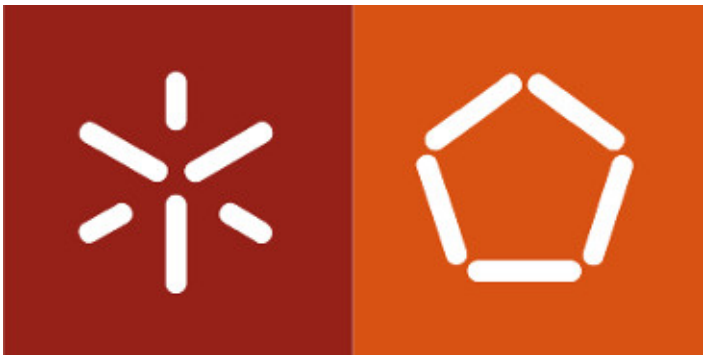


Universidade do Minho

Mestrado Integrado em Engenharia Informática
Laboratório em Engenharia Informática

Deteção de rumores em microblogs por via de interação de utilizadores

---

Orientadores
Professor Marco Gomes
Professor Paulo Novais

Autores
Adriana Meireles - A82582
Nuno Silva - A78156
Pedro Pinto - A82535

19 de Julho de 2020

Conteúdo

1 Introdução

Na última década, um dos desenvolvimentos tecnológicos mais evidenciado encontra-se relacionado com o crescimento de novos meios de comunicação de partilha de dados e informações. Em consequência dos desafios da globalização e com o avançar da tecnologia, denotou-se uma intensa capacidade de propagação de informação através destes meios mais digitais que contrasta com a decrescente disseminação de outros meios de comunicação social, como é o caso dos jornais, de rádios ou até mesmo da televisão. Assim sendo, é altamente propenso difundir informações, cujo conteúdo pode advir de fontes de informação fidedignas apresentando à priori um teor verdadeiro ou pelo contrário, podem derivar de fontes de informação mais duvidosa originando mensagens falsas, levando à criação de rumores.

A existência de rumores remonta aos primórdios da Humanidade, sendo que estes já conduziram a diversos impactos nefastos na sociedade. A título exemplificativo, podem destacar-se momentos como o Holocausto, que marcou toda uma sociedade, religião e rumo da vida Humana. Nessa época, os rumores transmitidos pela propaganda Nazi passaram pela disseminação de uma imagem distorcida e falsa da comunidade judaica, que levou ao genocídio em massa de cerca de seis milhões de judeus.

Em consequência do elevado grau de impacto que os rumores podem atingir, o foco deste trabalho centra-se na criação de um meio capaz de testar a veracidade de dados recolhidos, em plataformas de microblogging, neste caso o Twitter, e as consequentes informações divulgadas relativas à família SARS-CoV-2 que provocou o vírus da COVID-19 e que afetou a população a nível mundial no ano de 2020. O processo de recolha dos dados ocorrerá por via de uma ferramenta denominada *twint*.

2 Revisão Literária

Numa fase prévia a uma abordagem mais prática, foi feita uma análise sobre artigos científicos, correlacionados com foco no tema que se vai abordar. Assim sendo, de acordo com *Hamidian e Diab*, membros do Departamento de Ciências da Computação da Universidade de George Washington e autores do artigo científico *Rumor Detection and Classification for Twitter Data*, estes referem que o foco da sua investigação se centrou na deteção de rumores como forma de propagação de desinformação e execução de tarefas capazes de os classificar. Estes investigadores estudaram ainda o impacto que um rumor poderá atingir e como é que uma plataforma de *microblogging*, como é o caso do Twitter, poderá exponenciar o alcance de uma informação. No que concerne à fiabilidade e origem das informações, os autores tiveram em conta quatro fatores essenciais, nomeadamente a proveniência (origem do conteúdo), a fonte (quem foi o autor), o local e a data (quando e onde foi criada a informação).

Neste artigo, as informações recolhidas, cerca de 9000 registos através da API do *Twitter*, incidem sobre possíveis rumores acerca da religião de Barack Obama. Para além das *features* já presentes nos dados recolhidos, foi também adicionado conteúdo pragmático, revelando detalhes como por exemplo o impacto das *features* para a *task*, análise sentimental, análise de "*emojis*"utilizados, reconhecimento de nomes e entidades, entre outras. A partir destes, foram utilizadas as *frameworks* WEKA, com o intuito de treinar e testar os modelos gerados, e o J48, um classificador discriminativo que utiliza *decision trees*. Neste último, são realizadas duas abordagens, uma *multi-step* e uma *single-step*, já numa fase mais avançada, os dados são separados em 80% treino, 10% teste e 10% desenvolvimento. Por fim, os resultados apresentados pela abordagem *multi-step* são melhores que com *single-step*, revelando valores na ordem dos 82.9% e 74%, respetivamente, para *F-measure*.

