Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Análise de Dados em Gráficos com base na API do Twitter

Diego Marques Pazos[1]
Centro Universitário Academia, Juiz de Fora, MG

Tassio Ferenzini Martins Sirqueira[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de Pesquisa: Engenharia de Software

RESUMO

O avanço das tecnologias tem provocado diversas mudanças na sociedade em que vivemos, e uma área em exato é a de dados. As redes sociais como conhecemos hoje, tem prestado um importante papel nessa crescente gama de dados que conecta todo o mundo, possibilitando com isso analisar o comportamento e os sentimentos daqueles que utilizam essas plataformas. Visto isso, este presente trabalho busca explanar uma forma de analisar dados de forma rápida e eficiente trazendo um panorama geral de determinado assunto no presente momento, a partir de tecnologias atuais como Python e Pandas, atrelados a rede social Twitter. Como resultado, pretende-se apresentar um *dashboard* analítico com perspectivas sobre o assunto escolhido, que unido a conhecimentos da área ciência de dados possa trazer elucidações sobre o tema.

Palavra-Chave: Python. Dados. Pandas. *Twitter*

ABSTRACT

The advancement of technology has changed society in many ways and in particularly the data segment. Social networks as we know them has been playing an important role in this constantly growth of data through the world making it possible to analyze the behavior and sentiments of social network users. Having said that, this research aims to show a fast and efficient form of data analysis with current technologies like Python and Pandas united with the social network Twitter. As the result, we provide an analytic dashboard with insights of a chosen subject, that with the support of data science would able to bring insights from the given topic.

[1] Discente do Curso de Engenharia de Software do Centro Universitário Academia – UniAcademia. E-mail: dmarquespazos@gmail.com
[2] Docente do Curso de Engenharia de Software do Centro Universitário Academia. Orientador.

1. Introdução

Atualmente, cada vez mais se observa um aumento do fluxo de dados em todas as redes ao redor do mundo (HOLST, 2021). Entretanto, existe uma limitação perceptível do que o humano consegue absorver e analisar, e consequentemente transformar em informação. Assim como ocorreu na Revolução Industrial 4.0[3], que buscou a automação de processos dos meios tradicionais de produção, essa limitação humana supracitada faz com que seja imprescindível novas formas de processar a chamada *Big Data.*

Conforme explicam os autores Ishwarappa e Anuradha J (p.320, 2015) *Big Data* é comumente descrito como um determinado conjunto de dados que se adequam aos descritos "5 Vs" (Volume, Velocidade, Variedade, Veracidade e Valor). O volume que representa a crescente quantidade de dados, a velocidade que cada vez mais é uma necessidade humana e das empresas por obter informações em um ritmo contínuo, a variedade pois esses dados podem vir de diversas fontes seja ela uma rede social ou planilha de contabilidade, a veracidade pois tudo isso que é extraído e tratado deve possuir uma qualidade de dado adequada para consumo da informação e por fim o ponto principal que é o valor sobre as informações obtidas, valor que usuários buscam tornando importante novas maneiras de se processar a *Big Data.*

Adicionalmente, foi possível perceber que a forma como que o mundo se comunica mudou devido ao surgimento e vasta expansão das redes sociais[4]. Dessa maneira, começa-se a observar a necessidade por técnicas e aplicações para análise e extração desses dados gerados a todo tempo pelos bilhões de usuários ao redor do planeta.

Mesmo assim, como qualquer tecnologia precursora, esse novo ramo chamado de *data analytics* traz diversos desafios para serem superados. Dentre esses desafios, um que se destaca é o fato de que grande parte ou em alguns casos a totalidade desses dados extraídos não estão preparados para consumo. Exemplificando, Ali Hassan Sial, Syed Yahya Shah Rashdi e Dr. Abdul Hafeez Khan (p. 279, 2021), dizem que após o processamento e planejamento inicial dos dados, dentre os demais segmentos existentes, um deles é o *Data Cleaning*. Nessa fase, o dado bruto pode conter diversos erros, duplicatas ou até mesmo não ser necessário, dessa maneira se faz necessário a limpeza amostra de dados. A partir disso, diversas técnicas como extração, técnicas de normalização para remoção de *stopwords* e

[3] "A quarta revolução industrial não é definida por um conjunto de tecnologias emergentes em si mesmas, mas a transição em direção a novos sistemas que foram construídos sobre a infraestrutura da revolução digital (anterior)".

[4] How has social media emerged as a powerful communication medium?. Disponível em: https://www.ucanwest.ca/blog/media-communication/how-has-social-media-emerged-as-a-powerful-communication-medium Acesso em: 08 nov. 2021

headers, além da filtragem de dados irrelevantes são utilizadas para uma padronização do dado a ser aferido e para que seja possível transformar esses dados brutos em informações úteis e relevantes. Posteriormente esses dados já pré-processados podem passar por um processo de análise, seja ele de sentimento, classificação e até mesmo contextual.

Com isso, este trabalho busca demonstrar uma forma possível de, a partir uma rede social, qual seja o *Twitter,* e os *tweets* disponibilizados a partir de um tópico inserido pelo usuário final, gerar um *Dashboard* de análise em tempo real*,* o qual possibilitará uma análise inicial baseada em gráficos do sentimento e da popularidade do tema requisitado pelo usuário.

Baseado nisso, visa-se implementar uma ferramenta que possa possibilitar, juntamente a conhecimentos de ciência de dados, a tomada de decisão *"data-driven" (DDD)*, visto que conforme citam os autores Foster Provost e Tom Fawcett (2013), os benefícios das tomadas de decisão baseados em dados já se mostraram conclusivos, de maneira que decisões baseadas em análise de dados são mais assertivas que pura intuição, porém, não impedindo que experiências prévias deste campo de trabalhado analisado possam também suportar essas decisões. Concluído com isso que a ferramenta apresentada neste trabalho, possa vir a fazer parte de um universo maior, da ciência de dados, auxiliando com isso na tomada de decisão a partir de dados expostos.

2. Fundamentação Teórica

Nesta seção a seguir, serão apresentadas as tecnologias, conceitos e outros trabalhos que possibilitaram e apoiaram a elaboração do trabalho desenvolvido.

2.1. Python

Conforme explica o *Python Institute*, criado por Guido van Rossum, diferentemente de linguagens recentes que comumente são desenvolvidas por grandes empresas de tecnologia, o Python o qual é conhecido pela sua versatilidade, semântica dinâmica, ser interpretada e orientada a objetos, iniciou como um projeto individual e com o passar do tempo e a expansão pelo mundo a linguagem passou a ser desenvolvida e melhorada por milhares de especialistas e entusiastas.

