EP 1 - MAC5725 - Linguística Computacional

Calebe Rezende[1]

[1] Instituto de Matemática -- Universidade de São Paulo (USP)

[2]Departamento de Sistemas e Computação

calebe.rezende@usp.br

Abstract. *This article was written to measure outputs from bidirectional recurrent neural networks using LSTM and GRU architectures, and to evaluate the issues of underfitting and overfitting. In one of the key graphs, we will focus on comparing accuracy and epochs.*

Resumo. *Este artigo tem como objetivo medir as saídas geradas por redes neurais recorrentes bidirecionais utilizando as arquiteturas LSTM e GRU, e avaliar os problemas de subajuste (underfitting) e sobreajuste (overfitting). Nossa análise se concentrará na comparação da acurácia em relação às épocas..*

1. Introdução

Este artigo, relacionado ao trabalho de linguística computacional MAC5725, discute a implementação e análise de redes neurais recorrentes bidirecionais (LSTM e GRU), com foco na análise de sentimento. Ele também cobre dois problemas comuns de aprendizado de máquina: underfitting e overfitting, e estratégias para reduzir o overfitting. O ajuste inadequado geralmente ocorre quando a simplicidade impede que o modelo capture padrões nos dados. As soluções incluem exigir mais dados ou ajustar um modelo para lidar melhor com a complexidade. Por outro lado, o overfitting ocorre quando o modelo ajusta demais os dados de treinamento, incluindo os valores errados.. Isso pode ser evitado usando técnicas de regulação como o dropout, que desativa aleatoriamente os neurônios durante o treinamento. O artigo também destaca a importância de dividir os dados em conjuntos de treinamento, validação e teste. Os conjuntos de validação são importantes para monitorar o desempenho do modelo e evitar o overfitting, o conjunto de testes é usado para avaliar o modelo final. Além disso, o processo de treinamento, incluindo limitações de tamanho de vocabulário e agrupamento de dados, é abordado. Foram testadas duas arquiteturas de redes neurais recorrentes (unidirecional e bidirecional). O artigo conclui com uma análise dos resultados, incluindo gráficos de validação e identificação do modelo com maior precisão. Por fim, realizamos uma verificação final utilizando a melhor configuração do modelo, os resultados e gráficos são detalhados durante o artigo, incluindo as configurações e hiperparâmetros utilizados

2. Hiperparâmetros

Este artigo tem como objetivo realizar uma análise meticulosa das saídas obtidas através da utilização de redes neurais recorrentes bidirecionais, empregando as arquiteturas LSTM e GRU. O foco principal será a avaliação dos desafios enfrentados

pelo subajuste (underfitting) e pelo sobreajuste (overfitting). Este artigo tem como objetivo realizar uma análise meticulosa das saídas obtidas através da utilização de redes neurais recorrentes bidirecionais, empregando as arquiteturas LSTM e GRU. O foco principal será a avaliação dos desafios enfrentados pelo subajuste (underfitting) e pelo sobreajuste (overfitting). Um dos aspectos fundamentais que será abordado é a comparação da acurácia em relação às épocas, através de um gráfico de extrema relevância. A acurácia é uma métrica crucial para avaliar o desempenho dos modelos, uma vez que representa a taxa de sucesso em relação ao total de amostras analisadas. Por meio desse gráfico, será possível observar como o desempenho dos modelos evolui ao longo do tempo e a capacidade que possuem de se adaptar aos dados de treinamento.Através dessa análise da acurácia em função das épocas, será possível identificar a ocorrência de subajuste ou sobreajuste.

O subajuste ocorre quando o modelo não consegue aprender de forma adequada a partir dos dados de treinamento, resultando em uma baixa acurácia. Já no caso do sobreajuste, ocorre o memorização excessiva dos dados de treinamento, levando a uma acurácia alta nesses dados, porém uma baixa acurácia quando novos dados são apresentados.Através da comparação da acurácia ao longo das épocas, será possível identificar esses fenômenos e tomar medidas corretivas, como ajustar a complexidade dos modelos, aplicar técnicas de regularização (como o Dropout) ou expandir a quantidade de dados de treinamento disponíveis.Dessa forma, a análise cuidadosa da acurácia em relação às épocas é de suma importância para entender a capacidade de generalização dos modelos e tomar decisões que melhorem seu desempenho, além de evitar problemas como o subajuste e o sobreajuste.