Outro artigo estudado foi o *Learning Graph Influence From Social Interactions* dos autores *Vincenzo Matta*, *Virginia Bordignon*, *Augusto Santos* e *Ali H. Sayed*. Este trabalho considera o paradigma recente dos gráficos "fracos"em que a rede é dividida em componentes de envio e receção, com o primeiro a exercer um efeito dominador sobre o segundo sendo este efeito semelhante ao existente nas plataformas sociais. O artigo foca-se no mecanismo de aprendizagem dual ou inversa (que examina como os agentes aprendem). A partir da observação do que os agentes aprendem (SL,*Social Learning*) tenta descobrir-se como é que são influenciados pelos agentes de envio (TL, *Topology Learning*). Os autores estabeleceram que uma condição necessária para a TL ser consistente é que o número de hipóteses H seja pelo menos igual ao número de componentes S de envio.

Foram examinados dois modelos sendo que o primeiro é um modelo gaussiano estruturado no qual as sub-redes de envio usam a mesma família de probabilidades gaussianas, e as distribuições são escolhidas dentro dessa família e são distintas nas sub-redes de envio. Mostrou-se que, para este modelo, o TL só é viável quando S = 2, devido à diversidade limitada nas sub-redes enviadas. Nesse sentido, foi examinado um segundo modelo, em que as probabilidades e as distribuições exibem uma determinada diversidade. Para este caso, mostramos que o problema do TL é viável com probabilidade desde que $H \geq S$. Em suma, as duas características mais críticas que permitem consistência do TL são: mais hipóteses do que componentes enviados e um grau suficiente de diversidade.

Foi ainda analisado outro artigo, intitulado de *"Spreading Groups in Twitter: The Flow of Rumors about Stocks"* dos autores *Alon Sela, Orit Milo-Cohen, Irad Ben-Gal* e *Eugene Kagan.*

Este debruça-se sobre o método de difusão de mensagens nas redes sociais, através de aceleradores denominados por *spreading groups*. Estes grupos são responsáveis por iniciar a difusão da mensagem que, gradualmente, vai alcançando um maior número de utilizadores da rede.

Foram identificados três cenários de difusão de mensagens. No primeiro, chamado de *naive spreader*, é relativo à situação em que um utilizador desconhecido partilha uma mensagem interessante no *Twitter*, e esta torna-se viral. O segundo cenário, chamado de *guru spreader* está associado à existência de um "guru"que tem os seus seguidores. Neste cenário a mensagem é publicada pelo utilizador conhecido(guru) e é partilhada pelos seus seguidores, que acham a mensagem interessante. Finalmente, no terceiro cenário, o *spreading group*, onde um grupo definido de utilizadores pretende difundir a mensagem através de várias repetições dentro do grupo. Em suma, os *spreading groups* funcionam como catalisadores para a difusão da mensagem.

3 Caso de Estudo

Nesta secção irá ser realizada uma contextualização do caso de estudo, realizando-se ainda uma descrição das temáticas das próximas secções. Primeiramente, é introduzida uma definição de rumor, uma contextualização sobre a pandemia em questão e ,posteriormente, é descrita a plataforma de *microblogging* ao qual iremos retirar os dados que serão analisados.

3.1 Definição Rumor

Segundo o dicionário de *Oxford*, um rumor é

> "um pedaço de informação/história, não validada, comunicada entre pessoas". [7]

Já para os autores do artigo científico *Rumor Detection and Classification for Twitter Data*, Sardar Hamidian e Mona Diab,

> "(...) uma alegação cuja veracidade está em dúvida e não possui uma fonte clara, mesmo que suas origens e intenções ideológicas ou partidárias sejam claras." [1]

Assim sendo, pode concluir-se que um rumor é uma partilha de informação infundada, com a intenção de ser tomada como verdadeira.

