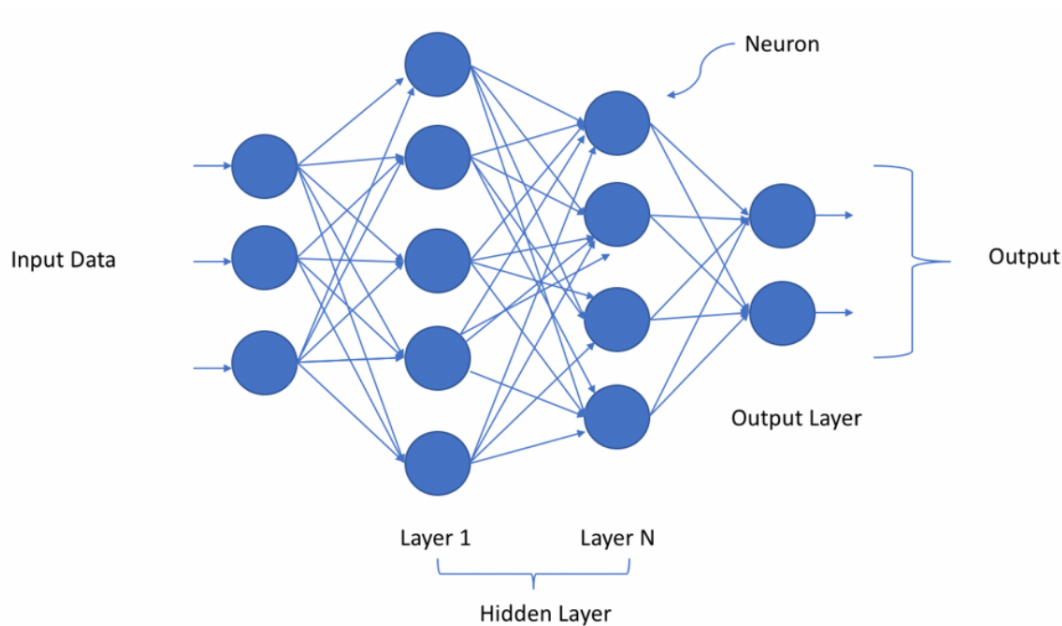


Fonte: Researchgate (2024).

DNN, normalmente, consiste de mais de uma camada escondida, organizado em uma rede profundamente interconectada. Além disso, é comum a presença de neurônios avançados neste tipo de rede, com a possibilidade de operações mais complexas, como a convolução, ou ativações múltiplas em um único neurônio. Essas características permitem que sejam alimentados com dados não tratados e automaticamente identifiquem a sua representação necessária para a aprendizagem. Essa é a principal capacidade de *Deep Neural Networks*, conhecida como *Deep Learning*. Com isso em mente, ANN simples podem ser chamadas de superficiais, uma vez que não contém tal capacidade, por outro lado, seu processo de tomada de decisão pode ser compreendido por humanos, sendo uma “caixa branca”, já ML mais complexos como DNN são de extrema difícil compreensão e por isso chamados de “caixa preta” (Janiesch; Zschech; Heinrich, 2021). A Figura 3 mostra de forma simplificada a estrutura de uma DNN.

DL são especialmente úteis quando se há muitos dados e muitas dimensionalidades, por essa razão que modelos DL performam melhor do que ML em cenários em que há texto, vídeo, áudio e fala (LeCun; Bengio; Hinton, 2015). De forma oposta, situações em que há uma quantidade limitada de dados e poucas dimensionalidades, ML superficiais são mais eficientes e produzem um resultado melhor (Zhang; Ling, 2018).

Figura 3 – Ilustração de uma DNN.



Fonte: Towardsdatascience (2024).

## 3.3 Natural Language Processing

*Natural Language Processing* é um domínio de pesquisa e desenvolvimento acadêmico, baseado em tecnologias, que objetiva possibilitar o entendimento de texto de linguagem natural humanas para máquinas (Chowdhary, 2020). Com isso em mente, a criação e treinamento de um modelo de NLP envolve diferentes etapas, cada um com suas particularidades.

### 3.3.1 Criação e aprimoramento de um Modelo NLP

#### 3.3.1.1 Escolha de Dataset Adequado

A escolha do conjunto de dados é crítico para o treinamento de modelos de ML, como os de NLP, uma vez que eles são responsáveis por prover informações na fase de treino e possibilitam a avaliação de desempenho dos modelos treinados. Por essa razão, a escolha de dataset feita de forma equivocada pode ter efeitos catastróficos no desempenho de modelos.

Nesse âmbito, a documentação da criação e uso dos dados se tornam relevantes para permitir a compreensão das fontes de dados, suas aplicações recomendadas e a extensão dos dados, de forma a identificar possíveis lacunas nas informações. Ademais, a documentação possibilita uma maior reprodutibilidade de resultados no treinamento de modelos, ao permitir a identificação das características dos dados aplicados durante o treino.

Além disso, é importante observar e especificar de forma adequada as licenças dos



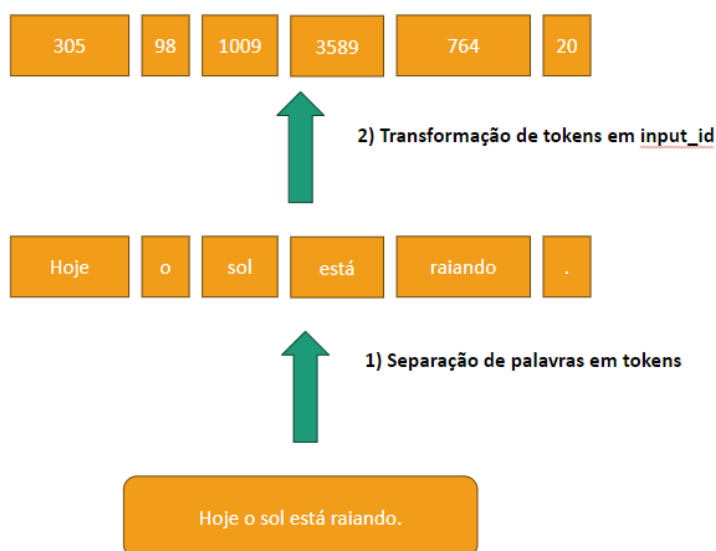
dados utilizados, de forma a evitar problemas legais ou mesmo éticos (Gebru et al., 2021).

### 3.3.1.2 Pré-Processamento

Apesar da grande capacidade de entendimento e geração de sentenças, os modelos NLP não são capazes de processar texto em sua forma pura, string, mas formatos específicos dependendo da técnica aplicada em sua criação. Nesse cenário, é necessário fazer a conversão de texto puro em uma abordagem aceitável pelos modelos selecionados, através do pré-processamento de dados e informações, o que leva em conta o formato, qualidade e padrões adotados.

Nesse sentido, é comum adotar a tokenização, o que envolve a quebra de sentenças em unidades menores, chamadas de tokens, que representam palavras e sinais de pontuação. Uma possível implementação é a partir da biblioteca *Transformers*, que implementa uma classe para tokenização, na qual ocorre a separação dos tokens. Após isso, ocorre a transformação do texto (string) em unidades numéricas (integer) - o `input_id`, de forma que cada um representa um significado de acordo com o contexto e um vocabulário contendo a relação texto-token, que pode ser modificado (Wolf et al., 2020). A Figura 4 representa o processo de tokenização de forma simplificada

Figura 4 – Processo de tokenização com transformers (simplificada).

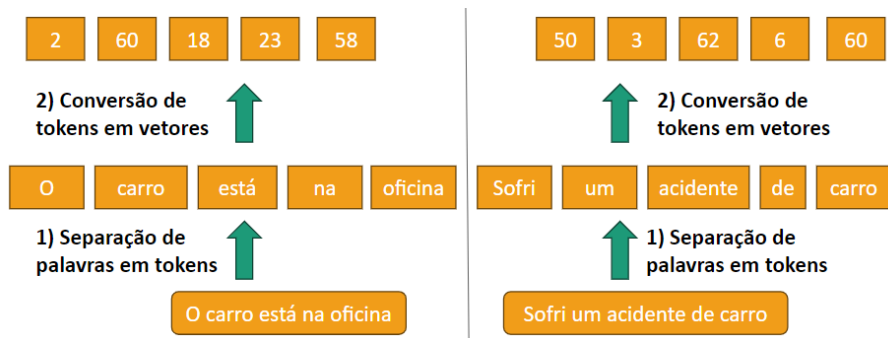


Fonte: Elaborado pelo Autor.

De forma similar, a vetorização, também chamada de *embeddings*, é o processo de representar palavras, tokens, ou mesmo frases a partir de números, chamados de vetores, de forma que palavras parecidas tenham vetores próximos. Apesar disso, diferente da tokenização com *Transformers*, a vetorização não leva em conta o contexto, representando de forma igual a mesma palavra em situações onde o seu significado pode não ser o mesmo. Dessa forma, essa estratégia foca no entendimento lexical das palavras de forma individual, sem levar em conta o conjunto que compõem a sentença (Neelakantan et al., 2015).

Nesse mesmo âmbito, as frases “O carro está na oficina” e “Sofri um acidente de carro” terão a palavra “carro” com o mesmo significado, apesar da primeira se referir ao veículo enquanto objeto e a segunda enquanto meio de transporte, assim como demonstrado na Figura 5.

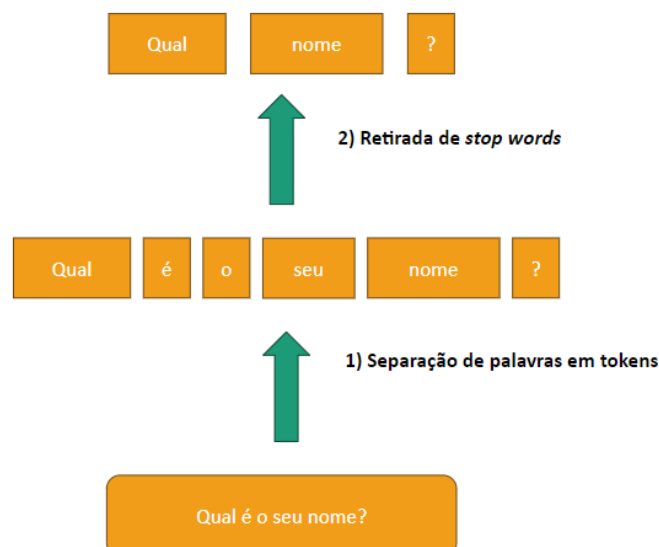
Figura 5 – Demonstração da vetorização em cenários diferentes.



Fonte: Elaborado pelo Autor.

Outra característica importante durante o pré-processamento é avaliar a necessidade das *stop words*. Essas são palavras cotidianas que são utilizadas para respeitar as regras gramaticais da língua, mas não carregam significado por si mesmas como “o”, “a”, “de”, “para”, “em” e “com”. Ao serem retiradas, o significado da sentença ainda se mantém o mesmo, apesar de não fazer sentido gramatical para humanos. Dessa forma, a remoção dessas palavras depende da atividade realizada e o método escolhido, em situações onde há grande sensibilidade a forma estrutural do texto, como identificação de autoria, essas palavras são extremamente relevantes, enquanto em abordagens de resumo elas podem adicionar complexidade ao texto e serem removidas. A Figura 6 mostra como o processo de retirada de *stop words* é feita.

Figura 6 – Processo de retirada de *stop words*.



Fonte: Elaborado pelo Autor.

### 3.3.1.3 Treinamento do Modelo

Uma vez que os dados estão prontos e já se escolheu os objetivos e tecnologias para a criação do modelo, chega a hora de treiná-lo. Dependendo da tecnologia adotada, o treinamento pode ocorrer de formas diferentes, como a implementação de estruturas de decisões e enlaçamentos com métodos de nível mais baixo como a biblioteca Python *Pytorch* e *Tensorflow*, como também pode ser feito com a chamada de funções e classes prontas, como a adoção da classe *Trainer* da biblioteca *Transformers*.

Apesar das diferenças em implementação, os hiperparâmetros são comuns a todas, sendo as características de alto nível de abstração que devem ser definidas antes de começar o treinamento do modelo, para especificar a arquitetura e estrutura será criada, pois diferente dos parâmetros, elas não podem ser aprendidas ao longo do treinamento (Andonie, 2019). Dessa forma, seguem os hiperparâmetros mais comuns no contexto de NLP e ANN:

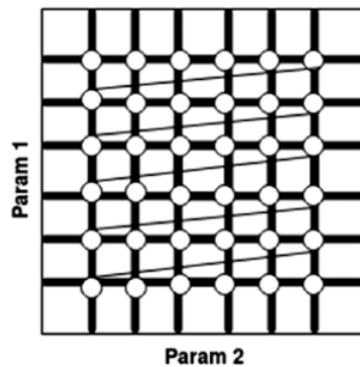
- *Learning Rate* - É um dos hiperparâmetros mais importantes, controla o tamanho dos passos do modelo feito em cada iteração (Yang; Shami, 2020).
- *Epochs* - Defini a quantidade de passagens por todos os dados de um conjunto de dados ao longo do treinamento.
- *Batch Size* - especifica o número de amostras retiradas do dataset necessários para atualizar os parâmetros do modelo.
- *Number of Layers* - Defini o número de camadas, representando a profundidade e habilidade de aprender recursos mais complexos (Agrawal, 2021).
- *Number of Nodes* - Representa o número de neurônios na primeira e última camada que devem ser iguais ao número de inputs e outputs.
- *Loss Function* - Uma função que tem como objetivo calcular a diferença entre os valores reais e os valores previstos.
- *Optimization Algorithm* - É um algoritmo que tem como objetivo auxiliar na definição dos hiperparâmetros, com o objetivo de alcançar os valores ótimos.

Com uma quantidade cada vez maior de hiperparâmetros, foi necessário a criação da otimização de hiperparâmetros que objetiva encontrar os hiperparâmetros que provem a melhor performance para um modelo em um dataset de validação (Andonie, 2019). Dentre os métodos de otimização, se destacam:

- *Grid Search* - É uma técnica baseada em força bruta na qual todas as possibilidades devem ser testadas. Apesar de, teoricamente, prover ao fim do processo a certeza

do melhor conjunto de hiperparâmetros, na prática isso se torna inviável para muitos hiperparâmetros, já que o trabalho aumenta exponencialmente para cada um adicionado (Andonie, 2019). A Figura 7 mostra o processo de procura com os parâmetros 1 e 2.

Figura 7 – Grid Seach.



Fonte: Agrawal (2021).

- *Random Search* - A busca aleatória, *Random Search*, permite uma procura mais rápida e eficiente do que a *Grid Search* em cenários de alta dimensionalidade de hiperparâmetros, por alterar-los de forma pontual, sem a necessidade de testar todas as possibilidades. Por outro lado, essa técnica pode deixar passar o valor ótimo de um hiperparâmetros, sendo necessário fazer um balanço de forma eficiente entre o tempo disponível e os resultados da busca (Andonie, 2019).
- *Bayesian optimization* - O *Bayesian optimization* (BO) é um algoritmo iterativo que determina os pontos de avaliação com base nos resultados obtidos anteriormente. Neste sentido, BO utiliza dois componentes chaves: Um modelo surrogate e uma chave de aquisição. O modelo surrogate objetiva incluir todos os pontos observados em uma função objetiva. Após obter a previsão de distribuição de probabilidades do modelo surrogate, a função de aquisição determina o uso de diferentes pontos a partir do balanço de exploração e exploração (Bourguet; Silva; Oliveira, 2021). Exploração é o processo de coleta de amostras de uma área que ainda não foi coletada e exploração é o processo de coleta de amostras de uma zona atualmente promissora com base nas interações passadas. O balanço entre exploração e exploração é essencial para detectar um hiperparâmetro que pode ser ótimo e não perder configurações ainda melhores em zonas não exploradas. Os procedimentos podem ser colocados na seguinte ordem:
  1. Criação de um modelo surrogate probabilístico com base em uma função objetiva.
  2. Detectar os hiperparâmetros ótimos com base no modelo surrogate.

3. Aplicar os hiperparâmetros na função objetiva para avaliá-los.
4. Atualizar o modelo surrogate com os novos resultados.
5. Repetir os passos 2-4 até chegar no número máximo de iterações possíveis (Yang; Shami, 2020).

#### 3.3.1.4 Avaliação

Após realizar o treinamento do modelo desejado, é chegada a hora de avaliar sua performance para averiguar seu desempenho em tarefas selecionadas. Neste sentido são utilizadas variáveis qualitativas - que tem como objetivo compreender experiência de um ou mais indivíduos, como perguntas sobre a experiência de uso, para identificar se ela foi satisfatória ou não, e as quantitativas - que apresentam as características de forma mecânica a partir de resultados numéricos que representam as métricas do sistema avaliado (Dalianis, 2018).

Ademais, com o avanço dos estudos que objetivam explicar modelos de ML, muitas métricas e técnicas estão surgindo ao decorrer dos últimos anos, na tentativa de possibilitar uma melhor compreensão dos modelos criados. Apesar de almejarem facilitar o entendimento dos resultados, não há um consenso sobre quais as métricas devem ser adotadas, de forma que os criadores de cada modelo podem ter uma abordagem diferente, criando assim uma divergência na forma de explicar suas propriedades (Zhou et al., 2021). Além disso, as métricas mais comuns adotadas para explicar modelos NLP são *Accuracy*, *Precision*, *Recall* e *F1 score* (Yacoub; Axman, 2020). O cálculo de cada um e seu significado segue abaixo:

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

$$Precision = \frac{TP}{(TP + FP)}$$

$$Recall = \frac{TP}{(TP + FN)}$$

$$F1Score = 2 \cdot \frac{(Precision \cdot Recall)}{(Precision + Recall)}$$

no qual:

- TP - True Positives (Verdadeiros Positivos)
- TN - True Negatives (Verdadeiros Negativos)
- FP - False Positives (Falsos Positivos)
- FN - False Negatives (Falsos Negativos)

Com isso em mente, a *Accuracy* reflete o total de resultados corretos, tantos os verdadeiros positivos, quanto os verdadeiros negativos, em comparação com todos os resultados da atividade avaliada. Já *Precision* demonstra a habilidade do modelo de não avaliar de forma equivocada uma amostra negativa como positiva e o *Recall* a capacidade de identificar todos os verdadeiros positivos. Por fim, *F1-Score* é uma média harmônica de *Precision* e *Recall*, sendo uma métrica ótima para avaliar a performance do modelo de forma geral.

### 3.3.1.5 *Fine-Tuning*

A utilização de modelos NLP pré treinados como BERT e GPT está cada vez mais comum e dominante, com a utilização de *Deep Neural Network* para o processo de *fine-tuning* (Min et al., 2024). Esse processo toma como base um modelo treinado em tarefas genéricas (pré treinado), como a previsão de palavras, com datasets grandes como Wikipedia e livros (Wu et al., 2020), para estabelecer modelos com capacidades linguísticas gerais úteis para diversas tarefas específicas. Dessa forma, o processo de *fine-tuning* é realizado para o refinamento de um modelo pré treinado para a realização de tarefas específicas.

Durante o processo de *fine-tuning* é importante levar em conta duas práticas:

- Redução do número de parâmetros treináveis - É comum congelar parâmetros para que não sejam alterados durante o *fine-tuning*, uma vez que isso possibilita especificar as partes do modelo que devem ser alteradas, ou mesmo melhoradas. Ademais, em situações onde há uma quantidade razoável, porém limitada de dados, é importante que o modelo foque em parâmetros relevantes, de forma a não comprometer outros previamente definidos (Grießhaber; Maucher; Vu, 2020).
- Análise de mudanças em parâmetros na camada - Ter consciência das mudanças feitas é extremamente importante, pois possibilita a análise do impacto dos parâmetros na performance do modelo nas atividades desejadas. Para isso, é importante ter salvo os parâmetros antes do *fine-tuning* e após, de forma que seja possível identificar as alterações feitas e, então, poder relacionar as mudanças e os impactos (Grießhaber; Maucher; Vu, 2020).