Além da semântica, será possível observar como todas as bibliotecas utilizadas para o desenvolvimento da aplicação se apoiam no Python, ou seja, a linguagem mostra sua importância no projeto a partir do fato da grande disponibilidade de bibliotecas.

2.2. Pandas

Pandas é uma ferramenta rápida e eficiente de manipulação e análise construída altamente otimizada para a performance, ponto importante quando se trata da análise de

dados.

Essa biblioteca tem como algumas de suas principais funções a criação de *DataFrames* para manipulação de grandes porções de dados, ferramentas próprias para leitura e escrita de dados assim como *reshaping*, *slicing* e *indexing* de grandes *datasets*.

No projeto em questão, a biblioteca Pandas será responsável pela criação de objetos *DataFrames* a serem tratados e analisados, a normalização de objetos Json, assim como à conversão de variáveis de tempo.

2.3. Natural Language Toolkit (*Python NLTK*)

Elizabeth D. Liddy (2001) explica que NLP ou *Natural Language Processing* são técnicas computacionais utilizadas com o objetivo de alcançar o entendimento linguístico humano a partir de técnicas e análises computacionais.

A *Natural Language Toolkit* é um software de código aberto, vastamente difundido e suportado pela comunidade que auxilia nos processos de categorização de textos, análise linguística, classificação entre outras características.

Essa ferramenta em questão é composta por diversos módulos, neste projeto foi utilizado o "*Vader Module"* que a partir de um dicionário Python faz a análise de sentimentos da base textual processada definindo pontuações entre negativas, neutras e positivas para se ter uma conclusão analítica.

2.4. Matplotlib

Segundo os autores Ali Hassan Sial, Syed Yahya Shah Rashdi e Dr. Abdul Hafeez Khan (p. 277, 2021), o matplotlib foi desenvolvido por John Hunter e seus diversos contribuidores que construíram uma das mais populares bibliotecas Python focada em visualização de dados. Vastamente difundida entre cientistas de dados, essa biblioteca se porta como um importante fator nos universos de dados com Python sendo suportada também por outras bibliotecas como por exemplo o NumPy.

No referido trabalho, a biblioteca Matplotlib foi utilizada para, após a transformação e tratamentos dos dados, criar um dashboard composto por gráficos diversos como por exemplo os gráficos de barras e setores, e consequentemente disponibilizar uma visualização efetiva das informações adquiridas.

2.5.Tweepy e Twitter API

O Twitter API e o Tweepy são tecnologias que juntas possibilitam a busca dos dados necessários para a construção da análise proposta.

Dentre as diversas funcionalidades que a API do Twitter possui e consequentemente

disponibiliza para extração, tem-se como por exemplo *Trends, Direct Messages,* mídias e usuários. O foco dessa aqui é na extração de Tweets para análise impessoal daquilo que está sendo tratado e discutido na rede social.

O Tweepy é uma biblioteca *Python based* que permite de maneira automatizada utilizar a API do Twitter para a busca de Tweets e outros dados que sejam relevantes para a pesquisa.

Neste trabalho, a união das tecnologias faz com que seja possível armazenar as informações em um objeto Json que futuramente será transformado para Data frames.

2.6 Extração, Transformação, Carregamento (*ETL*)

Dentre os elementos básicos de uma *Data Warehouse* temos a *Data Staging Area*. Nesta seção iremos dissertar sobre um conjunto de processos que forma o ETL (*extract-tranformation-load*) que juntamente com o armazenamento dos dados compõem a dita sessão da *Data Warehouse*. A *Data Staging Area* tem caráter subsidiário em relação à *Data Visualization*, de forma que todos os processos que não se encaixem nesta se enquadrarão naquela. Nesse trabalho será apresentado, por meio de um *Dashboard* de gráficos, tal plataforma visual, a qual utilizará como origem dados oriundos da rede social Twitter.

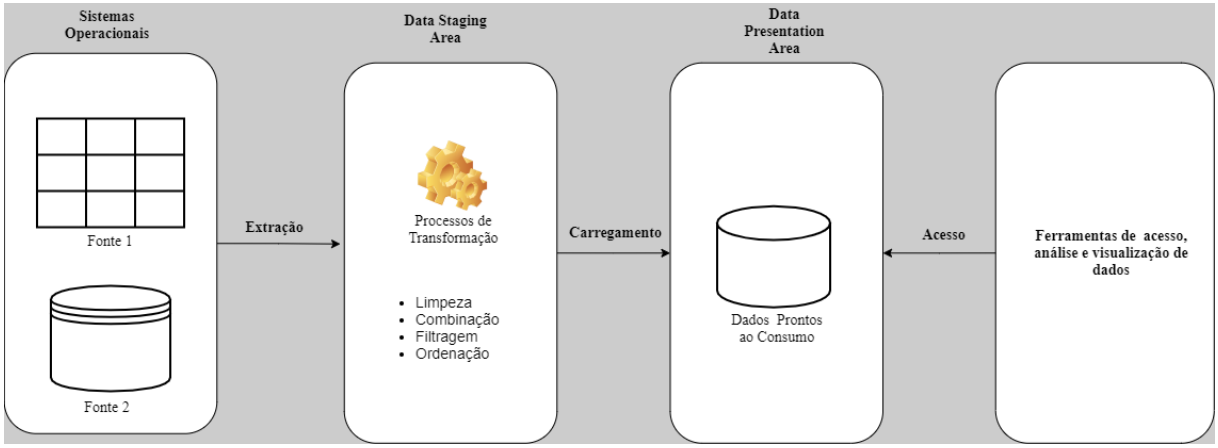
Falando em *Data Warehouse*, faz-se necessário esclarecimentos acerca do tema. Em suma, este repositório consiste em um sistema que possibilita o armazenamento de dados de diferentes modos e formatos para que seja possível transformações e, consequentemente, consultas analíticas.

Como pode ser percebido pela primeira letra da sigla ETL, o primeiro processo a ser conduzido é a extração dos dados, ou seja, buscar os dados que devem ser armazenados na *staging area* de maneira que eles possam passar pelas transformações necessárias.