3.2 COVID-19 no twitter

Coronavírus é uma família de vírus que podem causar infeções. Por norma, as mesmas afetam o sistema respiratório, podendo apresentar similaridades com a gripe ou até evoluir para doenças mais graves como, por exemplo, a pneumonia.[2]

O Covid19 é uma doença causada por este novo tipo de coronavírus, que foi primeiramente identificado em Wuhan, província de Hubei na China. [3]

Os portadores deste novo vírus apresentam diversos sinais e sintomas que poderão conduzir a infecções respiratórias agudas, ligeiras e moderadas. Podem apresentar febre ($T > 37.5^{o}C$), tosse e falta de ar. Nos casos mais alarmantes poderá causar pneumonia grave, falência renal e de outros órgãos e, eventualmente, a morte. Contudo, a maioria dos casos recuperam, podendo ou não ficar com sequelas, com exceção da população envelhecida que apresenta maior risco de letalidade. Com a evolução da pandemia, grupos etários mais jovens, que numa fase inicial da pandemia eram considerados de pouco risco, apresentam uma maior incidência nos dados estatísticos diários. [2] [8]

Twitter é uma plataforma de *microblogging* que permite a partilha de posts, os chamados *"tweets"*, limitados por 140 carateres, onde se poderá incluir links de outros websites e recursos. Um utilizador é assim capaz de criar os seus próprios *"tweets"* ou então *"retweetar"*, repartilhar, os posts de outras pessoas. [4] Os utilizadores podem ainda seguir outros utilizadores nos quais têm interesse para que os seus *"tweets"* apareçam na sua *"timeline"*, ou colocar *"like"* nos *"tweets"* e *"retweets"*.

Apesar de o Twitter não disponibilizar na totalidade o seu sistema de ranks de *tweets* da *timeline*, baseado numa publicação realizada num blog em 2017[5], é possível identificar quatro principais fatores de ponderação que dão relevância a um *tweet*:

- **Recently:** Quão recentemente o *tweet* foi publicado.

- **Engagement:** Relativo ao número de *Retweets*, *clicks*, favoritos que um *Tweet* alcançou.
- **Rich Media:** Que tipo de informação está no Tweet, como imagens, vídeos e *GIFs*.
- **Activity:** Relativo à atividade do utilizador, como por exemplo, quando foi a última vez que o utilizador foi visto no site, quantas vezes usa a plataforma ou quantos seguidores tem.[6]

Assim, é fácil perceber que os utilizadores com uma rede maior de seguidores conseguem ter um maior alcance com os seus *tweets* e, consequentemente, têm muito mais facilidade em espalhar eventuais rumores. Com estes fatores e dada a dimensão da plataforma em questão, torna-se altamente propensa a partilha de informações, principalmente com eventos de grande impacto como, por exemplo, o COVID19.

3.3 Fatores que incentivam a existência de rumores

Há diversos fatores que podem influenciar a existência e criação de rumores. Desde uma ação involuntária, derivada da falta de informação, até ações propositadas e mal intencionadas.

Com o poder emergente das redes sociais, torna-se cada vez mais fácil a criação de rumores pois qualquer utilizador pode tecer comentários relativamente a determinado assunto, que pode ser facilmente difundido através de *influencers*, que são utilizadores com uma grande quantidade de seguidores, ou através de *spreading groups*.

O poder de comunicação e persuasão pode ser uma arma muito poderosa, permitindo influenciar e manipular massas de população. Deste modo, os rumores podem ser utilizados com finalidades perigosas, uma vez que, têm a capacidade não só de difamação como a capacidade de influenciar as pessoas a tomar atitudes menos corretas como, por exemplo, a desestabilização de governos tentando tirar partido de vantagens políticas/financeiras, destruição de reputações de figuras públicas e até mesmo incentivos a práticas criminosas e violentas.

4 Dataset

Antes de se proceder a uma recolha dos dados no Twitter, dessa forma e com o sentido de recolher o maior número de dados possível, e por sua vez, resultados mais fidedignos, efetuámos o levantamento de fontes sobre o qual se irá testar a veracidade. Por exemplo, utilizadores que escreveram posts identificados com *"hashtags"* usualmente utilizadas sobre a matéria em questão. De seguida apresentamos uma tabela contendo contas de organizações nacionais ou mundiais ligadas por exemplo, à saúde pública.

Fontes credíveis	
Utilizador	**Razão**
@WHO	Organização Mundial Saúde
@UN	Nações Unidas
@amnesty	Amnistia Internacional
@ICRC	Cruz Vermelha
@UNESCO	Organização Educacional, Científica e Cultural das Nações Unidas
@NHSuk	Serviço Nacional Saúde Reino Unido
@GovCanHealth	Serviço Nacional Saúde Canadá
@CDCgov	Centro para Prevenção e Controlo de Doenças
@NIH	Institutos Nacionais de Saúde
@EuroRespSoc	Sociedade Europeia para Doenças Respiratórias

Outras fontes identificadas são, por exemplo, os jornais e/ou canal de informação, Correio da Manhã, SIC, SIC Noticias, TVI24, RTP Noticias assim como diversos *hashtags* que foram utilizados recorrentemente, entre os quais se verifica:

#COVID	#coronavirus
#stayhome	#nomask

Assim sendo procedemos à recolha dos dados, por via do *twint*. Nestes dados recolhidos, estão presentes informações tais como: datetime, username, verificação da conta, número de seguidores, conteúdo do tweet, número de gostos, número de retweets, hashtags, número de hashtags.

