

CENTRO UNIVERSITÁRIO CARIOCA
UNICARIOCA

LUCAS NUNES SOUZA

ANÁLISE DE SENTIMENTOS APLICADA EM TEXTOS DE USUÁRIOS DO
TWITTER RELACIONADO A COVID-19

Rio de Janeiro
2020

ANÁLISE DE SENTIMENTOS APLICADA EM TEXTOS DE USUÁRIOS DO TWITTER RELACIONADO A COVID-19

Trabalho de Conclusão de Curso,
apresentado como exigência para
obtenção de Título de Bacharel em
Ciência da Computação, pelo Centro
Universitário Carioca – UniCarioca, sob
Orientação do Professor Manuel Martins
Filho.

Rio de Janeiro
2020

S 719

Souza, Lucas Nunes.

Análise de sentimentos aplicada em textos de usuários do twitter relacionado a Covid-19 / Lucas Nunes Souza.
Rio de Janeiro, 2020.
91 f.

Orientador: Manuel Martins Filho.

Trabalho de Conclusão de Curso (Graduação em
Ciência da Computação) – Centro Universitário Unicarioca.
Rio de Janeiro, 2020.

1. Covid-19. 2. Análise de sentimentos. 3. Aprendizado de máquina. I. Martins Filho, Manuel. II. Título.

CDD 005

ANÁLISE SENTIMENTOS APLICADA EM TEXTOS DE USUÁRIOS DO TWITTER
RELACIONADO A COVID-19

LUCAS NUNES SOUZA

Este trabalho de Conclusão de Curso foi submetido ao processo de avaliação pela Banca Examinadora para a obtenção do Título de:

Bacharel em Ciência da Computação

E aprovada na sua versão final em _____, atendendo às normas da legislação vigente da Universidade do Contestado e Coordenação do Curso de Ciência da Computação.

André Luiz Avelino Sobral - MSc.
Coordenador do Curso de Ciência da Computação

Rogério Malheiros dos Santos - DSc.

Manuel Martins Filho - DSc.
Orientador

AGRADECIMENTOS

Agradeço a Deus por dar saúde e força para que eu pudesse seguir com este projeto de pesquisa, me mantendo no caminho certo até o fim. Sou grato à minha mãe pelo apoio que sempre me deu durante toda a minha vida.

Quero agradecer especialmente ao meu orientador, Manuel Martins Filho, pelo incentivo e dedicação do seu tempo limitado ao meu projeto de pesquisa.

Também gostaria agradecer a Unicarioca e a todos os professores do meu curso por oferecerem uma educação de alta qualidade.

“Nós podemos ver apenas uma curta distância à frente, mas podemos perceber que há muito a ser feito.”

(Alan Turing, tradução nossa)

RESUMO

O crescente volume de conteúdo multimídia desempenhou um papel importante na existência de grandes quantidades de dados sobre opiniões que podem ser encontrados em toda Internet como Redes Sociais, *Market Places*, Aplicativos e Blogs. A existência desses dados cria uma demanda de empresas, organizações e governos que é conhecer a opinião das pessoas para aprimorar produtos, serviços e políticas governamentais. A Análise de Sentimentos tem sido utilizada para atender a essas demandas. Atualmente avanços em aprendizado de máquina e técnicas de aprendizado profundo permitem que novas abordagens possam ser utilizadas para modelar e extrair informações. O objetivo deste trabalho é coletar e pré-processar dados em formato de texto relacionados a COVID-19 da rede social Twitter, mostrar o uso da biblioteca *TextBlob* e desenvolver um modelo de *Deep Learning* com a biblioteca *Keras* que pode classificar textos em Português para 7 emoções com base em dados rotulados na API do *Watson Tone Analyzer* IBM.

Palavras-chave: COVID-19, Análise de Sentimentos, Twitter, Aprendizado de máquina, Aprendizagem profunda, *Watson Tone Analyzer*, *TextBlob*, *Keras*

ABSTRACT

The growing volume of multimedia content played an important role in the existence of large amounts of data on opinions that can be found throughout the Internet, such as Social Networks, Market Places, Applications and Blogs. The existence of this data creates a new demand from companies, organizations and governments that is to know people's opinions to improve products, services and government policies. Sentiment Analysis has been used to meet these demands. Advances in machine learning and deep learning techniques today allow new approaches to be used to model and extract information. The objective of this work is to collect and pre-process data in text format related to COVID-19 from the social network Twitter, show the use of the TextBlob library and develop a model of Deep Learning with the Keras library that it can classify texts in Portuguese to 7 emotions based on data labeled in the IBM Watson Tone Analyzer API.

Keywords: COVID-19, Sentiment Analysis, Twitter, Machine learning, Deep learning, Watson Tone Analyzer, TextBlob, Keras

LISTA DE FIGURAS

Figura 2.1: Exemplo da polaridade positiva, negativa e neutro.....	18
Figura 2.2: Diagrama que representa o processo de treinamento e avaliação	21
Figura 2.3: Exemplo de tarefa de classificação binária.....	22
Figura 2.4: Exemplo de tarefa de classificação multiclasse	23
Figura 2.5: Aprendizado das representações em várias camadas para um modelo de classificação de dígitos	25
Figura 2.6: Exemplo do relacionamento das redes, camadas, função de perda e otimizador	26
Figura 2.7: Funções Relu e Sigmóide	27
Figura 2.8: Função de ativação da última camada e função de perda para o modelo	27
Figura 2.9: Exemplo de Vetorização de texto.....	28
Figura 4.1: Visão Geral dos dados brutos	33
Figura 4.2: Tweet_id verificando se há dados duplicados	33
Figura 4.3: Verificando se houve texto duplicado	34
Figura 4.4: Localização de usuários	34
Figura 4.5: Regiões variadas e mal formatadas.....	35
Figura 4.6: Hashtags do conjunto de dados.....	35
Figura 4.7: Word Cloud das <i>hashtags</i> no conjunto de dados.....	36
Figura 4.8: Exemplos dos textos (tweet_text, tweet_textEN, clean_textEN)	38
Figura 4.9: As 40 regiões com mais ocorrências não formatadas.....	40
Figura 4.10: As 20 regiões com mais ocorrências formatadas.....	41
Figura 4.11: Exemplo dos rótulos em textos	42
Figura 4.12: Exemplos de tweets classificados como positivos	43
Figura 4.13: Exemplos de tweets classificados como negativo.....	44
Figura 4.14: Exemplos de tweets classificados como neutro	45
Figura 4.15: Classificação total da polaridade e subjetividade.....	45
Figura 4.16: Word cloud da classificação neutra	46
Figura 4.17: Word cloud da classificação negativa	47
Figura 4.18: Word cloud da classificação positiva	48
Figura 4.19: Exemplo de texto vetorizado	49

Figura 4.20: Estrutura do modelo utilizado	51
Figura 4.21: Resumo do modelo.....	52
Figura 4.22: <i>Accuracy</i> do treinamento e validação	57
Figura 4.23: Perda do treinamento e validação	57
Figura 4.24: Matriz de confusão	59
Figura 4.25: Classes com a métrica AUC-ROC	60
Figura 4.26: Exemplo 1 do conjunto de teste	61
Figura 4.27: Exemplo 2 do conjunto de teste	61
Figura 4.28: Exemplo 3 no conjunto de teste	62
Figura 4.29: Exemplo 4 do conjunto de teste	62
Figura 4.30: Total das classes nos 527.236 <i>tweets</i>	63
Figura 4.31: Comparação de classes individualmente por regiões	63
Figura 4.32: Comparação de classes entre São Paulo e Rio de Janeiro.....	64
Figura 4.33: Exemplo 1 rotulado pelo modelo.....	64
Figura 4.34: Exemplo 2 rotulado.....	65
Figura 4.35: Exemplo 3 rotulado.....	65
Figura 4.36: Exemplo 4 rotulado.....	66
Figura 4.37: Textos próprios para testar modelo	66
Figura 4.38: Word clouds <i>analytical</i>	67
Figura 4.39: Word clouds <i>anger</i>	68
Figura 4.40: Word clouds <i>confident</i>	69
Figura 4.41: Word clouds <i>fear</i>	70
Figura 4.42: Word clouds <i>joy</i>	71
Figura 4.43: Word clouds <i>sadness</i>	72
Figura 4.44: Word clouds <i>tentative</i>	73
Figura 4.45: Word clouds <i>none</i>	74
Figura 4.46: Resumo dos dados rotulados pelo modelo.....	75
Figura 4.47: Resumo do treinamento do modelo em português	75
Figura 4.48: <i>Accuracy</i> do modelo para textos do português	77
Figura 4.49: <i>Loss</i> do modelo para textos do português.....	77
Figura 4.50: Matriz de confusão do modelo em português na decima época	78
Figura 4.51: AUC-ROC do modelo em português na decima época	79
Figura 4.52: Matriz de confusão do modelo em português na quinta época	79

Figura 4.53: AUC-ROC do modelo em português na decima época	80
Figura 4.54: Exemplo 1 modelo em português	80
Figura 4.55: Exemplo 2 modelo em português	81
Figura 4.56: Exemplo 3 do conjunto rotulado modelo em português	81
Figura 4.57: Exemplo próprios no modelo em português	81

SUMÁRIO

1 INTRODUÇÃO	13
1.1 Apresentação do tema	13
1.2 Objetivos.....	14
1.2.1 Objetivo geral	14
1.2.2 Objetivos específicos	14
1.3 Organização do trabalho.....	15
2 REFERENCIAL TEÓRICO	17
2.1 Análise de sentimentos	17
2.1.1 Classificação de polaridade e subjetividade	17
2.1.2 Classificação de emoção	18
2.1.3 Desafios da análise de sentimentos.....	19
2.2 Aprendizagem de máquina	19
2.2.1 Aprendizagem supervisionada	20
2.2.2 Overfitting	21
2.2.3 Tipos de classificação	22
2.3 Aprendizagem profunda.....	24
2.3.1 Um modelo de Deep Learning.....	25
2.3.2 Função de ativação e Função de perda.....	26
2.3.3 Otimizadores	28
2.3.4 Embeddings.....	28
2.3.5 Dropout	29
3 RECURSOS	29
4 DESENVOLVIMENTO	32
4.1 Coleta de dados	32
4.2 Preparação dos dados	36
4.2.1 Preparação dos textos para tradução	36
4.2.2 Tradução dos textos.....	38
4.2.3 Limpeza dos textos traduzidos	38

4.2.4 Remoção de tweets duplicados por usuário	39
4.2.5 Formatação das localizações dos usuários	39
4.2.6 Criar conjunto de dados rotulados.....	41
4.3 Classificação de polaridade e subjetividade com Textblob	43
4.4 Criando modelo para classificação de emoção.....	48
4.4.1 Entradas	49
4.4.2 Camadas	51
4.4.3 Compilação do Modelo	53
4.4.4 Treinamento do Modelo	54
4.4.5 Avaliação do modelo e previsões.....	57
4.4.6 Rotular novos dados	62
4.5 Criando o modelo para português	74
4.5.1 Avaliação do modelo em português	77
5 CONCLUSÃO.....	82
5.1 TRABALHOS FUTUROS	83
REFERÊNCIAS BIBLIOGRÁFICAS	84
ANEXO A – Código de desenvolvimento do modelo em Inglês.....	87

1 INTRODUÇÃO

1.1 APRESENTAÇÃO DO TEMA

Processamento de Linguagem Natural (PNL) tem como propósito a criação de técnicas e algoritmos que sejam capazes de processar e produzir dados não estruturados de linguagem natural. O crescimento das mídias sociais ajudou na criação de grandes quantidades de informações disponíveis na Internet que podem ser exploradas para o desenvolvimento de aplicações de PNL. Uma categoria em especial de dados são os que expressem opiniões sobre produtos, ideias e serviços. Entender a opinião das pessoas é um fator importante na tomada de decisões que é essencial para grupos como empresas, associações e governos. Com um melhor entendimento da opinião pública, esses grupos podem se beneficiar de uma melhor tomada de decisões estratégicas e, assim, reagir rapidamente às mudanças econômicas e sociais.

O crescimento de dados de opiniões nas mídias sociais ajudou no surgimento e desenvolvimento de um subcampo da PNL conhecido como análise de sentimento. Muitas aplicações podem ser desenvolvidas em torno da análise de sentimentos e geralmente os dados para essas aplicações podem ser extraídos de fontes como Twitter, Facebook, YouTube e blogs. Alguns exemplos para o uso dos dados em mídias sociais são medir o sentimento de iniciativas de desenvolvimento, por exemplo uma campanha de vacinação ou o processo político de um país.¹ Outro caso seria a utilização dos dados de uma rede social como o Twitter para responder rapidamente a crises emergentes, como epidemias ou desastres naturais.¹⁻²

Alguns dos fatores que impulsionam o desenvolvimento da PNL e da Análise de Sentimento incluem aumento do poder de computacional, disponibilidade de grandes quantidades de dados na internet, avanços em algoritmos de aprendizado de máquina (ML) e uma melhor compreensão do uso de estruturas de linguagem humanas.³ Devido a esses fatores, é possível observar várias aplicações sendo

¹ RUSSELL, M. A.; KLASSEN, M. Mining the Social Web. In: RUSSELL, M. A.; KLASSEN, M. **Mining the Social Web. Prefácio**. 3. ed. Sebastopol: O'Reilly Media Inc., 2019. p. xviii.

² MICHAILIDIS, M. **Social Machine Learning with H2O, Twitter, python**. LinkedIn, September 10, 2017. Disponível em: <<https://www.linkedin.com/pulse/social-machine-learning-h2o-twitter-python-marios-michailidis>>. Acesso em: 12 Novembro 2020.

³ HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. **Science**, New York, v. 349, n. 6245, p. 261-266, 16 jul. 2015. Disponível em: <<http://dx.doi.org/10.1126/science.aaa8685>>. Acesso em: 12 nov. 2020.

criadas por empresas, como assistentes virtuais (Alexa, Siri, Cortana), Chatbots, classificadores de spam, analisadores de sentimento e tradutores.

É evidente que a análise de sentimento pode trazer muitos benefícios devido à ampla gama de aplicações existentes e também pelo fato de que opiniões são fundamentais para a maioria das atividades humanas, o que justifica os grupos interessados em investir e realizar mais pesquisas nessa área o que promove um crescimento mais acentuado do mercado.⁴

A COVID-19 é uma doença respiratória, ela é causada pelo vírus SARS-CoV-2 que apresenta uma alta taxa de propagação e sintomas que variam de leves a graves. Diversas áreas da sociedade foram impactadas por causa da COVID-19, o que gerou uma mudança de comportamentos nas pessoas. Nesse cenário, a análise de sentimento é útil para trazer informações importantes para grupos interessados em entender a opinião pública sobre a tomada de decisões em resposta à pandemia. Sendo assim, este trabalho será dedicado a aplicação da análise de sentimentos em textos, também conhecidos como ‘*tweets*’ que foram coletados de usuários da rede social *Twitter* a fim de identificar a relevância de palavras relacionadas a COVID-19 no intervalo de tempo de 29 dias (07/09/2020 a 05/10/2020).

1.2 OBJETIVOS

1.2.1 Objetivo geral

Criar um modelo utilizando as bibliotecas *Keras* e *TensorFlow*, que seja capaz de classificar textos em português para 7 emoções com base em dados rotulados na API da IBM *Watson Tone Analyzer*.

1.2.2 Objetivos específicos

- Coletar *tweets* em português que tenham as palavras “#quarentena, #pandemia, #covid19, #covid19br, #coronavirus, #covid19, #covid-19, #ficaemcasa, #covid, quarentena, pandemia, covid19, covid19br, coronavirus, covid19, covid-19, ficaemcasa, covid” com a exclusão de retweets, utilizando a

⁴ LIU, B. **Sentiment Analysis**: Mining Opinions, Sentiments, and Emotions. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 21-22 p.

biblioteca do *Tweepy* (Python) em um intervalo de 29 dias de 07/09/2020 a 05/10/2020.

- Realizar o pré-processamento dos *tweets* para que possam ser utilizados com o *TextBlob* e o *Watson Tone Analyzer*, além do pré-processamento necessário para o treinamento dos modelos de classificação.
- Realizar uma Análise Exploratória de Dados nos conjuntos de dados de dados brutos, pré-processados e dados rotulados pelo modelo.
- Utilizar a biblioteca *TextBlob* para fazer uma classificação dos *tweets* em positivo, neutro e negativo.
- Criar um pequeno conjunto de dados rotulados pela API da IBM Tone Analyzer para serem utilizados no treinamento do modelo inicial.
- Criar um modelo inicial com as bibliotecas *Keras* e *TensorFlow* para classificar textos em inglês com a utilização de *Word Embeddings* pré-treinados que no caso será o GloVe (Global Vectors)
- Avaliar o desempenho do modelo inicial
- Rotular os conjuntos de dados restantes que não foram rotulados pelo *Watson Tone Analyzer* com o modelo inicial.
- Fazer uma análise do conjunto de dados totalmente rotulado por regiões de Estados.
- Criar um modelo final para classificar textos em português utilizando o conjunto de dados completamente rotulado.

1.3 ORGANIZAÇÃO DO TRABALHO

Capítulo 1 – Introdução: O trabalho é organizado em 5 capítulos, incluindo esta Introdução onde há a apresentação do tema, citação dos objetivos e organização do trabalho.

Capítulo 2 – Referencial Teórico: Onde os conceitos teóricos relacionados à construção e compreensão do trabalho serão brevemente descritos.

Capítulo 3 – Recursos: Descrição dos recursos utilizados para o desenvolvimento do trabalho.

Capítulo 4 – Explicação de como foi o desenvolvimento de cada etapa do projeto coleta, pré-processamento, análise exploratória de dados, criação dos modelos de

classificação, avaliação de desempenho dos modelos e análise final dos dados rotulados por regiões.

Capítulo 5 – Conclusão: Descreve os objetivos alcançados, o conhecimento adquirido, as limitações dos métodos usados para criar o modelo de classificação e as próximas etapas para a continuação do trabalho.

2 REFERENCIAL TEÓRICO

2.1 ANÁLISE DE SENTIMENTOS

A análise de sentimento, também chamada de mineração de opinião, é o campo de estudo que analisa as opiniões, sentimentos, avaliações, atitudes e emoções das pessoas em relação às entidades e seus atributos expressos em texto escrito. As entidades podem ser produtos, serviços, organizações, indivíduos, eventos, questões ou tópicos.⁵

Nos últimos anos, cada vez mais atenção tem sido dada a análise de sentimento. Esse interesse é motivado não apenas pela grande quantidade de textos disponíveis na internet como comentários, blogs, redes sociais e chats, mas também pelo grande número de aplicações potenciais e os novos desafios que este campo representa para a comunidade de pesquisa.

Existem muitas tarefas que a análise de sentimento é capaz de lidar, como por exemplo, classificação de polaridade, classificação de subjetividade e classificação de emoção. Para realizar essas tarefas diversas abordagens podem ser adotadas e vão desde a criação de dicionários de léxicos de opinião, que são lista de palavras que apresentam um sentimento positivo, negativo ou neutro, os quais são utilizados para atribuir uma polaridade a um texto, até métodos de classificação utilizados no aprendizado de máquina supervisionado.

2.1.1 Classificação de polaridade e subjetividade

Uma tarefa muito popular da análise de sentimentos é a classificação da polaridade de um texto que geralmente é feito com estratégias de dicionários léxicos de opinião. Pang e Lee ⁶ descrevem a classificação de polaridade possuindo as seguintes características: dado um texto com opiniões, onde se assume que a opinião geral é direcionada para um único tópico ou item, a classificação deve ser feita com escolha uma de duas polaridades opostas de sentimento ou localizar a posição no continuum entre as duas polaridades.

⁵ LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 1 p. Tradução Nossa.

⁶ PANG, B.; LEE, L. Opinion mining and sentiment analysis. **Foundations And Trends® In Information Retrieval**, [S.l.], v.2., n. 1-2., 14 nov. 2008. 1-135 p. Disponível em: <<http://dx.doi.org/10.1561/15000000011>>. Acesso em: 12 nov. 2020.

Basicamente a polaridade é uma maneira de definir o grau de satisfação ou insatisfação ao dizer se um sentimento ou opinião é positivo, negativo ou neutro. Desta forma pode ser feita a representação em uma reta numérica para mostrar a intensidade do sentimento na frase. Por exemplo, uma reta cuja polaridade vai de **-1 (Negativo)** a **+1 (Positivo)** conforme mostra a Figura 2.1.

Figura 2.1: Exemplo da polaridade positiva, negativa e neutro



Fonte: Próprio autor

Como pode ser visto na frase “**Eu odeio corrida**” representa uma intensidade negativa maior por causa da palavra “**odeio**”, enquanto o exemplo “**eu não gosto de corrida**” traz um sentimento menos intenso por combinar um termo positivo “**gostar**” e um negativo “**não**”. No caso do exemplo neutro “**Eu corro**” não apresenta uma palavra que traga um sentimento, agora nos exemplos positivos a intensidade da palavra **amar** é maior que **gostar**.

2.1.2 Classificação de emoção

Outra tarefa interessante da análise de sentimento é a classificação de emoção e humor. A classificação de emoção pode ser considerado umas das mais difíceis, porque possuem muitas classes para diferentes tipos de emoções e humores, portanto, são difíceis de distinguir por serem muito semelhantes.⁷

Algumas abordagens utilizadas para classificação de emoção ou humor podem ser feitas com a aprendizagem supervisionada. “Como a classificação de sentimento é um problema de classificação de texto, qualquer método de aprendizagem supervisionado existente pode ser aplicado diretamente, como a classificação com *Naive Bayes* ou *Support Vector Machine (SVM)*”⁸

⁷ LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 67-69 p.

⁸ JOACHIMS, 1999; SHAW-TAYLOR AND CRISTIANINI, 2000, apud LIU, 2015, p. 49. Tradução Nossa.

2.1.3 Desafios da análise de sentimentos

A análise de sentimento é um campo de pesquisa amplo e complexo, sua aplicação nas mídias sociais pode apresentar muitas dificuldades, pois são ambientes desafiadores por serem informais e heterogêneos, o que introduz mais complexidade as tarefas. Alguns exemplos dessas dificuldades são inconsistências (informações falsas e aleatórias), textos mal formatados, erros de digitação ou gramática, metadados (idade, gênero e localização) e a utilização de termos informais para abreviações como você (vc), muito (mt), obrigado (obg) etc.