Dentre essas manipulações, diversas são usuais dentro do cenário de *data analysis* como por exemplo remoção de duplicatas, combinação de dados de fontes diversas para formação de novos *datasets*, remoção de *stopwords* e correção erros ortográficos. Todas essas manipulações de dados citadas acima fazem parte dos processos de transformações que são os passos anteriores ao *Load* do ETL.

No último passo deste processo, ocorre o *Loading* dos dados aferidos. Neste estágio é de extrema importância assegurar a qualidade e integridade dos dados a serem publicados, de maneira que o usuário receba essa nova gama de informação com o resultado desejado. Conforme citado anteriormente, neste projeto a exemplificação desse resultado gerado ao usuário final será o *Dashboard* gráfico contendo as informações obtidas.

Figura 1: Demonstração do processo de ETL.



Fonte: Elaboração própria.

3. Metodologia

Nesta seção, será apresentado a metodologia empregada para o concebimento deste projeto. Exemplificando o tipo de estudo, e o passo-a-passo empregado para se obter o resultado.

3.1 Definição do Tipo de Estudo

O primeiro passo foi uma pesquisa qualitativa acerca de artigos e documentações para possibilitar o trabalho de conclusão de curso sobre o tema abordado, assim como para compreender a maneira ideal de se desenvolver a aplicação. Visto isso, foram selecionadas as tecnologias explanadas na introdução para serem utilizadas na criação.

Destarte, a aplicação por ter um resultado concreto e factual, fez com que fosse imprescindível que este trabalho possuísse pontos também de uma pesquisa quantitativa, visto que ao final é possível obter dados estruturados sobre o tema.

3.2 Escolha do Ambiente de Desenvolvimento

Para criação da aplicação, foram utilizadas as ferramentas (Tabela 1) a seguir e suas respectivas versões:

Tabela 1. Ferramentas e versões.

Ferramenta	Versão
Python	3.9.7
Visual Studio Code	1.60.1
Tweepy	3.10.0
Pandas	1.3.3
Matplotlib	3.4.3

NLTK	3.6.3
Twitter API	1.1

Fonte: Elaboração própria.

Demais bibliotecas e ou ferramentas utilizadas já são nativas do Python ou a versão escolhida não irá interferir no resultado.

3.3 Definição dos Processos de Autenticação e Buscas dos Dados

Para que seja possível a busca dos dados, se faz necessário que o usuário crie uma conta de desenvolvedor no *"Developer Portal - Twitter Developer".* Ao obter a conta de desenvolvedor, com a referência da documentação será possível compreender como fazer a busca dos Tweets ou outras informações desejadas utilizando a biblioteca Tweepy.

3.4 A Transformação dos Dados em Informação

Após a extração dos dados da rede social Twitter, foram utilizadas estruturas de controle, normalização, conversão de objetos dentre outras técnicas de *Data Cleaning* para que fosse possível iniciar a transformação desses dados extensos em informação concreta.

Cada gráfico resultante da pesquisa utiliza um determinado conjunto de informações. Ao ponto que, após uma primeira tratativa dos dados brutos que foram armazenadas as informações são direcionadas a um *DataFrame* genérico, se faz necessário operações, funções e métodos para que sejam armazenadas em variáveis as informações presentes em cada um dos gráficos a serem visualizados.

3.5 A Análise

Ao fim, as informações concretizadas são disponibilizadas em gráficos possibilitando para o usuário final uma visualização de todas as informações que foram obtidas a partir da extração e transformação dos dados. Dessa forma, proporciona que o final receptor da informação faça uma análise a partir de gráficos daquilo que foi solicitado. Dentre esses gráficos é possível obter elucidações de diversos aspectos como sentimentos, popularidade, idiomas e frequência.

4. Trabalhos Relacionados

Para desenvolvimento deste trabalho, foi feita uma pesquisa com intuito aplicado, ou seja, buscando o desenvolvimento de uma funcionalidade. Dessa maneira diversos artigos e documentações foram analisados para servir de embasamento teórico daquilo que seria desenvolvido.

Adicionalmente, para ainda confirmar a relevância do tema abordado, foi feita a busca por trabalhos relacionados para além de servir de embasamento teórico tenham-se exemplos do uso das tecnologias citadas na seção de Fundamentação Teórica.

Dessa maneira, nesta seção serão expostos alguns trabalhos que possuem arquitetura e desenvolvimento semelhantes a este trabalho e que sobretudo tem como fator relevante a análise de dados. Os projetos retratados serão *Sentiment Analysis in Machine Learning Using Twitter Data Analysis in Python*, *Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python*, *Twitter Sentiment Analysis as an Evaluation and Service Base On Python Textblob e Twitter Sentiment Analysis on Worldwide COVID-19 Outbreaks.*

4.1 Sentiment Analysis in Machine Learning Using Twitter Data Analysis in Python

J. Uma e Dr. K. Prabha apresentam um estudo que busca evidenciar como scripts de Machine Learning podem obter métricas relevantes a partir da análise de dados extraídos da rede social Twitter a partir da Twitter API o estudo também reflete e expõe sobre a crescente necessidade da análise de sentimento de dados em diversos setores da sociedade como *Online Commerce* e *Voice of The Market*, onde uma empresa ao lançar determinado produto pode se interessar por obter a reação dos consumidores ao seu mais novo empreendimento.

Ao concluir, o artigo mostra resultados para justificar a relevância que a extração de índices quantitativos de suposições pode ter ao ser analisados levantando que além de evidenciá-los é importante extraí-los. Enfatizando também que um trabalho futuro será feito um *streamlining* dos parâmetros obtidos para uma melhor visualização dos resultados expostos.

4.2 Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python

Neste estudo, duas proeminentes bibliotecas de visualização do Python foram debatidas: Matplotlib e Seaborn. Fatos foram expostos para comparar ambas as tecnologias e apontar em quais aspectos uma tecnologia pode se mostrar ou não superior a outra. Para possibilitar tais experimentos, são utilizados dados processados a partir de técnicas usuais de *Big Data* para que a partir das visualizações criadas a partir de estruturas de código a comparação possa ser feita.

Trazendo a conclusão, de que os cenários ideias para cada biblioteca podem variar, porém, se percebe que o Seaborn é ideal em casos de *datasets* mais complexos, enquanto o Matplotlib se encaixa melhor para visualização mais simples voltadas para a busca da expertise em ciências de dados e modelagem computacional.