4.1 Credibilidade de uma fonte

Para determinar a credibilidade de uma fonte devemos ter em conta vários fatores:[15]

- Deve ser realizada uma pesquisa sobre o autor com o objetivo de perceber se o mesmo é qualificado para falar sobre o assunto em questão;[16]
- Deve garantir-se que a informação é recente e está atualizada;
- Deve ser realizada uma pesquisa sobre a instituição que publicou a informação;
- Deve determinar-se qual o público-alvo da publicação para que a informação apresentada não seja mal interpretada;
- Deve analisar-se críticas relativas à informação escrita na publicação;
- Deve ser analisada a parcialidade da informação, assim como aferir se o seu conteúdo é de cariz objetivo ou subjetivo;

- Deve ser analisada a consistência da informação, garantindo que não se baseia em factos contraditórios.

4.2 Procedimento para a recolha de dados

Para a extração dos dados, utilizou-se um repositório encontrado no GitHub[1], relacionado com o *twint*, que é uma ferramenta avançada de *scraping* de Twitter, escrita em Python, que permite o *scraping* de *tweets* de perfis de utilizadores sem usar a API do Twitter.

O *Twint* utiliza operadores de pesquisa do Twitter para permitir o rastreio de *tweets* de determinados utilizadores relacionados com determinados tópicos, hashtags e tendências, ou classificar informações confidenciais de *tweets*, como email e números de telefone. Possui também capacidade para consultas especiais ao Twitter, permitindo rastrear os *tweets* dos seguidores de um determinado utilizador, os *tweets* que gostou e se seguem alguém sem nenhuma autenticação.

Existem várias opções de comando linha (CLI) disponíveis no repositório do GitHub, a que foi dada mais ênfase foi **twint -u username -s pal**, que rastreia os *tweets* que contenham a palavra "pal". De forma a obter *tweets* do nosso interesse, e referentes ao contexto em que nos focamos, utilizamos os hashtags referidos anteriormente, como meio de filtro. Como "username"usamos as fontes credíveis também mencionadas acima.

4.3 Junção dos dados recolhidos

Após a recolha da informação conforme referido anteriormente, possuíamos vários ficheiros .csv. Deste modo, foi necessário "juntá-los"num único ficheiro para um melhor tratamento dos dados. Para tal criou-se um script, em python, chamado **join_files**, como se pode observar em baixo:

```
import os
import sys
import pandas as pd
import csv

#Collect all the filenames
files = [f for f in os.listdir('.') if os.path.isfile(f)]
files.remove('join_files.py')

#Write the output
for f in files:
 open('train_validated.csv','a+', encoding="utf8").writelines([l for l in open(f,
  ↪  encoding="utf8").readlines()])

dataset = pd.read_csv('train_validated.csv', encoding="utf8")
text = list()
for line in dataset.values:
```

[1] https://github.com/twintproject/twint

```
    #Remove duplicated tweets
    if line[10] in text:
        dataset = dataset[dataset.tweet != line[10]]
    else:
        text.append(line[10])

dataset.to_csv('train_validated.csv', encoding="utf8", index=False)
```

Este script recolhe todos os ficheiros da diretoria atual, escrevendo-os para um ficheiro de output(*train_ validated*). Posteriormente, lê esse ficheiro e apenas considera os tweets que não sejam repetidos.

4.4 Tweets na mesma Língua

Como as fontes de informação são as mais variadas e relativas a alguns países, os tweets não se encontravam todos na mesma língua que queríamos, neste caso em Inglês. Para tal criou-se um script, em python, chamado **filterEn**,como se pode observar em baixo:

```
import pandas as pd
import googletrans
from googletrans import Translator

translator = Translator()

df_port = pd.read_csv('rumores.csv',encoding="utf8")

df_en = df_port.copy()

translations = {}
unique_elements = df_en['tweet'].unique()

for element in unique_elements:
    # add translation to the dictionary
        translations[element] = translator.translate(element).text

df_en.replace(translations, inplace = True)

df_en.to_csv('rumoresIngles.csv',encoding="utf8", index=False)
```

Para a construção deste script recorreu-se à biblioteca *Googletrans*. Primeiramente, é feita uma cópia do dataframe. Depois, para cada linha referente à coluna **tweets**, o tweet é traduzido no caso de não estar já em inglês. Por último, os termos do dataframe são modificados, aplicando o dicionário anteriormente criado como input da função.