O sarcasmo e a ironia também são difíceis de lidar por serem modos indiretos de comunicação. Mesmo textos bem formatados, podem ser mal interpretados por classificadores se não houver um contexto adicional fora da frase. Um exemplo de sarcasmo: "**Você corre incrivelmente rápido igual a uma tartaruga**" poderia confundir um classificador de polaridade, pois o termo "**incrivelmente**" traz um sentido positivo, embora se deseje transmitir um sentimento negativo ao fazer referência a tartaruga.

2.2 APRENDIZAGEM DE MÁQUINA

O aprendizado de máquina (ML – *Machine Learning*) é o campo de pesquisa da inteligência artificial (IA), que procura desenvolver algoritmos que possam aprender automaticamente com a experiência alcançada por meio do processamento de dados. Pode-se dizer que o algoritmo de aprendizado de máquina constrói um modelo que tenta encontrar representações adequadas aos dados de entrada realizando transformações para torná-los mais apropriados ao problema.

Desta maneira, o modelo é treinado em vez de ser programado explicitamente, permitindo que seja criado regras estatísticas das representações dos dados. Por exemplo, uma tarefa para classificar imagens de cães e gatos, o modelo poderia ser treinado com muitas imagens de gatos já rotuladas, então o modelo poderá aprender as regras para diferenciar gatos de cães, posteriormente o modelo pode ser utilizado para automatizar a tarefa de classificação em novas imagens.

A importância do aprendizado de máquina se mostra interessante pelo fato de que não é possível antecipar todas as situações de um sistema, o que implica na criação de várias regras para a resolução dos problemas. Além disso, pode não ser fácil criar uma solução para um problema utilizando as abordagens tradicionais como no caso do reconhecimento de imagem.

Fica evidente, portanto, que o ML pode ajudar na automatização da resolução de problemas e facilitar esse processo que geralmente não é trivial quando se utiliza abordagens tradicionais, além disso permite a descoberta de novos *insights*.

2.2.1 Aprendizagem supervisionada

A aprendizagem supervisionada busca desenvolver algoritmos capazes de construir modelos que possam ser treinados com dados de treinamento rotulados que são pares de entradas e saídas esperadas (rótulos). O modelo tenta encontrar uma regra geral que mapeie as entradas em saídas para que possa fazer previsões sobre dados não vistos ou futuros.

Portanto, o objetivo do modelo de aprendizado supervisionado é prever o rótulo correto para novas entradas. Por exemplo, as entradas podem ser textos, onde cada entrada possui uma saída que define o texto como positivo ou negativo. O modelo é treinado com os dados de treinamento para aprender os padrões, então ao receber um novo texto, ele tenta prevê o rótulo apropriado, isso é conhecido como uma tarefa de **classificação**. Caso o objetivo da tarefa seja fazer previsões de resultados contínuos, como a previsão de preços de imóveis, então essa será uma tarefa mais indicada para ser tratada por **Análise de Regressão**.

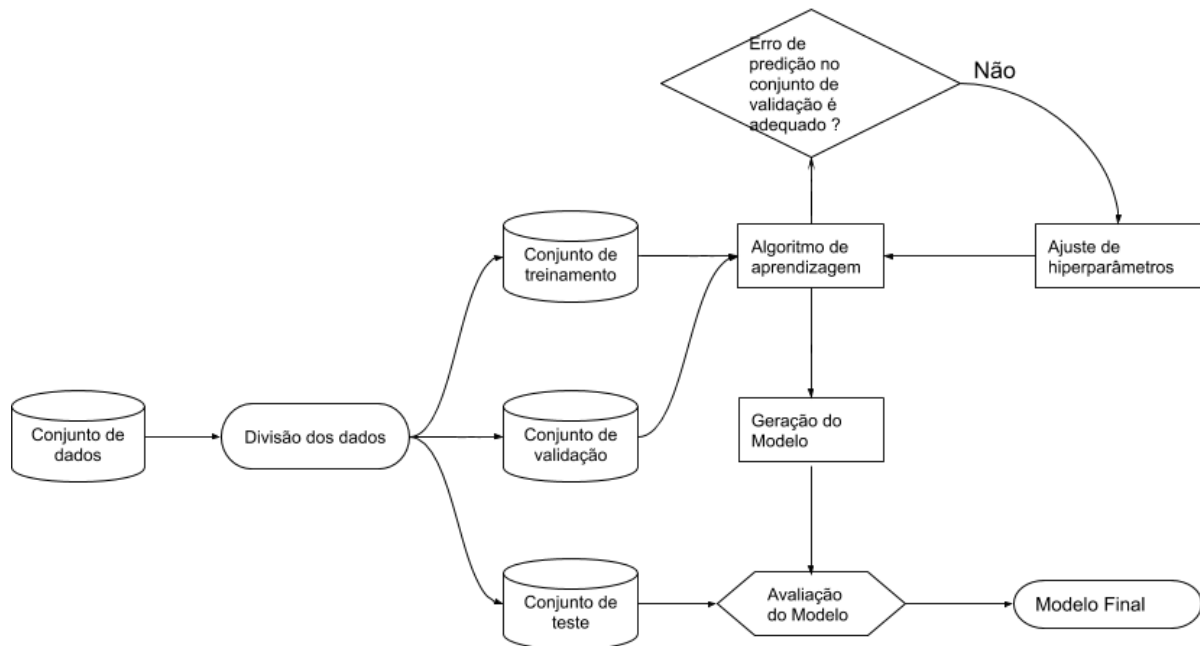
Normalmente, para realizar o aprendizado supervisionado, é necessário fazer a divisão do conjunto de dados para a serem utilizados nas etapas de treinamento e avaliação. Uma abordagem prática é dividir aleatoriamente o conjunto de dados em três partes, conjunto de treinamento, conjunto de validação e conjunto de teste.

O conjunto de treinamento é usado para ajustar os parâmetros de um modelo por exemplo, os pesos, a utilização do conjunto de validação é feita para ajustar os hiperparâmetros (por exemplo a taxa de aprendizagem) de acordo com as estimativas do erro de predição em relação ao conjunto de validação durante ou após o treinamento o que ajuda a evitar o *Overfitting* em novos dados. O conjunto de teste é usado para avaliação do erro de generalização do modelo final.⁹⁻¹⁰

⁹ HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning**: Data Mining, Inference, and Prediction. 2. ed. New York: Springer Verlag New York Inc, 9 Feb 2009. 222 p. Disponível em: <<https://web.stanford.edu/~hastie/ElemStatLearn/>>.

¹⁰ GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: The Mit Press, 18 Apr 2017. 120-122 p.

Figura 2.2: Diagrama que representa o processo de treinamento e avaliação



Fonte: Próprio autor

A figura 2.2 ilustra como uma ideia de como é a utilização dos conjuntos de dados com um algoritmo de aprendizagem.

2.2.2 Overfitting

De acordo com Goodfellow, Bengio e Courville “O principal desafio do aprendizado de máquina é que devemos ter um bom desempenho em entradas novas e não vistas, não apenas naquelas nas quais nosso modelo foi treinado”.¹¹ Se adaptar adequadamente a novas entradas que ainda não foram vistas pelo modelo é chamado de generalização. Para o modelo alcançar um bom desempenho em novas entradas é necessário que tenha a capacidade de reduzir tanto o erro de treinamento e quanto a diferença entre o erro do treinamento e o erro de generalização (erro de teste). No entanto dois desafios surgem a partir dessas questões que são o *underfitting* e *overfitting*.¹¹⁻¹²

O *overfitting* é um problema que ocorre quando o modelo se adapta muito bem aos dados de treinamento, mas não consegue generalizar bem para novas entradas.

¹¹ GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: The Mit Press, 18 Apr 2017. 108-112 p. Tradução Nossa.

¹² GÉRON, A. The Machine Learning Landscape. In: GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2ª. ed. Sebastopol: O'Reilly Media Inc., 2019. Cap. 1.

Isso geralmente acontece porque há muito ruído nos dados de treinamento ou há poucos dados para o treinamento o que introduz ruído de amostragem. Quando o modelo não consegue se adaptar aos dados de treinamento é considerado como *underfitting*, geralmente isso acontece porque o modelo é muito simples para aprender as estruturas subjacentes dos dados. É possível tomar algumas ações para lidar com *overfitting* como, por exemplo, simplificar o modelo para que ele tenha menos parâmetros, utilizar técnicas de regularização como parada antecipada, regularização L1 e L2, *dropouts* e aumentar a quantidade de dados de treinamento.¹²

2.2.3 Tipos de classificação

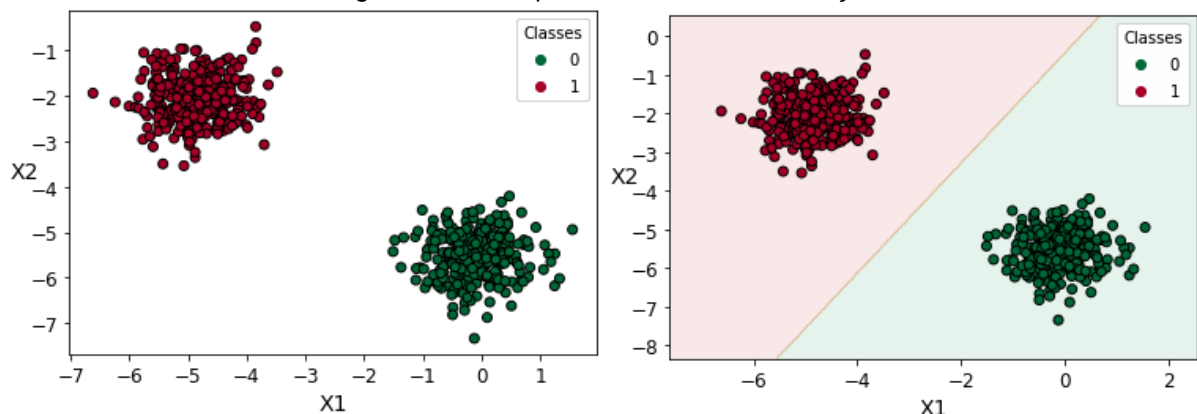
A classificação é uma tarefa de aprendizagem supervisionada em que um algoritmo de aprendizagem aprende regras para mapear entradas em saídas (rótulos), a fim de poder atribuir rótulos de classes a novos exemplos de problemas. Um exemplo disso seria classificar *e-mails* como *spam* ou *não spam*. Há 3 tipos principais para tarefas de classificação no aprendizado de máquina que são Classificação Binária, Classificação Multiclasse e Classificação Multirrótulo.

Classificação Binária

A classificação Binária acontece quando as tarefas de classificação possuem apenas dois rótulos de classe. Como no exemplo de detecção de *spam* de *e-mail*, onde há apenas duas classes *spam* ou/e *não spam*, geralmente nesses tipos de problemas uma classe é o estado normal (*não spam*) com rótulo 0 e outra classe que é o estado anormal (*spam*) com rótulo 1.

O exemplo na figura 2.3 exibe o conceito de uma tarefa de classificação binária, onde há 600 amostras em que 300 são para rótulos da classe negativa (vermelho) e o restante para a classe positiva (verde).

Figura 2.3: Exemplo de tarefa de classificação binária

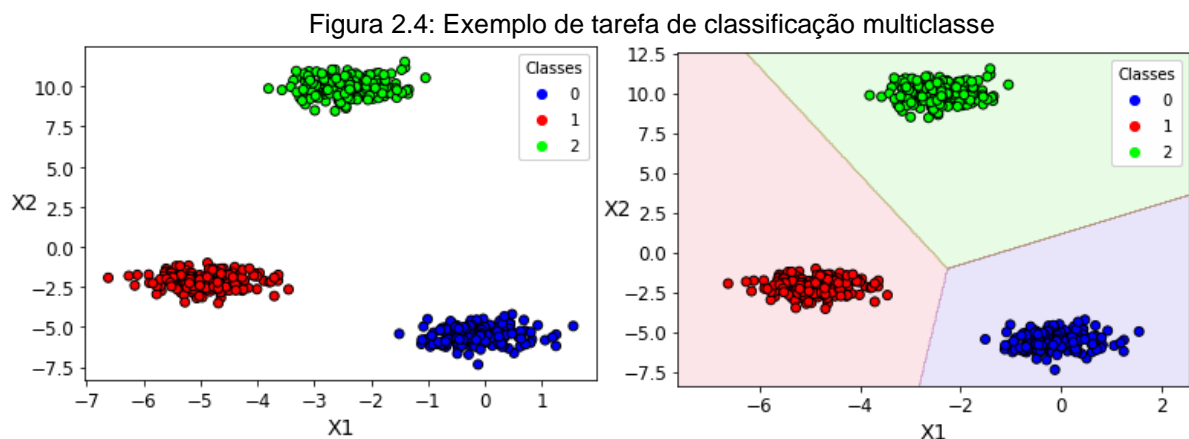


Fonte: Próprio autor

No exemplo da figura 2.3 é possível ver como é uma tarefa de classificação binária (figura esquerda) e como uma essa tarefa pode ser utilizada por um algoritmo de aprendizado (figura direita) para aprender uma regra que devida um limite de decisão representado pela linha que separa os dados. Nesses tipos de tarefas geralmente utiliza-se uma modelo capaz de fazer previsões das saídas como uma distribuição de Bernoulli o que dá a probabilidade de uma variável aleatória binária $X \in \{0, 1\}$ ser igual a 1.

Classificação Multiclasse

A classificação Multiclasse acontece quando as tarefas de classificação possuem mais de dois rótulos de classe. Diferente da classificação binária não há um estado anormal, um exemplo seria classificar imagens de dígitos com 10 classes que vão de 0 a 9. A figura 2.4 ilustra o conceito de uma tarefa de classificação Multiclasse de 3 classes com um total de 600 amostras cada classe com 200 amostras.



Fonte: Próprio autor

Neste exemplo foi utilizado um algoritmo máquina de vetores de suporte (SVM - *Support Vector Machine*) adaptado para estratégia um-contra-o-resto (OvR - *One-vs-Rest*) para realizar a classificação das três classes na figura 2.4 (direita).

Classificação Multirrótulo

A classificação Multirrótulo acontece quando as tarefas de classificação possuem um ou mais rótulos de classe para cada amostra. Um exemplo pode ser a classificação de emoções em um texto que pode ter vários rótulos para cada emoção, Considere um exemplo em que há um classificador que atribua três classes (Medo, Surpresa e Felicidade), então um texto como “**Nossa!** isso é **adorável**.” poderia ser classificado como “**Surpresa**” e “**Felicidade**”, mas não seria classificado como “**Medo**”.

A tabela 2.1 abaixo ilustra como seria um exemplo. Neste caso os textos são transformados em uma estrutura codificada de inteiros, depois as entradas são usadas para treinar um algoritmo com a abordagem do aprendizado supervisionado para gerar um modelo que mapeie as entradas x para vetores binários y , atribuindo um valor de 0 ou 1 para cada rótulo em y . Nesse tipo de tarefa também é usual utilizar uma função que permita ao modelo fazer previsões de cada saída como se faz quando se usa uma distribuição de probabilidade Bernoulli.

Tabela 2.1: Exemplo de como os dados são na classificação Multirrótulo

Texto/Entrada	Texto transformado	Y [Surpresa, Felicidade, Medo]
"Nossa! isso é adorável"	[10 12 5 16 0 0]	[1, 1, 0]
"Que horror! isso..."	[7 13 9 6 15 0]	[0, 0, 1]
"Outro dia normal."	[8 14 11 0 0 0]	[0, 0, 0]
"Meu Deus! Credo!"	[4 2 3 0 0 0]	[1, 0, 1]
"Meu Deus! Deus! Deus! Credo! Credo"	[4 2 2 2 3 3]	[1, 0, 1]

Fonte: Próprio autor

Neste exemplo foi utilizado a classe *TextVectorization*¹³ da biblioteca Keras para vetorizar os textos, transformando cada texto em uma sequência de inteiros podendo ser utilizado para uma rede neural artificial (RNA) com camada *Embedding* ou camada densa que será explicado posteriormente. A última linha da tabela 2.1 mostra que a palavra "Deus!" é representada pelo número 2, nesse caso o modelo teria que receber muitos exemplos com rótulos surpresa para poder encontrar uma representação que relacione o termo "Meu Deus!" com o rótulo surpresa.

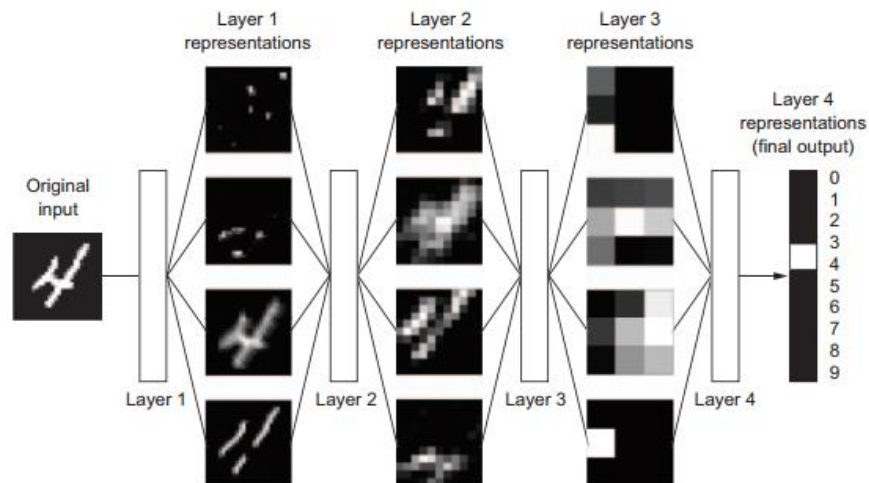
2.3 APRENDIZAGEM PROFUNDA

A Aprendizagem profunda (*Deep Learning* - DL) é um subcampo do aprendizado de máquina que usa uma nova abordagem para aprender representações de dados, onde o foco é o aprendizado de camadas sucessivas de representações. A aprendizagem profunda começou ficar popular com os avanços dos algoritmos de ML, o crescimento de dados e a evolução dos hardwares (Por exemplo as GPUs - *Graphic Processing Units*).

O significado de profundo na aprendizagem profunda está relacionado a quantidade de camada que compõem um modelo. A criação desses modelos de várias camadas é principalmente feita através da utilização de redes neurais artificiais

¹³ CHOLLET, F.; OMERNICK, M. Working with preprocessing layers. **Keras**, 25 julho 2020. Disponível em: <https://keras.io/guides/preprocessing_layers/>. Acesso em: 12 Novembro 2020.

(RNAs). Um exemplo que ilustra a ideia da abordagem em várias camadas é o perceptron multicamadas, que é composto por várias camadas de perceptron. O perceptron é uma RNA que pode ser definido como uma função matemática que mapeia um conjunto de valores de entrada para valores de saída.



Fonte: CHOLLET¹⁴

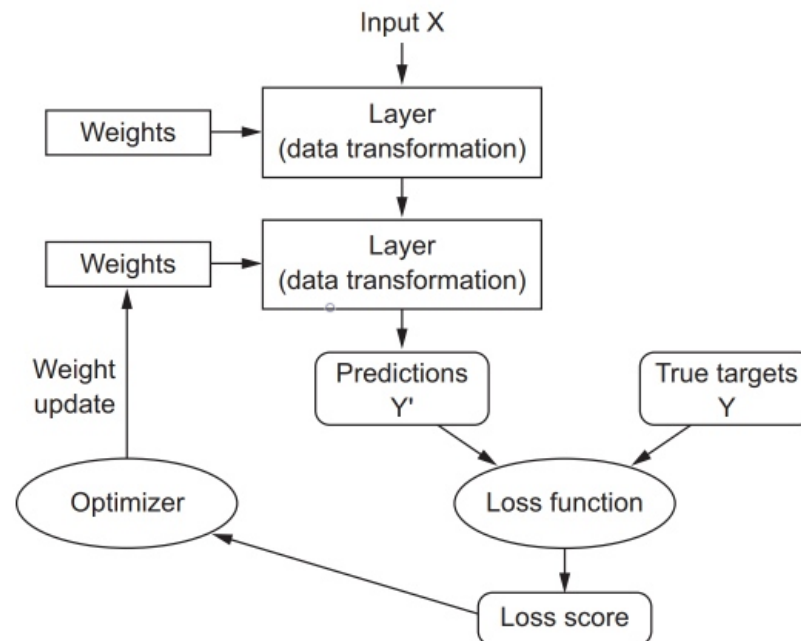
A figura 2.5 exibe o conceito de profundidade que é representado pela utilização de várias camadas "layers". A entrada (input) é considerada uma camada visível (visible layer), pois é possível visualizar as variáveis de entrada. Em seguida a entrada é passada por diversas camadas que são chamadas de camadas ocultas (hidden layers) já que não são observáveis diretamente, essas camadas buscam aprender diferentes representações dos dados. O exemplo da imagem mostra que cada camada tenta identificar alguma parte da imagem como bordas, arestas e cantos no final é emitido uma saída (output) que traz a identificação do objeto.

2.3.1 Um modelo de Deep Learning

A criação de modelos em *Deep Learning* possui alguns conceitos-chaves que estão relacionados com RNAs. Os principais conceitos para criação de um modelo são os dados de entrada e os alvos correspondentes, a combinação das camadas de RNAs, a definição da função de perda que é responsável pelo sinal de feedback usado na aprendizagem e o otimizador, que determina como a aprendizagem prossegue.

¹⁴ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 9 p.

Figura 2.6: Exemplo do relacionamento das redes, camadas, função de perda e otimizador



Fonte: CHOLLET¹⁵

A figura 2.6 exibe como esses conceitos estão interligados, primeiro há as entradas que são passadas pelas camadas da rede. Nessas camadas as entradas são processadas em funções (RNAs) que transformam as entradas para que as próximas camadas identifiquem novas representação, então os pesos ou parâmetros (Weights) que são os coeficientes das funções são salvos. As entradas são mapeadas nessas camadas ao realizar as previsões dos rótulos (*targets*), onde a cada ciclo de treinamento a função de perda (*Loss function*) produz um valor de perda (*Loss score*) da comparação das previsões feitas pelo modelo e os rótulos reais. O otimizador (*Optimizer*) faz a utilização do valor da perda para atualizar os pesos.¹⁵ Este processo é repetido até que o erro na rede caia abaixo de um limite aceitável.