4.3 Twitter Sentiment Analysis as an Evaluation and Service Base On Python Textblob

Utilizando bibliotecas como *Textblob* e *Worldcloud*, I Gede Susrama Mas Diyasa e colaboradores (2021), buscaram extrair e analisar informações sobre o sentimento dos usuários da empresa TELKOM no *Twitter.* Adicionalmente, os cientistas utilizaram métodos matemáticos como *Confusion Matrix Calculation* para testar a acurácia dos resultados obtidos. Ao concluir, o artigo mostra como é possível se obter uma análise concreta de sentimentos tendo por base dados de uma rede social assim como a visualização dessas métricas utilizando ferramentas de visualização.

4.3 Twitter Sentiment Analysis on Worldwide COVID-19 Outbreaks

Neste artigo, Kamaran H. Manguri, Rebaz N. Ramadhan e Pshko R. Mohammed Amin, buscam mostrar informações sobre o sentimento dos usuários do *Twitter* durante uma das semanas de maior alastramento do Corona Vírus.

Utilizando a API do *Twitter*, o *Tweepy* e a biblioteca *Textblob* os cientistas coletaram *tweets* e consequentemente plotaram gráficos para mostrar a eventual polaridade de sentimentos das informações obtidas contendo as *hashtags* #coronavirus e #COVID-19.

5. Apresentação da Aplicação

No decorrer desta seção, será demonstrado o funcionamento da aplicação, assim como o funcionamento interno do código até a obtenção dos gráficos finais como resultado.

Comumente para todos os *softwares* desenvolvidos em Python, são necessários alguns *imports* de bibliotecas para que seja possível obter métodos e funções essenciais para o trabalho (Figura 2).

Figura 2: *Imports* para aplicação

```
import tweepy
import json
import pandas as pd
import matplotlib.pyplot as plt
from textblob import TextBlob
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from datetime import datetime
from datetime import date
import re
import string
from collections import Counter
import sys
import os
import squarify
from PySimpleGUI import PySimpleGUI as sg
```

Fonte: Elaboração própria.

Conforme citado anteriormente, para que a aplicação funcione de maneira adequada se faz necessário que o usuário obtenha uma conta de desenvolvedor para utilizar a *Twittter API*. A partir deste fato, o desenvolvedor terá acesso a criação das chaves de acesso e *tokens* necessários para a autenticação da aplicação, essa criação é feita diretamente no "*Developer Portal - Twitter Developer"* ao criar uma aplicação. Após obtenção dessas *Keys* e *Secrets* é necessário armazená-las, para com isso poder criar a instância irá permitir possibilitar a busca dos tweets (Conforme a figura 3).

Figura 3: Chaves de acesso para busca dos dados

```
auth = tweepy.OAuthHandler('sochwawhF[redacted]',
'TMcyh5FaTdIw6ZgW2[redacted]')

auth.set_access_token('1400306796254879753-[redacted]',
'ZqAgJ3HGvucpUp9IE[redacted]')

ttapi = tweepy.API(auth)
```

Fonte: Elaboração própria.

Apesar de o *Tweepy* disponibilizar diversas formas de busca, este projeto utilizou a classe *tweepy.Cursor* que possibilita utilizar métodos da API em questão para fazer buscas paginadas de *tweets* que podem ser armazenados em uma varável.

A classe (Figura 4, *tweepy.Cursor*) aceita diversos métodos e atributos como parâmetros. Conforme pode ser observado na figura 4, neste caso foi utilizado o método *Search* para a busca dos *tweets* e os atributos "*q*" que contém o tópico escolhido pelo usuário, o *result_type* indicando que o desejado é uma busca a dados recentes e o método *items*

recebendo o total de *tweets* a ser buscado.

Figura 4: Método para busca dos dados

```
tweets = tweepy.Cursor(ttapi.search,
q=topic +' -filter:retweets',
result_type="recent").items(total_tweets)
```

Fonte: Elaboração própria.

Em seguida, a aplicação utiliza-se do *statement for*, uma ferramenta de controle de fluxo vastamente utilizada em Python. Esse *Loop* irá permitir que para cada iteração, um dos *tweets* buscados seja armazenado em um objeto Json utilizando a função *json.dumps* e classificado sentimentalmente utilizando a biblioteca *SentimentIntensityAnalyzer* do módulo *vader* da biblioteca NLTK (Figura 5).

Figura 5: Classificação dos *Tweets*

```
for tweet in tweets:
    json_data += json.dumps(tweet._json, separators=(',', ':'))
    analysis = TextBlob(tweet.text)
    score = SentimentIntensityAnalyzer().polarity_scores(tweet.text)
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']

    if neg > pos:
      negative += 1

    elif pos > neg:
      positive += 1

    elif pos == neg:
      neutral += 1
```

Fonte: Elaboração própria.

Posteriormente, já com o objeto *Json* populado se inicia o trabalho de transformação dos dados para que esse *dataset* bruto seja armazenado em um *dataframe* da biblioteca *Pandas.* Para isso é utilizada a função *json_normalize(),* essa atribuição permite a normalização de um objeto semiestruturado em um *dataframe* (Figura 6).

Figura 6: Transformação do *dataset* bruto.

```
json_correct_raw = '[' + json_data[:len(json_data)] + ']'
json_correct = json_correct_raw.replace('}{', '},{')

raw_data = json.loads(json_correct)
dataframe_raw = pd.json_normalize(raw_data)
```

Fonte: Elaboração própria.

O último passo antes de iniciar as transformações dos tweets é filtrar apenas os dados que são realmente necessários, pois nem todas as colunas do *dataframe* bruto serão utilizadas para a obtenção das informações. Para isso o nome das colunas desejadas é armazenado em uma variável e em seguida é utilizado o método *filter* da biblioteca Pandas retornando um *dataframe* contendo apenas o necessário (Figura 7).

Figura 7: Filtragem de colunas

```
selected_items = ['text', 'metadata.iso_language_code',
'retweet_count', 'favorite_count', 'created_at']

columns_filter = dataframe_raw.filter(items = selected_items)
```

Fonte: Elaboração própria.

5.1 Gráfico de *Ranking* de Idiomas

Para obter as informações necessárias para o gráfico de idiomas, gráfico o qual mostra em quais determinados idiomas o tópico escolhido está sendo mais falado, é utilizado o método *value_counts()*. Este meio irá retornar uma *Series* (*array* unidimensional indexada) que contém a contagem do total de ocorrências por valores únicos, ou seja, qual a distribuição de idiomas pelos *tweets* extraídos da rede social.