3. Preparação do Conjunto de Dados, Ajuste de Hiperparâmetros e Análise do Treinamento para um Modelo de Processamento de Linguagem Natural (NLP

Foi realizado o uso do conjunto de dados disponível no seguinte link: https://raw.githubusercontent.com/americanas-tech/b2w-reviews01/main/B2W-Reviews 01.csv. Para esse trabalho em específico, as colunas relevantes foram 'review_text' e 'overall_rating'. Os dados foram divididos idealmente em três partes: Treinamento, Validação e Teste, com proporções de 65, 10 e 25, respectivamente. A divisão foi realizada utilizando os parâmetros train_size = 0.61, testsize = 0.24 e randomstate = 42. Portanto, os tamanhos resultantes foram: Tamanho do conjunto de treinamento: 80747, Tamanho do conjunto de validação: 19856 e Tamanho do conjunto de teste: 31770.

Neste trabalho, foram realizadas as importações presentes na figura 1.

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import seaborn as sns
from keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout, Bidirectional
from keras.preprocessing.text import Tokenizer
import pandas as pd
import tensorflow as tf
from keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
from keras.preprocessing.text import Tokenizer
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix, classification_report
```

Figure 1. Importações feitas no Google Colab

Essas importações foram necessárias para a execução das etapas subsequentes. Um dos primeiros hiperparâmetros considerados foi o tamanho máximo de sequência, que foi definido como 795. O segundo hiperparâmetro foi o tamanho das palavras conhecidas: 50000. No entanto, visto que havia um documento orientador para este trabalho, foram utilizadas apenas 20000 palavras do vocabulário definido, considerando as demais como desconhecidas. Esse gráfico foi criado para entender aproximadamente o número de palavras para cada classe de palavras, o tamanho médio foi de 32.