Com isso em mente, o procedimento de *fine-tuning* se assemelha muito ao próprio treinamento de um modelo novo, com pré-processamento, treinamento e avaliação do modelo, porém utilizando como base um modelo pré-treinado, de forma a possibilitar uma performance ainda melhor em atividades específicas como análise de sentimentos e perguntas e respostas, assim, chegando a alcançar resultados de alto desempenho.

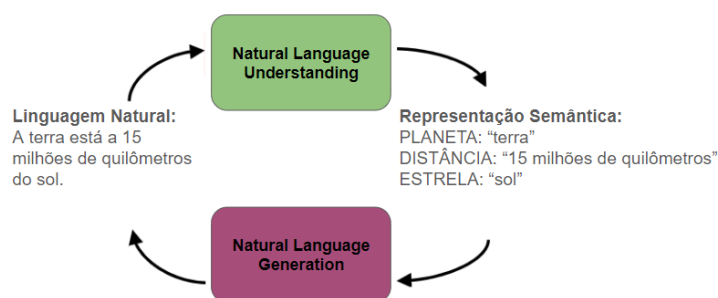
### 3.3.2 Natural Language Understanding e Natural Language Generation

*Natural Language Understanding* (NLU) e *Natural Language Generation* (NLG) são tópicos críticos ao se falar de NLP e para estruturar um diálogo. NLU objetiva extrair significado semântico de declarações, por outro lado, NLG constrói sentenças estruturadas correspondente a semântica extraída (Su; Huang; Chen, 2019). Nesse sentido, é possível simplificar e dividir o processo de sistemas de diálogo como mostrado abaixo.

1. Um sistema de reconhecimento de fala transcreve o que um usuário fala para texto.
2. Um módulo de NLU é utilizado para classificar o domínio, intenções associadas e formar uma representação semântica.
3. É utilizado um Dialogue State Tracker que prevê o estado do diálogo atual em uma conversa de múltiplos turnos.
4. Há uma política que determina os próximos passos baseados no estado atual.
5. O módulo de NLG é acionado para gerar uma resposta para o estado atual.

Como é possível ver, NLU e NLG trabalham em conjunto para resolver problema, de forma que o primeiro trabalha para a compreensão das situações e o ultimo para gerar uma resposta apropriada. Essa dualidade de interação e cooperação pode ser visto na Figura 8.

Figura 8 – Dualidade de NLU e NLG.

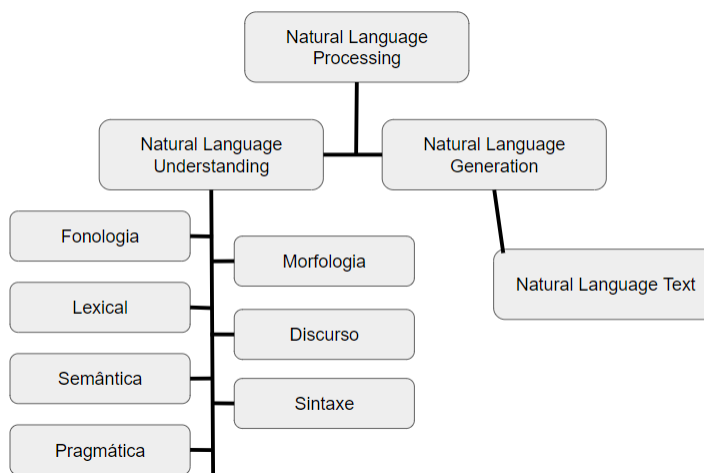


Fonte: Su, Huang and Chen (2019) (Adaptado pelo Autor).

#### 3.3.2.1 Principais componentes de NLU

A partir da integração de diferentes componentes NLU e NLG, é possível criar um sistema capaz de atividades complexas. Nesse cenário, a ambiguidade é um dos maiores problemas enfrentados em diversos níveis, como semântica, sintaxe e lexical. Para conseguir compreender tal problemática e as diversas análises possíveis no contexto de NLU, essa seção apresentará os principais componentes de NLU, conforme representado na Figura 9.

Figura 9 – Componentes de análise NLU.



Fonte: Khurana et al. (2023) (Adaptado pelo Autor).

#### 3.3.2.1.1 Fonologia

A fonologia é uma parte da língua que diz respeito à forma como os sons são organizados. O termo tem sua origem na Grécia antiga no qual “phōno” significa voz ou som e o sufixo “logía” implica a ideia de saber, proporcionando assim uma categoria da ciência. Neste sentido, a fonologia se apresenta como o estudo dos sons no sistema linguístico (Khurana et al., 2023).

#### 3.3.2.1.2 Morfologia

Os morfemas são as menores unidades com significado dentro de uma palavra. A morfologia investiga a natureza das palavras, que começa pelos morfemas. Um exemplo de morfema pode ser a separação morfológica da palavra desqualificado em 3 partes: o prefixo “des-”, o radical “-qualific-” e sufixo “-ado”. Neste, o prefixo “des-” indica negação e o sufixo “-ado” particípio passado, conforme ilustrado na Figura 10. Ademais, palavras que não podem ser divididas e são flexionadas e modificadas para criar novas são chamadas de morfemas lexicais como por exemplo “mar”. Já as adições feitas a eles para criar novas palavras são chamados de morfemas gramaticais como “-esia” em “maresia”. Os morfemas gramaticais que só aparecem em conjunto com outros são chamados de morfemas presos, como é o caso de “i-” na palavra “ilegal”. Estes por sua vez podem ser divididos em morfemas flexionais e derivados, no qual os flexionados ou desinências não influenciam a classe gramatical da palavra como “-mos” em “rimos”, já os derivados ou afixos modificam, como “-mente” na palavra “ferozmente” (Khurana et al., 2023).



Figura 10 – Análise morfológica da palavra “desqualificado”.

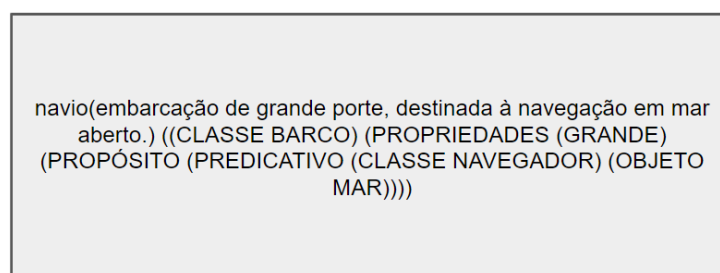


Fonte: Elaborado pelo Autor.

### 3.3.2.1.3 Lexical

Uma análise lexical é a interpretação do significado individual das palavras. Existem algumas formas de fazer o processamento no nível de palavras, sendo o primeiro a partir da classificação da palavra. Essa classificação objetiva definir a classe gramatical das palavras e, em casos de haver mais de uma possibilidade, classifica com base na classe mais provável. Ademais, caso a palavra tenha apenas um possível significado, ela pode ser substituída por uma representação semântica que varia dependendo da teoria semântica aplicada. Dessa forma, no nível lexical, a análise de estrutura das palavras é feita com base em seu significado lexical e classificação. Durante essas análises, o texto pode ser dividido em parágrafos, sentenças e palavras. As palavras são classificadas com base no contexto em que ocorrem (Khurana et al., 2023). A Figura 11 demonstra um exemplo da decomposição da palavra “navio” em propriedades mais básicas. A partir disso, é possível estabelecer um conjunto de primitivos semânticos usados em todas as palavras, criando uma representação lexical simples que permite a compreensão e unifica o significado das palavras, além de produzir uma interpretação complexa, semelhante ao dos humanos (Liddy, 2001).

Figura 11 – Análise lexical.



Fonte: Liddy (2001) (Adaptado pelo Autor).

#### 3.3.2.1.4 Sintaxe

Após a análise lexical, as palavras são agrupadas em frases e as frases são agrupadas para formar orações e então são combinadas em sentenças a nível sintático. A partir de uma análise gramatical da estrutura das sentenças é possível compreender a dependência entre as palavras, de forma a examinar a ordem das palavras, a interação entre elas, morfologia e classe gramatical, o que não são considerados no nível lexical. Dessa maneira, a ordem das palavras importa, tendo em vista que uma alteração pode mudar o significado da frase e a forma como as palavras se relacionam como em “O carro foi mais rápido que o ônibus” e “O ônibus foi mais rápido que o carro”, que apesar de terem as mesmas palavras com os mesmo significados, tem no conjunto um entendimento e dependências estruturais diferentes. Outrossim, não é possível durante esse processo utilizar técnicas como a remoção de *stop word*, extração de radicais ou extração da forma raiz das palavras, tendo em vista que isso mudaria a estrutura gramatical da frase (Khurana et al., 2023).

#### 3.3.2.1.5 Semântica

O nível semântico tem como principal objetivo determinar de maneira apropriada o significado de uma sentença. Seres humanos se baseiam no conhecimento da língua e conceitos presentes na sentença, já as máquinas não podem contar com tais técnicas. Para ultrapassar essa barreira, o processamento semântico é feito para identificar os possíveis significados de uma sentença a partir do processamento da estrutura lógica para reconhecer as palavras mais relevantes para o entendimento da interação entre palavras e conceitos. Por exemplo, um modelo é capaz de compreender que uma frase fala de “filmes”, mesmo que a palavra não esteja explícita, a partir de conceitos relacionados como “ator”, “atriz” e “roteiro”. Esse nível de processamento também inclui a desambiguação semântica de palavras com o mesmo significado com base no contexto, como a palavra “laranja” que pode se referir a uma fruta ou a uma cor. Ademais, nesse nível de processamento, a semântica é determinada pelo significado do dicionário ou interpretação derivada do contexto, como em “Elizabeth foi uma pessoa boa e honesta” onde “Elizabeth” pode se referir a uma pessoa qualquer com esse nome ou a rainha da Inglaterra (Khurana et al., 2023).

#### 3.3.2.1.6 Discurso

A sintaxe e semântica são utilizadas em conjunto com sentenças individualmente, diferente do discurso que lida com mais de uma sentença por vez. Nesse nível de NLP, são feitas análises de estrutura lógica a partir da conexão entre palavras e sentenças, para verificar a sua coerência com foco nas propriedades do texto que garantem significado ao interpretar a relação das estruturas linguísticas em diferentes níveis. Os dois níveis mais comuns são: Resolução de Anáfora e Resolução de Co-referência. A resolução de Anáfora

pode ser alcançada ao reconhecer uma entidade referenciada usando uma anáfora para resolver a referência contida dentro do texto. Por exemplo, considerando as frases:

1. Ruy é brasileiro.
2. Ele é alto.

Para um humano, compreender que “Ele” (2) se refere a “Ruy” (1) é uma tarefa trivial, porém essa interpretação depende da compreensão da relação entre a palavra “Ele” que se refere ao termo anterior “Ruy”. O entendimento de quem é alto depende da capacidade de fazer esse relacionamento, ou seja, de fazer uma Resolução de Anáfora. Por outro lado, a Resolução de Co-referência é alcançada ao se encontrar todas as referências a uma entidade em um texto. Esse é um passo muito importante para atividades NLP como resumo e extração de informações (Khurana et al., 2023).

#### 3.3.2.1.7 Pragmática

O nível pragmático foca nos conhecimentos e fatos que vêm de fora do documento analisado. Em outras palavras, é uma inferência que tem como objetivo compreender algo que não foi dito diretamente, levando em conta o contexto e representações do texto. Nesse sentido, a ausência de informações e contexto gera uma ambiguidade pragmática, fenômeno que causa pessoas diferentes a terem interpretações distintas de um mesmo texto. O contexto do texto pode incluir referências a outras partes do mesmo documento, o que influencia o entendimento dele, além do conhecimento prévio do leitor que garante o entendimento de conceitos contidos nele. A análise semântica foca no significado literal das palavras, diferente da análise pragmática que infere o significado com base em conhecimentos prévios. Como exemplo a frase “Você sabe que horas são?” pode ser referir a alguém que tem o objetivo de saber o tempo, ou pode ser uma crítica a alguém que está atrasado. Dessa forma, a análise pragmática ajuda a desvendar o real significado de um texto ao aplicar conhecimentos prévios que não estão contidos no texto de forma direta (Khurana et al., 2023).

#### 3.3.2.1.8 Resumo dos Componentes de NLU

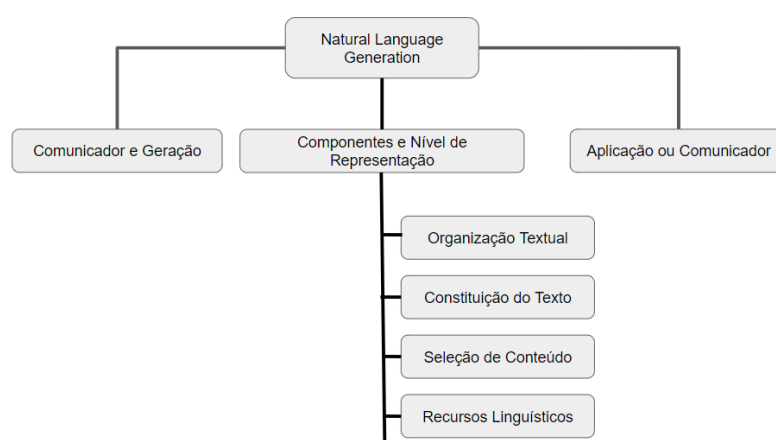
A divisão em componentes diferentes permite escolher quais os níveis de análise que um modelo NLP vai ter. Os níveis mais baixos focam em análises individuais de palavras e sentenças, como a morfológica, sendo baseadas em regras. Já nos níveis mais altos de processamento de linguagem lidam com textos e conhecimentos do mundo, como os conhecimentos prévios utilizados na análise pragmática (Liddy, 2001). Destarte, o desenvolvimento de um modelo NLP deve levar em conta quais as capacidades e níveis de

análises são necessárias, para assim possibilitar o desenvolvimento e treinamento com os componentes NLU adequados.

### 3.3.2.2 Principais componentes de NLG

NLG, sendo parte integrante de NLP, objetiva a produção de frases, sentenças e parágrafos significantes. Nesse sentido, o seu processamento pode ser dividido em três fases: identificação de objetivos, planejamento de como alcançar os objetivos a partir da situação e recursos de comunicação disponíveis e transformar o plano em texto (Khurana et al., 2023). A Figura 12 demonstra essa divisão.

Figura 12 – Componentes de análise NLG.



Fonte: Khurana et al. (2023) (Adaptado pelo Autor).

#### 3.3.2.2.1 Comunicador e Geração

Para gerar um texto é necessário ter um comunicador ou uma aplicação e um gerador de linguagem natural ou um programa que transcreve a intenção da aplicação para uma frase (Khurana et al., 2023).

#### 3.3.2.2.2 Componentes e Níveis de Representação

Para fazer a geração de linguagem natural existem tarefas inter-relacionadas que são essenciais (Khurana et al., 2023):

- Seleção de Conteúdo - Seleção das informações que devem ser incluídas. Dependendo da forma que elas devem ser passadas, pode ser necessário adicionar ou excluir partes.
- Organização Textual - A informação tem que ser textualizada seguindo regras gramaticais e sequências e relacionamentos lógicos.
- Recursos Linguísticos - Para dar suporte às informações, recursos linguísticos devem ser escolhidos, como idioma, construção sintática, palavras específicas etc.

- Constituição do Texto - Os recursos escolhidos e organizados devem constituir o texto ou áudio desenvolvido.

#### 3.3.2.2.3 Aplicação ou Comunicador

Essa parte é apenas para propósitos de manutenção. O comunicador inicia o processo, mas não participa do processo de geração de linguagem, apenas ouvindo e guardando informações potencialmente relevantes para criar uma representação do que sabe. O único requisito é que o comunicador compreenda a situação apresentada (Khurana et al., 2023).

#### 3.3.2.2.4 Resumo dos Componentes de NLG

A geração de linguagem a partir de um modelo NLP, utiliza diversos recursos de NLG, dos quais foram comentados os principais componentes. Esse é um processo complexo que pode ser dividido em tarefas, das quais podem se extrair outras subtarefas. Ainda assim, é a partir deste princípio que diversas soluções vêm sendo criadas como os chatbots.

### 3.3.3 Principais Atividades NLP

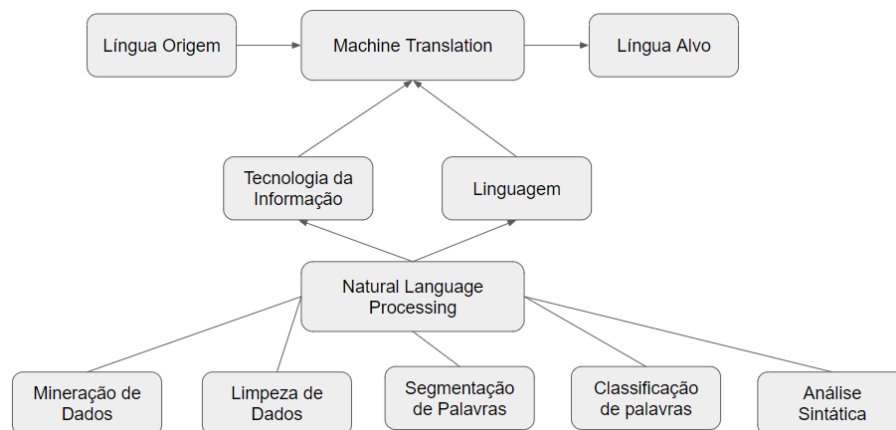
#### 3.3.3.1 *Machine Translation*

As atividades de *Machine Translation* datam seu início a mais de um século, com o começo na década de 1920, antes mesmo do NLP. Apesar disso, com a evolução das técnicas e melhorias de performance, essa área se transformou em uma das mais importantes dentro de NLP, consistindo da tradução de uma língua (origem) para uma língua (alvo) sem a ajuda de humanos. Nesse sentido, os modelos NLP são utilizados como ponto central, conforme mostrado na Figura 13 (Jiang; Lu, 2020).

#### 3.3.3.2 Perguntas e Respostas

A necessidade de acesso rápido à informação fez com que o segmento de perguntas e respostas em NLP recebesse muita atenção e crescesse em popularidade. Por existirem diferentes aplicações, os sistemas são desenvolvidos utilizando técnicas únicas e dataset de treinamentos específicos. Em um domínio limitado, a resposta a uma pergunta é mais simples, uma vez que é necessário menos esforço para estruturar os conhecimentos necessários. De forma antagônica, em um domínio aberto são necessários conhecimentos básicos gerais em conjunto com um dicionário semântico, como o English WordNet - um dicionário lexical utilizado para o idioma inglês. Nesse segmento, as capacidades de compreensão de texto (NLU) e geração (NLG) são mutuamente necessários para responder perguntas de forma correta e precisa. Ademais, a utilização de modelo Deep Learning,

Figura 13 – Uso de NLP em Machine Translation.



Fonte: Jiang and Lu (2020) (Adaptado pelo Autor).

ao invés de Machine Learning, possibilitou uma melhora significativa nos sistemas de pergunta e resposta, pois a capacidade de compreensão da língua a partir do treinamento não supervisionado obteve resultados superiores (Yu et al., 2020).