2.3.2 Função de ativação e Função de perda

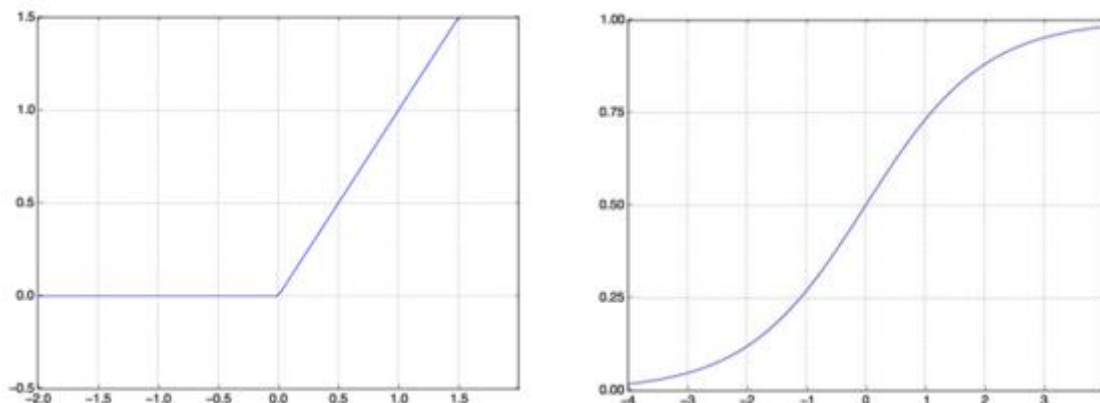
Função de ativação

A função de ativação é uma função utilizada para transformar a entrada ou a saída de uma RNA. Há vários tipos de funções de ativação destaque duas em especial para esse trabalho a sigmóide e Relu (*Rectified Linear Activation Unit*). As funções de ativação são importantes, pois adicionam não linearidades às redes neurais, o que beneficia as RNAs a fazer melhores representações dos dados. A função Relu é uma

¹⁵ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 58-59 p.

função utilizada para zerar os valores negativos, e a função sigmoide limita os valores em um intervalo de $[0, 1]$. Uma combinação das duas funções em camadas densas, onde a primeira camada densa usa a função Relu e, em seguida, a segunda camada densa usa a função sigmoide, permite que seja feita a geração de saídas com probabilidade entre 0 e 1.¹⁶

Figura 2.7: Funções Relu e Sigmoid



Fonte: CHOLLET¹⁶

Função de perda

A função de perda ou função de custo é uma função utilizada para medir a distância entre uma saída prevista para um valor esperado, sendo assim calculando uma pontuação que mostre o quão bom é a rede em fazer previsões. Essa pontuação é utilizada pelo otimizador para ajustar os pesos da rede.

Figura 2.8: Função de ativação da última camada e função de perda para o modelo

Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

Fonte: CHOLLET¹⁷

Cada tipo de problema possui uma função de perda e adequada, como pode ser visto na figura 2.8, os problemas de multirrótulo, utilizam a função de ativação sigmoide, para que as saídas da rede sejam probabilidades no intervalo de $[0, 1]$.

¹⁶ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 70-73 p.

¹⁷ Ibid, p. 114.

Enquanto a função de perda escolhida é a *binary_crossentropy*, pois o *crossentropy* ajuda a medir uma distância entre as distribuições de probabilidade, então sendo útil para a verificação das saídas previstas e saídas reais.¹⁵

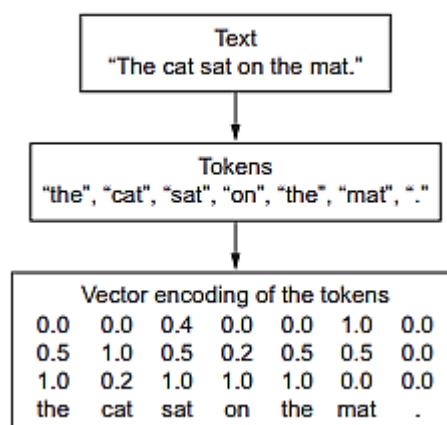
2.3.3 Otimizadores

Os otimizadores são algoritmos utilizados para atualizar os pesos com o objetivo de minimizar a função de perda. Muitos otimizadores estão disponíveis como Momentum, Nesterov, RMSProp e Adam etc. No capítulo de desenvolvimento será utilizado o Adam (*Adaptive Moment Estimation*)¹⁸ que é baseado na descida de gradiente estocástica, com características dos otimizadores RMSProp e momentum.

2.3.4 Embeddings

Os modelos de aprendizado profundo buscam mapear a estrutura estatística da linguagem escrita, sendo assim capaz de atribuir um rótulo para a tarefa de classificação supervisionada. Dessa maneira como as redes neurais não aceitam texto bruto de entrada é necessário que os textos passem por um processo de transformação em tensores numéricos que é conhecido como vetorização de texto. No processo de vetorização de texto as palavras são divididas em tokens o que é chamado de tokenização, em seguida esses tokens são associados vetores numéricos.¹⁹ A estratégia utilizada para a vetorização de texto no conjunto de treinamento será *word embedding*.

Figura 2.9: Exemplo de Vetorização de texto



Fonte: CHOLLET¹⁹

¹⁸ KINGMA, D. P.; BA, J. L. **Adam: A Method For Stochastic Optimization**, San Diego, 2015. 1-15 p. Disponível em: <<https://arxiv.org/abs/1412.6980v9>>. Acesso em: 22 nov. 2020.

¹⁹ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 180 p.

Há duas maneiras de obter Word Embedding, a primeira é deixar que o modelo aprenda os *embeddings* semelhante ao modelo aprender os pesos da rede neural, nesse caso o modelo pode ter uma camada de *embeddings* que pega inteiros como entrada e procura esses inteiros em um dicionário interno e retorna os vetores associados. A outra maneira seria carregar um modelo Word Embedding que já foi processado em outra tarefa de aprendizado de máquina diferente para ser utilizado em uma nova tarefa, esses Word Embeddings são chamados de *pretrained Word Embeddings*, sua utilização é útil para quando há poucos dados de treinamento.²⁰

O GloVe (Global Vectors for Word Representation) é um exemplo de *Word Embeddings* pré-treinados, que é usado para obter representações vetoriais de palavras.²¹ O GloVe tem a capacidade de identificar características estatísticas globais e estatísticas locais de um corpus (conjunto de textos), a fim de criar vetores de palavras.

2.3.5 Dropout

O *Dropout* é uma técnica de regularização que pode ser utilizada em uma camada de um modelo, sua função é descartar aleatoriamente (descartar significa definir o valor como 0) alguns valores da saída de uma camada durante o treinamento. Um exemplo seria pensar em uma camada que recebe uma entrada e retorna um vetor [0.6, 1, 0.4, 0.7, 1, 0.2, 0.5, 1]. Nesse exemplo se essa camada aplicasse *dropout* com uma taxa de 50% então metade dos valores seriam descartados (zerados) podendo ficar como [0, 1, 0, 0, 1, 0.2, 0, 1].²² O *dropout* ajuda a evitar o *overfitting*, pois esse descarte fará com que uma camada veja mais exemplos de dados diferentes, o que força com que as próximas camadas tenham que lidar com diferentes representações dos dados durante o treinamento de um modelo.

3 RECURSOS

Twitter

O **Twitter** é uma rede social, onde é possível publicar textos com até 140 caracteres chamados de **tweets**, também é possível publicar fotos e vídeos. Os usuários do Twitter publicam temas de variados assuntos como notícias e entretenimento para esportes, política e interesses cotidianos.

²⁰ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 184-188 p.

²¹ PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. **nlp.stanford.edu**. Disponível em: <<https://nlp.stanford.edu/projects/glove/>>. Acesso em: 10 Novembro 2020.

²² CHOLLET, F. Op cit. 109-110 p.

Google Colaboratory

O Google Colaboratory ou COLAB é um serviço de nuvem gratuito hospedado pelo Google e desenvolvido para incentivar o estudo e facilitar trabalhos de Inteligência Artificial e Ciência de Dados. O COLAB fornece acesso gratuito à GPU e permite fácil compartilhamento de código. Por meio do COLAB, é possível escrever código em linguagem Python ou R, e sua interface é semelhante à do *Jupyter Notebook*, portanto, consiste em uma série de células, que podem conter texto explicativo ou código executável e sua respectiva saída.

Watson Tone Analyzer

O *Watson Tone Analyzer* é um serviço IBM que usa análise linguística para detectar tons emocionais nos textos e realizando uma classificação desses tons. O *Tone Analyzer* funciona ao enviar um texto ou um arquivo JSON para API disponibilizada pela IBM, então o serviço retorna resultados em JSON com a classificação dos tons para o texto de entrada. A seguir há uma descrição das 7 classes que representam os tons no *Tone Analyzer*.

Tabela 4.2: Descrição das classes no Tone Analyzer

Classe	Descrição
<i>Analytical</i> (Analítico)	Classe que representa um raciocínio e atitude mais analítica do autor do texto sobre coisas. Relacionado a coisas intelectuais, racionais, impessoais.
<i>Anger</i> (Raiva)	Classe que representa que o autor do texto está com raiva. Relacionado a ofensas, sentimentos de tensão e hostilidade.
<i>Confident</i> (Confiança)	Classe que representa o grau de certeza do autor do texto sobre algo. Relacionado a confiança, afirmação, egoísmo.
<i>Fear</i> (Medo)	Classe que representa o medo do autor do texto sobre algo. Relacionado a algo negativo, perigos e fobias.
<i>Tentative</i> (Hesitante)	Classe que representa a hesitação do autor no texto. Relacionado a coisas questionáveis, duvidosas ou discutível.
<i>Sadness</i> (Tristeza)	Classe que representa a tristeza do autor no texto. Relacionado a coisas tristes, desaminastes e decepcionantes.
<i>Joy</i> (Alegria)	Classe que representa a alegria do autor no texto. Relacionado a coisas de amor, felicidade e segurança.

Fonte: IBM²³

²³ IBM. **Tone Analyzer**. Disponível em: <<https://tone-analyzer-demo.ng.bluemix.net/>>. Acesso em: 18 Novembro 2020.

Tweepy

Tweepy é uma biblioteca Python que facilita o acessar a API do Twitter. Ela é útil para realiza automação de *bots* e coleta de dados.

Googletrans

Googletrans é uma biblioteca Python que permite a utilização da API do Google Translate para realização traduções.

Pandas

Pandas é uma biblioteca Python que permite a manipulação e análise de dados, oferecendo estruturas de dados e operações para manipular tabelas numéricas e séries temporais.

Keras

Keras é uma biblioteca Python que permite desenvolver redes neurais artificiais em alto nível simples e intuitiva, que possui diversos recursos para trabalhos de Machine Learning e Deep Learning.

TensorFlow

O TensorFlow é um framework criado pelo Google para o desenvolvimento de trabalhos de aprendizado de máquina em geral, possui muitos recursos e inclui a biblioteca *keras* como parte do seu back-end.

Pandas Profiling

Pandas Profiling é uma biblioteca Python que oferece relatórios para que seja feita a etapa de análise exploratória de dados de maneira de rápida e resumida. Isso é muito útil, pois ajuda a poupar tempo para a visualização e compreender a distribuição de cada variável em um conjunto de dados.

Numpy

NumPy é uma biblioteca Python que oferece várias operações para arrays e matrizes, possuindo muitas funções matemáticas.

Matplotlib

Matplotlib é uma biblioteca de Python que permite a criação de gráficos e visualizações de dados.

Ploty

Plotly é uma biblioteca disponível para Python semelhante ao Matplotlib, ela oferece funções para criar gráficos e visualização de dados.

Scikit-learn

A scikit-learn é uma biblioteca disponível para Python, que possui vários recursos para aprendizado de máquina, como classificadores regressão e máquinas de vetor de suporte (SVM), *random forests*, etc.

NLTK (Natural Language Tool Kit)

NLTK é uma biblioteca de Python que oferece vários recursos para tarefas de NLP, como funções de *stopwords*, vocabulários de palavras, funções para o tratamento gramatical de textos etc.

TextBlob

Textblob é uma biblioteca de Python que oferece muitos recursos para tarefas de NLP, como análise de sentimento, correção ortográfica etc. Ela utiliza a NLTK para aproveitar alguns recursos já existentes.

4 DESENVOLVIMENTO

Este capítulo descreve o desenvolvimento do projeto. A execução foi feita através do COLAB do Google, os códigos fontes utilizados são baseados em exemplos das próprias documentações das bibliotecas utilizadas. O projeto tem 5 tarefas de implementação.

- Coleta de dados
- Preparação dos dados
- Classificação da polaridade e subjetividade com TextBlob
- Criação do modelo para classificação de emoção dos textos em inglês com os dados rotulado pelo *IBM Watson Tone Analyzer*.
- Criação do modelo para classificação de emoção dos textos em português

4.1 COLETA DE DADOS

A coleta de dados desse projeto foi realizada utilizando a biblioteca *Tweepy* do Python e o código criado foi baseado no que está disponível na documentação. Foram coletados cerca de 580 mil *tweets* em português durante um período de 29 dias nos dias (07/09/2020 a 05/10/2020) (em cada dia foi coletado 20 mil tweets). A coleta é feita procurando pelas palavras “#quarentena, #pandemia, #covid19, #covid19br, #coronavirus, #covid19, #covid-19, #ficaemcasa, #covid, quarentena, pandemia, covid19, covid19br, coronavirus, covid19, covid-19, ficaemcasa, covid” em tweets ou respostas a tweets, não foram coletado reetweets para tentar evitar mensagens repetidas.

Foi realizado uma análise exploratória de dados (EDA - *Exploratory Data Analysis*) inicial nos dados brutos com a biblioteca Pandas Profiling para verificar a existência de algum problema e o que poderia ser feito também durante o processo de preparação dos dados.

Figura 4.1: Visão Geral dos dados brutos

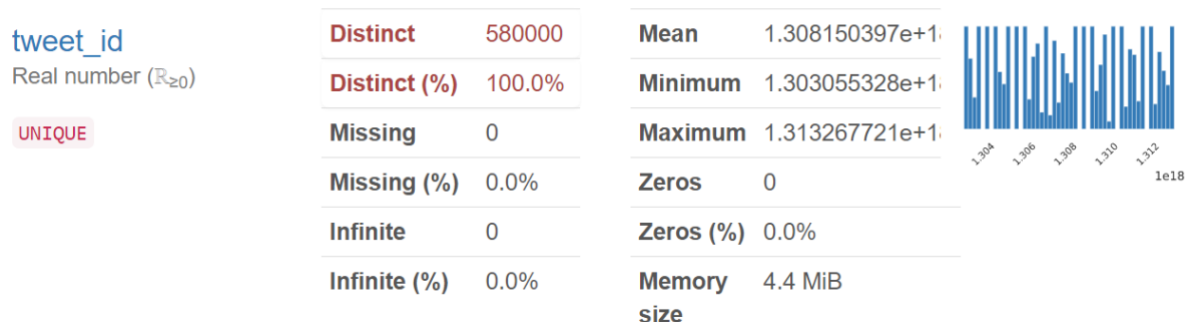
Dataset statistics		Variable types	
Number of variables	16	CAT	9
Number of observations	580000	NUM	6
Missing cells	619481	BOOL	1
Missing cells (%)	6.7%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	66.9 MiB		
Average record size in memory	121.0 B		

Fonte: Próprio autor

Como pode ser visto na figura 4.1 este é um conjunto de dados com 580 mil linhas e 16 colunas. Para o desenvolvimento será utilizado apenas três colunas que são `tweet_id`, `tweet_text` e `user_location` que passaremos a descrever.

tweet_id

Figura 4.2: Tweet_id verificando se há dados duplicados

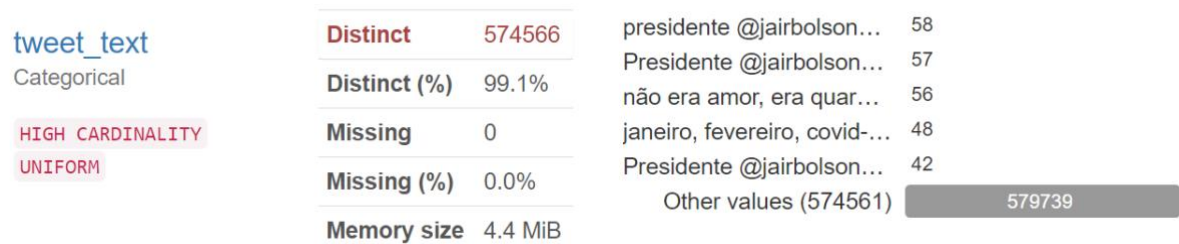


Fonte: Próprio autor

O `tweet_id` é um id que cada `tweet` possui e que foi utilizado para criação de alguns gráficos personalizados e verificação de duplicação de dados. A figura 4.2 exibe que inicialmente os dados são 100% distintos, pois variável `tweet_id` é única, o que é importante para saber que não houve nenhum tweet duplicado.

tweet_text

Figura 4.3: Verificando se houve texto duplicado

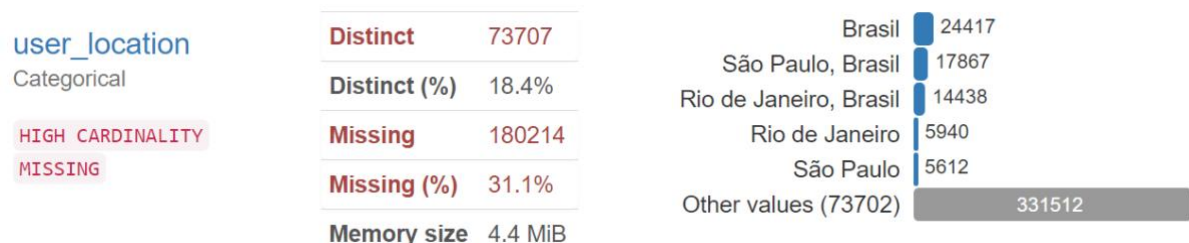


Fonte: Próprio autor

O **tweet_text** são as entradas que vão ser utilizadas para o treinamento do modelo. A figura 4.3 exibe que a coluna **tweet_text** possui alta cardinalidade o que significa que a maioria dos textos são únicos. Assim, os textos repetidos são **5434** representando **0.9%** do conjunto de dados. Algumas amostras de textos repetidos podem ser vistas do lado direito da tabela. Esses são textos que os usuários repetem sem realizar reetweets, ou seja, pode haver dois usuários que postaram o mesmo texto ou um único usuário postou a mensagem mais de uma vez.

user_location













Figura 4.4: Localização de usuários



Fonte: Próprio autor

O **user_location** é o local que o usuário define em seu perfil e que vai ser utilizado para criação de gráficos por regiões. Podemos ver que essa coluna possui alguns problemas o primeiro é que **31.1% das localizações não foram incluídas**. Isso mostra que muitos usuários preferem não colocar a localização. O segundo problema é a formatação desse campo é muito variada, por exemplo, São Paulo aparece em mais de uma representação como pode ser visto da figura 4.5. Isso exige um pré-processamento caso esse dado seja utilizado.

Figura 4.5: Regiões variadas e mal formatadas


Value	Count	Frequency (%)	
Brasil	24417	4.2%	
São Paulo, Brasil	17867	3.1%	
Rio de Janeiro, Brasil	14438	2.5%	
Rio de Janeiro	5940	1.0%	
São Paulo	5612	1.0%	
Belo Horizonte, Brasil	4508	0.8%	
Brasília, Brasil	4503	0.8%	
Brazil	4155	0.7%	
Porto Alegre, Brasil	3551	0.6%	
Portugal	3512	0.6%	
Other values (73697)	311283	53.7%	
(Missing)	180214	31.1%	

Fonte: Próprio autor

Algo que também pode ser avaliado nesse momento são as *hashtags*, embora não sejam utilizadas posteriormente nesse trabalho, elas podem ajudar a revelar algumas tendências nas mensagens.

tweet_hashtags

Figura 4.6: Hashtags do conjunto de dados

tweet_hashtags Categorical HIGH CARDINALITY	Distinct	19950	 531878 ['COVID19'] 1291 ['coronavirus'] 1116 ['Coronavirus'] 634 ['ficaemcasa'] 473 Other values (19945) 44608
	Distinct (%)	3.4%	
	Missing	0	
	Missing (%)	0.0%	
	Memory size	4.4 MiB	

Fonte: Próprio autor

É possível ver que a maioria das mensagens não possui *hashtags*. Outra maneira de se ver essas informações é utilizando uma nuvem de palavras (word cloud) que é uma representação visual de dados de texto, como mostra na figura 4.7.

português. Uma opção alternativa seria desenvolver uma estratégia para classificação de *lexicons* em português, no entanto por uma questão de simplificação do trabalho foi adotado essa abordagem.

Antes de iniciar a tradução os textos, foram realizadas algumas edições nos textos para garantir uma melhor tradução, como a remoção de *hashtags* e a correção de algumas palavras que geralmente são abreviações ou erros de português nas redes sociais.

Tabela 4.1: Palavras Modificadas

Palavra	Abreviação
você	vc
vocês	vcs
não	n, ã
mesmo	msm
ansioso	ancioso
também	tbm
sei lá	sla
que	q
depois	dps
de novo	dnv
com	c
verdade	vdd
muito	mt
nada	nd
estou	to
agora	agr
amigo	amg
quando	qnd
comigo	cmg
muitos	mts
faculdade	facul
pra	p
o que	oq
de	d

Fonte: Próprio autor

Essas foram apenas algumas modificações feitas, o que não reflete a melhor prática para tratar esse problema, mas por questão de simplificação foi realizado dessa forma, pois o não tratamento desse problema pode adicionar algum ruído durante o treinamento. Os tradutores hoje em dia são capazes de realizar correções em palavras por exemplo abreviação “vc”, o google tradutor é capaz de modificar para palavra “você” traduzindo posteriormente para a palavra “you”. No entanto, o tradutor não é capaz de tratar abreviações como “mt=muito, c=com, sla=sei lá”, o que representa um desafio para essa abordagem de tradução de texto, onde o ideal seria utilizar um corretor de textos personalizado.

4.2.2 Tradução dos textos

A segunda etapa foi realizar a tradução dos textos com a biblioteca *Googletrans*, a coluna dos *tweets* traduzidos será chamada de *tweet_textEN*.

4.2.3 Limpeza dos textos traduzidos

A terceira etapa é fazer a limpeza de dados que não são necessários para o treinamento do modelo que podem gerar ruído. As remoções foram com a biblioteca *re* para expressões regular e *unidecode* para acentuações, o texto traduzido depois dessa limpeza será chamado de *clean_textEN*.