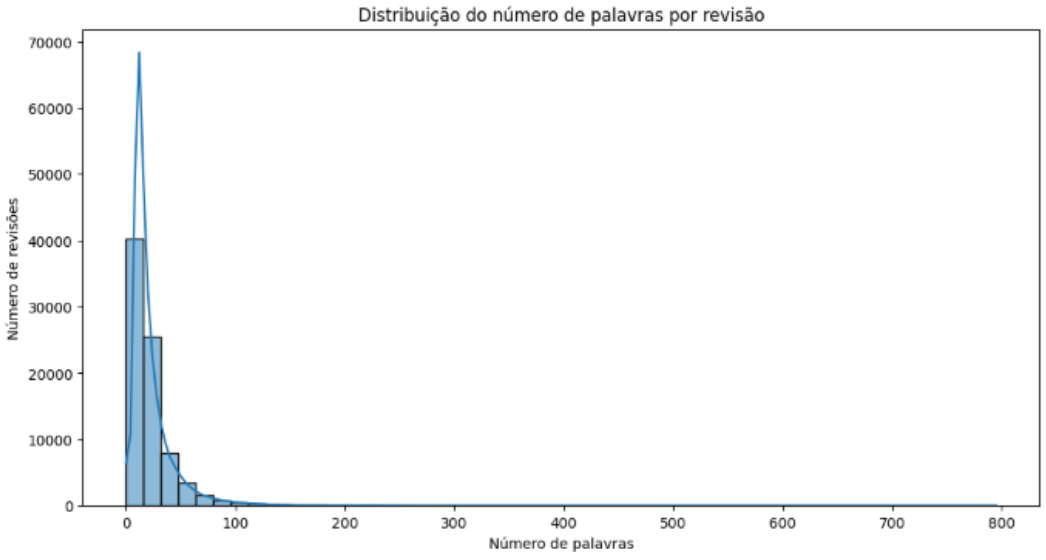


Figure 2: Distribuição do número de palavras

4. Validação e teste

Na parte de treinamento e teste, configuramos o ambiente de plotagem, definindo uma função para visualizar o histórico de treinamento de modelos de aprendizado profundo e, em seguida, realizamos um loop para avaliar vários modelos com diferentes configurações.

Definimos a função `plot_history' para representar o histórico de treinamento do modelo. Esse gráfico é dividido em duas áreas: uma para mostrar a acurácia do modelo e a outra para exibir a perda ao longo das épocas. Depois de configurar a função, entramos em um loop que percorre uma série de modelos de aprendizado profundo.

Cada modelo é avaliado com base em diferentes configurações, incluindo a escolha de ser bidirecional ou unidirecional e a taxa de dropout.

Nessa etapa foram gerados 6 gráficos, usando os valores de 0, 25 e 50. Em modelos unidirecionais e bidirecionais. A seguir os gráficos:

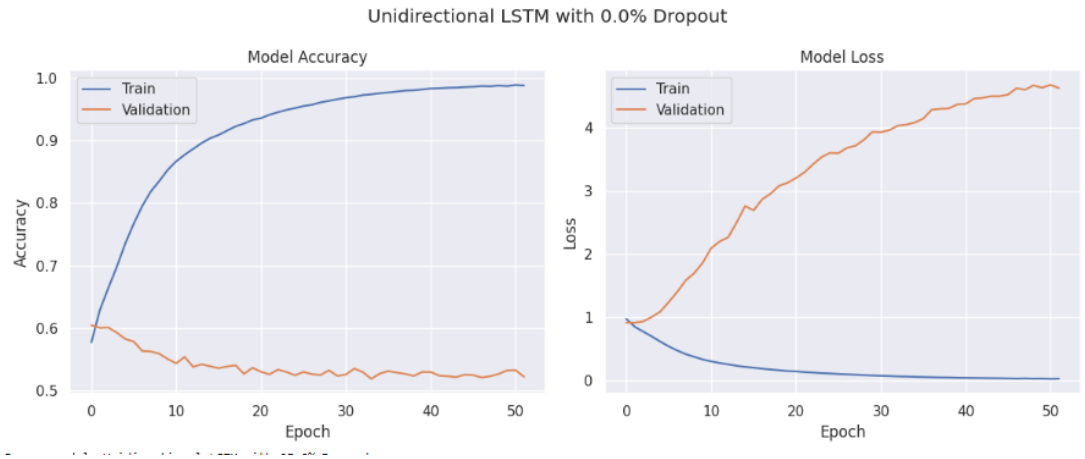


Figure 3: Dropout 0% Unidirecional

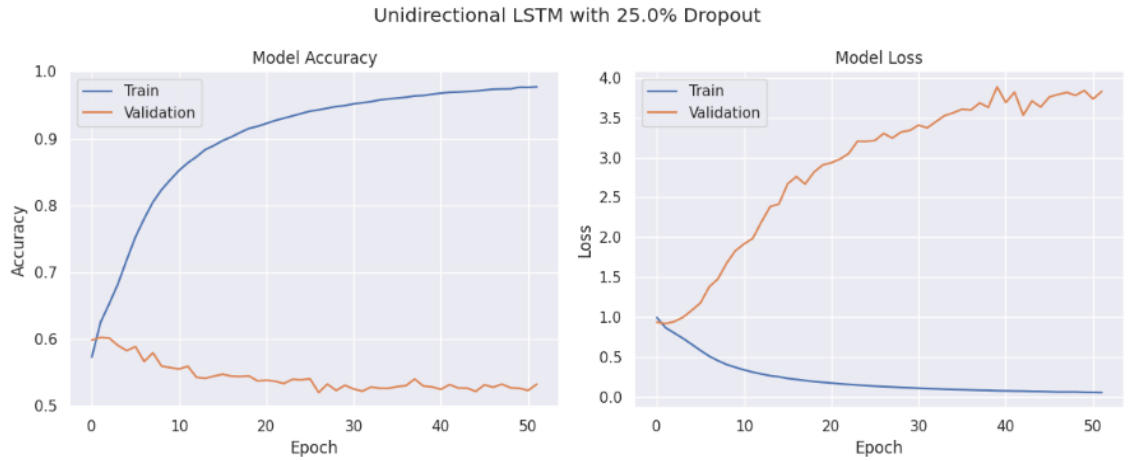


Figura 4: Dropout 25%Unidirecional

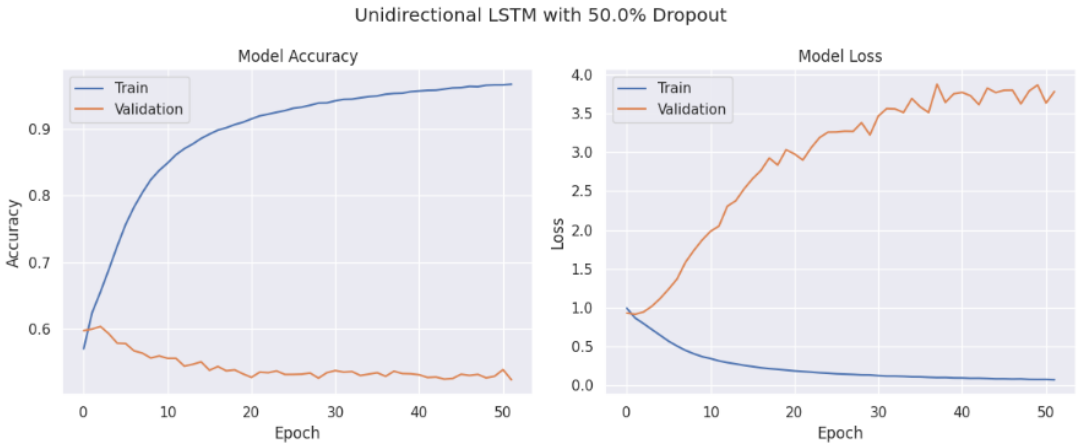


Figura 5: Dropout 50% Unidirecional

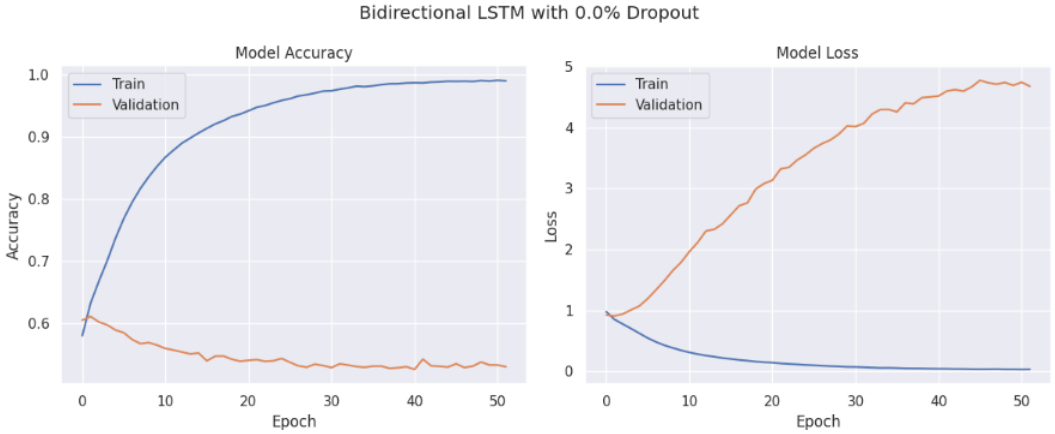


Figura 6: Dropout 0% Bidirecional

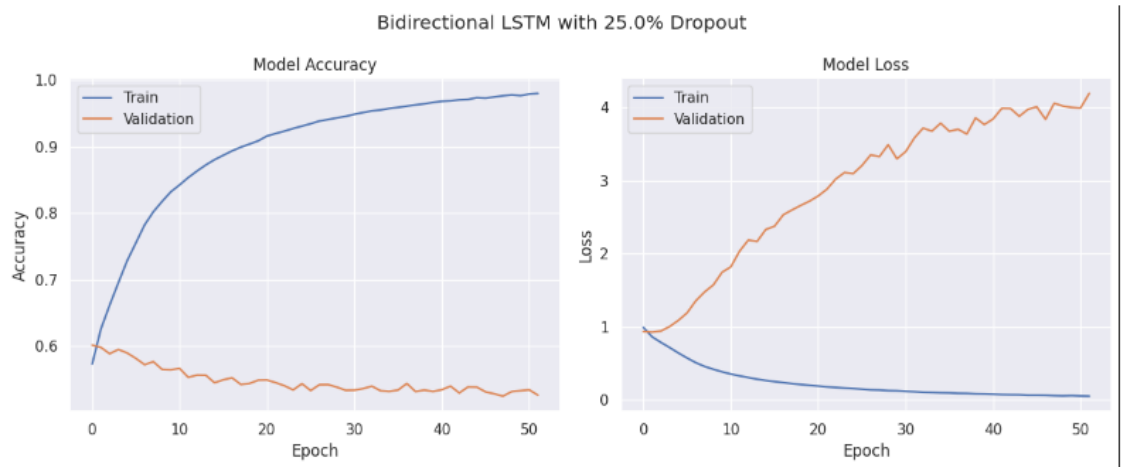


Figura 7: Dropout 25% Bidirecional

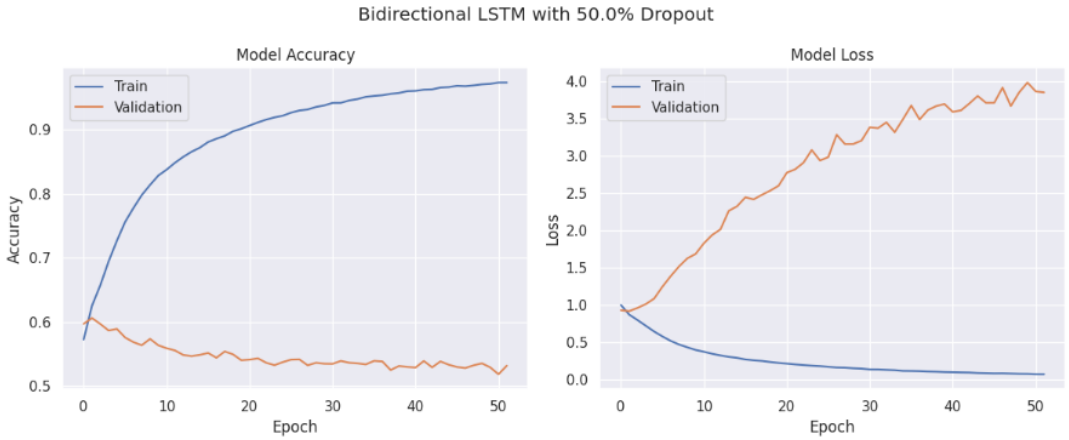


Figura 7: Dropout 50% Bidirecional

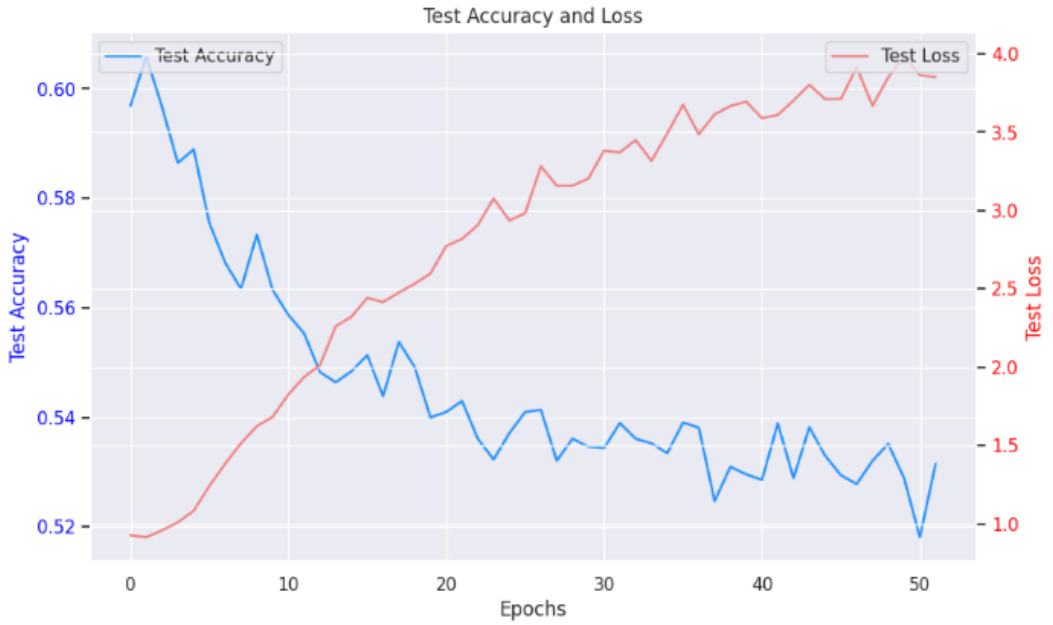
Os seguintes dados foram gerados:

- Para o modelo "Unidirectional LSTM with 25.0% Dropout", a melhor perda de validação na melhor época é 0.9176419973373413.
- Para o modelo Unidirectional LSTM with 0.0% Dropout: a melhor época: 2
- Perda de validação na melhor época: 0.9184806942939758