Prontamente, a partir dessa *Series* o método *tolist()* é utilizado para armazenar os índices em uma lista para os *labels* e os *values* em outra lista para os valores do futuro gráfico. Adicionalmente durante a criação da lista de *labels* é utilizado uma *list comprehension* para melhor visualização da legenda que será criada.

Figura 8: Variáveis gráfico de idiomas

```
language_ranking = columns_filter['metadata.iso_language_code'].value_counts()

language_labels = [x.upper() for x in language_ranking.index.tolist()]
language_values = language_ranking.values.tolist()
```

Fonte: Elaboração própria.

5.2 Gráfico de Principais Palavras

Inicialmente para a representação gráfica de principais palavras cria-se um filtro para buscar os *tweets* do idioma inglês (Devido a limitação que será citada nas considerações finais). Em seguida, são feitas transformações para tornar possível obter as principais palavras sendo utilizadas (*top words*) na base buscada no *Twitter*.

Dentre essas manipulações pode ser observado na figura 9 a remoção de caracteres indesejados usando *re.sub()* da biblioteca *Regex*, a conversão dos tweets para caixa baixa para meios de comparação consistentes utilizando *lower()*, a utilização da função *Counter* juntamente com o método *most_common()* para obter as palavras mais comuns e também a remoção de *stopwords* (Palavras usuais a serem ignoradas como *"the", "in" e "at"* por exemplo) utilizando o módulo *corpus* da biblioteca NLTK.

Após essas tratativas, devido a ordenação feita pela função *Counter*, é possível obter as 10 principais palavras armazenando-as em uma variável e suas respectivas quantidades de ocorrências em uma outra variável.

Figura 9: Variáveis gráfico de palavras

```
filter_language = columns_filter['metadata.iso_language_code'] == 'en'

dataframe_raw_en_only = columns_filter[filter_language]
dataframe_raw_en_only = dataframe_raw_en_only['text']

dataframe_no_user = dataframe_raw_en_only.apply(lambda tweet_txt: re.sub('@[^\s]+', '', tweet_txt))
dataframe_no_user = pd.DataFrame(dataframe_no_user)
dataframe_no_user.text = dataframe_no_user['text'].str.lower()

top_words = Counter(" ".join(dataframe_no_user.text).split()).most_common(5000)

selected_columns = ['Word', 'Appearances']
dataframe_words = pd.DataFrame(top_words, columns = selected_columns)

stop = stopwords.words('english')

dataframe_words['Word'] = dataframe_words['Word'].apply(lambda x: ''.join([Word for Word in x.split() if Word not in (stop)]))

len_min = 2
filter_words = dataframe_words['Word'].map(len) > len_min

dataframe_words = dataframe_words[filter_words]

dataframe_words = dataframe_words.head(10)

words = dataframe_words['Word'].str.upper()
appearances = dataframe_words['Appearances']
```

Fonte: Elaboração própria.

5.3 Gráfico de *Likes* e *Retweets*

Os *Likes* e *Retweets* são artifícios da rede social Twitter que demonstram popularidade e engajamento do *tweet,* com isso a partir da contagem dessas duas métricas presentes nos dados trazidos é feita uma soma total de ambas utilizando o método *sum()* e os valores de total de *likes* e total de *retweets* são armazenados em variáveis a serem utilizados no *plot* dos gráficos (Figura 10).

Figura 10: Variáveis gráfico de *likes e retweets*

```
total_likes = columns_filter.favorite_count.sum()
total_retweets = columns_filter.retweet_count.sum()

values_likes_retweets = [total_likes, total_retweets]
labels_likes_retweets = ['Likes' + ' [' + str(total_likes) +']', 'Retweets'  + ' [' + str(total_retweets) +']']
```

Fonte: Elaboração própria.

5.4 Gráfico de *Tweets* na Última Hora

Para obter um gráfico comparativo de quantos *tweets* dentro da gama obtida foram durante a última hora, primeiramente é necessário filtrar do *dataframe* apenas a coluna que se refere ao momento da criação daquele *post.*

Após isso, essa coluna é transformada em um novo *dataframe* Pandas que sofrerá uma manipulação de seus dados para que a variável de tempo esteja em um formato adequado para operações matemáticas, essa transformação ocorre utilizando o método *to_datetime()* que recebe a formatação desejada. Com isso, utilizando o módulo *datetime* do Python podemos obter o conjunto referente a última hora. Assim como nos exemplos anteriores, essas informações são armazenadas em variáveis para serem usadas posteriormente.

Figura 11: Variáveis do gráfico de última hora

```
dataframe_time = columns_filter.created_at

today = datetime.now()
dataframe_time = pd.DataFrame(dataframe_time)

dataframe_time["Convertido"] = pd.to_datetime(dataframe_time['created_at'], format= '%a %b %d %H:%M:%S +0000 %Y')

dataframe_time["Last Hour"] = ((dataframe_time["Convertido"] - today).dt.total_seconds() / 60)

ft_hr = dataframe_time["Last Hour"] >= -60
ft_ha_hr = dataframe_time["Last Hour"] >= -30
ft_qt_hr = dataframe_time["Last Hour"] >= -15
ft_min = dataframe_time["Last Hour"] >= -1

dataframe_lasthour = dataframe_time[ft_hr]

count_lh = dataframe_lasthour.shape[0]
count_total = total_tweets - count_lh
labels_tweets_time = ['Last hour ' + '[' +str(count_lh) + ' Tweets]', 'Previous Period' + ' ['+str(count_total) + ' Tweets]',]
values_tweets_time = [count_lh, count_total]
```

Fonte: Elaboração própria.

5.5 Gráfico de distribuição temporal

Em casos de assuntos extremamente populares, grande parte dos *tweets* extraídos ocorreram dentro da última hora. Com isso, foi desenvolvido um gráfico que busca expor a distribuição em camadas de tempo mais detalhadas, mostrando quantos ocorreram no último minuto, entre 1 e 15 minutos, entre 15 e 30 minutos, entre 30 e 60 minutos ou há mais de 1 hora.

Além de utilizar algumas varáveis obtidas a partir do “Gráfico de *Tweets* na Última Hora”, são feitas operações matemáticas de soma e subtração utilizando o *shape* na posição “0” dos respectivos *dataframes* que possibilitam obter os valores desejados de distribuição

por tempo.

Logo, é feito uma *list comprehension* para armazenar os valores e labels diferentes de zero em variáveis que serão utilizadas nas fatias do gráfico.