4.5 Adicionar a label rumor/não rumor ao dataset

Com o intuito de tornar o nosso algoritmo supervisionado, foi necessário acrescentar uma coluna ao dataset que pode assumir três valores:

- **0**: rumor
- **1**: não rumor
- **2**: inconclusivo

Foi criado um script, em python, chamado **addCol**, como se pode observar em baixo:

```
import os
import sys
import pandas as pd
import csv

df = pd.read_csv('train_validated.csv',encoding='utf-8')

if (os.path.isfile('train_validated.csv')):

        df.loc[df['username'] != None,'bool'] = 1

        df.loc[df['username'] == 'realdonaldtrump','bool'] = 0
        df.loc[df['username'] == 'cmjornal','bool'] = 0
        df.loc[df['username'] == 'frasesdem3rda','bool'] = 0
        df.loc[df['username'] == 'borisjohnson','bool'] = 0

        df.loc[df.hashtags.str.match('.*#nomasks.*'),'bool'] = 2

df.to_csv('train_val_col.csv', encoding="utf8", index=False)
```

Como se pode analisar, por default o valor da nova coluna é 1. Caso se encontre, na coluna *username*, os nomes **realdonaldtrump**, **cmjornal**, **frasesdem3rda** ou **borisjohnson** são considerados fontes não credíveis e, portanto, rumores e o seu valor é 0. Caso o tweet contenha o *hashtag* **nomasks** decidimos considerar como inconclusivo(atribuindo o valor 2) , uma vez que, poderiam ser pessoas a criticar o uso de máscaras bem como pessoas a criticar quem afirma isso.

4.6 Análise de Dados

Antes de se proceder à aplicação de qualquer algoritmo, é necessário uma análise sobre os dados recolhidos. Dessa forma, foi criado um perfil do *dataset*, através do *pandas-profiling*. Esta biblioteca demonstra-se extremamente útil, possibilitando uma análise mais gráfica sobre diversos detalhes. Dessa forma observou-se a seguinte tabela, que nos indica as estatísticas mais superficiais do nosso *dataset*.

Dataset statistics

Number of variables	35
Number of observations	10712
Missing cells	127645
Missing cells (%)	34.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	20.1 MiB
Average record size in memory	1.9 KiB

Variable types

CAT	16
UNSUPPORTED	11
NUM	4
URL	2
BOOL	2

Figura 1: Estatísticas sobre o dataset

Como podemos ver, o nosso *dataset* possui cerca de 10712 linhas e 35 colunas. Destas colunas, apenas iremos utilizar duas, referentes aos *tweets* e à sua *label*, ou seja, se é um rumor ou não, ou até mesmo algo inconclusivo. Num olhar sobre os utilizadores que mais contribuíram para a construção deste conjunto de dados, podemos constatar que as organizações com afiliações a áreas da saúde, partilham um maior número de *tweets* como seria esperado.

Value	Count	Frequency (%)
World Health Organization (WHO)	2839	26.5%
United Nations	782	7.3%
CDC	602	5.6%
UNESCO	569	5.3%
Health Canada and PHAC	534	5.0%

Figura 2: Top 5 de contribuições para os dados recolhidos

Como se pode observar graficamente, a **WHO**, **United Nations**, **CDC**, **UNESCO** e **Health Canada and PHAC** são as cinco principais entidades para os dados recolhidos.