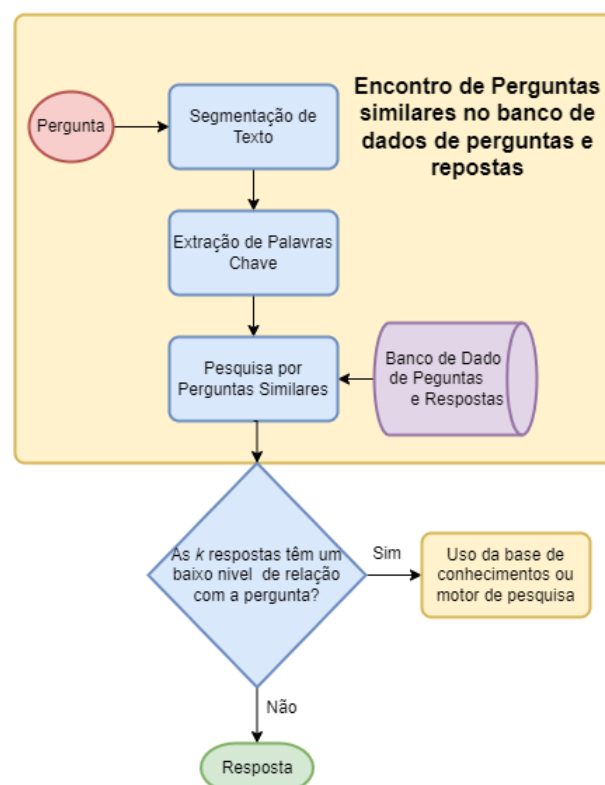
Para compreender melhor como o processo é feito, existem três técnicas principais utilizadas para perguntas e respostas, como descrito abaixo:

- Baseado em Base de Conhecimento - Sistemas que a partir de uma pergunta, criam uma consulta feita diretamente em uma base de conhecimento e retornam a resposta ao usuário (Costa et al., 2023). Existem diversas bases de dados grandes como o DBpedia, Freebase e YAGO que guardam informações em formato triplo RDF (Resource Description Framework). Em outras palavras, o objetivo aqui é traduzir a linguagem natural em consultas, porém isso só é efetivo em domínios específicos, caso contrário a estrutura do domínio se torna muito complexa.
- Baseado em Extração de Informações - A extração de informações é feita a partir de dados não estruturados, normalmente provenientes da internet. As perguntas são feitas em linguagem natural e processadas para extração das principais palavras e uma consulta é criada. Após isso, é feito a classificação e ordenação de documentos por relevância e então são retirados trechos que têm maior chance de ser uma resposta. Por fim, um módulo de processamento determina qual o tipo de resposta esperada e escolhe as melhores.
- Uso de Perguntas Guia - Ao considerar uma pergunta ambígua, o sistema pode gerar perguntas para direcionar e melhorar o entendimento sobre as intenções do usuário. Imagine a pergunta “Qual ônibus vai de Vitória para Belo Horizonte”, caso o sistema não faça uma pergunta guia, é provável que retorne todos os itinerários que fazem esse trajeto. Por outro lado, caso ele pergunte a data e horário desejado,

é provável que informe apenas as linhas referentes ao trajeto no período apropriado determinado pelo usuário.

A utilização em conjunto das três técnicas permite atingir os melhores resultados em um sistema completo de perguntas e respostas. Primeiramente, é feita a segmentação da pergunta com a extração das palavras consideradas relevantes que são estruturadas em vetores. Estes são utilizados em uma pesquisa por perguntas similares em um banco de dados de perguntas e respostas que retorna um número  $k$  de melhores respostas. Caso esse número  $k$  de melhores resposta apresente um grau baixo de relação com a pergunta do usuário, outros métodos de pesquisa são utilizados como bases de conhecimentos, motores de pesquisa ou mesmo perguntas guia para ajudar a compreender a pergunta do usuário. A Figura 14 apresenta esse processo (Yuan et al., 2019).

Figura 14 – O processo de perguntas e respostas.



Fonte: Yuan et al. (2019) (Adaptado pelo Autor).

### 3.3.3.3 Resumo de Textos

Desde o século XX, os dados são partes habituais na vida das pessoas. Com a invenção da internet, a quantidade de dados disponíveis é muito grande, dentre eles, os textos tendem a ser mais difícil de serem interpretados, o que traz a necessidade de uma forma capaz de resumir e selecionar os mais relevantes. Esse processo pode ser feito a partir de um ou múltiplos documentos, baseado em consulta, genérico e extrativo ou abstrativo.

Resumos extrativos são uma forma de resumir a partir do uso das mesmas sentenças do documento original. Por outro lado, os abstrativos são mais gerais, com foco nos conceitos principais apresentados. De forma similar, o resumo pode ser feito a partir de um documento ou ser gerado com vários. Ademais, existe uma necessidade de resumos baseados em consultas, que usam uma consulta criada pelo usuário para dar um foco em uma área específica. Diferentemente do genérico, que são em sua maioria abstrativos com foco em áreas gerais do texto.

Essa área de pesquisa tem atraído muita atenção de diversas áreas como ciência, medicina, direito, engenharia etc. Pesquisadores têm focado no resumo de prescrições médicas, assim como longos artigos novos para possibilitar que leitores absorvam conteúdos em tópicos de forma mais rápida e eficiente. As técnicas para resumo mais comuns são Machine Learning, Neural Networks e aprendizado por reforço. Além disso, a fusão de métodos também pode ser feita para alcançar modelos melhores em termos de *accuracy* (Rahul; Adhikari; Monika, 2020).

#### 3.3.3.4 Reconhecimento de Entidades Nomeadas

A atividade de reconhecimento de entidades nomeadas é uma importante para o NLP, como na extração de informações e perguntas e respostas (Oliveira et al., 2020). O objetivo dessa tarefa é identificar e classificar o nome de pessoas, localizações, organizações e expressões numéricas, incluindo datas, moedas e porcentagens (Costa et al., 2024). Inicialmente às línguas alvos eram as mais ricas em conteúdo disponível como inglês e chinses, porém atualmente há pesquisa e desenvolvimento em outras menos ricas como o indiano. Nesse contexto, existem 3 tipos de sistemas com relação às línguas alvos:

- Mono-lingual - Tem capacidade de identificar uma língua alvo. A principal língua pesquisada é e sempre foi o inglês. A pesquisa desse tipo de sistema é o mais comum.
- Bi-lingual - Sistemas capazes de identificar um par de línguas alvo. O mais comum é inglês-alemão.
- Multi-lingual - Sistemas com mais de dois alvos, como três ou seis que são os mais comuns.

Ademais, existem quatro formas de criar um sistema NLP para o reconhecimento de entidade nomeada. A escolha do adequado vai depender dos recursos linguísticos disponíveis, sua quantidade e qualidade (Sun et al., 2018). As quatro técnicas são:

- Baseado em Regras - Essa técnica mais simples foca no uso de regras com foco em aspectos lexicais, contextuais e morfológicos. Apesar de não ser considerada uma

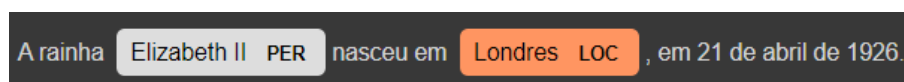


das principais técnicas, em situações adequadas ainda é possível obter resultados promissores.

- Baseado em Machine Learning - Essa técnica consiste no treinamento de modelos ML com base em um dataset de grande escala. Nesse contexto é possível utilizar os métodos supervisionados e não supervisionados de forma individual ou mesmo combinados.
- Baseado em Deep Learning - A partir de uma ANN e DL, muitos modelos foram criados com a intenção de possibilitar um maior desempenho nas atividades de reconhecimento de entidades nomeadas. A vantagem desses modelos é a capacidade de absorver conhecimento em um ambiente de textos não estruturados. Essa é a técnica que mais cresceu nos últimos anos.
- Outros - São 3: Base de conhecimentos, transferência de aprendizagem e aprendizagem conjunta baseada em multitarefas. As bases de conhecimento são utilizadas a partir da conexão feita com entidades externas que contém conhecimentos multi-lingual como Dbpedia e YAGO. Por outro lado, a transferência de aprendizagem parte de um modelo base com conhecimentos gerais que é feito o *fine-tunning* para superar problemas de falta de dados de treinamento. Já a aprendizagem conjunta baseada em multitarefas se trata de modelos capazes de concluir múltiplas tarefas simultaneamente como a identificação de entidades nomeadas e alinhamento de palavras.

Outrossim, a Figura 15 demonstra um exemplo de reconhecimento de entidades nomeadas usando a biblioteca Python spaCy. Nesse contexto, PER se refere a uma pessoa e LOC uma localização.

Figura 15 – Reconhecimento de entidade nomeada com spaCy.



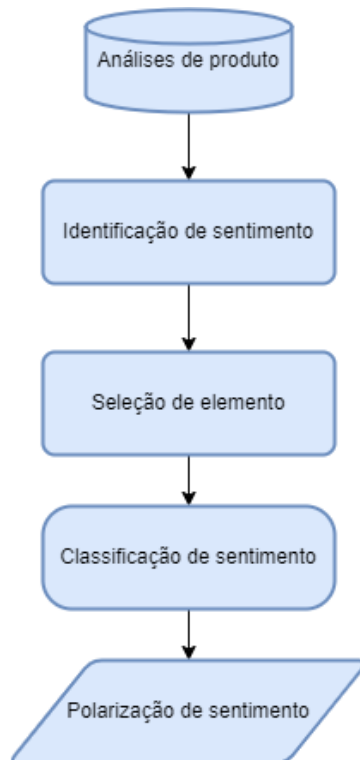
Fonte: Elaborado pelo Autor.

### 3.3.3.5 Análise de Sentimentos

Análise de sentimentos (AS) ou Mineração de Opinião (MO) é o estudo computacional da opinião e atitude das pessoas e suas emoções para com entidades. A entidade normalmente é representada por uma pessoa, evento ou tópico. Teoricamente AS e MS são intercambiáveis, representando o mesmo significado, porém alguns pesquisadores afirmam que há uma pequena diferença. (Tsytsarau; Palpanas, 2012) Mineração de Opinião extrai e analisa a opinião das pessoas sobre uma entidade, enquanto análise de sentimentos identifica os sentimentos expressados em um texto e então o analisa. Dessa forma, AS tem

como objetivo encontrar a opinião e identificar qual sentimento ela expressam e então classificar sua polaridade como mostrado na figura .

Figura 16 – Processo de análise de sentimentos de reviews de produto.



Fonte: Medhat, Hassan and Korashy (2014) (Adaptado pelo Autor).

O processo de AS pode ser considerado um processo de classificação. Existem 3 principais níveis de classificação: documentos, sentenças e aspecto. O nível de documentos objetiva classificar a opinião ou sentimento expressada através de um documento como positivo ou negativo. O documento inteiro é considerado como apenas uma unidade.

O nível de sentenças objetiva classificar o sentimento expresso por cada sentença. Primeiramente, a sentença é classificada como objetiva ou subjetiva, caso seja subjetiva será identificado se o sentimento é positivo ou negativo. Apesar disso, não há fundamentalmente nenhuma diferença entre o nível de documento e de sentença, uma vez que uma sentença pode ser interpretada como um documento curto.

Dessa forma, esses níveis de classificação não provem os aspectos necessários para compreender todos os detalhes da entidade em questão. O nível de aspecto objetiva classificar o sentimento levando em conta os aspectos específicos das entidade. O primeiro passo é identificar as entidades e seus aspectos, Um opinador pode apresentar diferentes aspectos sobre uma mesma entidade como “O mouse é bem preciso, porém os botões são duros” (Medhat; Hassan; Korashy, 2014).

### 3.3.4 Principais Modelos NLP

#### 3.3.4.1 GPT

O GPT - *Generative Pre-trained Transformer* é uma técnica de treino usado para a criação de modelos de NLP com o mesmo nome, criado pela OpenAI. A partir do treino semi-supervisionado, que mistura o treino supervisionado e não supervisionado, é criado um modelo com habilidades gerais em diversas tarefas. A estrutura utilizada é o *Transformer* pelas suas capacidades mostradas em atividades de *Machine Translation*, geração de documentos e análise sintática. O modelo apresenta grande quantidades de memória estruturada, o que permite que lide com tarefas específicas e *fine-tuning* de forma eficiente. Ademais, o modelo foi testado em quatro tarefas de NLU - inferência, perguntas e respostas, similaridade semântica e classificação de texto, demonstrando performance superior a modelos treinados especificamente para essas tarefas.

O processo de treino é dividido em duas partes. Primeiro, é feito o treino não supervisionado com grandes quantidades de texto e depois *fine-tuning* em tarefas específicas com textos manualmente categorizados. Na primeira etapa é utilizado um dataset contendo mais de 7.000 livros não publicados de diversos gêneros, como fantasia, romance e aventura. Nesse sentido, o dataset utilizado contém longos textos com uma variedade de vocabulários e permite que o modelo generativo compreenda a relação de informações relacionadas. Para o processo de *fine-tuning*, foram utilizados diferentes datasets, como o RACE, em tarefas de perguntas e respostas e entendimentos gerais, que dada uma passagem em inglês, são feitas perguntas relacionadas (Radford et al., 2018).

#### 3.3.4.2 BERT

O BERT - **B**idirectional **E**ncoder **R**epresentations from **T**ransformers é um modelo que tem como objetivo melhorar o desempenho de modelos *fine-tuned*, criado pelo Google. Diferente dos modelos que utilizam a arquitetura *Transformer* com análise de tokens de forma unidirecional, o BERT propõe uma análise bidirecional, o que leva em conta elementos anteriores ao token analisado e também posteriores, diferente, por exemplo do GPT que é unidirecional. Para tal, BERT usa um objeto chamado “Masked Language Model” (MLM) que aleatoriamente máscara alguns tokens e o modelo deve então prever eles com base no contexto. Essa implementação permite a criação de uma arquitetura *Transformer* bidirecional profunda. Adicionalmente, a tarefa de “previsão da próxima sentença” é usada para treinar o modelo com pares de sentença.

De forma semelhante ao GPT, BERT apresenta um processo de treino dividido em duas partes. Primeiro é feito o pré treino de forma não supervisionada e depois o *fine-tuning* supervisionado. O modelo base do BERT tem tamanho similar ao GPT e o modelo *fine-tuning* apresenta diferenças mínimas do base. Para o treinamento, BERT

aceita sequências, que podem conter um ou par de sentenças que são colocadas juntas (usado em perguntas e respostas por exemplo). Foi utilizado o WordPiece Embeddings com um vocabulário de 30.000 tokens, um dataset de livros com 800 milhões de palavras e Wikipédia em inglês com 2.5 bilhões de palavras. O primeiro token de cada sequência deve ser “[CLS]” e o token especial “[SEP]” é usado para diferenciar uma sentença de outra.

Os passos para o pré treino bidirecional do BERT são:

1. MLM - É o processo de esconder aleatoriamente 15% dos tokens para serem posteriormente previstos. Porém, ao invés de sempre substituir o token real pelo especial “[MASK]”, isso ocorre em 80% das vezes, 10% por um token aleatório e 10% não são alterados.
2. Next Sentence Prediction (NSP) - É o procedimento de prever uma próxima sentença *A* ou *B* de forma que em 50% das vezes é *A* e nos outros 50% *B*. Isso é extremamente útil para ajudar o modelo a compreender a relação entre sentenças, usado principalmente em perguntas e respostas e inferência.

*Fine-tuning* com BERT é muito mais simples e exige muito menos esforço computacional quando comparado ao pré treino. A partir da utilização do mecanismo de *self-attention* da arquitetura *Transformer*, BERT é capaz de especificar as partes de uma sentença em que deve prestar atenção e juntar sentenças para formar uma sequência, colocar os tokens especiais necessários e utilizar sua bidirecionalidade para compreender de forma eficiente tarefas e textos (Devlin et al., 2019).

#### 3.3.4.3 Llama 2

O Llama 2 é uma família de modelos LLM compostos por Llama 2 e Llama 2-Chat, sucessores do Llama 1, criados pela Meta para atividades de NLP. Llama 2 utiliza uma arquitetura *Transformer* e é um dos modelos com a menor taxa de violação de segurança para humanos. O modelo Llama 2 apresenta conhecimentos gerais, treinado em uma mistura de dados públicos e é o sucessor direto do modelo Llama 1, já o Llama 2-Chat é uma versão fine-tuned e otimizada para conversas do Llama 2. Seguindo a linha de outros LLM como BERT e GPT, ele apresenta um processo de treino não supervisionado para conhecimentos gerais e supervisionado para *fine-tuning*.

Para garantir maior segurança, após o processo de *fine-tuning*, o modelo passa por um treino por reforço com iterações feitas com humanos, a fim de permitir identificar ações consideradas inseguras e tratá-las. Também foram coletados dados de humanos para melhorar as respostas do modelo a partir de uma escolha binária com as opções “significativamente melhor”, “melhor”, “um pouco melhor”, “negligentemente melhor” e

“incerto”. Esses processos são feitos a fim de possibilitar a melhoria contínua da qualidade e segurança do modelo.

Quanto as aplicações, o modelo Llama 2 pode ser utilizado para *fine-tuning* ou aplicações em tarefas generalizadas, já Llama 2-Chat deve ser utilizado em atividades específicas como conversação e perguntas e respostas (Touvron et al., 2023).

### 3.4 *Large Language Models*

*Large Language Models* (LLM) são modelos de linguagem (NLP) que utilizam DP, com um número muito grande de parâmetros, treinados em grande volumes de dados de forma não supervisionada, como o GPT-4 com mais de 100 trilhões de parâmetros e Bard. Em questão de meses, LLM se tornou um assunto relevante na sociedade, tanto para o público geral, quanto para a comunidade científica, ao apresentar uma incrível capacidade em tarefas como perguntas e respostas (Birhane et al., 2023). Destarte, se tratam de modelos com componentes NLG capazes de atividades como inferência, usando uma distribuição estatística para definir uma resposta mais provável (Shanahan, 2024).

Apesar disso, existem diversas preocupações sobre a compreensão de ética e moral dos modelos, pois eles podem ser utilizados para propósitos ofensivos e que colocam a vida de pessoas em risco. Ademais, a capacidade generativa desses modelos não significa que as informações fornecidas sejam verídicas, dado o fato que o fenômeno de “alucinação” pode ocorrer quando um modelo passa uma informação incorreta com confiança de que é a verdade. Em razão destas, entre outras situações, muitas pessoas se sentem ameaçadas e com medo do uso dessa tecnologias disruptivas, assim como ocorreu com o surgimento dos automóveis, porém nem todas as preocupações têm fundamentos científicos, como o “iminente extermínio dos humanos pela IA”, que atualmente se trata apenas de ficção em uma distopia.

Ademais, LLM deve ser tratado no âmbito científico com cautela, sempre com uma postura disciplinada e humilde, procedendo com testes e desenvolvimentos com expectativas realistas. Esse procedimento se mostra relevante, a fim de evitar reivindicações falsas que se tornam corriqueiras em uma realidade de disputa por velocidade em uma esfera volátil da ciência atual. Não obstante, ainda há muito a ser desenvolvido, testado e compreendido sobre LLM, sua forma de funcionamento e aplicações (Birhane et al., 2023).

## 4 Projeto

Seguindo a problemática levantada e conforme especificado em 1.2.1, iremos apresentar uma proposta de solução objeto deste trabalho.

Para avaliar modelos de NLP e extrair métricas quantitativas, é necessário que o usuário tenha conhecimentos específicos de IA e ML. Nesse sentido, com o intuito de facilitar esse processo, a plataforma terá por padrão modelos, atividades e datasets contemplados. Estes serão utilizados para validar o funcionamento da plataforma e estarão disponíveis para uso na versão final. Porém, caso o usuário opte por um modelo, tarefa ou dataset não disponível por padrão, será necessário inserir de forma manual o caminho para o modelo, código para pré-processamento, código da tarefa ou mesmo dataset que devem ser utilizados. Nesse sentido, para inserir um novo modelo, deve ser fornecido um código de pré-processamento adequado a atividade e arquitetura específica ou utilizar um já outrora implementado.