- Para o modelo "Unidirectional LSTM with 50.0% Dropout", a melhor perda de validação na melhor época é 0.9159805774688721.
- Para o modelo "Bidirectional LSTM with 0.0% Dropout", a melhor perda de validação na melhor época é 0.9055081605911255.
- Para o modelo "Bidirectional LSTM with 25.0% Dropout", a melhor perda de validação na melhor época é 0.9267004728317261.
- Para o modelo "Bidirectional LSTM with 50.0% Dropout", a melhor perda de validação na melhor época é 0.9143401384353638.

As discrepâncias entre os resultados obtidos durante as etapas de validação e teste podem estar ligadas a diversos fatores, incluindo a estrutura do modelo, a taxa de exclusão (dropout), e as características dos conjuntos de dados utilizados na validação e no teste. As redes Bidirecionais, ao processarem a sequência de entrada em ambas as direções, têm a capacidade de capturar relações temporais de longo alcance, o que é particularmente crucial em tarefas que envolvem análise de sentimentos. Essa análise enfatiza a importância de avaliar os modelos em diferentes conjuntos de dados, ou seja, durante o treinamento, a validação e o teste. Isso permite obter uma compreensão mais abrangente do desempenho do modelo e sua capacidade de generalização

4. Acurácia versus loss

A próxima comparação que faremos nesse documento, é a comparação ao mesmo tempo, entre a acurácia e loss, ao encontrarmos o ponto em que ambas se encontram, daremos como o melhor resultado.



.

Figure 8: Acurácia versus loss

Nessa etapa foi criado um gráfico que desempenha um papel fundamental na análise do desempenho de um modelo de aprendizado de máquina avaliando a capacidade do modelo de generalização, identificando potenciais problemas de underfitting ou overfitting. Isso será usado para aprimorar a qualidade do modelo.