Figura 12: Variáveis gráfico de distribuição temporal

```
dataframe_percent = dataframe_time

dataframe_last_half_hour = dataframe_percent[ft_ha_hr]
dataframe_last_quarter_hour = dataframe_percent[ft_qt_hr]
dataframe_last_min = dataframe_percent[ft_min]

last_min_qt = dataframe_last_min.shape[0]
last_15min_qt = dataframe_last_quarter_hour.shape[0] - last_min_qt
last_halfhour_qt = dataframe_last_half_hour.shape[0] - (last_15min_qt + last_min_qt)
last_hour_qt = count_lh - (last_halfhour_qt + last_15min_qt + last_min_qt)

data_percent_time = {'Last Minute': last_min_qt, '1 and 15 minutes': last_15min_qt, '15 and 30 Minutes': last_halfhour_qt,
'30 minutos and 1 Hour': last_hour_qt, '1 Hour or more': count_total}

values_percent_posts = [value for value in data_percent_time.values() if value!=0]
labels_percent_posts = [key for key,value in data_percent_time.items() if value!=0]
```

Fonte: Elaboração própria.

5.6 Gráfico de Sentimentos

Para o gráfico de sentimentos, é apenas utilizado uma função para calcular a porcentagem de cada uma das possibilidades (positivo, negativo ou neutro) a partir dos valores obtidos no *Loop For* mencionado no início desta seção.

Figura 13: Variáveis gráfico de sentimentos

```
def percentage(part , whole):
  return 100 * float(part)/float(whole)

positive = percentage(positive, total_tweets)
negative = percentage(negative, total_tweets)
neutral = percentage(neutral, total_tweets)
positive = format(positive, '.1f')
negative = format(negative, '.1f')
neutral = format(neutral, '.1f')

labelsSentiments = ['Positive' , 'Neutral','Negative']
sizesSentiments = [positive, neutral, negative]
colorsSentiments = ['yellowgreen', 'blue','red']
```

Fonte: Elaboração própria.

5.7 *Interface* Gráfica do Usuário

Para criação da interface, foi utilizado o *PySimpleGUI,* esse layout desenvolvido é inicializado quando o sistema é executado e recebe do usuário um tópico de sua escolha. Caso este tópico não seja vazio, a estrutura de controle presente dá início a criação do *Dashboard*.

Figura 14: Código para interface gráfica

```
sg.theme('Reddit')
layout = [
    [sg.Text('Assunto'), sg.Input(key='topicIn')],
    [sg.Button('Pesquisar')]
    ]

janela = sg.Window('Dashboard Analytics', layout)

while True:
    eventos, valores = janela.read()
    if eventos == sg.WINDOW_CLOSED:
      break
    if eventos == 'Pesquisar':
      if  valores['topicIn'] != '':
        dash_func()
      else:
        print("Tópico inválido.")
```

Fonte: Elaboração própria.

5.8 *Plot* do *Dashboard*

Ao final, já com todas as informações necessárias para construção do *dashboard* de análise é feita o *plot* de todos os gráficos utilizando as variáveis concebidas a partir das diversas transformações mostradas ao longo desta seção. Cada gráfico tem sua particularidade de como receber essas variáveis se seus possíveis atributos o que pode ser observado na Figura 15.

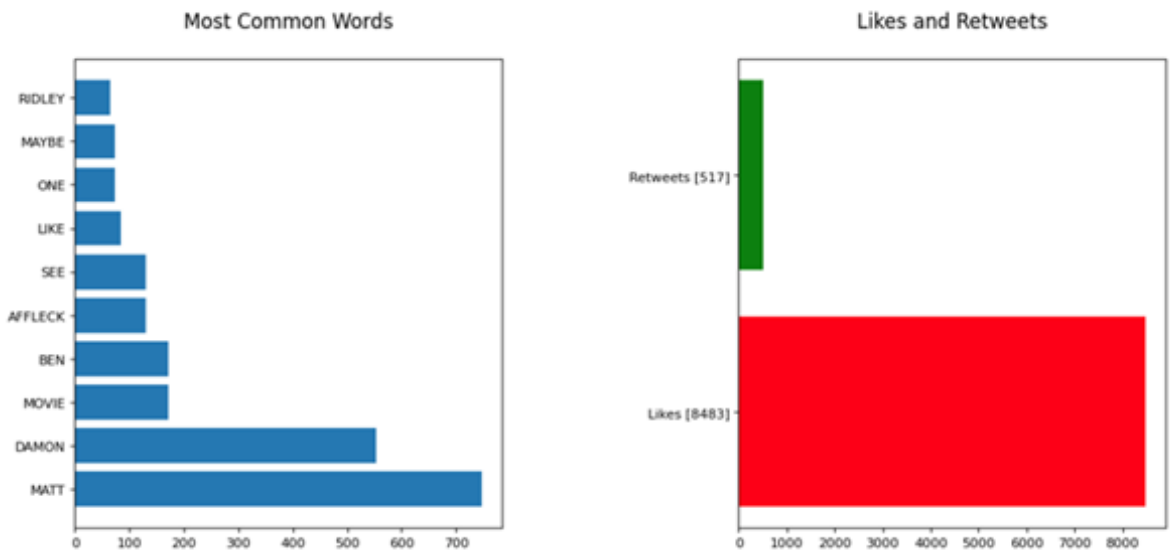
Figura 15: Código para plotagem dos gráficos

```
# Dashboard Final
final_dashboard = plt.figure(figsize=(10, 5), dpi=80)
final_dashboard.suptitle(str(topic) + ' Data Analysis', x = 1.5, y = 3.65 , fontsize=22)

# Plot de palavras
graph_words = final_dashboard.add_axes([0.25, 2.25, 0.45, 1])
graph_words.barh(words, appearances)
graph_words.set_title('Most Common Words',  y = 1.05, fontsize = 15)

# Plot de likes e retweets
graph_likes_retweets = final_dashboard.add_axes([1.25, 2.25, 0.45, 1])
graph_likes_retweets.barh(labels_likes_retweets, values_likes_retweets, color = ['red', 'green'])
graph_likes_retweets.set_title('Likes and Retweets', y = 1.05, fontsize = 15)

# Plot de idiomas
graph_language = final_dashboard.add_axes([2.25, 2.25, 0.45, 1])
graph_language = squarify.plot(sizes = language_values, label = language_labels[:5], alpha=.99 )
graph_language.set_title('Most Common Languages', y = 1.05, fontsize = 15)
graph_language.axis('off')

# Plot de sentimentos
graph_sentiments = final_dashboard.add_axes([0.25, 0.75, 0.45, 1], aspect=0.25)
graph_sentiments.pie(sizesSentiments, colors = colorsSentiments, autopct='%1.1f%%', radius = 1.5)
graph_sentiments.set_title('Sentiment Analysis', y = 1.15, fontsize = 15)
graph_sentiments.legend(labelsSentiments, bbox_to_anchor=(1.25, 1.0))

# Plot de distribuição percentual
graph_percentage_distribution = final_dashboard.add_axes([1.25, 0.75, 0.45, 1], aspect=0.25)
graph_percentage_distribution.pie(values_percent_posts, autopct='%1.1f%%', radius = 1.5)
graph_percentage_distribution.set_title('Tweet Posts Distribution', y = 1.15, fontsize = 15)
graph_percentage_distribution.legend(labels_percent_posts, bbox_to_anchor=(1.25, 1.0))

# Plot de contagem por períodos
graph_hour_count_time = final_dashboard.add_axes([2.25, 0.75, 0.45, 1])
graph_hour_count_time.bar(labels_tweets_time, values_tweets_time, color = ['Blue', 'Yellow'])
graph_hour_count_time.set_title('Last Hour Posts and Previous', y = 1.05, fontsize = 15)
```