Os modelos adicionados já devem ter sido treinados (*fine-tuned*), em um dataset similar ao que será usado para testes, para possibilitar que as métricas sejam calculadas e tenham resultados representativos das capacidades do modelo. Dessa forma, a plataforma terá capacidade para extrair dados quantitativos do desempenho do modelo, o que representa uma parte do processo de criação de um modelo NLP, não sendo contemplado a criação e treinamento do modelo.

A Figura 17 mostra um exemplo de como o código de pré-processamento pode ser implementado de forma que seja simples ajustar para as necessidades do usuário.

Figura 17 – Exemplo de código de pré-processamento.

```
def pre_processamento(opcao):  
    '''  
    DOCSTRING: Exemplo de seleção de código de pré-processamento.  
    '''  
    if opcao == "BERT_no_stopwords":  
        #...  
    elif opcao == "BERT_with_stopwords":  
        #....  
        #Colocar opções adicionais aqui como 'GPT_no_stopwords'  
    else:  
        #...
```

Fonte: Elaborado pelo Autor.

O código de pré-processamento pode contemplar diferentes técnicas, como a remoção ou não de stop words. A técnica utilizada dependerá da atividade que será realizada, por isso é necessário especificar qual técnica deve ser escolhida. Caso o pré-processamento necessário não tenha sido implementado, será preciso adicionar na função de pré-processamento a técnica requerida.

A modularidade da plataforma tem como objetivo possibilitar que o maior número de modelos e atividades sejam contemplados, uma vez que existem diversos modelos disponíveis e surgindo, cada um com suas peculiaridades de funcionamento e atividades com propósitos e características diferentes, além da escolha de datasets, os quais podem ser objeto de avaliação.

A Tabela 1 mostra um exemplo de como será o resultado das avaliações. Nesse sentido, as avaliações anteriormente realizadas ficaram salvas, de forma que seja possível realizar os testes em momentos diferentes e visualizar todo o histórico de resultados. Outrossim, a plataforma funcionará com o input de um modelo treinado (*fine-tuned*) para teste em um dataset semelhante ao de treino. Com base nisso, será feita a avaliação com o dataset de teste que irá gerar as métricas quantitativas. Vale ressaltar que as métricas serão padronizadas e não será possível fazer a alteração delas.

Tabela 1 – Exemplo de métricas da plataforma.

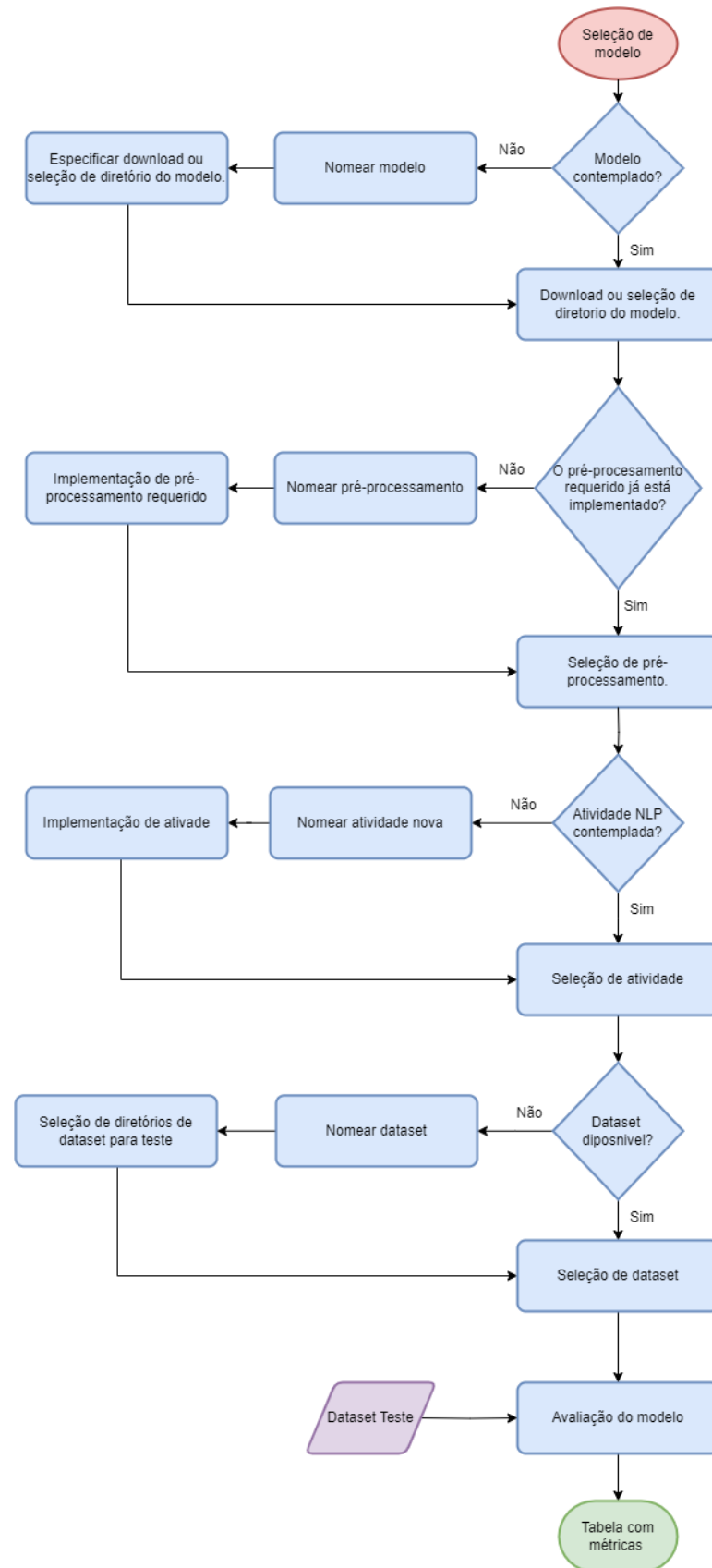
Modelo	Atividade	Dataset	Métrica t	Métrica s	Métrica p
A	Q&A	$\alpha$	<b>0.91</b>	0.89	0.73
A	NER	$\beta$	0.65	<b>1.00</b>	0.80
B	Q&A	$\alpha$	0.77	0.89	<b>0.99</b>

Fonte: Elaborado pelo Autor.

Por fim, dependendo da tarefa executada, as métricas utilizadas poderão fazer ou não sentido. Por essa razão, é importante compreender qual a natureza da atividade como classificação, resumo, geração de texto etc. Em outras palavras, por exemplo, utilizar acurácia para medir os acertos em uma atividade de NER faz sentido, porém, em uma atividade de resumo essa métrica se mostra ineficiente, sendo necessário recorrer a métricas e *benchmarks* específicos como o ROUGE - *Recall-Oriented Understudy for Gisting Evaluation* que foi desenvolvido para medir a qualidade de resumos. Destarte, apesar de ser modular, existem algumas tarefas que não podem ser avaliadas de forma satisfatória e compreensível pela plataforma.

Para facilitar a compreensão, a Figura 18 apresenta um fluxograma que demonstra a forma de uso da plataforma proposta.

Figura 18 – Fluxograma de funcionamento geral da plataforma.

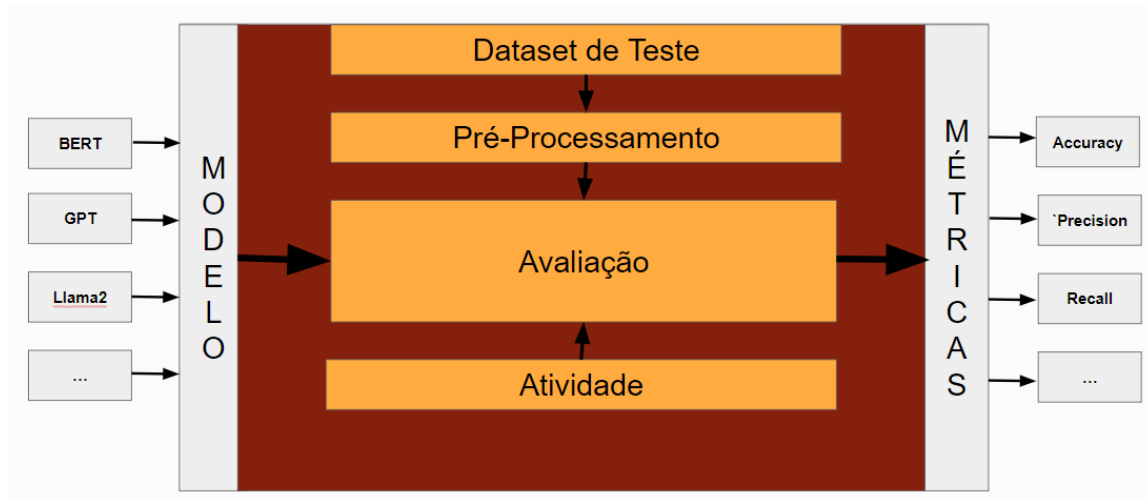


Fonte: Elaborado pelo Autor.



Por fim, a Figura 19 ilustra, em uma visão geral, os componentes que farão parte da plataforma e a relação entre si.

Figura 19 – Visão geral da plataforma.



Fonte: Elaborado pelo Autor.

## 5 Desenvolvimento

Ao longo deste capítulo serão apresentadas as principais tecnologias utilizadas, etapas do desenvolvimento e recursos de hardware utilizado para criar uma plataforma em conformidade com as especificações dispostas no Capítulo 4.

### 5.1 Hardware e Sistema Operacional

A plataforma foi desenvolvida utilizando a linguagem Python, em um ambiente de notebook Jupyter com bibliotecas como Transformers, Pytorch, Scikit-learn, entre outros. O desenvolvimento ocorreu em um computador com um Intel(R) Core(TM) i5-1135G7 CPU 2.40 GHz e 16Gb de RAM com o auxílio de um computador em nuvem com uma GPU NVIDIA T4 com 15Gb de VRAM e uma NVIDIA RTX 4060 ti com 8Gb de VRAM utilizados para treinar e avaliar modelos, uma vez que a utilização de uma GPU pode acelerar o processo consideravelmente (Baji, 2018). O sistema operacional utilizado foi o Microsoft Windows 11.

### 5.2 Tecnologias

#### 5.2.1 Python

Python<sup>1</sup> é uma linguagem de programação lançada em 20 de fevereiro de 1991 por Guido van Rossum. Ela foi criada em alto nível, ou seja é de alta abstração, se parecendo mais com a forma de pensar humana do que como o código de máquina funciona, o que facilita o entendimento para humanos. Além disso, o Python é uma linguagem orientada a objetos e interpretada com tipificação dinâmica - as variáveis não são instanciada com seu tipo associado de forma explícita.

Em 2017 Python foi a linguagem com o maior crescimento, sendo isso graças a sua versatilidade e possibilidade de uso em atividade de *Data Science* e *Machine Learning*. Outra característica que ajuda na acelerada adesão a língua é sua característica *Open Source*, sendo possível modificar, distribuir e ajudar no desenvolvimento da língua a partir de sua comunidade ativa. Com isso em mente, diversas empresas a utilizam, como Google, Intel, Cisco, Qualcomm e IBM por exemplo (Srinath, 2017).

Por ser referência na área de ML, o Python foi escolhido como linguagem de programação neste projeto. Destarte, todo projeto será codificado utilizando o mesmo.

---

<sup>1</sup> <<https://www.python.org/>>

### 5.2.2 Hugging Face

Hugging Face é uma empresa de inteligência artificial com foco em NLP que surgiu em 2016. Apesar de inicialmente oferecer apenas um chatbot para conversa, a empresa rapidamente se tornou referência em pesquisa e desenvolvimento de modelos NLP. Isso se deve a sua plataforma<sup>2</sup> que oferece bibliotecas *Open Source* e um Hub que possibilita a democratização do acesso à modelos, datasets, APIs, entre outros. Com isso, a plataforma é utilizada por milhares de pessoas, além de ter empresas de grande importância no âmbito de IA como parte de sua comunidade, como Google, Meta, OpenAI e Microsoft.

#### 5.2.2.1 Transformers

A biblioteca *Transformers* é dedicada ao suporte de arquiteturas baseadas em *Transformer*, para facilitar a distribuição de modelos. Para isso, ela pode ser utilizada em ambiente de pesquisa ou industrial, tendo um hub que centraliza a disponibilização de modelos. Ela é mantida pela comunidade e equipe do Hugging Face e está disponível a partir de uma licença Apache 2.0.

Seu desenvolvimento teve como base a comunidade de NLP que cria modelos *Open Source*, a biblioteca pioneira *tensor2tensor* e o modelo BERT. Seu propósito é ser fácil para usar, seguindo a tendência de bibliotecas como *spaCy*, *AllenNLP* e *Stanza*. Além disso, estas bibliotecas já utilizam *Transformers* como um framework em seus códigos.

O design inclui uma *pipeline* para pre-processamento de dados brutos, criação do modelo e predição. Ademais, é possível dividir sua estrutura em três blocos principais: tokenizador - converte o texto puro para codificações indexadas, *transformer* - transforma índices em embeddings contextuais, e *head* - utiliza os embeddings contextuais para predições em tarefas específicas.

O hub de modelos da comunidade permite a distribuição de modelos pré-treinados como BERT, GPT2 e DistilBERT. Por sua vez, esses modelos podem ser *fine-tuned* e disponibilizados na própria plataforma, fazendo assim com que seja possível disponibilizar tanto os modelos originais, quanto suas versões especializadas em tarefas (Wolf et al., 2020).

Como diversos modelos estão disponíveis no hub da plataforma e a biblioteca tem um extensivo suporte para modelos *fine-tuned*, foi optado por prosseguir com ela como principal ferramenta para lidar com o download e criação dos modelos.

#### 5.2.2.2 Datasets

Com o passar do tempo, os datasets que são parte essencial para criação de modelos de IA através de técnicas de *Machine Learning* se tornaram cada vez mais complexos,

---

<sup>2</sup> <<https://huggingface.co/>>

com diferentes metodologias de anotação, sendo também o particionamento uma etapa essencial. Com isso em mente, a biblioteca *datasets* foi criada, assim como um Hub que organiza e disponibiliza diversos datasets.

Assim como outras bibliotecas da Hugging Face, ela está disponível a partir de uma licença Apache 2.0 e sua documentação é extensa e simples, seguindo o princípio de uso fácil implementado pela empresa em seus serviços e produtos. Com ela é possível carregar, particionar, acessar, manipular e tokenizar os dados de forma simples, uma vez que foi pensada para isso. Outrossim, com a utilização de cartões, a documentação dos datasets disponíveis fica acessível e a biblioteca tem suporte para diversos tipos de arquivo como CSV, JSON e PARQUET (Lhoest et al., 2021).

Assim como a biblioteca *Transformers*, *datasets* será utilizada devido a grande quantidade de *datasets* ofertados a partir de seu hub, além da capacidade de lidar com variados tipos de arquivos.

### 5.2.2.3 Evaluate

Com o passar do tempo, a quantidade de métricas existentes cresceu consideravelmente, com cada vez mais sendo utilizadas em artigos e publicações. Com isso em mente, a heterogeneidade e complexidade aumentou, nesse cenário a biblioteca *evaluate* foi desenvolvida em conjunto com um hub para disponibilizar métricas.

Seguindo a trilha da Hugging Face, a biblioteca em questão segue uma licença Apache 2.0 e é sustentada por uma comunidade e funcionários da empresa. Ademais, ela tem como objetivo principal resolver 3 desafios: reprodutibilidade, centralização e cobertura.

- Reprodutibilidade - A avaliação de modelos é muito sensível a pequenas alterações, que são por muitas vezes não documentadas, como método de seleção de amostra, semente e hiperparâmetros escolhidos. Por essa razão, a comparação de modelos pode se tornar irrelevante e sem sentido.
- Centralização - Devido aos problemas de documentação e entendimento da comunidade de NLP, mudanças em datasets e métricas podem resultar em retrabalho com a divulgação de novos resultados de forma desnecessária. Além disso, a publicação de artefatos desatualizados pode se tornar comum.
- Cobertura - ML é uma área muito focada em resultados que são percebidos a partir da *accuracy* e métricas relacionadas e derivadas. Apesar de sua importância, informações relacionadas a eficiência, robustez e segurança não são considerados (Werra et al., 2022).



o cálculo de métricas como *accuracy* e *precision* a partir de diversas funções e também a criação de matrizes de confusão (Pedregosa et al., 2011).

Dentre as empresas que a utilizam<sup>4</sup> estão Spotify, Hugging Face, Evernote e Booking.com.

Para possibilitar o cálculo de forma mais dinâmica e sem depender exclusivamente do *evaluate*, sklearn foi utilizada para complementar e integrar na extração de métricas da plataforma.

### 5.2.5 Pandas

Pandas<sup>5</sup> está em desenvolvimento desde 2008, e está relacionada ao tratamento de dados como alinhamento e indexação hierárquica. Ela é uma das ferramentas mais robustas disponíveis para Python, utilizada principalmente para análise e ciência de dados, além de outras atividade relacionadas específicas de cunho estatístico e de banco de dados (McKinney et al., 2011).

A missão do pandas é ser uma ferramenta de alto nível que auxilie na análise de dados do mundo real em Python. Para além disso, há o notório objetivo de ser a ferramenta de análise e manipulação de dados *Open Source* mais poderosa e flexível do mundo.

O pandas foi escolhido para criar e manipular a tabela de métricas resultante do processo de avaliação dos modelos. Nesse âmbito, o pandas também permite a persistência dos dados ao salva-los em diversos formatos como o CSV.

### 5.2.6 Anaconda

Anaconda<sup>6</sup> é uma distribuição de R e Python criada em 2012 por cientistas de dados, com foco em projetos de data science. Atualmente com mais de 45 milhões de usuários no mundo, ela é utilizada por mais de 1 milhão e 90% das empresas da Fortune 500 com representantes como Microsoft, Lenovo, AWS, NVIDIA, IBM, Oracle e Red Hat. Além disso, são mais de 500 pacotes disponíveis (em 2024).

Foi escolhido o anaconda, mais especificamente sua versão básica e leve Miniconda, por sua confiabilidade, disponibilidade de pacotes e uso simplificado com comandos de alta abstração e compatibilidade com o Jupyter Notebook.

### 5.2.7 Jupyter Notebook

Os notebooks são uma ferramenta primeiramente utilizada por matemáticos que possibilita o ciclo REPL - read-evaluate-print-loop, ou seja, um ciclo que serve como base

---

<sup>4</sup> <<https://scikit-learn.org/stable/testimonials/testimonials.html>>

<sup>5</sup> <<https://pandas.pydata.org/>>

<sup>6</sup> <<https://www.anaconda.com/>>

para uma programação interativa (Kluyver et al., 2016). Nesse âmbito, o projeto Jupyter<sup>7</sup> surgiu em 2014 como parte do IPython Project para dar um suporte interativo em ciência de dados e computação científica. Vale destacar que o projeto é *Open Source* e gratuito.

O Jupyter Notebook é uma ferramenta utilizada em ciência de dados na qual é possível misturar texto em prosa com código em Python ou R. A partir dele é possível também executar apenas um trecho de código e verificar a sua saída (quando existente), fazendo com que seja uma opção para programar de um jeito mais interativo. Nesse sentido, ao trabalhar com células, a identificação de erros e problemas no código se torna mais dinâmica, pois não é necessário executar o programa inteiro para identificar um desvio, sendo possível visualizar o erro na célula específica e a linha de forma mais rápida e simplificada (Dombrowski; Gniady; Kloster, 2023).

Foi escolhido o Jupyter Notebook por ser ótimo para a manipulação de modelos de ML e dados, dois fatores primordiais para a execução do projeto.