- remoção de menções (Exemplo @Usuario2020): (@\w+|@ \w+)
- remoção de números: ([0-9]+)
- remoção de pontuações: (!"#\$%&'()*+,-.:/;=#@[?[\]^_`{|}~]*',')
- remoção de acentuações (Exemplo são paulo -> sao paulo)

Figura 4.8: Exemplos dos textos (*tweet_text*, *tweet_textEN*, *clean_textEN*)

	<i>tweet_text</i>	<i>tweet_textEN</i>	<i>clean_textEN</i>
0	as professoras já se perdiam pra saber se era eu ou a @dudahgnoatto, imagine depois da quarentena kkkkkkkkkkk	the teachers were already lost to know if it was me or @dudahgnoatto, imagine after the quarantine kkkkkkkkkkk	the teachers were already lost to know if it was me or imagine after the quarantine kkkkkkkkkkk
1	estou no último ano do ensino médio se eu não me formar por causa da pandemia vo ficar muito [REDACTED]	I'm in the last year of high school if I don't graduate because of the pandemic I'm going to be really [REDACTED]	Im in the last year of high school if I dont graduate because of the pandemic Im going to be really [REDACTED]
2	lembrando aqui que no início da quarentena eu surtei e raspei a cabeça, até hoje me arrependo kkkk	remembering here that at the beginning of the quarantine I freaked out and shaved my head, until today I regret it kkkk	remembering here that at the beginning of the quarantine I freaked out and shaved my head until today I regret it kkkk
3	enquanto alguns nao estao nem aí para pandemia. como vai ficar a vida meu deus.... crise, fome, desemprego... o relacionamento e a convivência das pessoas estao se distanciando.	while some don't care about a pandemic. how will life be, my god ... crisis, hunger, unemployment ... the relationship and the coexistence of people are distancing.	while some dont care about a pandemic how will life be my god crisis hunger unemployment the relationship and the coexistence of people are distancing
4	o surto desse professor que pediu a cura do covid em uma prova kkkkkkkkkk	the outbreak of this teacher who asked for the cure of covid in a test kkkkkkkkkk	the outbreak of this teacher who asked for the cure of covid in a test kkkkkkkkkk
...

Fonte: Próprio autor

Algumas modificações podem ser notadas na coluna *clean_textEN* exibido figura 4.8 como a remoção das meções e pontuções. Durante esse processo de remoções houve 3 mensagens que só tinham menções “@user”, isso gerou mensagens vazias (NaN) que foram removidas.

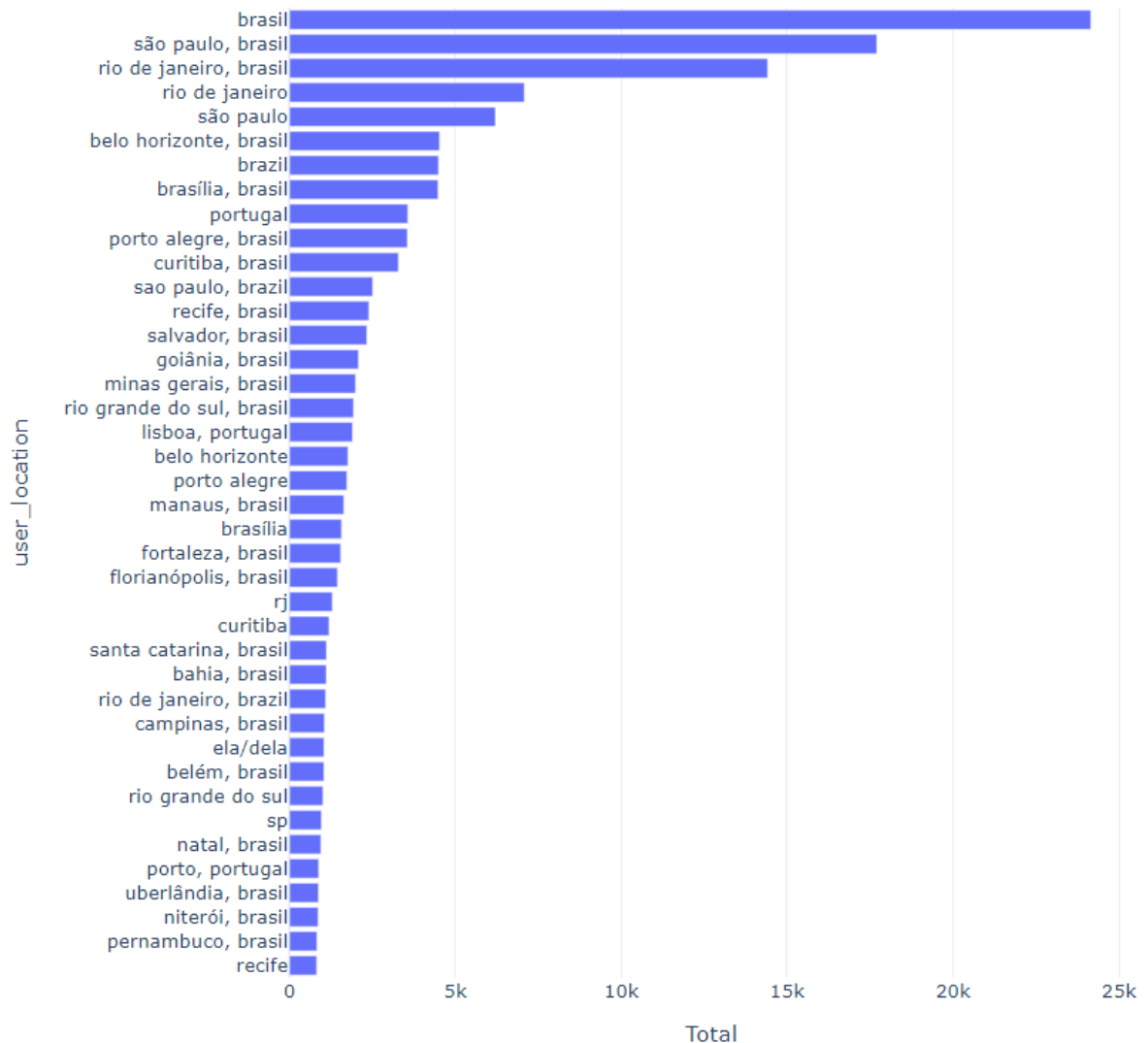
4.2.4 Remoção de tweets duplicados por usuário

A quarta etapa é fazer a remoção dos tweets duplicados por usuários, ou seja, se um usuário postou a mesma mensagem duas vezes. Antes de realizar esse processo foi separado 40 mil *tweets* dos dias 27/09/20 e 28/09/20 que vão ser utilizados para criar um conjunto de dados rotulados, esses dados separados não vão passar pelo processo de remoção de tweets duplicados, pois só existem 465 *tweets* duplicados. Depois da remoção ter sido feita 12764 tweets foram removidos, ficando 527.236 tweets, se for considerar os 40 mil o total de tweets agora é 567.236.

4.2.5 Formatação das localizações dos usuários

A quinta etapa é realizar a formatação das regiões, necessário para permitir que a localização seja utilizada para criar gráficos que compare as classes do *Tone Analyzer* por região. No entanto, essa é uma tarefa complicada, pois como foi visto na figura 4.5 há 31.1% de regiões faltando. Além disso, outros problemas podem ser vistos na figura 4.9 na página seguinte, como valores que não representam regiões, regiões com mais que uma representação (ex: São Paulo, sp, São Paulo brasil), regiões que possuem apenas a cidade (existem cidades com o mesmo nome). Devido a esses problemas a formatação foi simplificada, separando apenas regiões que tinham o estado, enquanto as outras que não foram capazes de serem classificadas foram definidas como OUTROS.

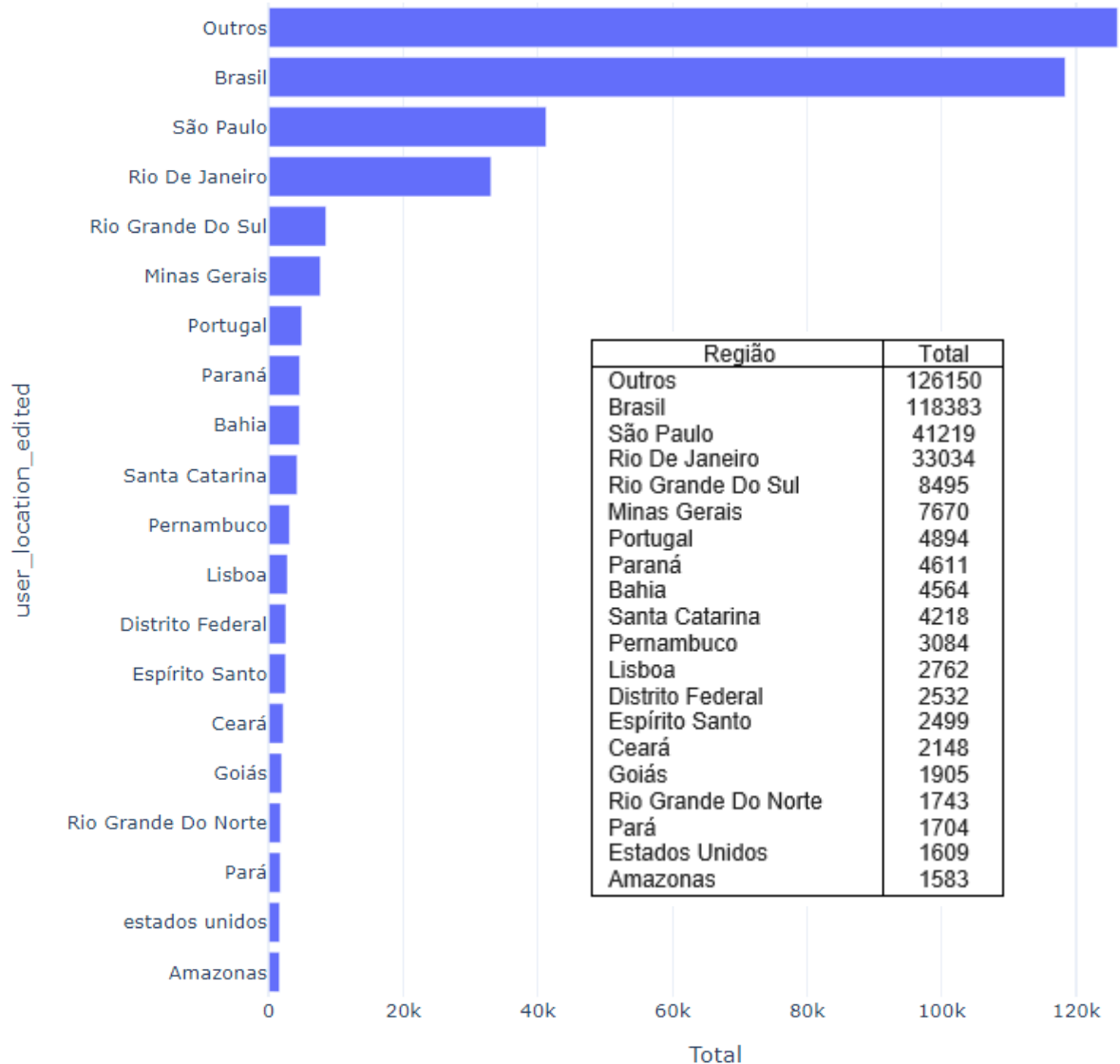
Figura 4.9: As 40 regiões com mais ocorrências não formatadas



Fonte: Próprio autor

Também foram consideradas regiões de Portugal, como pode ser visto na figura 4.10 onde as regiões já estão formatadas. Nessa figura estão exibidas as 20 regiões com maior frequência. Essa pode não ser melhor maneira de tratar os dados, mas como foi dito anteriormente ela foi adotada para simplificar o processo.

Figura 4.10: As 20 regiões com mais ocorrências formatadas



Fonte: Próprio autor

4.2.6 Criar conjunto de dados rotulados

A sexta etapa é preparar os 40 mil dados separados para a classificação de emoção que vão ser utilizados para treinar o modelo de aprendizado de máquina. Os dados separados foram rotulados com a API da IBM *Watson Tone Analyzer* que foi escolhida para essa finalidade por uma questão de simplificação, pois obter dados rotulados geralmente é caro e demorado. Embora também poderia ser utilizado outros conjuntos de dados existentes na internet para o treinamento como o conjunto de dados da competição *Toxic Comment Classification Challenge*²⁴.

²⁴ JIGSAW. Toxic Comment Classification Challenge. **kaggle**, 2017. Disponível em: <<http://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/2---diodo-semicondutor.pdf>>. Acesso em: 18 Novembro 2020.

O *Tone Analyzer* classifica o tom de um texto de entrada em 7 classes que são *Analytical* (Analítico), *Anger* (Raiva), *Confident* (Confiante), *Fear* (Medo), *Tentative* (Hesitante), *Sadness* (Tristeza) e *Joy* (Alegria). No caso dos 40 mil *tweets* o *Tone Analyzer* gerou uma probabilidade de um texto pertencer a cada classe e se não pertencer a uma dessas classes ele não emite uma probabilidade. Em seguida, 7 colunas foram criadas no conjunto de dados. Se o texto tiver mais de 50% de chance de pertencer a uma determinada classe, o valor será atribuído a 1, caso contrário, o valor será atribuído a 0.

Figura 4.11: Exemplo dos rótulos em textos

	tweet_text	clean_textEN	joy	sadness	tentative	analytical	anger	fear	confident
0	as professoras já se perdiam pra saber se era eu ou a @dudahgnoatto, imagine depois da quarentena kkkkkkkkkkk	the teachers were already lost to know if it was me or imagine after the quarantine kkkkkkkkkkk	0.0	1.0	1.0	1.0	0.0	0.0	0.0
1	estou no último ano do ensino médio se eu não me formar por causa da pandemia vo ficar muito [REDACTED]	Im in the last year of high school if I dont graduate because of the pandemic Im going to be really [REDACTED]	0.0	0.0	0.0	1.0	1.0	0.0	0.0
2	lembrando aqui que no início da quarentena eu surtei e raspei a cabeça, até hoje me arrependo kkkk	remembering here that at the beginning of the quarantine I freaked out and shaved my head until today I regret it kkkk	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Fonte: Próprio autor

Como pode ser visto na figura 4.11 o *Tone Analyzer* pode não ser capaz de lidar com casos específicos, pois nunca foi treinado para certos dados como a repetição da letra “k”, o que pode trazer uma ideia de alegria. Também é possível ver que quando um texto não pertencer a nenhuma classe ele será classificado como 0 em todas as classes.

Uma visão geral da frequência dos dados rotulados pode ser vista na tabela 4.2, levando em consideração que um texto pode ter mais que uma classe, então o valor mostrado na tabela é o valor total individual de cada classe, além disso o valor **none** seria os textos classificados como não pertencendo a nenhuma das classes, ou seja, o rótulo é [0, 0, 0, 0, 0, 0, 0].

Tabela 4.2: Total da classificação no conjunto de treinamento

Classificação	Total
none	12209
analytical	11306
sadness	8766
joy	6331
tentative	5456
confident	2940
anger	2378
fear	1252

Fonte: Próprio autor

4.3 CLASSIFICAÇÃO DE POLARIDADE E SUBJETIVIDADE COM TEXTBLOB

A classificação da polaridade foi realizada com o TextBlob, que possui um analisador de sentimento que retorna dois valores para um texto de entrada. O primeiro é a polaridade, que é um ponto flutuante entre $[-1,1]$, onde -1 significa sentimento negativo e $+1$ significa sentimento positivo. O segundo é a subjetividade, que também é um ponto flutuante entre $[0,1]$, em que $0,0$ é muito objetivo (fatos) e $1,0$ é muito subjetivo (sentimentos pessoais).

Para a apresentação do TextBlob foi feito uma simplificação em relação a polaridade que foi: considerar valores abaixo de -1 como negativos, valores acima de 0 são positivos enquanto valores iguais a 0 são neutros. Em relação a subjetividade foi considerado que valores maiores ou iguais 0.5 são subjetivos enquanto abaixo de 0.5 são objetivos.

Figura 4.12: Exemplos de tweets classificados como positivos

tweet_text	clean_textEN	polarity	subjectivity
preocupado se estou tendo sintomas de coronavirus ao mesmo tempo em que acendo outro delicioso cigarro	worried if i am having coronavirus symptoms while lighting another delicious cigarette	1.0	1.0
apesar da pandemia meu aniversário esse ano foi um dos melhores de todos	despite the pandemic my birthday this year was one of the best of all	1.0	0.3
@jairbolsonaro você usou isso em detrimento das pessoas como retórica para que os cidadãos não se isolassem diante da maior pandemia que esse mundo já teve. canalha!	you used this to the detriment of people as a rhetoric so that citizens do not isolate themselves in the face of the greatest pandemic this world has ever had scoundrel	1.0	1.0
eu no maior tédio sem sentir gosto de comida é foda, covid do [REDACTED]	me in the greatest boredom without feeling like food is badass covid do [REDACTED]	1.0	1.0
o mark falando que a casa da jennifer é o melhor lugar ora passar a quarentena eu queria vey	mark saying that jennifers house is the best place now to quarantine i wanted vey	1.0	0.3
sair do insta por tempo indeterminado foi a melhor coisa que fiz nessa quarentena	leaving the insta indefinitely was the best thing i did in this quarantine	1.0	0.3
sob pressão plantão covid vai ser foda demais	under pressure covid on duty will be awesome	1.0	1.0
o melhor bolo para quem fez aniversário na quarentena	the best cake for those who had a birthday in the quarantine	1.0	0.3
2021 + bts no brasil é a melhor combinação do mundo se essa pandemia passar	bts in brazil is the best combination in the world if this pandemic passes	1.0	0.3
pandemia não existe, esse mídia são foda	pandemic does not exist this media is awesome	1.0	1.0

Fonte: Próprio autor

A figura 4.12 mostra alguns exemplos de *tweets* positivos, é possível ver na linha 4 uma **tarja preta** onde foi omitido uma palavra ofensiva que estava abreviada e não foi feito o tratamento. Se tivesse sido feito o tratamento a tradução correta poderia trazer um grau mais negativo a frase, isso mostra a dificuldade de filtrar manual abreviações nas mídias sociais. Também é possível ver que o classificador é capaz de realizar alguns acertos como visto na linha 2 onde deve ter identificado a parte “best” como positivo. No entanto o classificador também possui uma margem de erro que pode ser percebido na linha 3, o qual traz um sentido negativo, mas devido a palavra “greatest” o classificador entendeu como positivo.

Figura 4.13: Exemplos de tweets classificados como negativo

tweet_text	clean_textEN	polarity	subjectivity
estou furando quarentena vindo no dentista gente quem vem me cancelar	Im boring quarantine coming to the dentist people who come to cancel me	-1.0	1.0
reinaldo pós pandemia voltou a ser o péssimo reinaldo 2013-2015	post pandemic reinaldo was again the terrible reinaldo	-1.0	1.0
no final, furar a quarentena foi uma questão de quem aguenta/aguentou por mais tempo. uma hora ou outra todo mundo vai acabar furando.	in the end sticking to the quarantine was a matter of who can stand it eventually everybody will end up boring	-1.0	1.0
estou com uma dor horrível no corpo, não sei se é covid dengue ou estresse	I have a horrible pain in my body I dont know if its covid dengue or stress	-1.0	1.0
@hlfloret agora que estão com covid estão preocupados. são muito nojentos. além disso a torcida ainda usa esse discurso pra defender esses genocidas mercenários mulambus.	now that they are with covid they are worried they are very disgusting in addition the fans still use this speech to defend these genocidal mulambus mercenaries	-1.0	1.0
isto num período que apanhou os 3 piores meses da pandemia e se cumpriu com todos os compromissos assumidos! saia lá mais um caso contra benfica para abafar este resultado estrondoso.	this in a period that caught the worst months of the pandemic and was fulfilled with all the commitments assumed there is yet another case against benfica to stifle this resounding result	-1.0	1.0
@testiculo_de @changemy_mind_ @leticiashirakin e é obrigado a ir em debate ? convenhamos que é chatão	and you have to go into debate lets say its boring	-1.0	1.0
mercado explodiu no pior pib da história dos us, mas cai quando o presidente pega covid.. 🤮🤮	The market exploded at the worst gib in the history of but it falls when the president takes covid	-1.0	1.0
gente quero pedir orações por favor 😊 minha tia está pessima no hospital com covid!!!! por favor	people I want to ask for prayers please my aunt is terrible at the hospital with covid please	-1.0	1.0
eles inventam que as pessoas estão com covid e simplesmente matam elas?? que horror	do they invent that people have covid and just kill them how horrible	-1.0	1.0

Fonte: Próprio autor

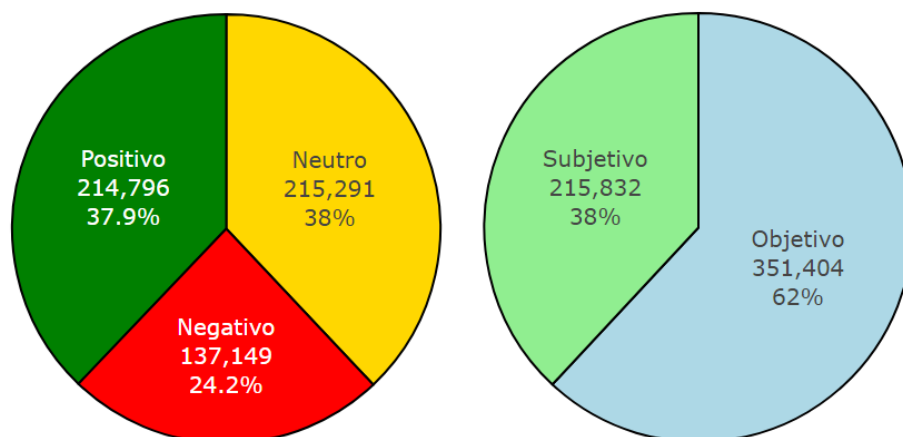
Figura 4.14: Exemplos de tweets classificados como neutro

tweet_text	clean_textEN	polarity	subjectivity
antes da quarentena // na quarentena só fiquei ruiva e arranjei um trabalho	before the quarantine in the quarantine i just got red and got a job	0.0	0.000000
e de novo hoje. até as 20h58 de 5 de outubro de 2020. o painel rio com dados sobre a pandemia do na cidade do rio está ainda em atualização.	and again today until pm on October the rio panel with data on the pandemic in the city of rio is still being updated	0.0	0.000000
meu spotify resume como eu pirei nesses últimos dias de quarentena	my spotify sums up how i freaked out these last quarantine days	0.0	0.066667
@arsenal_brasil até estranhei o kroenke liberar essa contratação em plena pandemia, agora é torcer pra dar certo	I even thought it was kroenke to release this contract in the middle of a pandemic now it is hoping to work	0.0	0.000000
sequelas da covid-19 sobrecarregam o sus e entidades cobram orçamento maior	sequels of covid overload the sus and entities charge larger budget	0.0	0.500000
@lucaswebero atual campeão turco e na fase de grupos da champions, acho que até sem pandemia ele não viria	current Turkish champion and in the group stage of the champions I think that even without a pandemic he would not come	0.0	0.400000
@thalesrassi sdd papa br session quando acabar a quarentena	sdd papa br session when quarantine ends	0.0	0.000000
antes da quarentena / agora só aprendi a me maquiar e mudei a cor do cabelo	before quarantine now i just learned to put on makeup and changed my hair color	0.0	0.000000
@preconceitu não te preocupes que ele não tem covid, isto e tudo campanha.	dont worry that he doesnt have covid this and everything campaign	0.0	0.000000
há um provérbio africano que diz que é preciso uma aldeia para se educar uma criança. em tempos de pandemia, fica ainda mais claro que aldeia é muito mais que escola > família. a aldeia está no caminho que a criança faz de casa à escola. todos são responsáveis pelo conhecimento.	ha um proverbio africano que diz que e preciso uma aldeia para se educar uma criança em tempos de pandemia fica ainda mais claro que aldeia e muito mais que escola gt familia a aldeia esta no caminho que a crianca faz de casa a escola todos sao responsaveis pelo conhecimento	0.0	0.000000

Fonte: Próprio autor

Apesar de alguns problemas que podem ser vistos nessas classificações, o classificador mostrou ser capaz de identificar textos como positivo e negativo corretamente. Algumas modificações futuras, poderia melhorar a precisão como a utilização do corretor ortográfico personalizado e a modificações de partes da tradução através de algoritmos de ML. A figura 4.15 mostra uma visão geral dos dados.