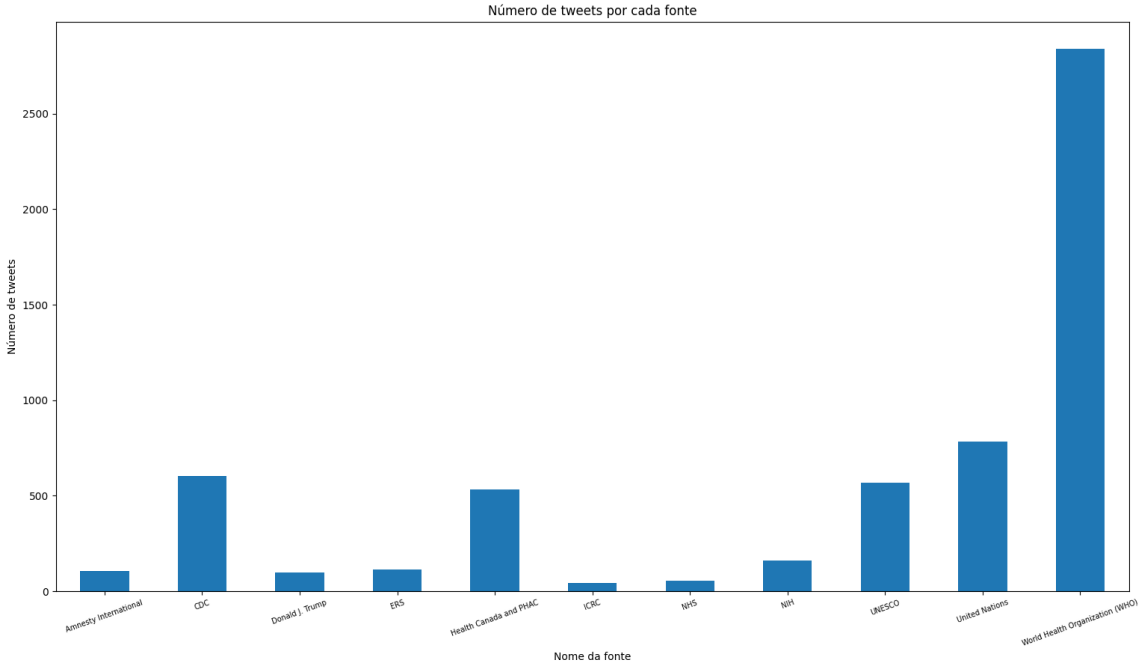


Figura 3: Número de tweets por fonte de informação

Para observar a evolução da pandemia, foi interessante analisar um gráfico, construído a partir de um script em python, que mostra a evolução dos tweets sobre covid-19 desde Janeiro a Julho de 2020. É possível notar que se atinge o pico no mês de Abril, o qual coincide com o período de quarentena obrigatória um pouco por todo o mundo.

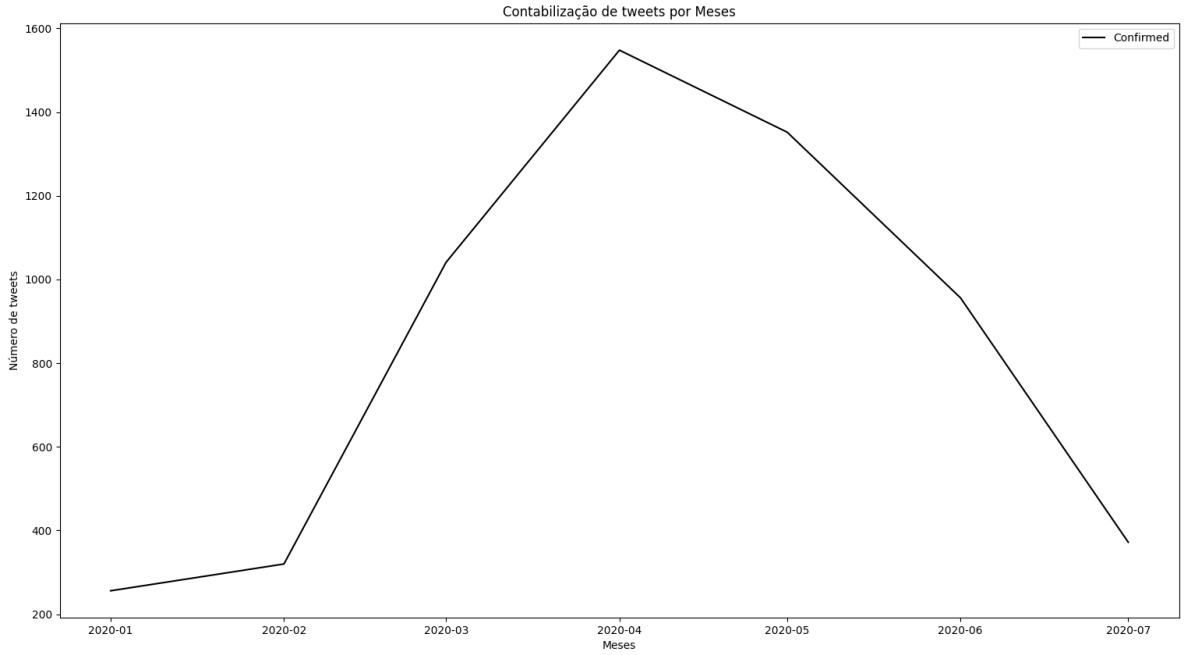


Figura 4: Evolução dos tweets ao longo dos meses

4.7 Tratamento de Dados

Numa fase prévia à aplicação do algoritmo, é necessário um tratamento sobre os dados. Este irá basear-se na remoção de colunas desnecessárias, para o nosso caso de estudo, assim, apenas ficaremos com as colunas referentes aos *tweets* e à sua *label*. Posteriormente, o *dataframe* importado foi dividido em *dataset* de treino e de teste, sendo que a divisão foi feita em cerca de 70% e 30%, respectivamente. Contudo, foram feitas mais duas divisões similares a estas, com o intuito de otimizar a *accuracy*, porém para o *dataset* de treino ficou com 60% e 80%, e para o teste o restante dessas divisões. De seguida iremos explicitar a forma como os dados dos *datasets* de treino foram tratados, de modo ao nosso modelo tirar o melhor partido deles.