### 5.2.8 PySimpleGUI

PySimpleGUI é uma biblioteca baseada em Tkinter, biblioteca integrada ao Python, que permite a criação de uma GUI (*Graphical User Interface*) de forma simples e eficiente. Criada por Mike Campbell, ela tem uma ótima documentação, com leitura fácil e apresenta uma pequena curva de aprendizagem, além disso há o suporte para as bibliotecas Tkinter, Qt, wxPython e Remi.

A biblioteca disponibiliza seu código fonte, porém não é *Open Source*, tendo uma licença proprietária. Para utilizá-la é necessário se inscrever em seu site<sup>8</sup> e selecionar a categoria de uso. Para este projeto foi feito o registro como *Hobbyist Developers* que inclui estudantes que utilizam o produto para trabalhos pessoais e estudo. Com isso é possível utilizá-la pelo período de 1 ano sem custos.

Foi escolhido utilizar o PySimpleGUI por ser uma forma de criar uma GUI sem a necessidade de seguir uma grande curva de aprendizagem, além de ser possível utilizá-la sem custos (para o caso especificado).

## 5.3 Modelos Utilizados

Para demonstrar as capacidades da plataforma desenvolvida, foram escolhidos dez modelos NLP. Dentre estes estão modelos pré-treinados e *fine-tuned*. Esta sessão explicará quais os modelos e em quais atividades eles foram utilizados.

---

<sup>7</sup> <<https://jupyter.org/>>

<sup>8</sup> <<https://www.pysimplegui.com/>>

### 5.3.1 BERT

Assim como outrora explicado, o BERT foi escolhido devido a suas capacidades em diversas atividades, como a inferência, com sua arquitetura com transformer bidirecionais. Ademais, foram utilizadas as versões “bert-base-uncased” e “bert-large-uncased” tendo 117 milhões e 336 milhões de parâmetros, respectivamente. Os dois utilizam o tipo F32 de tensor e estão disponíveis através do hub de modelos do Hugging Face.

Foi feito o processo de *fine-tuning* com o modelo “bert-base-cased” com o dataset CoNLL2003 para a atividade de NER com  $2 \times 10^{-5}$  de learning rate, 16 de batch size, weight decay de 0.01 e 3 epochs. Além disso, foi utilizado o modelo “bert-base-uncased-squad-v1” e “bert-large-uncased-whole-word-masking-finetuned-squad” treinado para atividades de Q&A. Todos esses modelos foram instanciados com *Transformers*.

### 5.3.2 ALBERT

ALBERT é uma versão modificada do BERT criado pelo Google, com o objetivo de reduzir seu tamanho sem um grande impacto na performance de maneira geral. Essa ideia surge devido as limitações de hardware presentes para a maioria das pessoas que não tem acesso aos recursos de grandes empresas como o Google. Com isso em mente, a diminuição quantidade de parâmetros pode solucionar problemas como falta de memória (RAM e VRAM) durante o treino e uso dos modelos. Dessa forma, o ALBERT - **A Little BERT** foi criado com uma quantidade significativa de menos parâmetros.

Esse modelo incorpora duas técnicas para redução de parâmetros. O primeiro foi a decomposição da matriz de embedding do vocabulário em duas menores, separando as camadas escondidas do embedding do vocabulário. A segunda é o compartilhamento de parâmetros através das camadas. Esta técnica impendi que parâmetros cresçam com a profundidade da rede. A partir dessas técnicas, ALBERT tem 18 vezes menos parâmetros que BERT-Large e pode ser treinado 1.7 vezes mais rápido (Lan et al., 2019).

Para comprovar essa tese, foi utilizado o modelo “albert-base-v2-sst2” treinado para análise de sentimentos com o dataset SST2 - Stanford Sentiment Treebank. Para se ter uma ideia, o albert-base-v2 disponível no hub tem 11.8 milhões de parâmetros.

### 5.3.3 DistilBERT

De forma similar ao ALBERT, DistilBERT - a **distilled** version of **BERT** - é uma versão modificada e menor do BERT que segue a tendencia de diminuir a quantidade de parâmetros de grandes modelos sem grandes impactos na performance. O modelo em questão foi criado pela Hugging Face e é 40% menor que BERT e 60% mais rápido em atividades de inferência. A técnica de compressão chamada destilação de conhecimentos



consiste em um modelo compacto - o estudante - treinado para reproduzir o comportamento de um modelo maior - o professor.

Em termos de arquitetura, o estudante tem uma arquitetura bem parecida com o professor - BERT. O *token-type embeddings* e *pooler* foram removidos e o número de camadas foi diminuído por um fator de 2. Apesar das mudanças e diminuição do tamanho, o modelo reteve 97% do desempenho do BERT (Sanh et al., 2019).

Foi treinado o DistilBERT, “distilbert-base-uncased” e 67 milhões de parâmetros, no dataset CoNLL2003 (NER) com o uso do *Transformers* puro e em conjunto com o PyTorch seguindo os mesmos hiperparâmetros usados no BERT. Também foram utilizados o “distilbert-base-uncased-finetuned-sst-2-english” e “distilbert-base-cased-distilled-squad” para análise de sentimentos e Q&A, respectivamente.

### 5.3.4 GPT2

O GPT2 foi desenvolvido pela OpenAI como um modelo generalista treinado com um dataset constituído de documentos de qualidade extraídos da internet. Foram usados 8 milhões de documentos com um total de 40 GB de texto, sem incluir nenhum documento da Wikipedia por ser uma fonte comum a diversos modelos. Ademais, o modelo foi criado com a arquitetura transformer unidirecional em 4 versões como é possível ver na Tabela 2 (Radford et al., 2019).

Tabela 2 – Arquitetura dos 4 modelos.

Parâmetros	Camadas
117M	12
345M	24
762M	36
762M	48

Fonte: Radford et al. (2019) (Adaptado pelo Autor).

A versão com 12 camadas foi treinado no dataset CoNLL2003 (NER) com learning rate  $5 \times 10^{-5}$ , batch size 8, weight decay 0.01 e 3 epochs. Além disso, foi escolhido o modelo de 24 camadas “gpt2-medium-finetuned-sst2-sentiment” (Q&A).

## 5.4 Atividades Contempladas

Foram escolhidas três atividades para serem contempladas de forma nativa pela plataforma:

- NER - O reconhecimento de entidades nomeadas irá apresentar a capacidade dos modelos selecionados em identificar de forma correta os substantivos que representam

nomes próprios como pessoas, organizações e locais.

- Q&A - A atividade de perguntas e resposta terá como princípio a identificação da resposta para uma pergunta a partir de um contexto (texto).
- Análise de sentimentos - A análise de sentimentos envolve classificar um segmento de texto, texto, paragrafo ou documento a partir de classificações pré-determinadas como positivo, negativo e neutro por exemplo.

Ademais, todas as atividades escolhidas podem ter seu desempenho demonstrado a partir das métricas selecionadas apresentadas na Seção 5.6 e é possível adicionar novas métricas a partir da inserção da opção.

## 5.5 Datasets

Os datasets são parte primordial para avaliar os modelos, uma vez que é a partir dos resultados obtidos nas tarefas aplicadas a eles que é possível calcular e extrair as métricas. Essa seção irá explicar quais são os datasets usados de forma direta e indireta e em quais atividades eles são utilizados.

### 5.5.1 CoNLL2003

A Message Understanding Conferences (MUC) ofereceu a desenvolvedores a oportunidade de avaliar sistemas em um mesmo dataset. Foi determinado um esquema de anotação com quatro tipos de entidades nomeadas: pessoas, locais, organizações e nomes diversos que não fazem parte das classes anteriores. O dataset, CoNLL-2003, consiste em 8 arquivos divididos em 2 linguas - Inglês e Alemão - para cada uma há um arquivo de treino, desenvolvimento, teste e um sem anotações (labels). O texto em inglês foi retirado do Reuters Corpus e o alemão do ECI Multilingual Text Corpus. Os 4 tipos de entidades nomeadas são (PER), (ORG), (LOC) e (MISC) para pessoas, organizações, locais e diversos, respectivamente (Sang; Meulder, 2003).

Foi utilizado uma versão com apenas a parte em inglês do dataset, dividido em treino (14041), validação (3250) e teste (3453) e disponível no hub com nome “conll2003” para atividades de NER. Para avaliar os modelos foi utilizado a parte de teste. Além disso, existe a classificação “O” - não é uma entidade nomeada - e no início da classificação da palavra foi adicionado “I” quando está dentro de uma frase e “B” quando está no começo, como exemplo “ORG” ficou “B-ORG” e “I-ORG”.

### 5.5.2 SST2

O SST2 - Stanford Sentiment Treebank - é composto por um conjunto de 215.154 frases únicas, anotadas manualmente por 3 jurados humanos. Nesse sentido, o dataset apresenta um label como o sentimento da frase - 0 (negativo) e 1 (positivo). As frases foram retiradas do *rottentomatoes.com* e publicados por Pang and Lee (2005). O dataset original continha 10.662 sentenças, metade positiva e a outra metade negativa. A partir delas, foi utilizado o Stanford Parser que dividiu cada uma delas, em 1.100 casos foram criadas múltiplas frases (Socher et al., 2013).

Esse dataset foi utilizado para treinar os modelos “distilbert-base-uncased-finetuned-sst-2-english”, “gpt2-medium-finetuned-sst2-sentiment” e “albert-base-v2-sst2”.

### 5.5.3 IMDB

O IMDB - Internet Movie Database - é um dataset com análises de filmes da internet. Normalmente, é dado uma pontuação representada em estrelas (1, 2 ..., 10), porém foram substituídas para os labels 0 (negativo) e 1 (positivo). O dataset de treino e teste apresentam 25 mil análises, além de 50 mil sem labels. Este ultimo conjunto dispõe de análises neutras. Foi filtrado para que não houvesse mais de 30 análise por filme e há um número igual de positivas e negativas para prevenir muitos acertos aleatórios. Para o conjunto de treino e teste só foram selecionadas análises altamente polarizadas, negativo com uma pontuação  $\leq 4$  e positivo uma pontuação  $\geq 7$  de 10 (Maas et al., 2011).

Foi utilizado o dataset disponível a partir do hub com o nome “imdb”, seguindo a mesma divisão do original. Para os testes foi utilizada a divisão “test” com 25 mil análises para avaliar análise de sentimentos.

### 5.5.4 SQUaD

O SQUaD (v1.0)- Stanford Question Answering Dataset - é um dataset de alta qualidade, com tamanho considerável (107.785 pares) composto por pares de perguntas e resposta e um contexto de onde resposta pode ser extraída. Criado por empregados da internet em 536 artigos da Wikipedia, diferente de outros datasets ele apresenta apenas uma resposta para cada pergunta no dataset de treino. Por outro lado, o dataset de validação apresenta um conjunto com 3 respostas para cada pergunta. Os artigos foram escolhidos da pagina interna do projeto Nayuki para obter os melhores 1000 artigos em inglês. Destes, foram selecionados aleatoriamente 536 e escolhido um paragrafo com menos de 500 caracteres de cada. Os parágrafos resultantes são sobre diversos tópicos, desde celebrações musicais até conceitos abstratos. Foi feita a divisão em treino (80%), desenvolvimento (10%) e teste (10%) (Rajpurkar et al., 2016).

O dataset foi escolhido para ser utilizado na avaliação de Q&A a partir da divisão de validação. Ele está disponível com o nome “squad” no hub.

## 5.6 Métricas utilizadas

Para explicar as capacidades dos modelos avaliados nos datasets descritos e nas atividades selecionadas, foram escolhidas 4 métricas: *Accuracy*, *Precision*, *Recall* e *F1-Score*. Como já mencionado anteriormente, essas são as métricas mais comuns em NLP e, apesar de terem problema em cenários de avaliação complexos, em atividades que podem ser interpretadas de forma binária, são uma excelente opção. As atividades escolhidas podem ser descritas a partir de uma interpretação de certo e errado que vem de encontro com a definição de TP, TN, FP e FN. Com isso em mente, segue as particularidades de cada atividade:

- **NER** - A atividade de reconhecimento de entidade nomeada depende da correta classificação das palavras presentes em uma sentença, texto ou documento com base nas características que o modelo foi treinado (como local, pessoa, organização, data etc.). Com isso em mente, calcular as métricas se torna simples, pois basta comparar a saída do modelo em sua classificação com aquela correta para verificar se há uma correspondência que deve ser exata.
- **Análise de Sentimentos** - Para análise de sentimentos, uma tarefa, no cenário proposto, de natureza binária - o resultado é 0 (negativo) ou 1 (positivo) a tarefa a verificação de um acerto é feita a partir da comparação entre o resultado esperado e a saída do modelo. O resultado deve ser exato para que a classificação seja considerada correta, caso contrario está errado.
- **Q&A** - Ao avaliar a atividade de perguntas e resposta é necessário levar em conta que é possível que haja mais de uma resposta correta no texto, como por exemplo a pergunta “Onde você mora?” pode ser respondida de algumas maneiras como “Vila Velha”, “Vila Velha, Espirito Santo” e “Vila Velha, ES”. Com isso em mente, é observado as 3 opções de respostas presentes no dataset utilizado, SQUAD, e caso alguma das possíveis repostas coincida com a apresentada pelo modelo, então será considerado como certo, caso não esteja, estará errado.

## 5.7 Etapas de Desenvolvimento

Essa sessão irá descrever as etapas envolvidas no processo de desenvolvimento da plataforma.

### 5.7.1 Avaliação Individual

Para começar foram estabelecidas quais as atividades seriam utilizadas: NER, análise de sentimentos e Q&A. A partir disso, começou a busca por modelos, no hub de modelos Hugging Face, que pudessem ser utilizados e surgiu a ideia de fazer o processo de *fine-tuning* manualmente. Para isso foram escolhidos BERT, DistilBERT e GPT2 e foi criado o código para importar e instanciar cada modelo, além do pré-processamento de cada um. Com todos prontos, foi escolhido começar com a atividade NER e o dataset CoNLL2003, antes de treinar, foram definidas as métricas para serem utilizadas, que fizessem sentido nas 3 atividades propostas, com a escolha de *accuracy*, *precision*, *recall* e *f1-score*. A partir dessas escolhas, uma função foi definida para extrair as métricas que fosse comum a todos os modelos. O treinamento *fine-tuning* começou com BERT, seguido por GPT2 e por ultimo DistilBERT.

Com o intuito de acelerar o processo de avaliação e possibilitar que todas as atividades tenham 3 modelos cada, foi pulado o processo de *fine-tuning*, com a escolha de modelos já treinados nas tarefas específicas. Isso se deve, principalmente, as dificuldades de treinar com recursos escassos e o grande tempo necessário para o processo. Apesar dessa maior facilidade, ainda foi necessário criar o código para pré-processamento, escolha de datasets e função de avaliação para cada tipo de atividade. Seguindo NER, foi feito a avaliação de modelos de análise de sentimentos e por fim Q&A.

Apesar de já ter um processo de avaliação funcionando, na prática ainda se tratava de células em um Jupyter Notebook sem relação entre si, o que leva a próxima etapa, a agregação 5.7.2. O Código 1 mostra como foi feito o treinamento e avaliação com DistilBERT, note que se trata de um código extenso e requer diversas funções.

Código 1 – Código Python responsável por treinar e avaliar o modelo DistilBERT

```
1 # Carregar o dataset CoNLL-2003
2 dataset = load_dataset("conll2003", trust_remote_code=True)
3
4 # Carregar o tokenizer e o modelo DistilBERT pré-treinado
5 tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")
6 model = AutoModelForTokenClassification.from_pretrained("distilbert-base-uncased", num_labels=len(dataset['train'].features['ner_tags'].feature.names))
7
8 label_list = dataset['train'].features['ner_tags'].feature.names
9
10 # Função de tokenização
11 # https://gist.github.com/acoadmarmon/20c1167692da23f946a8da39904ec6b9
12 def tokenize_and_align_labels(examples):
```

```

13     tokenized_inputs = tokenizer(examples['tokens'], truncation=True,
padding='max_length', is_split_into_words=True, max_length=128)
14     labels = []
15     for i, label in enumerate(examples['ner_tags']):
16         word_ids = tokenized_inputs.word_ids(batch_index=i)
17         previous_word_idx = None
18         label_ids = []
19         for word_idx in word_ids:
20             if word_idx is None:
21                 label_ids.append(-100)
22             elif word_idx != previous_word_idx:
23                 label_ids.append(label[word_idx])
24             else:
25                 label_ids.append(-100)
26                 previous_word_idx = word_idx
27         labels.append(label_ids)
28     tokenized_inputs["labels"] = labels
29     return tokenized_inputs
30
31 # Tokenizar o dataset
32 tokenized_datasets = dataset.map(tokenize_and_align_labels, batched=True
    )
33
34 # Carregar a métrica
35 metric = load_metric("segeval", trust_remote_code=True)
36
37 # Função de cálculo das métricas
38 def compute_metrics(p):
39     predictions, labels = p
40     predictions = np.argmax(predictions, axis=2)
41
42     true_labels = [[label_list[l] for l in label if l != -100] for label
in labels]
43     true_predictions = [
44         [label_list[p] for (p, l) in zip(prediction, label) if l !=
-100]
45         for prediction, label in zip(predictions, labels)
46     ]
47
48     results = metric.compute(predictions=true_predictions, references=
true_labels)
49     return {
50         "accuracy": results["overall_accuracy"],
51         "precision": results["overall_precision"],
52         "recall": results["overall_recall"],
53         "f1": results["overall_f1"],
54     }

```

```
55
56 # Configurações de treinamento
57 training_args = TrainingArguments(
58     output_dir="./results",
59     eval_strategy="epoch",
60     learning_rate=2e-5,
61     per_device_train_batch_size=16,
62     per_device_eval_batch_size=16,
63     num_train_epochs=3,
64     weight_decay=0.01,
65 )
66
67 # Inicializar o Trainer
68 trainer = Trainer(
69     model=model,
70     args=training_args,
71     train_dataset=tokenized_datasets["train"],
72     eval_dataset=tokenized_datasets["validation"],
73     tokenizer=tokenizer,
74     compute_metrics=compute_metrics,
75 )
76
77 # Treinar o modelo
78 trainer.train()
79
80 # Avaliar o modelo
81 eval_results = trainer.evaluate()
82
83 print(eval_results)
```

### 5.7.2 Agregação e Primeira Versão

A partir do código distribuído em células, foram criadas funções para agregar o código em um só lugar, dando origem a primeira versão da plataforma. Funções como “criar\_modelo”, “escolha\_dataset”, “escolha\_pre\_processamento” e “avaliar\_modelo” foram criadas, cada um com o objetivo de realizar um dos processos feitos outrora juntos. O encontro das funções era feito em na função principal que chamava internamente as outras com o nome de “main”. O processo para fazer a avaliação era feita com o uso de print e input, nativos do Python, com uma serie de repostas do usuário para escolher o modelo e dataset, avaliar e o resultado era salvo em um arquivo CSV local com o nome “result”. É importante ressaltar que nesse estágio ainda não era possível utilizar um modelo, dataset, pré-processamento ou função de avaliação próprios sem a alteração do código fonte.

A Figura 21 mostra a resposta à primeira pergunta e como a plataforma funcionava em suas versões iniciais.