Fonte: Elaboração própria.

Com o fim da execução, uma imagem é salva no diretório em que se encontra a aplicação.

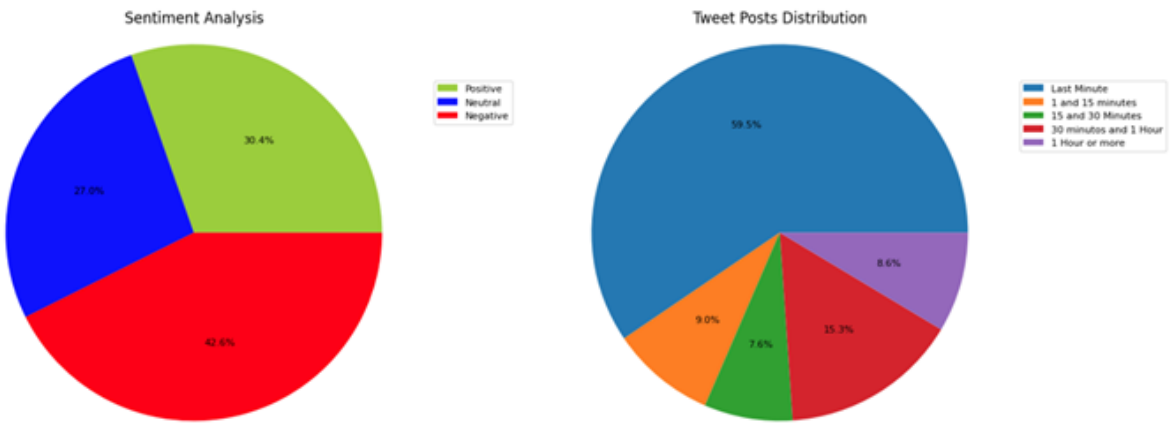
Figura 16: *Dashboard* analítico parte 1



Fonte: Elaboração própria.

Conforme pode ser observado na Figura 16, na parte superior do *Dashboard* gerado são apresentados 3 gráficos. O primeiro com título “Most Common Words” trata-se de um gráfico de barras, que contém um *ranking* das palavras que são mais utilizadas nos *tweets* extraídos. O segundo gráfico sinalizado por “Likes and Retweets”, apresenta também um gráfico de barras contendo a contagem de *Likes* e *Retweets* totais.

Figura 17: *Dashboard* analítico parte 2

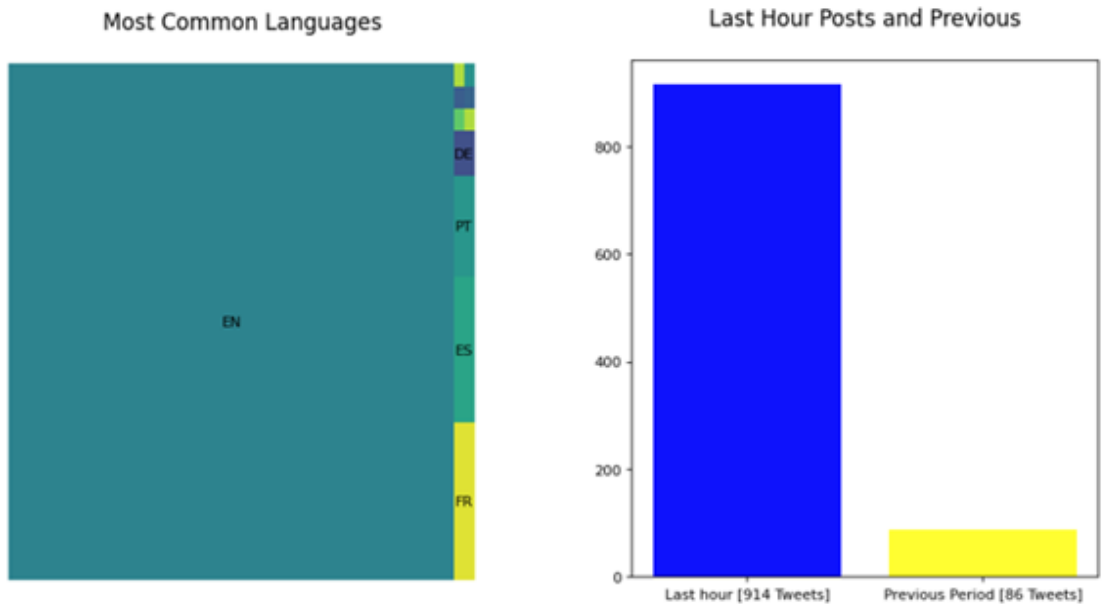


Fonte: Elaboração própria.

Na Figura 17, podemos observar os gráficos “Sentiment Analysis” e “Tweets Posts Distribution”. Ambos sendo um gráfico de setores, o primeiro faz a representação dos

sentimentos presentes no *tweets* separando por *Positive* (Positivo), *Negative* (Negativo) e *Neutral* (Neutro). Já o segundo retrata a distribuição temporal dos *tweets,* sinalizando entre as fatias dos gráficos a porcentagem por tempo das ocorrências das postagens.