Relativamente aos *tweets*, poderão apresentar um conteúdo de todo o tipo, possuindo diversos caracteres, *emojis*, *urls*, imagens, etc. Dessa forma é importante "normalizar"o máximo possível o *tweet*, removendo tudo o que poderá diferenciá-lo perante os outros. Assim, é feita uma substituição de todo o conteúdo que não sejam palavras e/ou letras, por um espaço. Daí todas as palavras são diminuídas e separadas individualmente, tornando um *tweet* numa lista de *strings*.

Após este tratamento sobre os *tweets*, é criado um vocabulário que irá conter todas as *palavras* que estão na lista de *strings* mencionada anteriormente. De seguida, é criado um dicionário cuja chave será cada uma das *strings* presentes no vocabulário, e o valor será uma lista, em que cada índice dessa mesma corresponderá ao número de registos dessa string num certo *tweet*. Gerando o seguinte *dataframe*:

	chair	38d3tg5	intensify	propose	braiets	landscape
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
...	...	...	...	...	...	...
7493	0	0	0	0	0	0
7494	0	0	0	0	0	0
7495	0	0	0	0	0	0
7496	0	0	0	0	0	0
7497	0	0	0	0	0	0

Figura 5: Excerto do dataframe gerado

Este *dataframe* será posteriormente concatenado com o *dataframe* de treino, formando um novo conjunto de dados que será sujeito ao nosso algoritmo.

	Complete_Message	Class_label	chair	38d3tg5	intensify	propose	braiets	landscape
0	[don, t, care, what, he, has, to, say, about, ...	2.0	0	0	0	0	0	0
1	[ers, covid19, blog, series, is, written, by, ...	1.0	0	0	0	0	0	0
2	[plandemic, scamdemic, coronabollox, nomasks, ...	2.0	0	0	0	0	0	0
3	[while, many, are, staying, inside, during, co...	1.0	0	0	0	0	0	0
4	[nomasks, https, twitter, com, mareq16, status...	2.0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...

Figura 6: Excerto do novo dataframe de treino

5 Algoritmo

No sentido de obter os melhores resultados, com pouco treino, dado a escassez de dados recolhidos, foi escolhido o uso do algoritmo *Naive Bayes*. Este é um algoritmo probabilístico, que nos permite calcular probabilidades condicionais, de ocorrências de dois eventos, baseados nas probabilidades individuais.[9]

5.1 Classificadores Naive Bayes

Como indicado anteriormente, o algoritmo que iremos utilizar para proceder a estas classificações é o *Naive Bayes*. Este faz uso do teorema de *Bayes*, que procura a probabilidade de um evento ocorrer na probabilidade de um outro evento que já ocorreu.[10]

Supondo **B**, uma classe dependente e **A**, que representa um evento prévio a esse, poderemos calcular **B**, segundo o teorema:[11][12]

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Pela independência condicional assume-se que:

$$P(A_i|B, A_1, ..., A_n) = P(A_i|B)$$

Onde podemos simplificar para todo o i, ficando com:

$$\frac{P(B)\prod_{i=1}^{n} P(A_i|B)}{P(A_i,...,A_n)}$$

Como a probabilidade de um evento acontecer já é dada, podemos então concluir que:

$$\widehat{B} = \arg\max_{y} P(B)\prod_{i=1}^{n} P(A_i|B)$$

Ou seja, a probabilidade de uma classe B acontecer, será o produto máximo entre a probabilidade de B pelo produtório entre as probabilidades condicionadas A dependente de B. Por exemplo, a seguinte frase: *"O João gosta de jogar futebol"*, a partir do cálculo mencionado anteriormente podemos determinar a sua veracidade da seguinte forma:

P(O João gosta de jogar futebol|Verdade) = P(O|Verdade) * P(João|Verdade) * P(gosta|Verdade) * P(de|Verdade) * P(jogar|Verdade) * P(futebol|Verdade)

P(O João gosta de jogar futebol|Falso) = P(O|Falso) * P(João|Falso) * P(gosta|Falso) * P(de|Falso) * P(jogar|Falso) * P(futebol|Falso)

Após a execução destes cálculos, a classe que apresentar uma maior probabilidade para esta situação, determina a sua veracidade ou não. Contudo, existe ainda a possibilidade de acontecer uma probabilidade nula, ou seja, um dado evento não se suceder para nenhuma das classes. Uma possível solução é a aplicação de uma suavização de Laplace, em que a ideia base é aumentar a probabilidade de todos os eventos, através do uso de um pseudo-contador, α, tendo como valor pré-definido 1, no entanto, poderá não se apresentar como o melhor valor. Com a adição deste novo contador, a probabilidade condicionada de um evento para uma classe passará a ser calculada da seguinte forma:[11]

$$P(B_i|A) = \frac{N_{Bi}+\alpha}{N_B+\alpha n}$$

Em que N_{Bi} representa o número de vezes que um evento **i** se apresenta na classe A, já N_B apresenta a soma do número total de eventos existentes em A. Por fim o n, indica o número de eventos.