A primeira parte do gráfico enfoca a acurácia do modelo, que é uma métrica importante para avaliar o quão corretamente o modelo classifica os dados. O eixo x representa o número de épocas de treinamento, enquanto o eixo y mostra a acurácia no conjunto de teste (validação).

Nessa etapa podemos dizer que a acurácia definida como o melhor que conseguimos, foi de 0.55, logo em seguida começa o overfitting enquanto o loss estava em 2.0;

O gráfico exibe a evolução da acurácia à medida que o modelo é treinado. Idealmente, esperamos ver um aumento na acurácia durante as primeiras épocas, indicando que o modelo está aprendendo com os dados. No entanto, se a acurácia estagnar ou diminuir nas épocas posteriores, isso pode ser um sinal de overfitting, sugerindo que o modelo está se ajustando demais aos dados de treinamento e não generalizando bem para novos dados. Perda do modelo no conjunto de teste (Validação): A segunda linha do gráfico se concentra na perda do modelo, que é uma medida do erro

cometido pelo modelo em suas previsões. O eixo y representa a perda no conjunto de teste (validação), enquanto o eixo x representa o número de épocas.

Similar à acurácia, observar a perda é fundamental, uma queda constante na perda é desejável, indicando que o modelo está melhorando suas previsões, entretanto, um aumento na perda após um ponto de mínimo pode ser indicativo de overfitting. A utilização de dois eixos verticais, um para acurácia e outro para perda é útil porque o aumento da acurácia não garante necessariamente um modelo de alta qualidade; é importante considerar a perda como um indicador de erro. Em resumo, esse gráfico desempenha uma avaliação e ajuste de modelos de aprendizado de máquina, fornecendo informações detalhadas sobre a evolução do modelo durante o treinamento e sua capacidade de generalização.

5. Matriz de confusão