Figura 18: *Dashboard* analítico parte 3



Fonte: Elaboração própria.

Por final, na imagem acima (Figura 18), podemos observar o gráfico “Most Common Languages” que exibi um *TreeMap*, este gráfico mostra os idiomas mais presentes sendo retratados por tamanho com base em sua ocorrência nos *tweets* extraídos, sendo quanto mais presente maior a área representada. E o gráfico “Last Hour Posts and Previous”, esta representação é um gráfico de barras que mostra quantos desses *tweets* ocorreram na última hora ou anteriormente.

6. Considerações Finais

Como resultado deste projeto, obteve-se um *Dashboard* analítico que possibilita de maneira visual um entendimento inicial a respeito do sentimento geral de uma rede social no atual momento, observando que as tecnologias atuais podem acelerar e melhorar a forma com o ser humano recebe e transforma grande quantidade de dados em um curto período. Apresentando-se com isso uma ferramenta que visa expor as possibilidades de implementação visando a análise de dados com base em extrações de dados via API.

O presente trabalho, para versões futuras dessa aplicação cabe transpor algumas limitações existentes, assim como melhorias para a ferramenta. Dentre essas limitações, está a falta de tratativa de erros decorrentes de possíveis falhas ou sobrecarga da API do Twittter, ressaltando também que o número de *tweets* extraídos por vez é limitado de acordo com a

versão gratuita da Twitter API que pode variar, porém, tais valores podem ser consultados diretamente no portal de desenvolvedores do *Twitter*. Outra limitação existente é a análise de sentimentos e o gráfico de principais é limitado ao idioma inglês, sendo assim uma limitação para a geração do *Dashboard* com os demais idiomas.

Citando futuras melhorias, além das limitações acima citadas, é sugestivo a melhoria da interface gráfica presente, trazendo mais campos a serem preenchidos para que a interação do usuário com o código fonte seja menor. Dentre esses campos, podem ser citados o número de *tweets* desejados, as *Keys* e *Tokens*, assim como o idioma desejado para geração do *Dashboard*.

A ferramenta apresentada neste trabalho, pode ser acessada no repositório https://github.com/diegompazos/DashboardAnaliticoTwitter, assim como seu funcionamento pode ser observado no link https://youtu.be/g240lzilVSM.

Referências

OpenEDG PYTHON INSTITUTE. **About Python.** Online. 2021. Disponível em: <https://pythoninstitute.org/what-is-python/>

PYTHON SOFTWARE FOUNDATION. **The Python Tutorial.** Online. 2021. Disponível em: <https://docs.python.org/3.9/tutorial/index.html>

THE MATPLOTLIB DEVELOPMENT TEAM. **Matplotlib: Visualization with Python.** Online. 2021. Disponível em: <https://matplotlib.org/stable/index.html>

NLTK PROJECT. **Natural Language Toolkit.** Online. 2021. Disponível em: <https://www.nltk.org/index.html#natural-language-toolkit>

PANDAS DEVELOPMENT TEAM. **User Guide.** Online. 2021. Disponível em: <https://pandas.pydata.org/docs/user_guide/index.html>

Roesslein, Joshua. **Tweepy Documentation.** Online. 2021. Disponível em: <https://docs.tweepy.org/en/stable/#>

TWITTER, INC. **Twitter API Documentation.** Online. 2021. Disponível em: <https://developer.twitter.com/en/docs/twitter-api>

KIMBALL, Ralph; ROSS, Margy. **The Data Warehouse Toolkit:** The Complete Guide to Dimensional Modeling. 2nd ed. Wiley Computer Publishing, 2002.

BIRD, Steven; KLEIN, Ewan; LOPER, Edward. **Natural Language Processing with Python**. O'Reilly Media Inc, 2009.

TAO, Dandan; YANG, Pengkun; FENG, Hao. **Utilization of text mining as a big data analysis tool for food science and nutrition**. Comprehensive Reviews In Food Science and Food Safety: WILEY, 2020, p. 875-894.

WATSON, Richard B.; RYAN, Peter J. **Big Data Analytics in Australian Local Government**. Smart Cities, 2020, p. 657-675.

UMA, J.; PRABHA, Dr. K. **SENTIMENT ANALYSIS IN MACHINE LEARNING USING TWITTER DATA ANALYSIS IN PYTHON**. International Journal of Advanced Research in Engineering and Technology, Volume 11, Issue 12, 2020, p. 3042-3053.

SAPRE, Atharva; VARTAK, Shubham. **Scientific Computing and Data Analysis using NumPy and Pandas.** International Journal of Advanced Research in Engineering and Technology, Volume 7, Issue 12, 2020, p. 1334-1346.

MANGURI, Kamaran H.; RAMADHAN, Rebaz N.; AMIN, Mohammed P. R. **Twitter Sentiment Analysis on Worldwide COVID-19 Outbreaks.** Kurdistan Journal of Applied Research, Special Issue on Coronavirus (COVID-19), 2021.

DIYASA, I Gede Susrama M. *et al***. Twitter Sentiment Analysis as an Evaluation and Service Base On Python Textblob.** IOP Conference Series: Materials Science and Engineering, 2021.

SIAL, Ali Hassan; RASHDI, Syed Y. Shah; KHAN, Dr. Abdul H. **Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python**. International Journal of Advanced Trends in Computer Science and Engineering, Volume 10, Nº 1, 2021, p. 277-281

ISHWARAPPA; J, ANURADHA. **A Brief Introduction on Big Data 5 Vs Characteristics and Hadoop Technology**. International Conference on Intelligent Computing, Communication & Convergence, Procedia Computer Science 48, 2015, p. 319-324

PERASSO, V. **O que é a 4ª revolução industrial - e como ela deve afetar nossas vidas.** BBC News. 2016. Acesso em 10/09/2021. Disponível em: <https://www.bbc.com/portuguese/geral-37658309>

HOLST, A. **Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2025.** Statista, 2021. Acesso em 08/09/2021. Disponível em: <https://www.statista.com/statistics/871513/worldwide-data-created>

PROVOST, F.; FAWCETT, T. **DATA SCIENCE AND ITS RELATIONSHIP TO BIG DATA AND DATA-DRIVEN DECISION MAKING.** Mary Ann Liebert, Inc, Volume 1, Nº1, 2013, p51-58