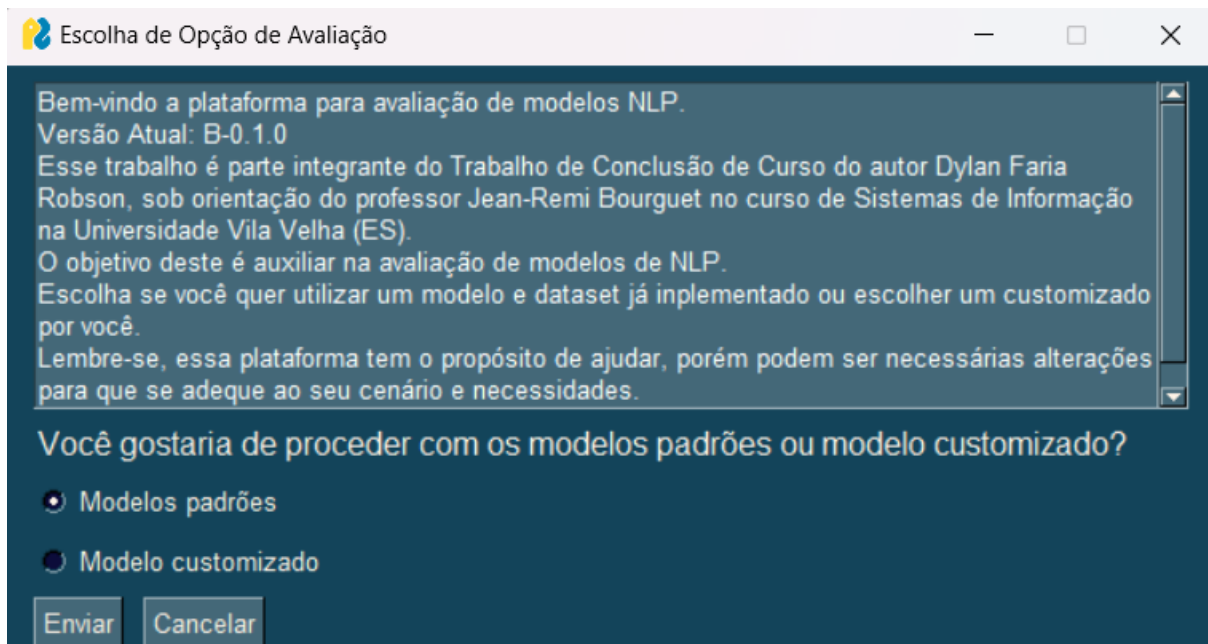




claro com alto contraste de cores e a possibilidade de rever todas as respostas antes de prosseguir com o botão “Enviar”.

A Figura 23 mostra a interface grafica criada. A janela apresenta a primeira sessão de perguntas e suas respectivas opções.

Figura 23 – Uso da GUI na plataforma.



Fonte: Elaborado pelo Autor.

### 5.7.5 Documentação

Uma vez que a plataforma já estava funcional em um estado considerado aceitável, enquanto MVP (*Minimum Viable Product*), foi dado prosseguimento para a documentação da solução proposta. Isso foi feito a partir da criação de docstring - uma documentação em código utilizada em Python que mostra o que uma função faz e como deve ser utilizada, podendo conter informações como a saídas (*return*) (Dainese; Ilin; Marttinen, 2024). Foi seguido um padrão com uma breve descrição do que a função faz, seguido de mais detalhes em um parágrafo subjacente, caso haja parâmetros, uma descrição do que representam e por fim as saídas. Ademais, durante essa fase foram criados diversos comentários com a explicação do código.

A Figura 24 é da função “criar\_modelo” que recebe um parâmetro e retorna um modelo.

### 5.7.6 Melhorias Adicionais

Seguindo o conceito reconhecido na Gestão da Qualidade Total, “nada é tão bom que não possa ser melhorado” (Galvão, 2011), a plataforma ainda podia passar por diversas

Figura 24 – Documentação da função `criar_modelo`.

```
def criar_modelo(nome):  
    """  
    Cria e retorna um modelo fine-tuned e a atividade associada com base no nome fornecido.  
  
    A função aceita um nome de modelo específico, que é utilizado para carregar um modelo fine-tuned  
    correspondente da biblioteca Hugging Face. Com base no modelo selecionado, a função também retorna  
    a atividade correspondente, como NER, análise de sentimentos ou Q&A. Se o nome do modelo fornecido  
    não for encontrado, uma exceção será levantada.  
  
    Parameters:  
    | - nome (str): O nome do modelo fine-tuned a ser carregado.  
  
    Returns:  
    | - modelo (transformers.PreTrainedModel): O modelo pré-treinado correspondente.  
    | - atividade (str): A atividade associada ao modelo (NER, SENTIMENT, ou Q&A).  
    """
```

Fonte: Elaborado pelo Autor.

melhorias. Primeiro, foram criadas diversas exceções em locais que foram identificados os erros mais comuns que os usuários podem cometer. Essas exceções foram adicionadas como uma informação no docstring das funções modificadas. Ademais, foi feito o processo de refatoração do código com técnicas como decomposição de condicional e extração de métodos com o intuito de facilitar o entendimento do código e manutenção (Python..., 2024).

A Figura 25 ilustra o processo de decomposição de condicional e extração de métodos.

Figura 25 – Exemplo de refatoração de código.

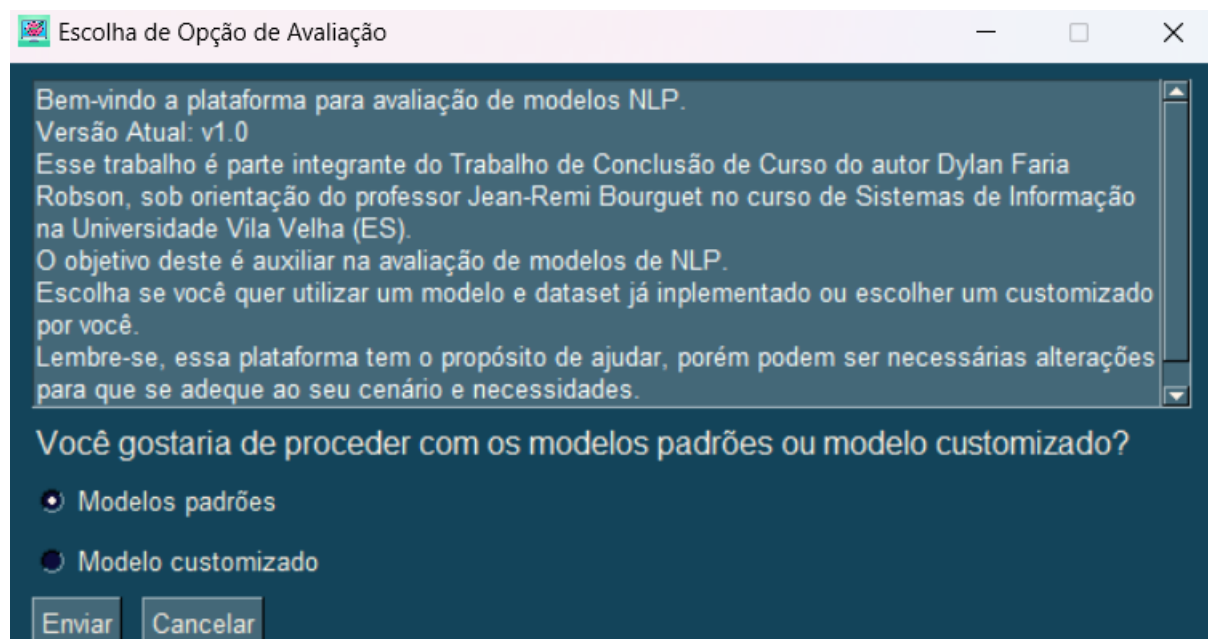
Antes:	Após:
<pre>def calcular_preco(categoria, desconto):     if categoria == 'eletrônico':         if desconto:             return 900         else:             return 1000     elif categoria == 'móvel':         if desconto:             return 450         else:             return 500     elif categoria == 'roupa':         if desconto:             return 90         else:             return 100     else:         return "Categoria desconhecida"</pre>	<pre>def calcular_preco(categoria, desconto):     preco_base = obter_preco_base(categoria)     return aplicar_desconto(preco_base, desconto)  def obter_preco_base(categoria):     if categoria == 'eletrônico':         return 1000     elif categoria == 'móvel':         return 500     elif categoria == 'roupa':         return 100     else:         return "Categoria desconhecida"  def aplicar_desconto(preco_base, desconto):     if isinstance(preco_base, str): # Se for uma mensagem de erro         return preco_base     return preco_base * 0.9 if desconto else preco_base</pre>

Fonte: Elaborado pelo Autor.

### 5.7.6.1 Versão Final

Após as melhorias anteriormente mencionadas, foi criado um ícone e refinado a documentação a partir da criação de textos descritivos para as seções utilizadas no Jupyter Notebook. A Figura 26 mostra a tela inicial já com o ícone aplicado. O código e outros arquivos relevantes estão disponíveis a partir do link<sup>9</sup>.

Figura 26 – GUI na versão final.



Fonte: Elaborado pelo Autor.

<sup>9</sup> <<https://github.com/MrRobson9/TCC-II/tree/main>>

## 6 Resultados

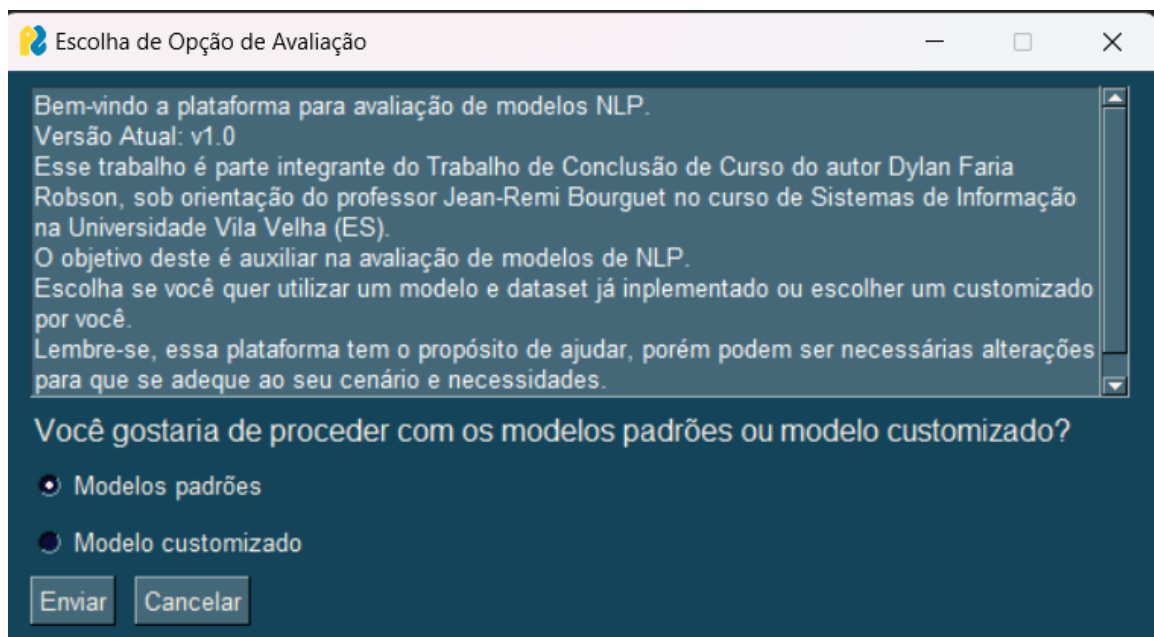
Esse capítulo tem como propósito demonstrar os resultados obtidos durante o desenvolvimento da solução proposta. Não foi utilizado o ícone criado na versão apresentada na Figura 26, mostrando que isso pode ser desativado sem alterar qualquer outro aspecto da plataforma por ser opcional.

### 6.1 Fluxo da Plataforma

#### 6.1.1 Fluxo de Modelos Padrões

Para selecionar o fluxo com modelo, pré-processamento, *dataset* e método de avaliação já implementados, basta selecionar a opção “Modelos padrões” e clicar em “Enviar”, conforme mostra a Figura 27.

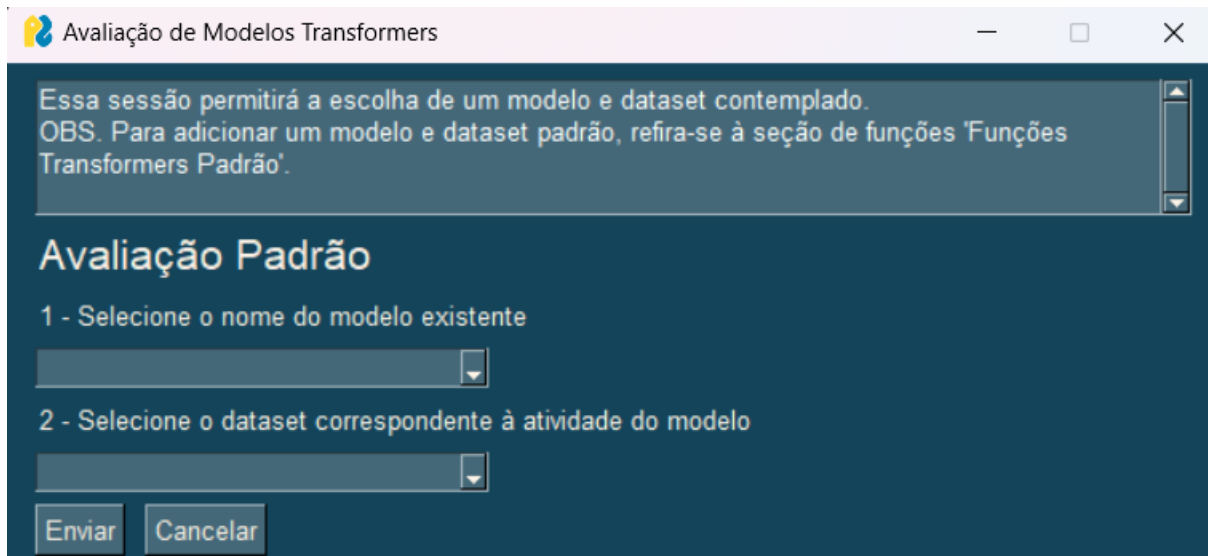
Figura 27 – Fluxo padrão - 1.



Fonte: Elaborado pelo Autor

Após isso será aberta uma nova tela com um campo para selecionar um modelo de uma lista pré-criada e um dataset. A Figura 28 mostra a interface utilizada. As opções de modelo são “BERT\_NER”, “GPT\_NER”, “DISTILBERT\_NER”, “DISTILBERT\_SENTIMENT”, “GPT\_SENTIMENT”, “ALBERT\_SENTIMENT”, “DISTILBERT\_Q&A”, “BERT\_BASE\_Q&A” e “BERT\_LARGE\_Q&A”. Já os datasets são “conll2003” (NER), “IMDB” (Sentiment) e “SQUaD” (Q&A).

Figura 28 – Fluxo padrão - 2.



Avaliação de Modelos Transformers

Essa sessão permitirá a escolha de um modelo e dataset contemplado.  
OBS. Para adicionar um modelo e dataset padrão, refira-se à seção de funções 'Funções Transformers Padrão'.

### Avaliação Padrão

1 - Selecione o nome do modelo existente

2 - Selecione o dataset correspondente à atividade do modelo

Enviar Cancelar

Fonte: Elaborado pelo Autor.

Após clicar em “Enviar” começará o processo de carregar os modelo, *dataset* e qualquer outra informação relevante e então o processo de avaliação será iniciado. A Figura 29 mostra o processo em execução.

Figura 29 – Fluxo padrão - final.



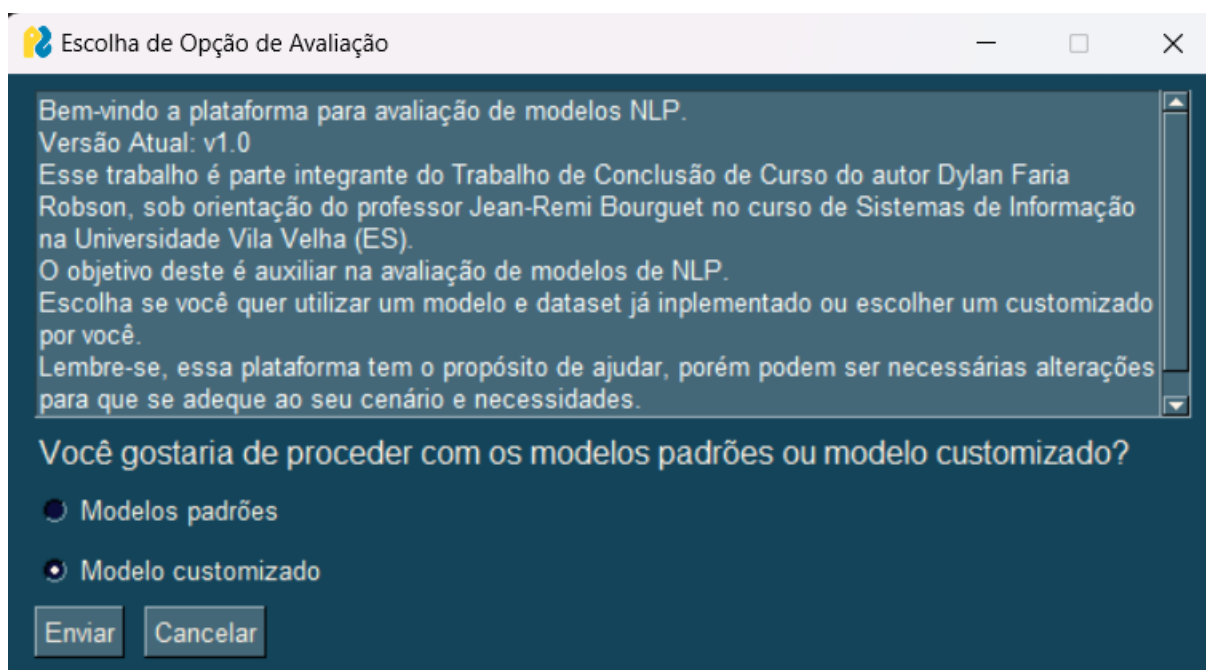
Fonte: Elaborado pelo Autor.

No final será retornada uma tabela com as métricas, conforme explicado em 6.2.

### 6.1.2 Fluxo de Modelos Customizados

Para utilizar os modelos customizados a partir da modularidade da plataforma, primeiramente, selecione a opção “Modelo customizado” conforme exemplificado na Figura 30.

Figura 30 – Fluxo customizado - 1.



Fonte: Elaborado pelo Autor.

Agora é necessário especificar um nome para o modelo utilizado, atividade e *dataset*. Essas informações serão essenciais para compor a tabela com os resultados da avaliação, uma vez que o usuário tem liberdade para escolher o modelo da arquitetura de sua preferencia, atividade (dentro das opções apresentadas com possibilidade de alterar o código para adicionar mais) e dataset em qualquer formato. Com isso em mente, não é possível prever os itens utilizados na avaliação e foi escolhido não proceder com a geração de nomes automáticos como “Modelo 1” já que seria muito genérico e confuso. A Figura 31 mostra essa etapa.

Na próxima etapa é necessário criar algumas funções customizadas pelo usuário:

- Função de criação do modelo - Deve ser possível chamar sem utilizar parâmetros e retornar um modelo (no formato que será avaliado).
- Função de criação do dataset - Similar a função de criação do modelo, porém retorna um dataset.
- Função de pré-processamento (opcional) - Uma função que recebe o dataset e o retorna pós-processado.
- Função de Avaliação - A forma que o modelo será avaliado. Recebe o modelo e o dataset (processado ou não), retornar um dicionario ou *padas.Series* com os campos “eval\_accuracy”, “eval\_precision”, “eval\_recall” e “eval\_f1”.

Figura 31 – Fluxo customizado - 2.

Configuração do Modelo Customizado

Essa sessão tem como objetivo ajudar na avaliação de um modelo personalizado. Para isso será necessário, no mínimo, uma função que retorne um modelo já treinado, uma outra que retorne o dataset que será utilizado e uma função de avaliação. É opcional a utilização de pré-processamento (atente-se a necessidade do seu modelo e dataset)

Primeiramente escolha um nome para o seu modelo e dataset. Essa será a forma que eles apareceram na tabela com as métricas.