A matriz de confusão é uma ferramenta que permite representar de forma visual os erros de classificação de um modelo durante os testes. Essa abordagem possibilita identificar se o modelo tem uma inclinação em favor de uma classe específica e também ajuda a identificar quais classes estão enfrentando maior dificuldade de classificação**.**

Com essa matriz de confusão, é gerado também o gráfico de interpretação de seus resultados. Savietto em Google Developers para Machine Learning**,** nos apresenta como o recall serve para entender quantos positivos foram compreendidos corretamente .Uma vez que o cálculo é feito da seguinte forma:

$$recall = \frac{TP}{TP + FN}$$

	precision	recall	f1-score	support
0	0.75	0.89	0.81	6424
1	0.41	0.02	0.04	1997
2	0.42	0.41	0.42	3965
3	0.44	0.48	0.46	8046
4	0.67	0.68	0.68	11843
accuracy			0.60	32275
macro avg	0.54	0.50	0.48	32275
weighted avg	0.58	0.60	0.58	32275

Figure 9: Gráfico de dados da matriz de confusão

Apesar desta seção trabalharmos sobre recall, e a ideia é procurarmos o menor valor, e perceptivelmente a época 4 possui um compilado desses maiores

valores. Para não tratarmos apenas do gráfico gerado, anexo também a matriz de confusão gerada. Apresentando o resultado de verdadeiros positivos, falsos positivos, falso negativo e verdadeiro negativo

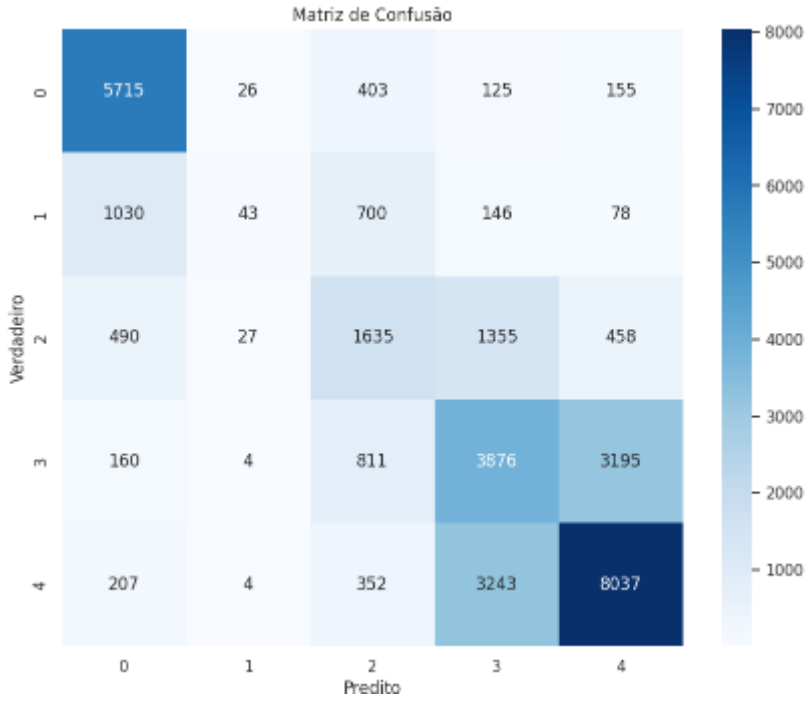


Figure 10: Matriz de confusão

Por fim, algumas melhorias não foram inseridas aqui, pela dificuldade que houve em rodar no Google Colab, todos os movimentos aqui, foram pagos, pois na forma gratuita que entramos na fila para esperar ser rodado, estouraram os parâmetros e perdiam o processo todo de treinamento e teste. O que serviu de aprendizado também para o tempo gasto em um trabalho como esse, ao reler o que foi aprendido buscarmos referências, foi muito mais que a metade do prazo total da atividade.

7. Conclusões