Figura 4.15: Classificação total da polaridade e subjetividade



Fonte: Próprio autor

Os gráficos da classificação total da polaridade mostram que o classificador identificou mais tweets positivos, o que pode trazer um indicio de que as pessoas

4.4.1 Entradas

Os 40 mil dados rotulados foram divididos em três conjuntos para o treinamento do modelo que são o conjunto de treinamento (28000), conjunto de validação (7800) e conjunto de teste (4200). Cada conjunto possui dois valores, um valor “**X**” (X_train, X_valid e X_test) que representa as estradas (*tweets*) e um valor “**y**” (y_train, y_valid e y_test) que se refere aos rótulos de cada entrada.

Tabela 4.3: Resumo da distribuição individual entre os três conjunto de dados

Classificação	y_train	y_valid	y_test
none	8622	2351.0	1236.0
analytical	7857	2246.0	1203.0
sadness	6155	1691.0	920.0
joy	4412	1247.0	672.0
tentative	3860	1053.0	543.0
confident	2038	599.0	303.0
anger	1648	465.0	265.0
fear	871	242.0	139.0

Fonte: Próprio autor

Antes de iniciar a criação do modelo é necessário fazer a vetorização das entradas para isso foi feito a utilização da classe *TextVectorization* com base no código fonte encontrado no site do keras²⁶. A seguir pode ser visto o exemplo de um texto vetorizado na figura 4.19.

Figura 4.19: Exemplo de texto vetorizado

```
output = vectorizer(['the teachers were already lost to know if it was me or imagine after the quarantine kkkkkkkkkk'])
output.numpy()

array([[ 2, 775, 80, 64, 228, 3, 62, 32, 14, 20, 31,
        68, 330, 65, 2, 13, 1353, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Fonte: Próprio autor

Como pode ser visto na figura 4.19, a função *vectorizer* ela transforma um texto em um tensor de inteiros, no caso há um processo anterior em que é criado um

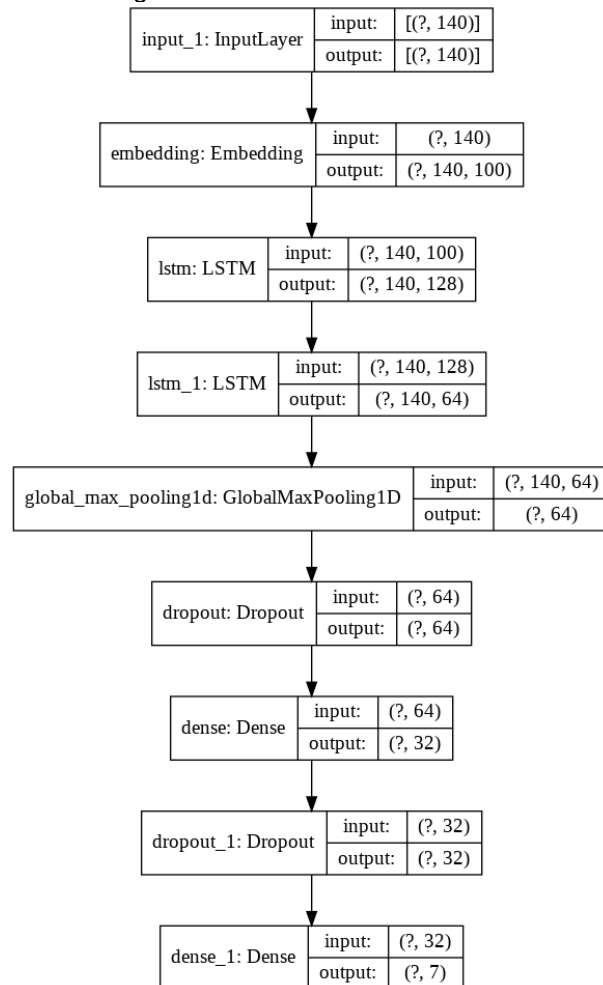
²⁶ CHOLLET, F. Using pre-trained word embeddings. **keras**, 5 mai 2020. Disponível em: <https://keras.io/examples/nlp/pretrained_word_embeddings/>. Acesso em: 19 Novembro 2020.

vocabulário com um número definido das principais palavras do **conjunto de treinamento**, nesse caso foi escolhido 20 mil palavras. Além disso, como pode ser visto o vetor imprimido possui um tamanho de 140, onde cada um elemento do vetor é um *token* (palavra do texto) associado a palavra uma palavra das 20 mil palavras vocabulário definido. Os valores com o dígito 0 são para preencher a matriz caso o texto não alcance 140 palavras isso é chamado *padding*, esse processo de vetorização é aplicado nos textos dos três conjuntos de entrada (X_train, X_valid e X_test).

Depois que as entradas são vetorizadas, é necessário criar uma camada de *embedding* para a rede neural, essa camada de é criada ao carregar o *word embedding* pré-treinado, no caso o GloVe isso pode ser feito baixando um arquivo que contém vetores codificados de textos em vários tamanhos, nesse caso foi escolhido a matriz *glove.6B.100d*, A camada de embeddings possui um valor chamado *trainable* que é definido como *false* para evitar que a camada seja treinada e perca os pesos pré-treinados.

4.4.2 Camadas

Figura 4.20: Estrutura do modelo utilizado



Fonte: Próprio autor

A figura 4.20 mostra a estrutura utilizada para o modelo *Keras*, ela é uma estrutura simples, e pode não refletir a melhor solução para o problema, mas será o suficiente para o desenvolvimento do projeto. A seguir é mostrado o resumo do modelo na figura 4.21, e logo em seguida há a descrição de cada camada utilizada no modelo.

Figura 4.21: Resumo do modelo

Model: "functional_1"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 140)]	0
embedding (Embedding)	(None, 140, 100)	2000200
lstm (LSTM)	(None, 140, 128)	117248
lstm_1 (LSTM)	(None, 140, 64)	49408
global_max_pooling1d (Global	(None, 64)	0
dropout (Dropout)	(None, 64)	0
dense (Dense)	(None, 32)	2080
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 7)	231
Total params: 2,169,167		
Trainable params: 168,967		
Non-trainable params: 2,000,200		

Fonte: Próprio autor

Explicação das camadas:

1ª Camada de entrada: A primeira camada recebe as entradas que são as listas dos textos codificados possuindo uma dimensão de 140 onde cada é uma palavra.

2ª Camada de *Embeddings*: A segunda camada é carregada com o GloVe (glove.6B.100d), possuindo 3 argumentos que são `input_dim=20002` que representa o tamanho do vocabulário definido no início da vetorização de textos, `output_dim=100` é a dimensão da camada que é igual a dimensão da matriz do GloVe, `input_length=140` que é o comprimento das sequências de entrada, nesse caso também são 140 unidades ocultas na camada onde cada representa uma palavra. A camada possui 2 milhões (20002×100) de parâmetros (os parâmetros são os pesos em cada camada) não treináveis, pois foi definido no início da criação do modelo *keras* (*trainable=False*).

3ª Camada LSTM: Essa camada é uma LSTM (Long short-term memory) que é o tipo de uma rede neural recorrente artificial (Recurrent neural network – RNN). Uma RNN funciona ao alimentar recursivamente a saída da rede anterior na entrada da rede atual, onde após um número N de recursões é obtido a saída final. Embora em alguns casos seja melhor alterar esse comportamento, permitindo que a RNN transfira cada saída recursiva para a próxima camada, este processo é chamado de *unrolled network*

²⁷. O modelo do Keras pode funcionar dessa maneira ao definir o argumento da camada LSTM *return_sequences* como True. A camada possui 117.248 parâmetros treináveis ($4 \times [128(128+100) + 128]$).

4ª Camada LSTM: Outra camada LSTM com o número de unidades reduzida para 64, também possui *return_sequences* definido como True, a camada tendo 49.408 parâmetros treináveis ($4 \times [64(64+128) + 64]$).

5ª Camada GlobalMaxPooling1D: A camada de pooling geralmente é utilizada em CNNs para reduzir a dimensionalidade dos dados de imagens, mas nesse caso ela será utilizada para remodelar o tensor de entrada de 3 dimensões para 2 dimensões, além disso será pego os valores máximos de cada patch do tensor.

6ª Camada de *Dropout*: Essa camada aplica a técnica de regularização *dropout* com uma taxa de 20% (0,2), fazendo com que algumas unidades de saída da camada sejam descartadas, obrigando a próxima camada a lidar com a representação dos dados descartados, o que pode resultar em uma melhor generalização e ajuda a reduzir o overfitting.

7ª Camada densa: Essa é uma camada densa, ou seja, uma camada de RNA totalmente conectada onde cada unidade (Nó) de entrada é conectado a unidade de saída. Esta camada recebe os valores da camada de *dropout*, possui 32 unidades e aplica a função RELU para os valores da saída, o número de parâmetros na camada são 2080 parâmetros treináveis ($32 \times (64+1)$).

8ª Camada de *Dropout*: Outra camada de *dropout* com a mesma taxa de 20% (0.2).

9ª Camada densa: A última camada aplica uma função sigmoide que é adequada para o problema de multirrótulos, pois assim vai tratar como uma distribuição de probabilidade Bernoulli onde a saída final é um vetor de 7 valores em que cada valor é a probabilidade entre os limites de 0 a 1 de um texto pertencer a uma das classes.

4.4.3 Compilação do Modelo

Antes de iniciar o treinamento, as configurações de treinamento devem ser especificadas, desta forma a função de perda, otimizador e métricas são definidos

²⁷ GÉRON, A. Processing Sequences Using RNNs and CNNs. In: **GÉRON, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2ª. ed. Sebastopol: O'Reilly Media Inc., 2019. Cap. 15.

para a compilação do modelo. A escolha do otimizador foi o Adam mantendo a taxa de aprendizado padrão (0,001). A função de perda escolhida foi `binary_crossentropy`³⁰, por ser uma classificação binária é adequada para problemas de vários multirrótulo, a métrica definida foi `binary_accuracy`.

4.4.4 Treinamento do Modelo

O treinamento de um modelo *keras* é realizado através de lotes (*batches*) que é o número de amostras processadas antes da atualização do modelo. Esse processo é feito através de interações chamada de épocas que são o número de passagens completas pelo conjunto de dados de treinamento. O *keras* oferece um recurso chamado de *callbacks* que é um objeto capaz de realizar ações que alteram o modelo *Keras* durante o processo de treinamento. Sendo que essas alterações podem ser no início ou no final de uma época, antes ou depois de um único lote.²⁸

Antes de iniciar o treinamento é criado um conjunto de 4 *callbacks* que são: *Model checkpointing* responsável por salvar os pesos atuais do modelo ao monitorar se o `val_loss` foi reduzido. *Early stopping* responsável por interromper o treinamento quando a perda de validação não melhorar mais, realizando o salvamento do melhor modelo obtido durante o treinamento. Por último há um *callback* personalizado chamado *RocAucScore* que utiliza a biblioteca para o calcula do score da métrica AUC-ROC e exibe o valor durante o treinamento.

O treinamento do modelo *keras* é realizado ao chamar a função `fit()` que recebe os conjuntos de treinamento (28000) e validação (7800), de maneira que é feito uma interação por lotes (`batch_size=219`) durante um determinado números de épocas (`Epoch=15`). O treinamento acontece durante 15 épocas para cada é exibido os valores acurácia (`binary_accuracy` e `- val_binary_accuracy`), perda (`loss` e `val_loss`) e ROC-AUC. Durante o treinamento os *callbacks* *Model checkpointing* e *Early stopping* são ativados caso o valor da perda no conjunto de validação (`val_loss`) seja reduzido ou não. A seguir a tabela 4.4 mostra o resumo do treinamento em cada época.

Tabela 4.4: Resumo do treinamento

Epoch 1/15 219/219 [=====] - ETA: 0s - loss: 0.3883 - binary_accuracy: 0.8578 ROC-AUC - epoch: 1 - score: 0.577799 Epoch 00001: val_loss did not improve from 0.20041 219/219 [=====] - 50s 227ms/step - loss: 0.3883 - binary_accuracy: 0.8578 - val_loss: 0.3664 - val_binary_accuracy: 0.8618

²⁸ CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 249 p.

Epoch 2/15
 219/219 [=====] - ETA: 0s - loss: 0.3726 - binary_accuracy: 0.8629
 ROC-AUC - epoch: 2 - score: 0.628539
 Epoch 00002: val_loss did not improve from 0.20041
 219/219 [=====] - 46s 212ms/step - loss: 0.3726 - binary_accuracy: 0.8629 - val_loss: 0.3588 - val_binary_accuracy: 0.8621

Epoch 3/15
 219/219 [=====] - ETA: 0s - loss: 0.3454 - binary_accuracy: 0.8647
 ROC-AUC - epoch: 3 - score: 0.762425
 Epoch 00003: val_loss did not improve from 0.20041
 219/219 [=====] - 46s 212ms/step - loss: 0.3454 - binary_accuracy: 0.8647 - val_loss: 0.3146 - val_binary_accuracy: 0.8684

Epoch 4/15
 219/219 [=====] - ETA: 0s - loss: 0.3058 - binary_accuracy: 0.8743
 ROC-AUC - epoch: 4 - score: 0.831919
 Epoch 00004: val_loss did not improve from 0.20041
 219/219 [=====] - 46s 211ms/step - loss: 0.3058 - binary_accuracy: 0.8743 - val_loss: 0.2835 - val_binary_accuracy: 0.8863

Epoch 5/15
 219/219 [=====] - ETA: 0s - loss: 0.2697 - binary_accuracy: 0.8917
 ROC-AUC - epoch: 5 - score: 0.878957
 Epoch 00005: val_loss did not improve from 0.20041
 219/219 [=====] - 46s 212ms/step - loss: 0.2697 - binary_accuracy: 0.8917 - val_loss: 0.2430 - val_binary_accuracy: 0.9021

Epoch 6/15
 219/219 [=====] - ETA: 0s - loss: 0.2403 - binary_accuracy: 0.9045
 ROC-AUC - epoch: 6 - score: 0.906563
 Epoch 00006: val_loss did not improve from 0.20041
 219/219 [=====] - 46s 212ms/step - loss: 0.2403 - binary_accuracy: 0.9045 - val_loss: 0.2182 - val_binary_accuracy: 0.9120

Epoch 7/15
 219/219 [=====] - ETA: 0s - loss: 0.2157 - binary_accuracy: 0.9150
 ROC-AUC - epoch: 7 - score: 0.923608
 Epoch 00007: val_loss did not improve from 0.20041
 219/219 [=====] - 47s 212ms/step - loss: 0.2157 - binary_accuracy: 0.9150 - val_loss: 0.2016 - val_binary_accuracy: 0.9197

Epoch 8/15
 219/219 [=====] - ETA: 0s - loss: 0.1975 - binary_accuracy: 0.9219
 ROC-AUC - epoch: 8 - score: 0.929805
 Epoch 00008: val_loss improved from 0.20041 to 0.18933, saving model to weights.hdf5
 219/219 [=====] - 47s 213ms/step - loss: 0.1975 - binary_accuracy: 0.9219 - val_loss: 0.1893 - val_binary_accuracy: 0.9251

Epoch 9/15
 219/219 [=====] - ETA: 0s - loss: 0.1794 - binary_accuracy: 0.9296
 ROC-AUC - epoch: 9 - score: 0.937849
 Epoch 00009: val_loss improved from 0.18933 to 0.17888, saving model to weights.hdf5
 219/219 [=====] - 47s 213ms/step - loss: 0.1794 - binary_accuracy: 0.9296 - val_loss: 0.1789 - val_binary_accuracy: 0.9283

Epoch 10/15
 219/219 [=====] - ETA: 0s - loss: 0.1673 - binary_accuracy: 0.9354
 ROC-AUC - epoch: 10 - score: 0.940087
 Epoch 00010: val_loss did not improve from 0.17888


```

219/219 [=====] - 47s 213ms/step - loss: 0.1673 - binary_accuracy:
0.9354 - val_loss: 0.1793 - val_binary_accuracy: 0.9296

Epoch 11/15
219/219 [=====] - ETA: 0s - loss: 0.1527 - binary_accuracy: 0.9416
ROC-AUC - epoch: 11 - score: 0.940920
Epoch 00011: val_loss improved from 0.17888 to 0.17556, saving model to weights.hdf5
219/219 [=====] - 46s 212ms/step - loss: 0.1527 - binary_accuracy:
0.9416 - val_loss: 0.1756 - val_binary_accuracy: 0.9305

Epoch 12/15
219/219 [=====] - ETA: 0s - loss: 0.1436 - binary_accuracy: 0.9449
ROC-AUC - epoch: 12 - score: 0.944400
Epoch 00012: val_loss improved from 0.17556 to 0.17285, saving model to weights.hdf5
219/219 [=====] - 47s 214ms/step - loss: 0.1436 - binary_accuracy:
0.9449 - val_loss: 0.1729 - val_binary_accuracy: 0.9329

Epoch 13/15
219/219 [=====] - ETA: 0s - loss: 0.1285 - binary_accuracy: 0.9514
ROC-AUC - epoch: 13 - score: 0.943863
Epoch 00013: val_loss did not improve from 0.17285
219/219 [=====] - 47s 213ms/step - loss: 0.1285 - binary_accuracy:
0.9514 - val_loss: 0.1749 - val_binary_accuracy: 0.9318

Epoch 14/15
219/219 [=====] - ETA: 0s - loss: 0.1162 - binary_accuracy: 0.9560
ROC-AUC - epoch: 14 - score: 0.946058
Epoch 00014: val_loss did not improve from 0.17285
219/219 [=====] - 47s 213ms/step - loss: 0.1162 - binary_accuracy:
0.9560 - val_loss: 0.1751 - val_binary_accuracy: 0.9349

Epoch 15/15
219/219 [=====] - ETA: 0s - loss: 0.1061 - binary_accuracy: 0.9607
ROC-AUC - epoch: 15 - score: 0.943163
Epoch 00015: val_loss did not improve from 0.17285
219/219 [=====] - 47s 213ms/step - loss: 0.1061 - binary_accuracy:
0.9607 - val_loss: 0.1916 - val_binary_accuracy: 0.9323