Digite o nome do modelo (como será apresentado na tabela de métricas):

Llama2\_NER\_conll2003

Qual atividade será avaliada (escolha um número):

☒ NER ☐ Análise de Sentimentos ☐ Q&A

Nome do dataset (como aparecerá na tabela de métricas):

conll2003

Próximo

Fonte: Elaborado pelo Autor.

Todas as funções supracitadas devem estar salvas em memória (ter sido especificadas antes) e são fornecidas como *String* com o nome da função como “criar\_modelo” que será chamado como “criar\_modelo()”. A Figura 32 mostra a seleção de não utilizar o pré-processamento. Já a Figura 33 demonstra a opção para utilizar.

Figura 32 – Fluxo customizado - 3.

Configuração das Funções Customizadas

Agora você deve ter salvo em memória as funções:

- 1 - Uma função que seja chamada sem argumentos e retorne um modelo como 'criar\_modelo\_1'
- 2 - Uma função que seja chamada sem argumentos e retorne o dataset como 'carregar\_dataset\_A'
- 3 (Opcional) - Uma função que seja chamada passando o dataset como argumento e retorne sua versão processada como 'tokenizacao' que será utilizado como 'tokenizacao(dataset)'.

Nome da função que retorna o modelo carregado:

criar\_modelo\_generico

Nome da função que retorna o dataset carregado:

carregar\_dataset\_conll2003

O dataset precisará passar pelo processo de pré-processamento?

☐ Sim ☒ Não

Próximo

Fonte: Elaborado pelo Autor.

Figura 33 – Fluxo customizado - 4.

Configuração das Funções Customizadas

Agora você deve ter salvo em memória as funções:

- 1 - Uma função que seja chamada sem argumentos e retorne um modelo como 'criar\_modelo\_1'
- 2 - Uma função que seja chamada sem argumentos e retorne o dataset como 'carregar\_dataset\_A'
- 3 (Opcional) - Uma função que seja chamada passando o dataset como argumento e retorne sua versão processada como 'tokenizacao' que será utilizado como 'tokenizacao(dataset)'.

Nome da função que retorna o modelo carregado:

Nome da função que retorna o dataset carregado:

O dataset precisará passar pelo processo de pré-processamento?

☒ Sim ☐ Não

Nome da função de pré-processamento:

Próximo

Fonte: Elaborado pelo Autor.

A Figura 34 mostra a janela para escrever o nome da função de avaliação.

Figura 34 – Fluxo customizado - 5.

Configuração da Avaliação Customizada

Por fim, deve ser fornecido o nome de uma função que receba como parametros o modelo e dataset (nessa ordem) e retorne um pandas.Series ou dicionario contendo 'eval\_accuracy', 'eval\_precision', 'eval\_recall' e 'eval\_f1'.  
Ex.: 'avaliar\_modelo' que será utilizado como 'avaliar\_modelo(modelo, dataset)'.

Nome da função para Avaliar o modelo:

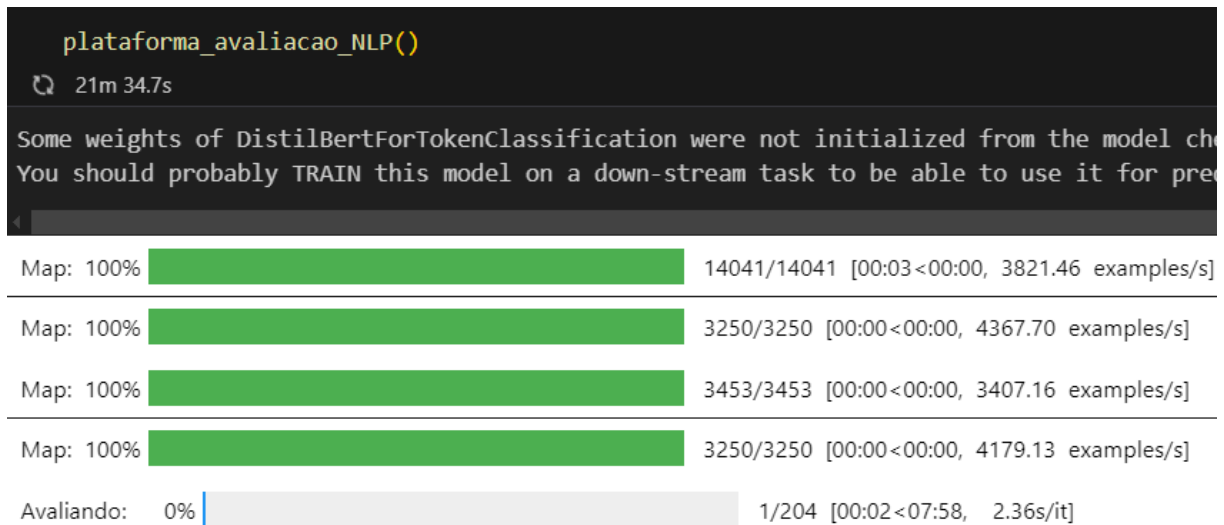
Enviar

Fonte: Elaborado pelo Autor.

Após isso, é mostrado para o usuário o progresso da avaliação, após clicar em “Enviar”, conforme a Figura 35



Figura 35 – Fluxo customizado - 6.



Fonte: Elaborado pelo Autor

Por fim, os resultados dos testes anteriores são mostrados. A Figura 36 mostra a saída final da plataforma com a tabela que já foi salva em um arquivo CSV local.

Figura 36 – Fluxo customizado - final.

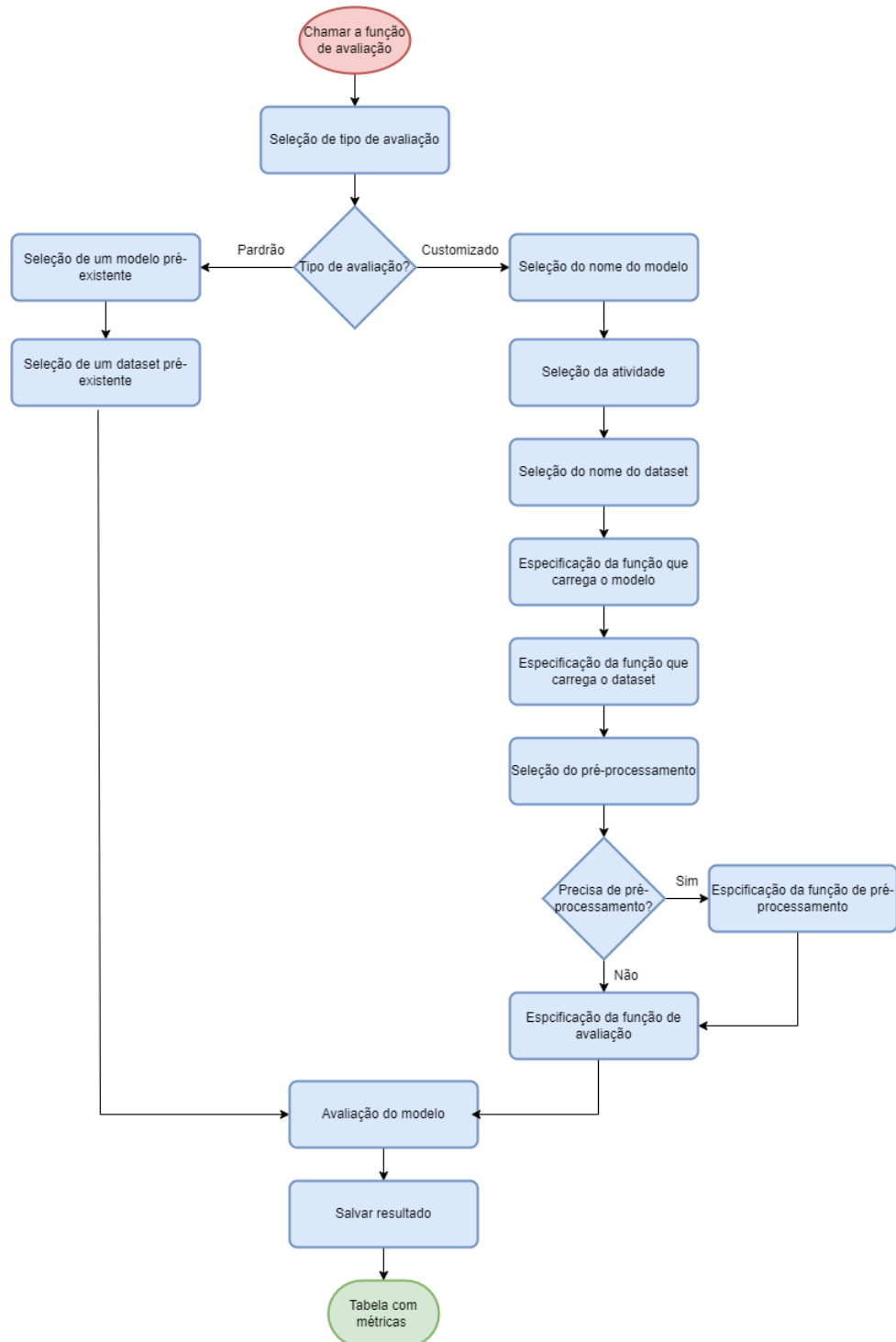
	Modelo	Atividade	Dataset	accuracy	precision	recall	f1-score
1	DISTILBERT_NER_PT	NER	conll2003	0.987	0.987	0.987	0.987
0	BERT_NER	NER	conll2003	0.991	0.946	0.953	0.950

Fonte: Elaborado pelo Autor.

### 6.1.3 Fluxograma

A Figura 37 apresenta um fluxograma que demonstra as maneiras que a plataforma pode ser utilizada.

Figura 37 – Fluxograma da plataforma.



Fonte: Elaborado pelo Autor.

Conforme mostrado no fluxograma, é possível utilizar funções e modelos já prontos ou utilizar os customizados e, no fim do processo, sempre ocorre a avaliação do modelo

para extrair as métricas, salva-las e as mostrar.

## 6.2 Métrica com Resultados

Ao longo do desenvolvimento da plataforma, foi possível testar um total de 10 modelos apresentados. Agora as métricas serão apresentadas e será feita uma análise destas.

### 6.2.1 Reconhecimento de Entidades Nomeadas

Foram avaliados 4 modelos, BERT, DistilBERT (*Transformers*), GPT2 e DistilBERT (Pytorch), referidos, respectivamente como BERT\_NER, GPT\_NER, DTLBERT\_NER e DTLBERT\_NER\_PT. Todos foram treinados com o dataset CoNLL2003. A Tabela 3 mostra o desempenho de cada um.

Tabela 3 – Métricas NER.

Modelo	Atividade	Dataset	accuracy	precision	recall	f1-score
DTLBERT_NER_PT	NER	cll2003	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>
BERT_NER	NER	cll2003	0.99	0.95	0.95	0.95
DTLBERT_NER	NER	cll2003	0.99	0.94	0.94	0.94
GPT_NER	NER	cll2003	0.97	0.78	0.84	0.81

Fonte: Elaborado pelo Autor.

É possível perceber que o treinamento do modelo DistilBERT utilizando Pytorch obteve os melhores resultados, empatando ou sendo superior aos concorrentes em todas as métricas, com um *f1-score* quase perfeito. Isso pode ser devido a vários fatores como o *Data Collection* utilizado, que foi configurado manualmente, enquanto o *Transformers* já vem pré-definido. De qualquer forma, a utilização de uma biblioteca de mais baixo nível se mostrou melhor no caso específico estudado.

Além disso, a bidirecionalidade do BERT se mostrou bem efetiva, ao possibilitar um desempenho bem melhor que o GPT2 *fine-tuned*. Outro ponto interessante é que mesmo com um tamanho consideravelmente menor, DISTILBERT treinado com *Transformers* conseguiu um desempenho praticamente idêntico ao seu irmão maior.

### 6.2.2 Análise de Sentimentos

Foram avaliados 3 modelos - ALBERT, GPT2 e DISTILBERT, apelidados de ALBERT\_SENT, GPT\_SENT, e DTLBERT\_SENT. A Tabela 4 demonstra os resultados.

A partir da análise da Tabela 4 é possível concluir, que apesar das diferenças em arquitetura dos modelos utilizados e tamanho, todos tiveram resultados muito similares

Tabela 4 – Métricas Análise de Sentimentos.

Modelo	Atividade	Dataset	accuracy	precision	recall	f1-score
ALBERT_SENT	SENTIMENT	imdb	<b>0.91</b>	<b>0.94</b>	0.87	<b>0.90</b>
GPT_SENT	SENTIMENT	imdb	0.90	0.91	<b>0.88</b>	0.90
DTLBERT_SENT	SENTIMENT	imdb	0.89	0.91	0.86	0.88

Fonte: Elaborado pelo Autor.

na atividade proposta, de forma que a pode-se considerar um empate técnico, uma vez que os resultados são tão próximos que a utilização de outro dataset pode permitir que o modelo com a menor pontuação seja o melhor classificado e vice-versa.

### 6.2.3 Perguntas e Respostas

Os modelos BERT Large, BERT Base e DistilBERT *fine-tuned* com o dataset SQUaD foram testados, com os nomes BERT\_LARGE\_Q&A, BERT\_BASE\_Q&A e DTLBERT\_Q&A, respectivamente. Os resultados são apresentados na Tabela 4.

Tabela 5 – Métricas de Perguntas e Respostas.

Modelo	Atividade	Dataset	accuracy	precision	recall	f1-score
BERT_LARGE_Q&A	Q&A	squad	<b>0.83</b>	<b>0.84</b>	<b>0.83</b>	<b>0.84</b>
BERT_BASE_Q&A	Q&A	squad	0.78	0.78	0.78	0.78
DTLBERT_Q&A	Q&A	squad	0.77	0.77	0.77	0.77

Fonte: Elaborado pelo Autor.

Como todos utilizam a mesma arquitetura, é possível perceber que BERT Large se beneficiou da maior quantidade de parâmetros e tokens para assegurar a liderança contra seus concorrentes. Já a disputa pelo segundo lugar foi bem acirrada, com uma diferença em pontuação de apenas 1% em cada métrica para BERT Base contra o DistilBERT, mostrando que apesar da diferença de tamanho do modelo, a versão *distilled* conseguiu um desempenho quase idêntico a versão maior e original.

### 6.2.4 Todas as Atividades e Modelos

A Tabela 6 mostra o desempenho de todos os modelos testados, nas diferentes tarefas e com seus respectivos *datasets* e métricas.

Tabela 6 – Métricas de todas as atividades.

Modelo	Atividade	Dataset	accuracy	precision	recall	f1-score
DTLBERT_NER_PT	NER	cll2003	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>	<b>0.99</b>
BERT_NER	NER	cll2003	0.99	0.95	0.95	0.95
DTLBERT_NER	NER	cll2003	0.99	0.94	0.94	0.94
GPT_NER	NER	cll2003	0.97	0.78	0.84	0.81
ALBERT_SENT	SENTIMENT	imdb	0.91	0.94	0.87	0.90
GPT_SENT	SENTIMENT	imdb	0.90	0.91	0.88	0.90
DTLBERT_SENT	SENTIMENT	imdb	0.89	0.91	0.86	0.88
BERT_LARGE_Q&A	Q&A	squad	0.83	0.84	0.83	0.84
BERT_BASE_Q&A	Q&A	squad	0.78	0.78	0.78	0.78
DTLBERT_Q&A	Q&A	squad	0.77	0.77	0.77	0.77

Fonte: Elaborado pelo Autor.

Com base nas informações e métricas obtidas, é possível concluir que no ambiente de avaliação criado, as atividades de NER tiveram desempenho melhor, seguida pela Análise de Sentimentos e por fim Perguntas e Respostas. A atividade de NER requer a previsão de muitos tokens e a maioria é classificada como “O” por não representar uma entidade nomeada, com isso a quantidade de acertos é maior, o que reflete em métricas melhores.

No caso de análise de sentimentos utilizada é necessário definir entre duas possibilidades - positivo (1) e negativo (0) - o que significa que há poucas possibilidades de classificação e mesmo se o modelo não tiver grande confiança, ele ainda terá 50% de chance de acertar.

Por fim, perguntas e respostas é o cenário mais desafiador, uma vez que só é possível ter uma resposta correta de forma consistente se o modelo for capaz de interpretar a pergunta e localizar a informação no texto, algo que envolve mais passos e é muito difícil de acertar por sorte.

## 7 Conclusão

### 7.1 Considerações Finais

Seguindo o projeto apresentado em 4, foi possível desenvolver uma plataforma que atingisse os objetivos gerais especificados em 1.2.1, com a extração de métricas padronizadas em diferentes atividades, conjuntos de dados e modelos de NLP. Ademais, quanto aos objetivos específicos retratados em 1.2.2, foi feito um estudo de NLP, definidas e conceituadas métricas, atividades, *datasets* e modelos, o desenvolvimento da plataforma e a análise das métricas extraídas.

A partir desse trabalho foi possível compreender a importância do NLP, suas principais aplicações e o longo caminho que ainda há no desenvolvimento e aperfeiçoamento de modelos de linguagem. Nesse mesmo âmbito, a avaliação dos modelos é parte fundamental do processo de criação e melhoria contínua de modelos.

Conforme a plataforma proposta, utilizar métricas de forma padronizada é extremamente relevante para possibilitar a comparação de forma efetiva entre modelos, de forma a evitar que todos "tenham o melhor modelo" com base em metodologias de avaliação diferentes. Apesar disso, é importante ressaltar que não existe uma métrica universal para avaliar todos os modelos em qualquer atividade, sendo necessário compreender a natureza da tarefa realizada para seleção desta.

Com base nisso, foi escolhido modularizar a plataforma, de forma a dar controle ao usuário final do código para que possa fazer ajustes e adequações às suas preferências e especificidades. Com a quantidade de atividades e modelos NLP surgindo em velocidade cada vez maior, essa propriedade garante que a plataforma não se torne rapidamente obsoleta, podendo ser atualizada de forma simples. Dito isto, é importante considerar que a criação, teste e manuseio de modelos exige conhecimentos técnicos específicos usuário ou mesmo desenvolvedor.

O estudo de caso apresentado levou em consideração três atividades: Reconhecimento de Entidades Nomeadas, Perguntas e Respostas e Análise de Sentimentos, com 10 modelos que utilizam a arquitetura *transformer*, variando de 12 milhões a 380 milhões de parâmetros, lançados entre 2018 e 2019. Nesse sentido, os resultados alcançados foram promissores, com uma análise dos resultados extraídos.

Ademais, o trabalho foi limitado devido ao tempo e recursos computacionais necessários para avaliar modelos maiores como Llama 3.1, Gemma, Phi-3.5, entre outros LLM.

## 7.2 Trabalhos Futuros

Para trabalhos futuros, pode ser feita uma plataforma WEB que possibilite o compartilhamento do código criado por usuários da comunidade e que permita mostrar o ranking dos melhores modelos com filtros interativos. Como inspiração, pode-se utilizar as ideias já validadas por *benchmarks* como GLUE e SuperGLUE.

Além disso, o teste com modelos maiores e computacionalmente mais exigentes deve ser considerado, a partir de uma análise da evolução dos modelos ao longo do tempo, levando em conta seu ano de lançamento, número de parâmetros e performance alcançada.

Ademais, há a possibilidade de construir uma API para testes de modelos em servidores, possibilitando o uso de hardware mais potente com a utilização de *cluster*.

Por fim, é válido o estudo de novas métricas, atividades e *datasets* para enriquecer o repertório criado para uma análise mais completa e ampla da performance de modelos NLP.