O modelo aqui trabalhado, não teve uma boa acurácia, uma vez que vimos na figura 8, a comparação entre a acurácia e loss ficou em 0,55 e 2.0 respectivamente,, podemos ter várias explicações para isso o banco de dados usado como base para o treinamento do modelo pode conter dados conflitantes. Isso inclui situações em que as notas de avaliação estão em desacordo com os comentários, tornando o treinamento e teste de uma rede neural uma tarefa desafiadora. Além disso, a interpretação individual desses tipos de comentários pode ser complicada, uma vez que palavras e frases podem ter significados diferentes para diferentes pessoas.

A inconsistência nas etiquetas de classificação também é um fator relevante. Divergências nas definições das categorias ou rótulos de classificação podem resultar em erros de classificação. A quantidade de dados de treinamento disponíveis é crucial para o desempenho de modelos de aprendizado de máquina, como redes neurais. Modelos se beneficiam de conjuntos de dados extensos para aprender padrões e generalizar com precisão. Quando o conjunto de dados é limitado em tamanho, o modelo pode lutar para interpretar variações nos dados.

A qualidade dos dados de treinamento é igualmente importante. Dados ruidosos ou mal rotulados podem prejudicar o desempenho do modelo, levando-o a aprender informações incorretas ou irrelevantes. A escolha da arquitetura da rede neural e dos hiperparâmetros de treinamento também desempenha um papel significativo no desempenho do modelo. Arquiteturas inadequadas ou configurações de hiperparâmetros mal ajustados podem resultar em resultados insatisfatórios.

A função de perda categorical_crossentropy é apropriada quando se está trabalhando com saídas categóricas e desejamos verificar a perda entre a distribuição de probabilidade prevista pelo modelo. Ela é usada para treinar o modelo a minimizar a diferença entre as probabilidades previstas. Essa escolha de função de perda faz sentido, dada a natureza da tarefa de classificação de avaliações em cinco categorias diferentes.