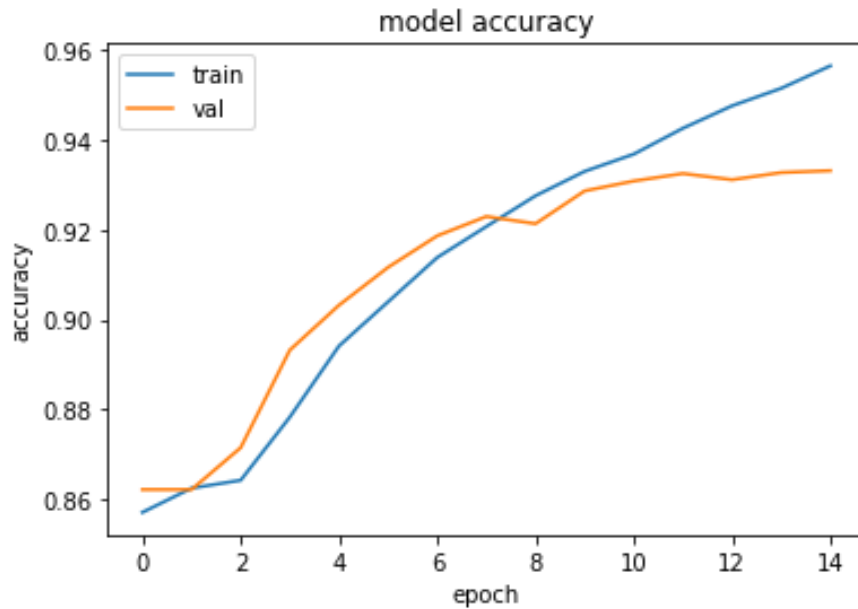
```

Fonte: Próprio autor

No final do treinamento a função `fit()` gera um objeto chamado *history* que registra os valores de perda e métricas usadas durante o treinamento, podendo ser visualizado nos gráficos a seguir.

4.4.5 Avaliação do modelo e predições

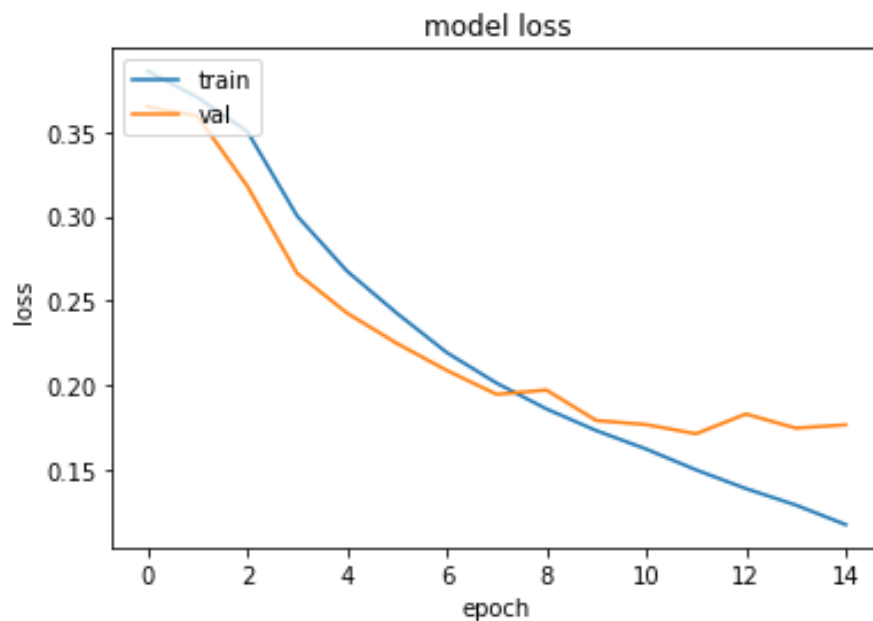
Figura 4.22: *Accuracy* do treinamento e validação



Fonte: Próprio autor

A partir do gráfico de *Accuracy* (figura 4.22), pode ser visto que o modelo começa ter overffiting a partir da 7 época, pois a *accuracy* do conjunto de treinamento começa a superar a do conjunto de validação, porem a acurácia no conjunto de validação melhorou um pouco até chegar na época 15 onde começar ter uma perda de *accuracy*.

Figura 4.23: Perda do treinamento e validação



Fonte: Próprio autor

O gráfico da perda (figura 4.23) mostra que o modelo provavelmente não irá melhorar mais a partir da oitava época, então o ideal seria carregar o modelo a partir da época 12 onde apresenta um valor melhor da perda no conjunto da validação (*val_loss*), mesmo possuindo um pouco de *Overfitting*, como o callback da *ModelCheckpoint* para antecipada salvou os pesos na décima segunda época, eles serão utilizados para construção do modelo a seguir.

Depois que o modelo está treinado é ideal que seja feita uma avaliação do desempenho do modelo para saber sua eficácia em dados que não foram vistos durante o treinamento, nesse caso conjunto de teste (4200) separado no início serve para essa finalidade. Para isso foi utilizado duas métricas a primeira é a matriz de confusão e a segunda métrica é a área sob a curva (Area Under Curve - AUC) da característica de operação do receptor (Receiver Operating Characteristic - ROC) chamado de AUC-ROC.

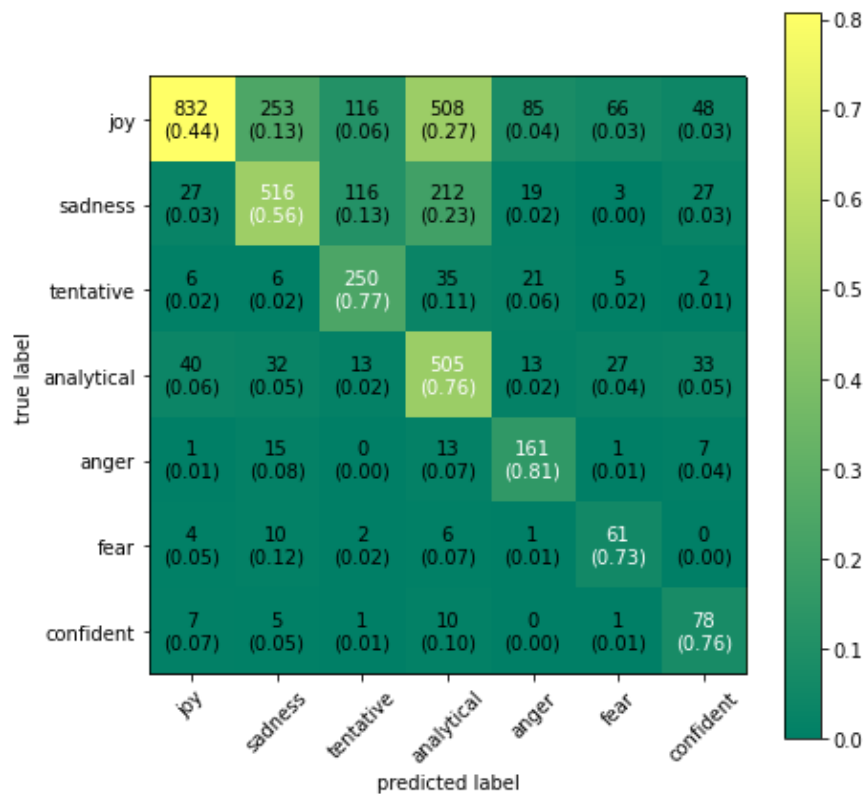
A matriz de confusão é uma matriz bidimensional que permite visualizar o número de classificações corretas ou incorretas para cada classe, de maneira que permita verificar o desempenho do modelo, além de permitir calcular outras métricas, como *accuracy*, *precision*, *recall*, *specificity*, *F1-Score* etc. Existem 4 valores principais na matriz de confusão, verdadeiro positivo (*true positive* – TP), falso positivo (*false positive* – FP), verdadeiro negativo (*true negative* – TN) e falso negativo (*false negative* – FN).

Descrição

- *True Positive* (TP): O classificador previu corretamente a classe como positivo.
- *True Negative* (TN): O classificador previu corretamente a classe como negativo.
- *False Positive* (FP): O classificador previu incorretamente a classe negativa como positivo.
- *False Negative* (FN): O classificador previu incorretamente a classe positiva como negativo.

Quando a matriz de confusão possui múltiplas classes os valores TP, TN, FP e FN devem ser calculados individualmente para cada classe.

Figura 4.24: Matriz de confusão



Fonte: Próprio autor

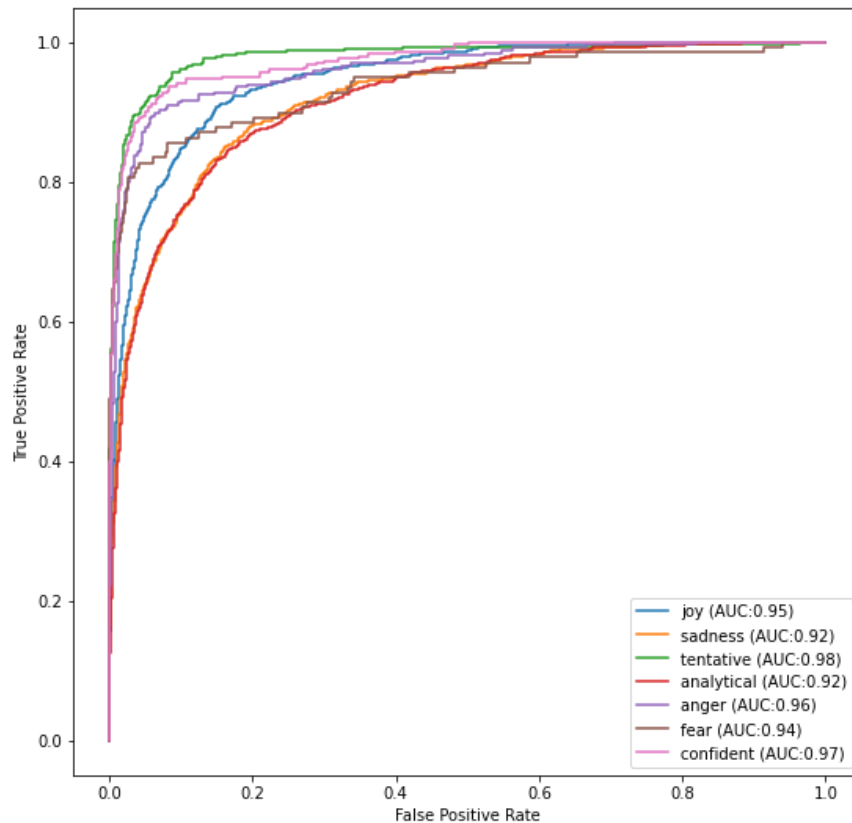
Na figura 4.24 é possível ver que a classe *joy* teve 832 rótulos são classificados corretamente (TP) enquanto 85 (27+6+40+1+4+7) foram classificados em outras classes (FN), além disso ela possui cerca de 1076 (253+116+508+85+66+48) rótulos de outras classes sendo classificado como *joy* (FP). Isso mostra que o modelo possui alguma capacidade de generalização para o projeto, no entanto o rótulo *joy* mostra ter um forte fator relação as outras classes, o que pode ser ruim para o modelo.

O ideal seria ter uma diagonal com um alto índice de classificações, o que representaria que houve mais classificações sendo feitas corretamente (TP), mas como pode ser visto algumas classes não foram tão bem classificadas por exemplo a classe *analytical* que classificou 508 rótulos erroneamente como *joy*. Para ter uma visão melhor dos resultados da matriz de confusão a utilização da métrica AUC–ROC pode ajudar a compreender melhor a classificações das **taxas de verdadeiro positivo** contra **taxa de falso positivo**.

A curva ROC é gráfico que ajuda a compreender o desempenho do modelo de classificação em todos os limites de classificação, onde o gráfico é feito com a taxa positivo verdadeiro contra a taxa falso positivo. A AUC é uma medida para a do classificador em diferenciar entre as classes positivas e negativas, quanto mais alto a

AUC melhor será a capacidade do modelo na diferenciação das classes positivas e negativas. A combinação das duas métricas AUC-ROC, oferece uma medida de desempenho em todos os limites de classificação. Basicamente pode se dizer que se o valor da AUC está o mais próximo de 1 melhor será a capacidade do modelo, se estiver em 0.5 o modelo não possuirá nenhuma habilidade, e se estiver abaixo de 0.5 significa que o modelo não tem nenhuma habilidade para realizar classificação.

Figura 4.25: Classes com a métrica AUC-ROC



Fonte: Próprio autor

A figura 4.25 exibe que o modelo possui alguma capacidade de distinguir entre classes, tendo destaque para as classes *tentative* e *confident* que possuem poucos dados no conjunto de teste, mostrando que são casos mais fáceis de serem classificados. Enquanto a classe *analytical* mostra um valor menor embora tenha mais exemplos no conjunto geral, mostrando que pode haver alguma dificuldade para o modelo em lidar com o problema desse tipo, onde talvez seja necessária futura modificações.

Depois verificar o desempenho das métricas, ver alguns exemplos dos textos classificados do conjunto de teste pode ser interessante, a seguir separei alguns exemplos para observações.

Figura 4.26: Exemplo 1 do conjunto de teste

```
impossível continuar a faculdade na pandemia sem fazer tratamento psicológico
impossible to continue college in the pandemic without undergoing psychological treatment
joy sadness tentative analytical anger fear confident
0.0      1.0      0.0      1.0      0.0      0.0      1.0
```

- joy, Prediction: 0.04%
- sadness, Prediction: 97.92%
- tentative, Prediction: 0.47%
- analytical, Prediction: 1.36%
- anger, Prediction: 0.01%
- fear, Prediction: 0.10%
- confident, Prediction: 0.30%

Fonte: Próprio autor

A figura 4.26 exibe o texto em português não formatado, o texto em inglês formatado do conjunto de teste, os rótulos originais conseguidos pelo *tone analyzer* e as probabilidades que o modelo identificou para o texto em inglês. O modelo foi capaz de identificar o primeiro valor “sadness”, mas não conseguiu identificar o valor “analytical” e “confident”, pela ideia que a mensagem passa pode ser dito que o *tone analyzer* classificou bem a mensagem já que “impossível” pode trazer um tom de certeza “confident”.

Figura 4.27: Exemplo 2 do conjunto de teste