## Referências

- AGRAWAL, T. Hyperparameter optimization in machine learning. *Springer*, Apress, 2021. Citado 2 vezes nas páginas 41 e 42.
- ANDONIE, R. Hyperparameter optimization in learning systems. *J. Membr. Comput.*, v. 1, n. 4, p. 279–291, 2019. Available on: <<https://doi.org/10.1007/s41965-019-00023-0>>. Citado 2 vezes nas páginas 41 e 42.
- BAJI, T. Evolution of the gpu device widely used in ai and massive parallel processing. In: *2018 IEEE 2nd Electron Devices Technology and Manufacturing Conference (EDTM)*. 2018. p. 7–9. Citado na página 65.
- BIRHANE, A.; KASIRZADEH, A.; LESLIE, D.; WACHTER, S. Science in the age of large language models. *Nature Reviews Physics*, Nature Publishing Group UK London, v. 5, n. 5, p. 277–280, 2023. Citado na página 59.
- BISHOP, C. M. Pattern recognition and machine learning. *Springer google schola*, v. 2, p. 645–678, 2006. Citado 2 vezes nas páginas 35 e 36.
- BOURGUET, J.; SILVA, W.; OLIVEIRA, E. de. Minimalist fitted bayesian classifier-based on likelihood estimations and bag-of-words. In: BERGET, G.; HALL, M. M.; BRENN, D.; KUMPULAINEN, S. (Ed.). *Linking Theory and Practice of Digital Libraries - 25th International Conference on Theory and Practice of Digital Libraries, TPDL 2021, Virtual Event, September 13-17, 2021, Proceedings*. Springer, 2021. (Lecture Notes in Computer Science, v. 12866), p. 17–28. Available on: <[https://doi.org/10.1007/978-3-030-86324-1\\_2](https://doi.org/10.1007/978-3-030-86324-1_2)>. Citado na página 42.
- CAMPBELL, M.; HOANE, A.; HSU, F. hsiung. Deep blue. *Artificial Intelligence*, v. 134, n. 1, p. 57–83, 2002. ISSN 0004-3702. Available on: <<https://www.sciencedirect.com/science/article/pii/S0004370201001291>>. Citado na página 27.
- CHOWDHARY, K. *Fundamentals of artificial intelligence*. : Springer, 2020. Citado 3 vezes nas páginas 27, 35 e 38.
- COSTA, M. Z. et al. Automated semantic annotation pipeline for brazilian judicial decisions. In: *Legal Knowledge and Information Systems*. : IOS Press, 2024. p. 226–238. Citado na página 54.
- COSTA, M. Z.; VIEIRA, T. B. P.; BOURGUET, J.; GUIZZARDI, G.; ALMEIDA, J. P. A. Enhancing access to legal data through ontology-based representation: A case study with brazilian judicial appeals. In: BARCELLOS, M. P.; MOREIRA, J. L. R.; FONSECA, C. M.; SOUZA, A. D. de (Ed.). *Proceedings of the XVI Seminar on Ontology Research in Brazil (ONTOBRAS 2023) and VII Doctoral and Masters Consortium on Ontologies (WTDO 2023), Brasília, Brazil, August 28 - September 01, 2023*. CEUR-WS.org, 2023. (CEUR Workshop Proceedings, v. 3564), p. 79–92. Available on: <<https://ceur-ws.org/Vol-3564/paper6.pdf>>. Citado na página 52.
- DAINESE, N.; ILIN, A.; MARTTINEN, P. Can docstring reformulation with an llm improve code generation? In: *Proceedings of the 18th Conference of the European Chapter*



of the Association for Computational Linguistics: Student Research Workshop. 2024. p. 296–312. Citado na página 80.

DALIANIS, H. *Clinical Text Mining - Secondary Use of Electronic Patient Records*. Springer, 2018. ISBN 978-3-319-78502-8. Available on: <<https://doi.org/10.1007/978-3-319-78503-5>>. Citado na página 43.

DEVLIN, J.; CHANG, M.; LEE, K.; TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. In: BURSTEIN, J.; DORAN, C.; SOLORIO, T. (Ed.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019. p. 4171–4186. Available on: <<https://doi.org/10.18653/v1/n19-1423>>. Citado na página 58.

DOMBROWSKI, Q.; GNIADY, T.; KLOSTER, D. Introdução ao jupyter notebook. *The Programming Historian em Português*, ProgHist Ltd, 2023. Citado na página 70.

GALVÃO, E. B. *GQT: "nada é tão bom que não possa ser melhorado"*. 2011. <<https://www.administradores.com.br/artigos/gqt-nada-e-tao-bom-que-nao-possa-ser-melhorado>>. Citado na página 80.

GEBRU, T. et al. Datasheets for datasets. *Commun. ACM*, v. 64, n. 12, p. 86–92, 2021. Available on: <<https://doi.org/10.1145/3458723>>. Citado na página 39.

GRIESSHABER, D.; MAUCHER, J.; VU, N. T. Fine-tuning BERT for low-resource natural language understanding via active learning. In: SCOTT, D.; BEL, N.; ZONG, C. (Ed.). *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. International Committee on Computational Linguistics, 2020. p. 1158–1171. Available on: <<https://doi.org/10.18653/v1/2020.coling-main.100>>. Citado na página 44.

HAENLEIN, M.; KAPLAN, A. A brief history of artificial intelligence: On the past, present, and future of artificial intelligence. *California Management Review*, v. 61, n. 4, p. 5–14, 2019. Available on: <<https://doi.org/10.1177/0008125619864925>>. Citado na página 27.

HAO, X.; ZHANG, G.; MA, S. Deep learning. *Int. J. Semantic Comput.*, v. 10, n. 3, p. 417, 2016. Available on: <<https://doi.org/10.1142/S1793351X16500045>>. Citado na página 35.

JANIESCH, C.; ZSCHECH, P.; HEINRICH, K. Machine learning and deep learning. *Electron. Mark.*, v. 31, n. 3, p. 685–695, 2021. Available on: <<https://doi.org/10.1007/s12525-021-00475-2>>. Citado 3 vezes nas páginas 35, 36 e 37.

JIANG, K.; LU, X. Natural language processing and its applications in machine translation: A diachronic review. In: IEEE. *2020 IEEE 3rd International Conference of Safe Production and Informatization (IICSPI)*. 2020. p. 210–214. Citado 2 vezes nas páginas 51 e 52.

KETKAR, N.; MOOLAYIL, J.; KETKAR, N.; MOOLAYIL, J. Deep learning with python: Learn best practices of deep learning models with pytorch. *Apress LP*, Springer, 2020. Citado na página 68.

KHURANA, D.; KOLI, A.; KHATTER, K.; SINGH, S. Natural language processing: state of the art, current trends and challenges. *Multim. Tools Appl.*, v. 82, n. 3, p. 3713–3744, 2023. Available on: <<https://doi.org/10.1007/s11042-022-13428-4>>. Citado 7 vezes nas páginas 27, 46, 47, 48, 49, 50 e 51.

KIELA, D. et al. Dynabench: Rethinking benchmarking in NLP. In: TOUTANOVA, K. et al. (Ed.). *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*. Association for Computational Linguistics, 2021. p. 4110–4124. Available on: <<https://doi.org/10.18653/v1/2021.naacl-main.324>>. Citado na página 32.

KLUYVER, T. et al. Jupyter notebooks - a publishing format for reproducible computational workflows. In: LOIZIDES, F.; SCHMIDT, B. (Ed.). *Positioning and Power in Academic Publishing: Players, Agents and Agendas, 20th International Conference on Electronic Publishing, Göttingen, Germany, June 7-9, 2016*. IOS Press, 2016. p. 87–90. Available on: <<https://doi.org/10.3233/978-1-61499-649-1-87>>. Citado na página 70.

LAN, Z. et al. ALBERT: A lite BERT for self-supervised learning of language representations. *CoRR*, abs/1909.11942, 2019. Available on: <<http://arxiv.org/abs/1909.11942>>. Citado na página 71.

LECUN, Y.; BENGIO, Y.; HINTON, G. E. Deep learning. *Nat.*, v. 521, n. 7553, p. 436–444, 2015. Available on: <<https://doi.org/10.1038/nature14539>>. Citado na página 37.

LHOEST, Q. et al. Datasets: A community library for natural language processing. In: ADEL, H.; SHI, S. (Ed.). *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2021, Online and Punta Cana, Dominican Republic, 7-11 November, 2021*. Association for Computational Linguistics, 2021. p. 175–184. Available on: <<https://doi.org/10.18653/v1/2021.emnlp-demo.21>>. Citado na página 67.

LIDDY, E. D. Natural language processing. 2001. Citado 2 vezes nas páginas 47 e 49.

MAAS, A. L. et al. Learning word vectors for sentiment analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, 2011. p. 142–150. Available on: <<http://www.aclweb.org/anthology/P11-1015>>. Citado na página 74.

MCKINNEY, W. et al. pandas: a foundational python library for data analysis and statistics. *Python for high performance and scientific computing*, Seattle, v. 14, n. 9, p. 1–9, 2011. Citado na página 69.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, Elsevier, v. 5, n. 4, p. 1093–1113, 2014. Citado na página 56.

MIN, B. et al. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, v. 56, n. 2, p. 30:1–30:40, 2024. Available on: <<https://doi.org/10.1145/3605943>>. Citado na página 44.

- NEELAKANTAN, A.; SHANKAR, J.; PASSOS, A.; MCCALLUM, A. Efficient non-parametric estimation of multiple embeddings per word in vector space. *CoRR*, abs/1504.06654, 2015. Available on: <<http://arxiv.org/abs/1504.06654>>. Citado na página 39.
- OLIVEIRA, E. de; SILVA, W.; PIROVANI, J. P. C.; BOURGUET, J. A prototypical semantic annotator for A tribuna newspaper. In: LEMOS, D. L. da S.; SALES, T. P.; CAMPOS, M. L. M.; FIORINI, S. R. (Ed.). *Proceedings of the XIII Seminar on Ontology Research in Brazil and IV Doctoral and Masters Consortium on Ontologies (ONTOBRAS 2020)*, Vitória, Brazil, November 23-26, 2020. CEUR-WS.org, 2020. (CEUR Workshop Proceedings, v. 2728), p. 22–34. Available on: <<https://ceur-ws.org/Vol-2728/paper2.pdf>>. Citado na página 54.
- PANG, B.; LEE, L. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*, 2005. Citado na página 74.
- PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011. Citado na página 69.
- Python Refactoring: Techniques, Tools, and Best Practices. 2024. <<https://acesse.dev/g6spK>>. Citado na página 81.
- QIN, C. et al. Is chatgpt a general-purpose natural language processing task solver? In: BOUAMOR, H.; PINO, J.; BALI, K. (Ed.). *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics, 2023. p. 1339–1384. Available on: <<https://aclanthology.org/2023.emnlp-main.85>>. Citado na página 28.
- RADFORD, A.; NARASIMHAN, K.; SALIMANS, T.; SUTSKEVER, I. et al. Improving language understanding by generative pre-training. OpenAI, 2018. Citado 2 vezes nas páginas 27 e 57.
- RADFORD, A. et al. Language models are unsupervised multitask learners. 2019. Citado na página 72.
- RAHUL; ADHIKARI, S.; MONIKA. Nlp based machine learning approaches for text summarization. In: *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*. 2020. p. 535–538. Citado na página 54.
- RAJPURKAR, P.; ZHANG, J.; LOPYREV, K.; LIANG, P. SQuAD: 100,000+ questions for machine comprehension of text. In: SU, J.; DUH, K.; CARRERAS, X. (Ed.). *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Austin, Texas: Association for Computational Linguistics, 2016. p. 2383–2392. Available on: <<https://aclanthology.org/D16-1264>>. Citado na página 74.
- RAUPP, F. M.; BEUREN, I. M. Metodologia da pesquisa aplicável às ciências. *Como elaborar trabalhos monográficos em contabilidade: teoria e prática*. São Paulo: Atlas, p. 76–97, 2006. Citado na página 29.
- Researchgate. 2024. <<https://www.researchgate.net>>. Citado na página 37.

- SANG, E. F. T. K.; MEULDER, F. D. "introduction to the CoNLL-2003 shared task: Language-independent named entity recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003. p. "142–147". Available on: <<https://www.aclweb.org/anthology/W03-0419>>. Citado na página 73.
- SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019. Citado na página 72.
- SELLTIZ, C.; JAHODA, M.; DEUTSCH, M.; COOK, S. W.; LEITE, D. M. Métodos de pesquisa nas relações sociais. In: *Métodos de pesquisa nas relações sociais*. 1975. p. 690–690. Citado na página 29.
- SHANAHAN, M. Talking about large language models. *Commun. ACM*, v. 67, n. 2, p. 68–79, 2024. Available on: <<https://doi.org/10.1145/3624724>>. Citado na página 59.
- SOCHER, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, 2013. p. 1631–1642. Available on: <<https://www.aclweb.org/anthology/D13-1170>>. Citado na página 74.
- SRINATH, K. Python—the fastest growing programming language. *International Research Journal of Engineering and Technology*, v. 4, n. 12, p. 354–357, 2017. Citado na página 65.
- SU, S.; HUANG, C.; CHEN, Y. Dual supervised learning for natural language understanding and generation. In: KORHONEN, A.; TRAUM, D. R.; MÀRQUEZ, L. (Ed.). *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*. Association for Computational Linguistics, 2019. p. 5472–5477. Available on: <<https://doi.org/10.18653/v1/p19-1545>>. Citado na página 45.
- SUN, P.; YANG, X.; ZHAO, X.; WANG, Z. An overview of named entity recognition. In: DONG, M. et al. (Ed.). *2018 International Conference on Asian Language Processing, IALP 2018, Bandung, Indonesia, November 15-17, 2018*. IEEE, 2018. p. 273–278. Available on: <<https://doi.org/10.1109/IALP.2018.8629225>>. Citado na página 54.
- TAULLI, T.; ONI, M. *Artificial intelligence basics*. : Springer, 2019. Citado 2 vezes nas páginas 35 e 36.
- TOOSI, A.; BOTTINO, A.; SABOURY, B.; SIEGEL, E. L.; RAHMIM, A. A brief history of AI: how to prevent another winter (a critical review). *CoRR*, abs/2109.01517, 2021. Available on: <<https://arxiv.org/abs/2109.01517>>. Citado na página 27.
- TOUVRON, H. et al. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. Available on: <<https://doi.org/10.48550/arXiv.2307.09288>>. Citado na página 59.
- Towardsdatascience. 2024. <<https://towardsdatascience.com>>. Citado na página 38.
- TSYTSARAU, M.; PALPANAS, T. Survey on mining subjective data on the web. *Data Min. Knowl. Discov.*, v. 24, n. 3, p. 478–514, 2012. Available on: <<https://doi.org/10.1007/s10618-011-0238-6>>. Citado na página 55.

TURING, A. M. Computing machinery and intelligence. *Mind*, LIX, n. 236, p. 433–460, 1950. Available on: <<https://doi.org/10.1093/mind/LIX.236.433>>. Citado na página 27.

WANG, A. et al. Superglue: A stickier benchmark for general-purpose language understanding systems. In: WALLACH, H. M. et al. (Ed.). *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 2019. p. 3261–3275. Available on: <<https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html>>. Citado na página 31.

WANG, A. et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. Available on: <<https://openreview.net/forum?id=rJ4km2R5t7>>. Citado na página 31.

WERRA, L. von et al. Evaluate & evaluation on the hub: Better best practices for data and model measurements. In: CHE, W.; SHUTOVA, E. (Ed.). *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022 - System Demonstrations, Abu Dhabi, UAE, December 7-11, 2022*. Association for Computational Linguistics, 2022. p. 128–136. Available on: <<https://doi.org/10.18653/v1/2022.emnlp-demos.13>>. Citado na página 67.

WOLF, T. et al. Transformers: State-of-the-art natural language processing. In: LIU, Q.; SCHLANGEN, D. (Ed.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020. p. 38–45. Available on: <<https://doi.org/10.18653/v1/2020.emnlp-demos.6>>. Citado 2 vezes nas páginas 39 e 66.

WU, C.; HOI, S. C. H.; SOCHER, R.; XIONG, C. TOD-BERT: pre-trained natural language understanding for task-oriented dialogue. In: WEBBER, B.; COHN, T.; HE, Y.; LIU, Y. (Ed.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*. Association for Computational Linguistics, 2020. p. 917–929. Available on: <<https://doi.org/10.18653/v1/2020.emnlp-main.66>>. Citado na página 44.

WU, L.; DODOO, N. A.; WEN, T. J.; KE, L. Understanding twitter conversations about artificial intelligence in advertising based on natural language processing. *International Journal of Advertising*, Taylor & Francis, v. 41, n. 4, p. 685–702, 2022. Citado na página 36.

XU, L. et al. CLUE: A chinese language understanding evaluation benchmark. In: SCOTT, D.; BEL, N.; ZONG, C. (Ed.). *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*. International Committee on Computational Linguistics, 2020. p. 4762–4772. Available on: <<https://doi.org/10.18653/v1/2020.coling-main.419>>. Citado na página 32.

XU, L.; LIU, J.; PAN, X.; LU, X.; HOU, X. Dataclue: A benchmark suite for data-centric NLP. *CoRR*, abs/2111.08647, 2021. Available on: <<https://arxiv.org/abs/2111.08647>>. Citado na página 32.



- YACOUBY, R.; AXMAN, D. Probabilistic extension of precision, recall, and F1 score for more thorough evaluation of classification models. In: EGER, S.; GAO, Y.; PEYRARD, M.; ZHAO, W.; HOVY, E. H. (Ed.). *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems, Eval4NLP 2020, Online, November 20, 2020*. Association for Computational Linguistics, 2020. p. 79–91. Available on: <<https://doi.org/10.18653/v1/2020.eval4nlp-1.9>>. Citado na página 43.
- YANG, L.; SHAMI, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, v. 415, p. 295–316, 2020. Available on: <<https://doi.org/10.1016/j.neucom.2020.07.061>>. Citado 2 vezes nas páginas 41 e 43.
- YU, X.; FENG, W.; WANG, H.; CHU, Q.; CHEN, Q. An attention mechanism and multi-granularity-based bi-lstm model for chinese q&a system. *Soft Comput.*, v. 24, n. 8, p. 5831–5845, 2020. Available on: <<https://doi.org/10.1007/s00500-019-04367-8>>. Citado na página 52.
- YUAN, C.; XUE, M.; ZHANG, L.; WU, H. Robustness analysis on natural language processing based AI q&a robots. In: ZHAI, X. B.; CHEN, B.; ZHU, K. (Ed.). *Machine Learning and Intelligent Communications - 4th International Conference, MLICOM 2019, Nanjing, China, August 24-25, 2019, Proceedings*. Springer, 2019. (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, v. 294), p. 695–711. Available on: <[https://doi.org/10.1007/978-3-030-32388-2\\_57](https://doi.org/10.1007/978-3-030-32388-2_57)>. Citado na página 53.
- ZHANG, Y.; LING, C. A strategy to apply machine learning to small datasets in materials science. *Npj Computational Materials*, Nature Publishing Group UK London, v. 4, n. 1, p. 25, 2018. Citado na página 37.
- ZHONG, Q. et al. Toward efficient language model pretraining and downstream adaptation via self-evolution: A case study on superglue. *CoRR*, abs/2212.01853, 2022. Available on: <<https://doi.org/10.48550/arXiv.2212.01853>>. Citado na página 33.
- ZHOU, J.; GANDOMI, A. H.; CHEN, F.; HOLZINGER, A. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, MDPI, v. 10, n. 5, p. 593, 2021. Citado 2 vezes nas páginas 28 e 43.
- ZHOU, Z.-H. *Machine learning*. : Springer nature, 2021. Citado na página 35.