A avaliação de modelos de análise de sentimentos é um processo iterativo, que deve levar em consideração métricas além da acurácia, como a perda, a precisão, e revocação e a pontuação F1, a fim de proporcionar uma visão completa do desempenho do modelo. Por fim, é importante reconhecer que a tarefa de análise de sentimentos em comentários é complexa e multidimensional. Pode haver casos em que a classificação ideal é subjetiva. Portanto, é fundamental manter expectativas realistas em relação ao desempenho do modelo e estar ciente das limitações inerentes a esse tipo de tarefa. Há formas de melhorar esse resultado, criando embeddings diferentes por exemplo o método

GloVe, que são representações vetoriais de alta dimensão, essas representações são amplamente utilizadas em processamento de linguagem natural (PLN) e aprendizado de máquina para codificar informações significativas sobre palavras, frases, documentos ou entidades.

Em um espaço de embeddings, palavras ou entidades com significados semelhantes são mapeadas para vetores próximos no espaço, o que significa que as relações semânticas entre elas são preservadas.

8. References

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. Disponível em: https://www.researchgate.net/publication/234131319_Efficient_Estimation_of_Word_Representations_in_Vector_Space Acesso em 17 de out. 2023.

Normas ABNT 2023. Disponível em: https://www.normasabnt.org/normas-abnt-2023/ Acesso em: 15 de out. 2023.

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543). Disponível em: https://aclanthology.org/D14-1162. Acesso em 21 de out. 2023

Savietto, João. Machine Learning: Métricas, Validação Cruzada, Bias e Variância 2021, Disponível em:https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=pt-br,Acesso em: 19 de out. 2023.