```
acho que a pandemia acabou e não me avisaram
I think the pandemic is over and I was not warned
joy sadness tentative analytical anger fear confident
0.0      0.0      0.0      1.0      0.0      0.0      0.0
```

- joy, Prediction: 87.42%
- sadness, Prediction: 1.02%
- tentative, Prediction: 76.34%
- analytical, Prediction: 92.87%
- anger, Prediction: 0.10%
- fear, Prediction: 0.04%
- confident, Prediction: 0.24%

Fonte: Próprio autor

No exemplo da figura 4.27 o modelo classificou o texto ainda em duas classes diferente do *tone analyzer*, pode ser dito que a mensagem possui um tom de humor difícil de ser identificado, que não foi identificado pelo *tone analyzer*, embora o modelo tenha classificado como 87%, não creio que tenha realmente identificado isso, embora o texto também demonstre uma ideia de incerteza (*tentative*) com a palavra “acho”, o que talvez traga sim algum indicio de que o modelo possa ter encontrado outros padrões que o *tone analyzer* não tenha conseguido

Figura 4.28: Exemplo 3 no conjunto de teste

medo de me apaixonar na quarentena
 fear of falling in love in quarantine
 joy sadness tentative analytical anger fear confident
 0.0 0.0 0.0 0.0 0.0 1.0 0.0

- joy, Prediction: 0.01%
- sadness, Prediction: 39.88%
- tentative, Prediction: 0.04%
- analytical, Prediction: 0.40%
- anger, Prediction: 0.03%
- fear, Prediction: 0.10%
- confident, Prediction: 0.13%

Fonte: Próprio autor

Nesse caso da figura 4.28 o modelo não foi capaz de identificar uma classe com muita exatidão, enquanto o tone analyzer identificou como *fear*.

Figura 4.29: Exemplo 4 do conjunto de teste

o morango tá branco pq deixei de molho no álcool pra tira o covid
 the strawberry is white because I let it soak in alcohol to remove the covid
 joy sadness tentative analytical anger fear confident
 0.0 0.0 0.0 1.0 0.0 0.0 0.0

- joy, Prediction: 1.42%
- sadness, Prediction: 0.84%
- tentative, Prediction: 0.36%
- analytical, Prediction: 91.19%
- anger, Prediction: 0.03%
- fear, Prediction: 0.03%
- confident, Prediction: 0.17%

Fonte: Próprio autor

A figura 4.29 mostra um texto bastante difícil de classificar, pois claramente traz um tom de humor, embora o modelo tenha sido capaz de fazer a mesma classificação que o *tone analyzer*.

4.4.6 Rotular novos dados

Com o término da avaliação do modelo, e verificando sua viabilidade para rotulagem de novos dados, apesar de apresentar alguns problemas o modelo se mostra ter alguma capacidade generalização. Então esse modelo foi utilizado para rotular os 527.236 tweets restantes, onde se uma classe tivesse mais que 60%, então seria classificada como 1 se não seria classificada como 0, pode ser visto um resumo da contagem na tabela 4.30 das classes individualmente por texto. No fim desse processo todos os dados tratados foram rotulados totalizando os 567.236 tweets.

A seguir é mostrado uma tabela com apenas as regiões do brasil que foram possíveis de serem formatadas, então foi deixado de lado a categoria Outros que

também possui muitas regiões de brasil difíceis de serem formatadas, também foi deixado de lado as regiões de outros países como por exemplo Portugal.

Figura 4.30: Total das classes nos 527.236 tweets

Classes	Total
none	188623
analytical	133695
sadness	87703
joy	77131
tentative	73476
confident	31681
anger	22264
fear	13245

Fonte: Próprio autor

Figura 4.31: Comparação de classes individualmente por regiões

	Fear	Tentative	Anger	Sadness	Analytical	Joy	Confident
Locations							
São Paulo	1002.0	5380.0	1496.0	7039.0	10958.0	6327.0	2271.0
Paraná	105.0	526.0	129.0	913.0	1290.0	637.0	267.0
Mato Grosso Do Sul	21.0	173.0	32.0	189.0	265.0	158.0	61.0
Distrito Federal	52.0	289.0	75.0	398.0	753.0	358.0	145.0
Santa Catarina	79.0	566.0	130.0	692.0	1118.0	638.0	268.0
Rio Grande Do Norte	33.0	182.0	33.0	306.0	462.0	228.0	78.0
Amapá	3.0	33.0	4.0	48.0	80.0	36.0	20.0
Rondônia	12.0	73.0	15.0	136.0	184.0	68.0	43.0
Alagoas	16.0	79.0	22.0	178.0	214.0	89.0	40.0
Pará	46.0	226.0	35.0	286.0	474.0	248.0	85.0
Sergipe	15.0	110.0	19.0	149.0	207.0	112.0	34.0
Mato Grosso	16.0	88.0	19.0	122.0	179.0	91.0	24.0
Rio De Janeiro	776.0	4320.0	1500.0	5430.0	8426.0	4946.0	2138.0
Goiás	56.0	207.0	48.0	372.0	516.0	287.0	114.0
Espírito Santo	55.0	325.0	90.0	461.0	627.0	333.0	151.0
Ceará	54.0	255.0	51.0	390.0	588.0	338.0	132.0
Bahia	98.0	576.0	102.0	818.0	1166.0	685.0	288.0
Amazonas	28.0	155.0	27.0	280.0	425.0	231.0	86.0
Tocantins	15.0	90.0	26.0	171.0	261.0	198.0	81.0
Roraima	1.0	33.0	7.0	29.0	56.0	30.0	13.0
Piauí	32.0	119.0	14.0	188.0	265.0	118.0	61.0
Brasil	2897.0	16159.0	4799.0	20063.0	30302.0	17856.0	7362.0
Paraíba	31.0	167.0	27.0	308.0	430.0	177.0	77.0
Rio Grande Do Sul	163.0	1012.0	244.0	1350.0	2501.0	1327.0	500.0
Maranhão	30.0	133.0	33.0	177.0	327.0	135.0	65.0
Acre	11.0	90.0	22.0	145.0	162.0	97.0	45.0
Pernambuco	62.0	332.0	97.0	633.0	855.0	458.0	163.0
Minas Gerais	178.0	1034.0	230.0	1362.0	2102.0	1156.0	470.0

Fonte: Próprio autor

A figura 4.31 demonstra que as regiões com mais ocorrências são Rio de Janeiro e São Paulo. A categoria Brasil são várias regiões que não eram fáceis de formatar, mas que representavam partes do Brasil, com uma possível melhor formatação dos dados seria possível distribuir melhor os dados, porem essa foi uma

formatação simplificada, logo será utilizado São Paulo e Rio de Janeiro para uma comparação em um gráfico de radar, outra possibilidade seria exibir esses valores em um mapa de calor apesar de não ter sido utilizado nesse trabalho.

Figura 4.32: Comparação de classes entre São Paulo e Rio de Janeiro



Fonte: Próprio autor

Pode ser visto que São Paulo possui mais valores do que Rio de Janeiro, embora Rio de Janeiro demonstre uma tendência maior que São Paulo em relação à classe *sadness*. A seguir pode ser verificado alguns exemplos de mensagens rotuladas pelo modelo.

Figura 4.33: Exemplo 1 rotulado pelo modelo
 antes da quarentena // na quarentena só fiquei ruiva e arranjei um trabalho
 before the quarantine in the quarantine i just got red and got a job

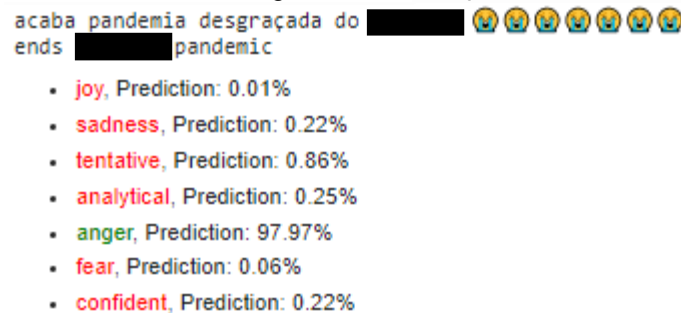
- joy, Prediction: 1.15%
- sadness, Prediction: 0.73%
- tentative, Prediction: 99.17%
- analytical, Prediction: 2.62%
- anger, Prediction: 0.28%
- fear, Prediction: 0.29%
- confident, Prediction: 0.00%

Fonte: Próprio autor

No primeiro exemplo rotulado na figura 4.33 o modelo identificou como *tentative* o texto, não possível dizer se essa seria a melhor classificação, pois esse texto possui um desafio interessante encontrado nas mídias sociais que é a utilização de imagens. A forma de como o texto é escrito passa uma ideia de comparação sobre antes e depois, provavelmente deveria haver uma imagem, muitas mensagens desse tipo podem ser difíceis de serem classificadas, pois há a falta do contexto da imagem. Talvez nesses

tipos de casos poderia ser utilizado a combinação de um classificador de texto com algum classificador de imagem.

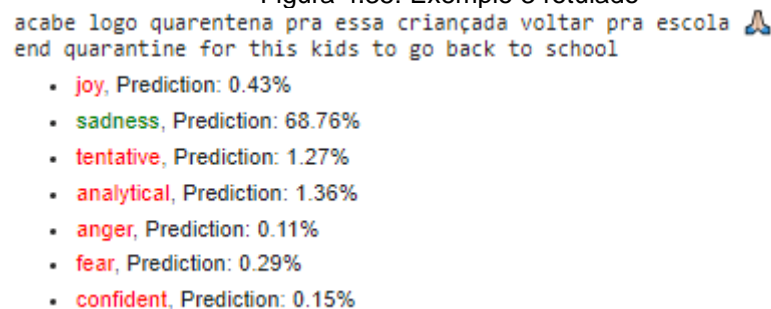
Figura 4.34: Exemplo 2 rotulado



Fonte: Próprio autor

O segundo exemplo da figura 4.34 mostra um texto com um tom negativo e o classificador identificou como *anger* o que pode ser dito como correto. Embora nesse trabalho não tenha sido utilizado os *emotes*, como pode ser visto no texto em inglês, o modelo ideal também deveria ser capaz de lidar com essa questão, não seria muito difícil realizar a vetorização dos *emotes*, mas tive que optar por remover para simplificar o trabalho por questões de formatação de textos.

Figura 4.35: Exemplo 3 rotulado



Fonte: Próprio autor

No terceiro exemplo da figura 4.35 o modelo classifica o texto como *sadness*, a minha avaliação do texto me diz que ele possui um tom de tristeza, mas talvez não sei o suficiente para ser classificado dessa maneira, como o modelo mostrou uma taxa de 68.76% pode se dizer que foi uma classificação adequada.

Figura 4.36: Exemplo 4 rotulado

num guento mais os mortos do tinder ressuscitando nessa pandemia
 in a wind plus the dead of the tinder rising in this pandemic

- joy, Prediction: 0.08%
- sadness, Prediction: 68.92%
- tentative, Prediction: 0.66%
- analytical, Prediction: 2.16%
- anger, Prediction: 0.21%
- fear, Prediction: 0.29%
- confident, Prediction: 0.24%

Fonte: Próprio autor

O último exemplo na figura 4.36 o modelo classificou como *sadness*, mas esse é um exemplo a ser destacado pela dificuldade de tradução, pois a parte “num guento” é entendida como “in a wind”, o que mostra um dos desafios da abordagem utilizando traduções. Outra observação que pode ser feita é o significado do texto que pode ser interpretado de várias maneiras, talvez a classificação *sadness* não seja a mais adequada nesse caso.

O último teste do modelo foi realizado com quatro exemplos de textos próprio, que foram criados para verificar como o modelo iria se comportar, como pode ser visto na figura 4.37.

Figura 4.37: Textos próprios para testar modelo

<p>Im tired I need a break Estou cansado preciso de uma pausa</p> <ul style="list-style-type: none"> • joy, Prediction: 0.02% • sadness, Prediction: 98.88% • tentative, Prediction: 2.62% • analytical, Prediction: 1.93% • anger, Prediction: 0.01% • fear, Prediction: 0.21% • confident, Prediction: 0.61% 	<p>I hate this quarantine Eu odeio essa quarentena</p> <ul style="list-style-type: none"> • joy, Prediction: 0.05% • sadness, Prediction: 0.80% • tentative, Prediction: 9.55% • analytical, Prediction: 5.19% • anger, Prediction: 97.15% • fear, Prediction: 0.39% • confident, Prediction: 0.21%
<p>You are incredibly fast as a turtle Você é incrivelmente rápido como uma tartaruga.</p> <ul style="list-style-type: none"> • joy, Prediction: 98.77% • sadness, Prediction: 0.07% • tentative, Prediction: 0.75% • analytical, Prediction: 6.54% • anger, Prediction: 0.01% • fear, Prediction: 0.01% • confident, Prediction: 0.84% 	<p>You are fast as a turtle Você é rápido como uma tartaruga.</p> <ul style="list-style-type: none"> • joy, Prediction: 0.84% • sadness, Prediction: 2.60% • tentative, Prediction: 0.61% • analytical, Prediction: 0.45% • anger, Prediction: 0.26% • fear, Prediction: 0.40% • confident, Prediction: 0.05%

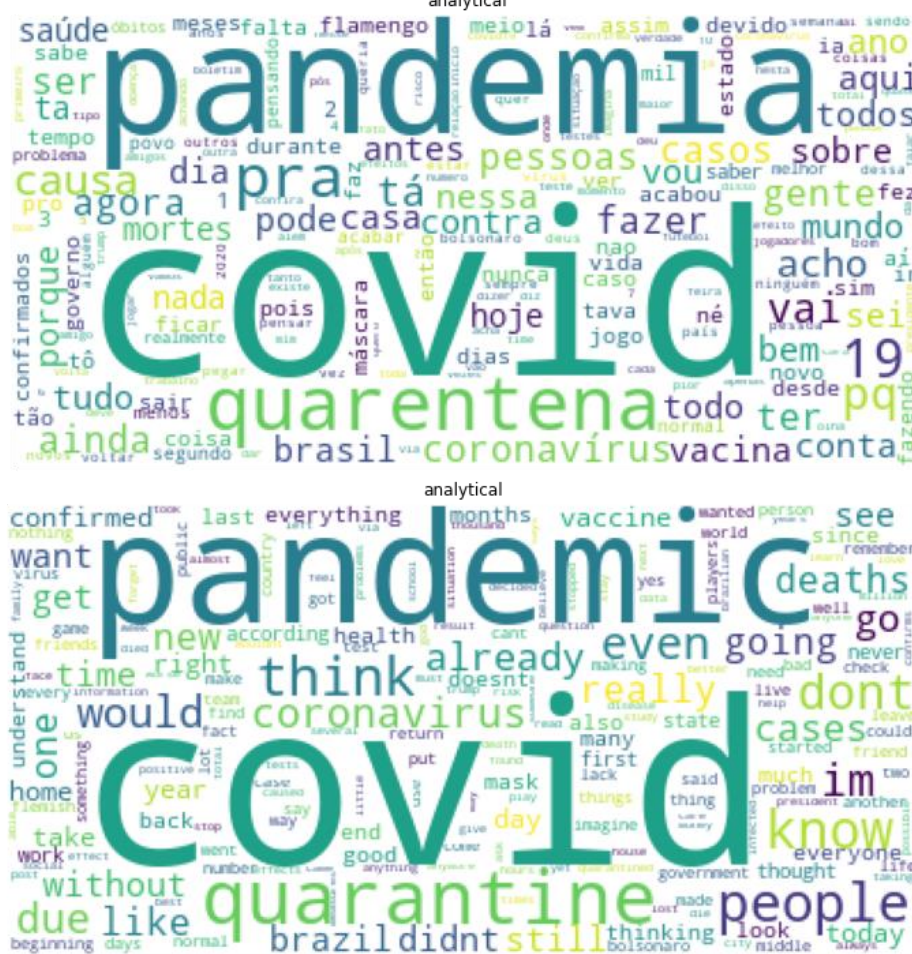
Fonte: Próprio autor

O primeiro texto (canto sup. esquerdo) na figura 4.37 é classificado como *sadness*, pode se dizer que é uma boa classificação. O segundo (canto sup. direito) foi classificado como *anger* a palavra “odeio” intensifica justifica essa classificação. O

terceiro e o quarto exemplos são uma tentativa de ver o impacto de uma única palavra na classificação, pode ser visto que no terceiro foi classificado como *joy*, mas no quarto quando é removido a palavra “incrivelmente” o modelo classifica o texto pertencendo a nenhuma das classes.

A utilização de word clouds também pode ser feita para avaliar os principais textos individualmente de cada uma dessas classes, também poderia ser feito por regiões, mas nos exemplos a seguir será feito apenas utilizando os textos que sejam classificados em uma das classes.

Figura 4.38: Word clouds *analytical*



Fonte: Próprio autor

O primeiro Word Cloud (figura 4.38) da classe *analytical* mostra palavras muito variadas, tirando as principais palavras (pandemia, covid, quarentena) que vão aparecer em todos os word cloud, não há algo relevante, talvez as palavras mais significativas sejam “pessoas, acho e mortes” que pode indicar análise sobre esses temas, como textos de notícias.

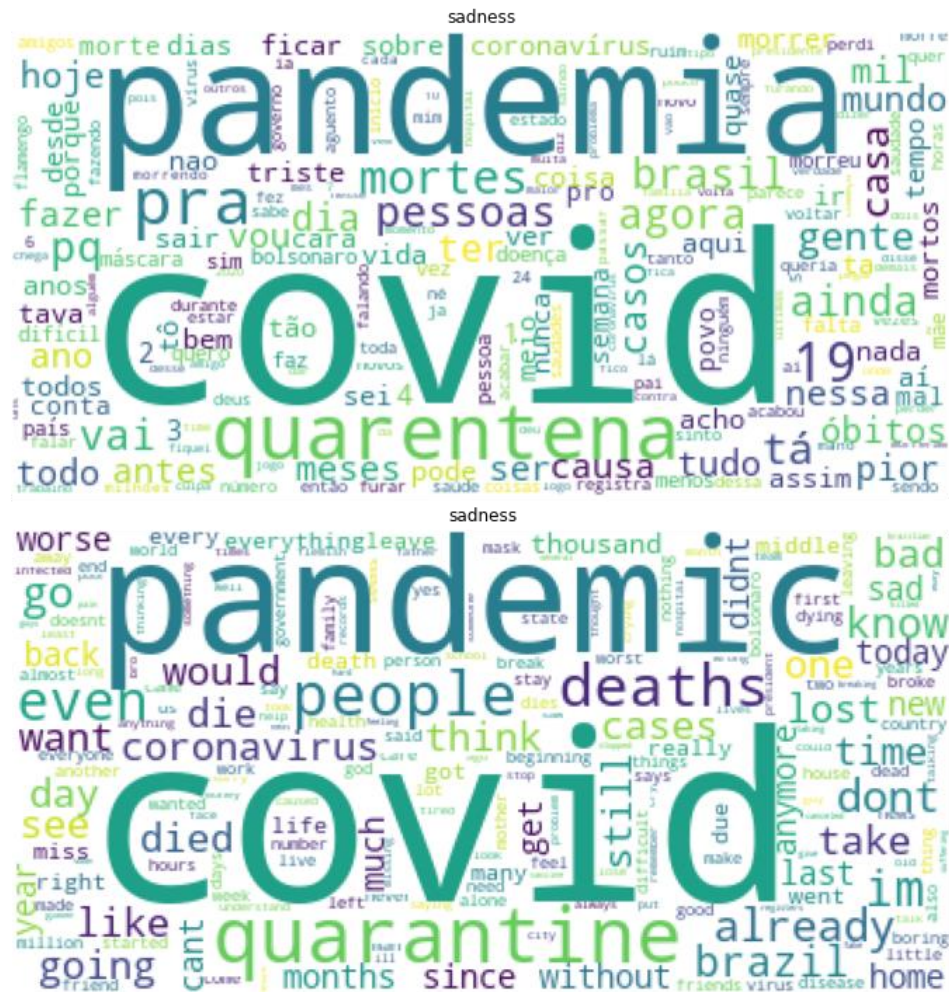
Figura 4.40: Word clouds *confident*



Fonte: Próprio autor

O *Word Cloud* da figura 4.40 representa a classe confidente (confiança), pode ser visto algumas palavras significativas como “nunca, sempre, agora, tudo e todos”.

Figura 4.43: *Word clouds sadness*



Fonte: Próprio autor

O *Word Cloud* (figura 4.43) exibe as principais palavras da classe *sadness*. Algumas palavras significativas são “mortes, morrer, óbitos, casos, pior”.

da camada de *embeddings* é realizado através de um aprendizado em que a própria camada busca desenvolver sua própria representação, a camada de *embeddings* também recebe um vocabulário das principais palavras do texto em português, e os textos recebem um tratamento de limpeza semelhante aos textos em inglês.

Um resumo geral dos dados são existe cerca de 393.613 dados rotulados, nesses dados existem cerca de 188.623 textos classificados como *none*, foi decidido então reduzir essa quantidade para 55 mil valores aleatórios classificados como *none*, isso para não criar uma tendência de classificação *none*, embora essa redução não seja algo recomendado para dados reais. O valor total de dados rotulados são 338.613 que vão ser dividido em dois conjuntos treinamento (255848) e validação (137765), o conjunto de teste será os 40 mil dados do *tone analyzer*.

Figura 4.46: Resumo dos dados rotulados pelo modelo

Reduzir a classe none para 55 mil	Depois
none 188623	analytical 133695
analytical 133695	sadness 87703
sadness 87703	joy 77131
joy 77131	tentative 73476
tentative 73476	none 55000
confident 31681	confident 31681
anger 22264	anger 22264
fear 13245	fear 13245

Fonte: Próprio autor

Figura 4.47: Resumo do treinamento do modelo em português

```
Epoch 1/10
507/507 [=====] - ETA: 0s - loss: 0.3598 - binary_accuracy: 0.8570
ROC-AUC - epoch: 1 - score: 0.871116
Epoch 00001: val_loss improved from inf to 0.29284, saving model to weights.hdf5
507/507 [=====] - 386s 761ms/step - loss: 0.3598 -
binary_accuracy: 0.8570 - val_loss: 0.2928 - val_binary_accuracy: 0.8840

Epoch 2/10
507/507 [=====] - ETA: 0s - loss: 0.2148 - binary_accuracy: 0.9155
ROC-AUC - epoch: 2 - score: 0.946510
Epoch 00002: val_loss improved from 0.29284 to 0.18532, saving model to weights.hdf5
507/507 [=====] - 384s 758ms/step - loss: 0.2148 -
binary_accuracy: 0.9155 - val_loss: 0.1853 - val_binary_accuracy: 0.9253

Epoch 3/10
507/507 [=====] - ETA: 0s - loss: 0.1777 - binary_accuracy: 0.9294
ROC-AUC - epoch: 3 - score: 0.952804
Epoch 00003: val_loss improved from 0.18532 to 0.18141, saving model to weights.hdf5
507/507 [=====] - 384s 757ms/step - loss: 0.1777 -
binary_accuracy: 0.9294 - val_loss: 0.1814 - val_binary_accuracy: 0.9267

Epoch 4/10
507/507 [=====] - ETA: 0s - loss: 0.1617 - binary_accuracy: 0.9359
ROC-AUC - epoch: 4 - score: 0.956688
Epoch 00004: val_loss improved from 0.18141 to 0.17066, saving model to weights.hdf5
507/507 [=====] - 385s 760ms/step - loss: 0.1617 -
binary_accuracy: 0.9359 - val_loss: 0.1707 - val_binary_accuracy: 0.9311
```

Epoch 5/10
507/507 [=====] - ETA: 0s - loss: 0.1477 - binary_accuracy: 0.9414
ROC-AUC - epoch: 5 - score: 0.958679
Epoch 00005: val_loss improved from 0.17066 to 0.16967, saving model to weights.hdf5
507/507 [=====] - 382s 754ms/step - loss: 0.1477 -
binary_accuracy: 0.9414 - val_loss: 0.1697 - val_binary_accuracy: 0.9320

Epoch 6/10
507/507 [=====] - ETA: 0s - loss: 0.1374 - binary_accuracy: 0.9458
ROC-AUC - epoch: 6 - score: 0.957893
Epoch 00006: val_loss did not improve from 0.16967
507/507 [=====] - 381s 752ms/step - loss: 0.1374 -
binary_accuracy: 0.9458 - val_loss: 0.1753 - val_binary_accuracy: 0.9299

Epoch 7/10
507/507 [=====] - ETA: 0s - loss: 0.1285 - binary_accuracy: 0.9499
ROC-AUC - epoch: 7 - score: 0.955525
Epoch 00007: val_loss did not improve from 0.16967
507/507 [=====] - 382s 753ms/step - loss: 0.1285 -
binary_accuracy: 0.9499 - val_loss: 0.1827 - val_binary_accuracy: 0.9292

Epoch 8/10
507/507 [=====] - ETA: 0s - loss: 0.1207 - binary_accuracy: 0.9531
ROC-AUC - epoch: 8 - score: 0.954751
Epoch 00008: val_loss did not improve from 0.16967
507/507 [=====] - 381s 751ms/step - loss: 0.1207 -
binary_accuracy: 0.9531 - val_loss: 0.1872 - val_binary_accuracy: 0.9289

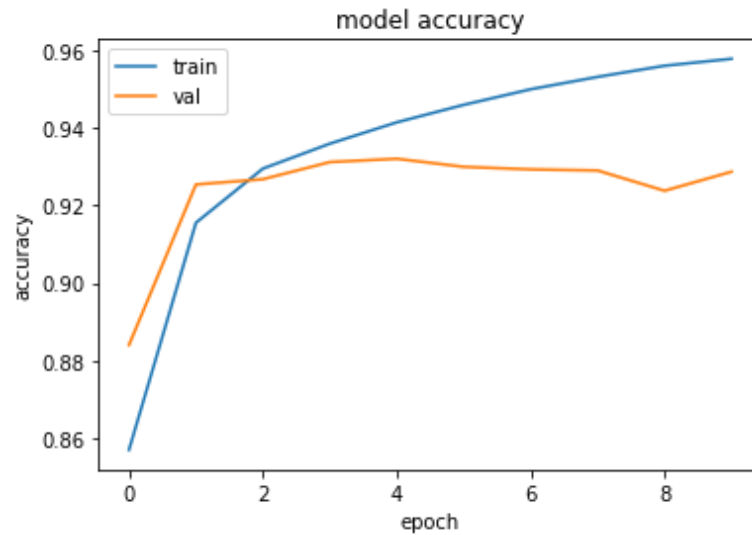
Epoch 9/10
507/507 [=====] - ETA: 0s - loss: 0.1139 - binary_accuracy: 0.9559
ROC-AUC - epoch: 9 - score: 0.949491
Epoch 00009: val_loss did not improve from 0.16967
507/507 [=====] - 381s 752ms/step - loss: 0.1139 -
binary_accuracy: 0.9559 - val_loss: 0.2121 - val_binary_accuracy: 0.9237

Epoch 10/10
507/507 [=====] - ETA: 0s - loss: 0.1097 - binary_accuracy: 0.9577
ROC-AUC - epoch: 10 - score: 0.948992
Epoch 00010: val_loss did not improve from 0.16967
507/507 [=====] - 382s 753ms/step - loss: 0.1097 -
binary_accuracy: 0.9577 - val_loss: 0.1987 - val_binary_accuracy: 0.9286

Fonte: Próprio autor

4.5.1 Avaliação do modelo em português

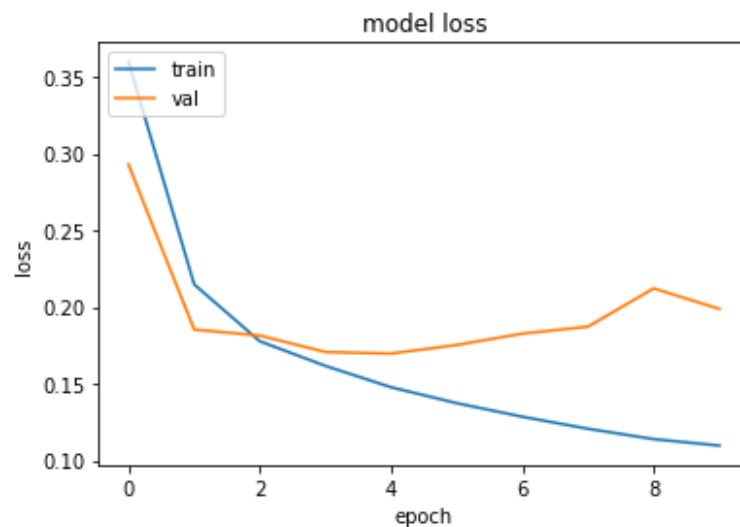
Figura 4.48: Accuracy do modelo para textos do português



Fonte: Próprio autor

O modelo foi treinado durante 10 épocas como mostrado na tabela 4.47. O *overffting* no modelo acontece muito rápido a partir da terceira época a acurácia (Figura 4.48) do conjunto de treinamento já é superior ao conjunto de validação. O *callback* do *model checkpoint* realiza o salvamento dos pesos na quinta época, a partir da quinta época a acurácia do conjunto de validação começa a piora, mostrando que o modelo não vai melhorar mais.

Figura 4.49: Loss do modelo para textos do português

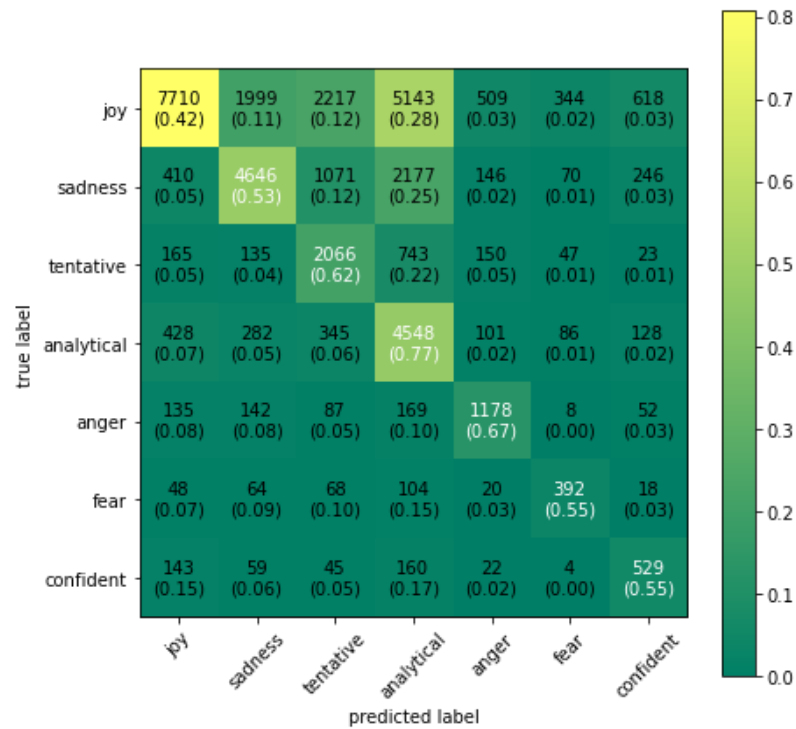


Fonte: Próprio autor

A perda do modelo no conjunto de validação começa piorar cada vez mais a partir da quinta época, o que mostra que o modelo não vai melhorar.

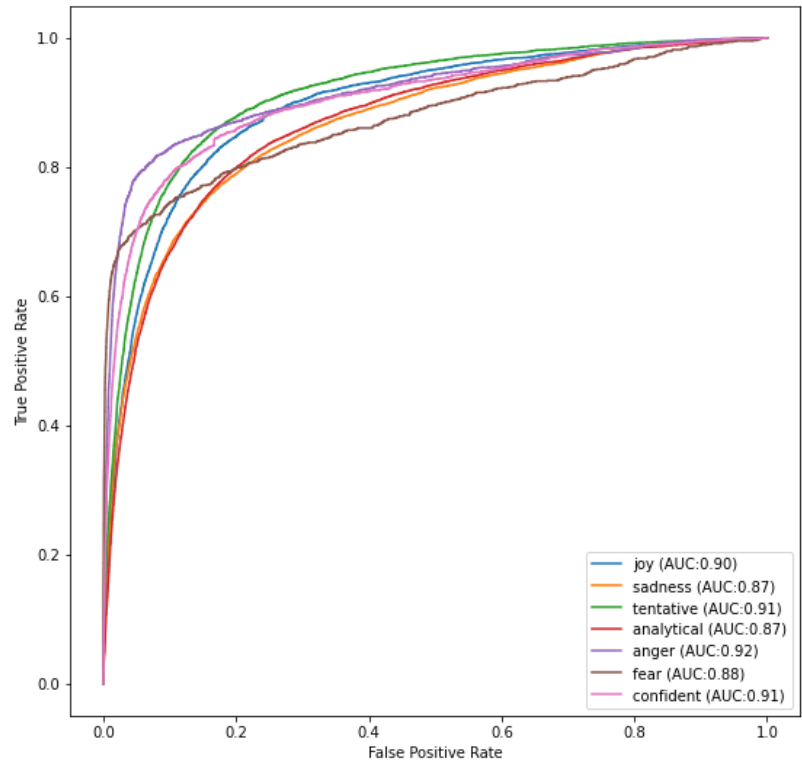
A realização dos testes será feita com os pesos tanto da decima época quanto da quinta época, isso foi feito para verificar o efeito de deixar o modelo com alto overfitting e alta perda na validação.

Figura 4.50: Matriz de confusão do modelo em português na decima época



Fonte: Próprio autor

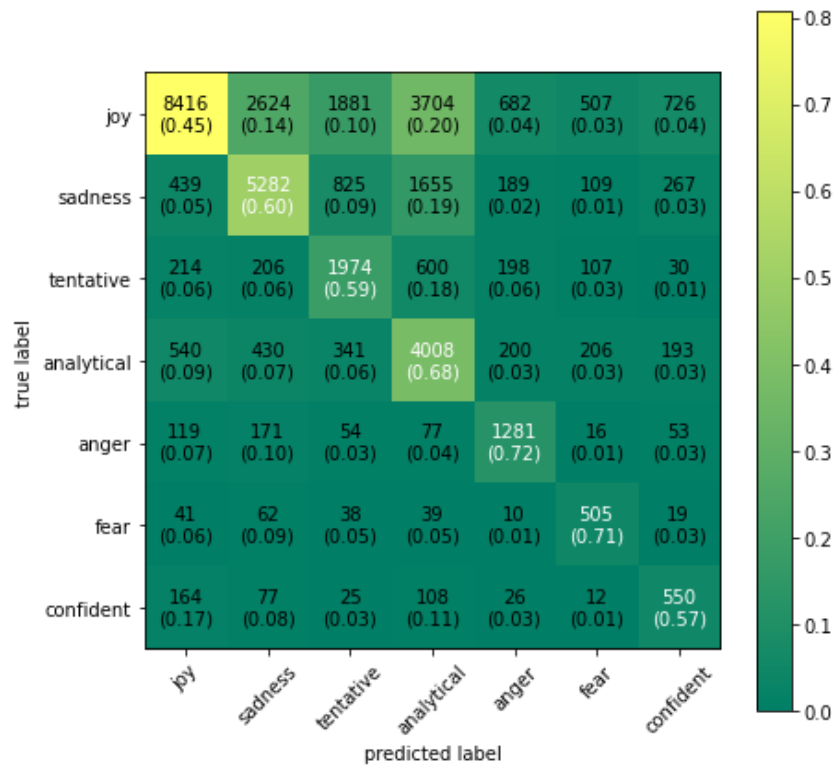
Figura 4.51: AUC-ROC do modelo em português na decima época



Fonte: Próprio autor

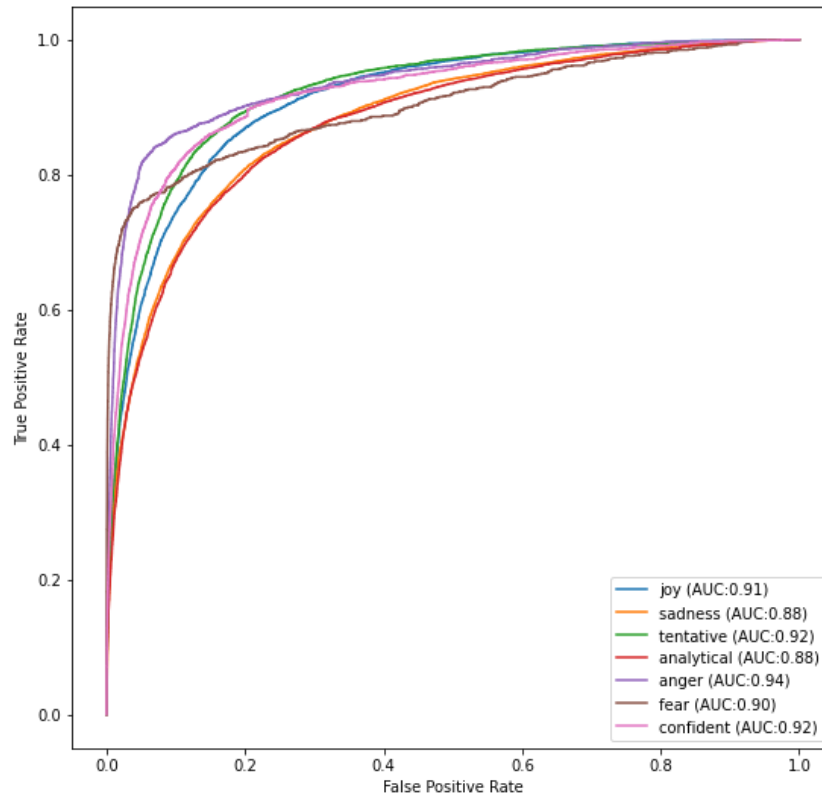
Pode ser visto pela matriz de confusão (Figura 4.50) e o gráfico AUC-ROC (Figura 4.51) que esse modelo é pior que o modelo para classificação de textos em inglês, apesar de ser ruim ainda possui alguma capacidade de classificação.

Figura 4.52: Matriz de confusão do modelo em português na quinta época



Fonte: Próprio autor

Figura 4.53: AUC-ROC do modelo em português na decima época



Fonte: Próprio autor

O carregamento dos pesos da quinta época melhorou o desempenho do modelo, isso mostra que realmente faz diferença monitorar o *loss* no conjunto de validação e tentar evitar ter muito overfitting na criação do modelo. Alguns exemplos vistos anteriormente nos testes do modelo que classifica em inglês foram utilizados nesse modelo para verificar sua capacidade.

Figura 4.54: Exemplo 1 modelo em português

impossível continuar a faculdade na pandemia sem fazer tratamento psicologico

- joy, Prediction: 0.16%
- sadness, Prediction: 88.53%
- tentative, Prediction: 0.61%
- analytical, Prediction: 39.23%
- anger, Prediction: 0.11%
- fear, Prediction: 0.08%
- confident, Prediction: 0.65%

Fonte: Próprio autor

O modelo de classificação em inglês tinha conseguido 97% no texto da figura 4.54, embora o modelo em português tenha piorado 88%, ele ainda é capaz de reconhecer o rótulo *sadness*.

Figura 4.55: Exemplo 2 modelo em português

o morango esta branco porque deixei de molho no alcool pra tira o covid

- joy, Prediction: 8.36%
- sadness, Prediction: 1.09%
- tentative, Prediction: 5.95%
- analytical, Prediction: 49.27%
- anger, Prediction: 0.67%
- fear, Prediction: 0.20%
- confident, Prediction: 2.06%

Fonte: Próprio autor

O modelo de classificação em inglês tinha conseguido 91% no texto da figura 4.55 realizando a classificação para a *analytical*, nesse caso o modelo em português não alcançou um valor maior que 60%, embora o modelo tenha identificado o texto como *analytical* com 49.27%, isso mostra que o modelo realmente piorou.

Figura 4.56: Exemplo 3 do conjunto rotulado modelo em português

acabe logo quarentena pra essa criançada voltar pra escola

- joy, Prediction: 7.02%
- sadness, Prediction: 18.15%
- tentative, Prediction: 5.66%
- analytical, Prediction: 2.98%
- anger, Prediction: 1.78%
- fear, Prediction: 0.40%
- confident, Prediction: 2.97%

Fonte: Próprio autor

O modelo de classificação em inglês tinha alcançado 68.76% no texto da figura 4.56 classificando como a classe *sadness*, o modelo em português alcançou 18.15% outro valor inferior ao esperado.

Figura 4.57: Exemplo próprios no modelo em português

eu estou cansado preciso de uma pausa eu odeio essa quarentena

- | | |
|---------------------------------|---------------------------------|
| • joy, Prediction: 0.11% | • joy, Prediction: 0.07% |
| • sadness, Prediction: 97.79% | • sadness, Prediction: 0.12% |
| • tentative, Prediction: 3.52% | • tentative, Prediction: 0.34% |
| • analytical, Prediction: 0.87% | • analytical, Prediction: 0.14% |
| • anger, Prediction: 0.08% | • anger, Prediction: 97.79% |
| • fear, Prediction: 0.10% | • fear, Prediction: 0.05% |
| • confident, Prediction: 0.68% | • confident, Prediction: 0.32% |

voce e incrivelmente rapido como uma tartaruga voce e rapido como uma tartaruga

- | | |
|----------------------------------|----------------------------------|
| • joy, Prediction: 86.36% | • joy, Prediction: 4.04% |
| • sadness, Prediction: 0.81% | • sadness, Prediction: 2.92% |
| • tentative, Prediction: 4.40% | • tentative, Prediction: 22.00% |
| • analytical, Prediction: 14.71% | • analytical, Prediction: 13.40% |
| • anger, Prediction: 0.41% | • anger, Prediction: 1.07% |
| • fear, Prediction: 0.08% | • fear, Prediction: 0.38% |
| • confident, Prediction: 5.09% | • confident, Prediction: 3.71% |

Fonte: Próprio autor

O último exemplo da figura 4.57, exibe os mesmos exemplos criados para testar o modelo em inglês, no primeiro exemplo (canto sup. esquerdo) o modelo em português alcançou 97.79% contra 98.88% do modelo em inglês. O segundo exemplo (canto sup. direito) o modelo em português alcançou 97.79% contra 97.15% do modelo em inglês. O terceiro exemplo o modelo em português alcançou 86.36% contra 98.77%. O último exemplo ambos os modelos não classificaram o texto como pertencendo a nenhuma classe.

5 CONCLUSÃO

Neste trabalho, foi apresentado a abordagem de análise de sentimento e construção de um método para classificação de emoção aplicado em mensagens do Twitter em português brasileiro relacionados ao COVID-19.

Para isso, foram realizadas algumas etapas básicas para a construção de um projeto de *machine learning* como; coleta de dados, análise exploratória dos dados, pré-processamento dos dados, construção de uma base de dados rotulada, criação de um classificador de textos, avaliação do classificador e rotulagem em novos dados. Duas abordagens foram utilizadas, a primeira foi a classificação de polaridade utilizando a biblioteca *TextBlob* (Python) e a segunda foi a criação de um modelo de *deep learning* com a biblioteca Keras também do Python.

Os testes mostraram que é viável a abordagem de tradução dos textos, pois os classificadores do modelo *TextBlob* e *Keras* foram capazes de encontrar certas representações para realizar generalizações. Essa abordagem mostra uma opção para falta de dados rotulados, buscando aproveitar ferramentas que podem rotular dados ou conjuntos de dados rotulados que já existem em inglês. No entanto, é possível perceber que a abordagem apresenta alguns desafios, como a dificuldade de tratar os textos ou palavras que geram ruídos para os classificadores.

5.1 TRABALHOS FUTUROS

O trabalho demonstrou apenas uma pequena parte do que pode ser obtido através da análise de sentimento, além disso, muitas técnicas da análise de sentimentos estão disponíveis em bibliotecas como o *TexBlob* e outras para NLP. Com o avanço dos métodos de aprendizado de máquina e do aprendizado profundo, é esperado que no futuro abordagens mais avançadas sejam desenvolvidas dentro da NLP e análise de sentimentos.

Para possíveis trabalhos futuros;

- Utilizar melhores abordagens para o tratamento de textos, como a utilização de um corretor ortográfico personalizado para tratar casos especiais das redes sociais, podendo ser feito através de técnicas de aprendizado de máquina.
- Estudar meios de lidar com sarcasmo
- Aplicar métodos conhecidos como a criação de um dicionário para compilar palavras de sentimento em português para realização da classificação de polaridade.²⁹
- Utilizar diferentes classificadores simples que podem alcançar bons desempenhos em tarefas que não precisem de modelos complexos, o que ajudaria a evitar *overffing*.
- Melhorar o modelo de *deep learning* para ter uma precisão melhor ao utilizar outras opções de camada como os *Transformers* que são especiais para tarefas da NLP e muito utilizados em traduções e simplificação de textos.
- Expandir a utilizar das técnicas da análise de sentimento para outros tipos de problemas, como a combinação da classificação de polaridade de um texto para reforçar um *chatbot*.

O trabalho pode ser encontrado no link do github.

<https://github.com/lnunesAI/Unicarioca-TCC>

²⁹ LIU, B. **Sentiment Analysis**: Mining Opinions, Sentiments, and Emotions. 1. ed. Cambridge: Cambridge University Press., 4 June 2015. 190-191 p.

REFERÊNCIAS BIBLIOGRÁFICAS

1. RUSSELL, M. A.; KLASSEN, M. Mining the Social Web. In: RUSSELL, M. A.; KLASSEN, M. **Mining the Social Web. Prefácio**. 3. ed. Sebastopol: O'Reilly Media Inc., 2019. p. xviii.
2. MICHAILIDIS, M. Social Machine Learning with H2O, Twitter, python. **Linkedin**, September 10, 2017. Disponível em: <<https://www.linkedin.com/pulse/social-machine-learning-h2o-twitter-python-marios-michailidis>>. Acesso em: 12 Novembro 2020.
3. HIRSCHBERG, J.; MANNING, C. D. Advances in natural language processing. **Science**, New York, v. 349, n. 6245, p. 261-266, 16 jul. 2015. Disponível em: <<http://dx.doi.org/10.1126/science.aaa8685>>. Acesso em: 12 nov. 2020.
4. LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 21-22 p.
5. LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 1 p.
6. PANG, B.; LEE, L. Opinion mining and sentiment analysis. **Foundations And Trends® In Information Retrieval**, [S.l.], v.2., n. 1-2., 14 nov. 2008. 1-135 p. Disponível em: <<http://dx.doi.org/10.1561/15000000011>>. Acesso em: 12 nov. 2020.
7. LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 67-69 p.
8. LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press, 4 June 2015. 49 p.
9. HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2. ed. New York: Springer Verlag New York Inc, 9 Feb 2009. 222 p. Disponível em: <<https://web.stanford.edu/~hastie/ElemStatLearn/>>.
10. GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: The Mit Press, 18 Apr 2017. 120-122 p.
11. GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge: The Mit Press, 18 Apr 2017. 108-112 p.

12. GÉRON, A. The Machine Learning Landscape. In: GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2^a. ed. Sebastopol: O'Reilly Media Inc., 2019. Cap. 1.
13. CHOLLET, F.; OMERNICK, M. Working with preprocessing layers. **Keras**, 25 julho 2020. Disponível em: <https://keras.io/guides/preprocessing_layers/>. Acesso em: 12 Novembro 2020.
14. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 9 p.
15. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 58-59 p.
16. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 70-73 p.
17. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 114 p.
18. KINGMA, D. P.; BA, J. L. **Adam: A Method For Stochastic Optimization**, San Diego, 2015. 1-15 p. Disponível em: <<https://arxiv.org/abs/1412.6980v9>>. Acesso em: 22 nov. 2020.
19. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 180 p.
20. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 184-188 p.
21. PENNINGTON, J.; SOCHER, R.; MANNING, C. D. GloVe: Global Vectors for Word Representation. **nlp.stanford.edu**. Disponível em: <<https://nlp.stanford.edu/projects/glove/>>. Acesso em: 10 Novembro 2020.
22. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 109-110 p.
23. IBM. Tone Analyzer. Disponível em: <<https://tone-analyzer-demo.ng.bluemix.net/>>. Acesso em: 18 Novembro 2020.

24. JIGSAW. Toxic Comment Classification Challenge. **kaggle**, 2017. Disponível em: <<http://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/2---diodo-semicondutor.pdf>>. Acesso em: 18 Novembro 2020.
25. REDAÇÃO GE. Flamengo divulga mais dois casos positivos de Covid no departamento de futebol. **globoesporte**, 22 Setembro 2020. Disponível em: <<https://globoesporte.globo.com/futebol/times/flamengo/noticia/flamengo-divulga-mais-dois-casos-positivos-de-covid-no-departamento-de-futebol.ghml>>. Acesso em: 18 Novembro 2020.
26. CHOLLET, F. Using pre-trained word embeddings. **keras**, 5 mai 2020. Disponível em: <https://keras.io/examples/nlp/pretrained_word_embeddings/>. Acesso em: 19 Novembro 2020.
27. GÉRON, A. Processing Sequences Using RNNs and CNNs. In: GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2^a. ed. Sebastopol: O'Reilly Media Inc., 2019. Cap. 15.
28. CHOLLET, F. **Deep Learning with Python**. New York: Manning Publications, 10 Jan 2018. 249 p.
29. LIU, B. **Sentiment Analysis: Mining Opinions, Sentiments, and Emotions**. 1. ed. Cambridge: Cambridge University Press., 4 June 2015. 190-191 p.

ANEXO A – CÓDIGO DE DESENVOLVIMENTO DO MODELO EM INGLÊS

Importação de bibliotecas utilizadas

```
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.layers import LSTM, GlobalMaxPooling1D, Dropout, Embedding, Dense, Input
from tensorflow.keras.layers.experimental.preprocessing import TextVectorization
from tensorflow.keras.callbacks import Callback
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
```

Carregamento dos dados para treinamento

```
#Carrega o conjunto de dados com 40 mil valores via Colab
df=pd.read_csv("/content/drive/My Drive/TCC/Datasets/Final/tones/27-28_tones_Edited_tweets_ENPT_covid.csv")
#Criar uma lista com as 7 classes
class_names = list(df.columns[19:26])
#Os textos que não possui uma classe estão com valor NaN, então é modificado para ser 0.
df.iloc[:,19:26] = df.iloc[:,19:26].fillna(0)
#Separação dos dados em (Treinamento 28000 | Validação 7800 | Test 4200)
df_train, df_valid_test = train_test_split(df, test_size = 0.30, random_state = 42)
df_test, df_valid, = train_test_split(df_valid_test, test_size=0.65, random_state=42)
```

Vetorização dos textos

```
#O código da vetorização pode ser encontrado com explicações na referência 26 da bibliografia.
vectorizer = TextVectorization(max_tokens=20000, output_sequence_length=140)
```



```
text_ds = tf.data.Dataset.from_tensor_slices(df_train.clean_textEN).batch(128)
vectorizer.adapt(text_ds)
```

```
voc = vectorizer.get_vocabulary()
word_index = dict(zip(voc, range(len(voc))))
```

```
path_to_glove_file='/content/drive/MyDrive/pt_540k/glove/glove.6B.100d.txt'
embeddings_index = {}
```

```
with open(path_to_glove_file) as f:
    for line in f:
        word, coefs = line.split(maxsplit=1)
        coefs = np.fromstring(coefs, "f", sep=" ")
        embeddings_index[word] = coefs
print("Found %s word vectors." % len(embeddings_index))
```

```
num_tokens = len(voc) + 2
embedding_dim = 100
hits = 0
misses = 0
```

```
# Prepare embedding matrix
```

```
embedding_matrix = np.zeros((num_tokens, embedding_dim))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # Words not found in embedding index will be all-zeros.
        # This includes the representation for "padding" and "OOV"
        embedding_matrix[i] = embedding_vector
        hits += 1
    else:
        misses += 1
print("Converted %d words (%d misses)" % (hits, misses))
```

#Criado a camada de Embeddings para ser utilizado na criação do modelo

```
from tensorflow.keras.initializers import Constant
embedding_layer = Embedding(
    num_tokens,
    embedding_dim,
    embeddings_initializer = Constant(embedding_matrix),
    trainable=False,
)
```

#Vetorizando as entradas X

```
X_train = vectorizer(df_train.clean_textEN)
X_valid = vectorizer(df_valid.clean_textEN)
X_test = vectorizer(df_test.clean_textEN)
```

#Criando uma lista com os rótulos corretos (y) de cada entrada (X)

```
y_valid = df_valid[class_names].values
y_train = df_train[class_names].values
y_test = df_test[class_names].values
```

Criação do Modelo

#Criação da estrutura do modelo

```
inp = Input(shape=(140, ), dtype='int32')
x = embedding_layer(inp)
x = LSTM(128, return_sequences=True)(x)
x = LSTM(64, return_sequences=True)(x)
x = GlobalMaxPooling1D()(x)
x = Dropout(0.2)(x)
x = Dense(32, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(7, activation="sigmoid")(x)
model = Model(inputs=inp, outputs=x)
```

#Compilação do modelo

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['binary_accuracy'])
```

Lista de Callbacks

#Callback personalizado que utiliza a métrica roc_auc_score por época

```
class RocAucEvaluation(Callback):
```

```
    def __init__(self, validation_data=(), interval=1):
```

```
        super(Callback, self).__init__()
```

```
        self.interval = interval
```

```
        self.X_val, self.y_val = validation_data
```

```
    def on_epoch_end(self, epoch, logs={}):
```

```
        if epoch % self.interval == 0:
```

```
            y_pred = self.model.predict(self.X_val, verbose=0)
```

```
            score = roc_auc_score(self.y_val, y_pred)
```

```
            print("\n ROC-AUC - epoch: {:d} - score: {:.6f}".format(epoch+1, score))
```

```
RocAuc = RocAucEvaluation(validation_data=(X_valid, y_valid), interval = 1)
```

#Callback que salva os pesos ao monitorar o val_loss do modelo caso diminua durante o treino

```
file_path="weights.hdf5"
```

```
checkpoint = ModelCheckpoint(file_path, monitor='val_loss', verbose=1, save_best_only=True, mode='min')
```

#Callback que para o treino caso não tenha mudança no val_loss durante 8 épocas

```
early = EarlyStopping(monitor="val_loss", mode="min", patience=8)
```

#Lista com os callbacks criados para ser utilizado na criação do modelo

```
callbacks_list = [RocAuc, checkpoint, early]
```

Treinamento do modelo

```
epochs = 15
history = model.fit(
    X_train, y_train,
    batch_size= 128,
    epochs=epochs,
    validation_data=(X_valid, y_valid),
    verbose=1,
    callbacks=callbacks_list)
```

Exibindo o histórico do treinamento do modelo

```
plt.plot(history.history['binary_accuracy'])
plt.plot(history.history['val_binary_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```