5.2 Aplicação Algoritmo

Com a exposição das bases do classificador Naive Bayes, é de extrema importância o cálculo da probabilidade de acontecimento de uma classe, o número de registos de cada uma e o tamanho do vocabulário total.

O desenvolvimento do algoritmo apresentado tem por base um repositório do Github[2]. Dessa forma, foram criadas quatro funções, uma com a finalidade de calcular as probabilidades condicionadas de cada palavra, outra para classificar os *tweets* do *dataset* de teste, e as duas restantes têm a finalidade de calcular a *accuracy* do nosso modelo.

Assim sendo, iremos demonstrar de seguida como o cálculo dessas probabilidades condicionadas é feito.

```
def calc_conditional_probability(alpha,training_rumour,training_valid,vocabulary,
↪  n_rumour,n_vocabulary,n_valid,training_uncertain,n_uncertain):
    #Calculation of the divisors for eache class
    divRumour = (n_rumour + alpha * n_vocabulary)
    divValid = (n_valid + alpha * n_vocabulary)
    divUncertain = (n_uncertain + alpha * n_vocabulary)

    #Creation of the dictionary, with the finality
    #of storing a word and its probability
    p_rumour_dict = dict()
    p_valid_dict = dict()
    p_uncertain_dict = dict()

    #Iterate by the words present in the vocabulary
    for word in vocabulary:
        #Get the total occurences of a word in each class,
        #if there is no value it will be assumed as 0
        auxR = training_rumour[word].sum(min_count = 0)
        auxV = training_valid[word].sum(min_count = 0)
        auxU = training_uncertain[word].sum(min_count = 0)
        #Calculate the probability of a given word and store it in the dictionary
        p_rumour_dict[word] = (auxR + alpha)/divRumour
        p_valid_dict[word] = (auxV + alpha)/divValid
        p_uncertain_dict[word] = (auxU + alpha)/divUncertain

    return p_rumour_dict,p_valid_dict,p_uncertain_dict
```

Após a definição de como é feito o cálculo da probabilidade condicionada de ocorrer uma palavra num dado evento, falta ainda a demonstração de como é feito o produtório das probabilidades condicionais, que será demonstrado em baixo.

[2] shorturl.at/abluG

```
def classify_messages(message,p_rumour_dict,p_valid_dict,p_uncertain_dict,pRumour
↪  ,pValid,pUncertain):
    #Treat the tweet as it was done for the train dataset
    message = re.sub('\W', ' ', message)
    message = message.lower()
    message = message.split()

    p_rumour_message = pRumour
    p_valid_message = pValid
    p_uncertain_message = pUncertain

    #Calculation of the probability of a tweet in a each class
    for word in message:
        if word in p_rumour_dict:
            p_rumour_message *= p_rumour_dict[word]
        if word in p_valid_dict:
            p_valid_message *= p_valid_dict[word]
        if word in p_uncertain_dict:
            p_uncertain_message *= p_uncertain_dict[word]

    #print('P(Rumour|message):', p_rumour_message)
    #print('P(Valid|message):', p_valid_message)

    #Determination of the class for the calculated tweet
    if p_valid_message > p_rumour_message and p_valid_message >
    ↪  p_uncertain_message:
        return int(1)
    elif p_rumour_message > p_valid_message and p_rumour_message >
    ↪  p_uncertain_message:
        return int(0)
    elif p_uncertain_message > p_valid_message and p_uncertain_message >
    ↪  p_rumour_message:
        return int(2)
    else:
        return int(2)
```

Por fim, estas duas funções são executadas numa função *get_ accuracy*, que segue uma linha de pensamento muito simples. Baseia-se na iteração pelos valores de uma lista de α, pré-definida, daí são criados os dicionários de probabilidade condicionada para cada uma das classes, através da função *calc_ conditional_ probability* e, de seguida, através do output obtido, classifica-se cada um dos *tweets* do *dataset* teste. Por último, estas classificações são comparadas com as *labels*, com o intuito de calcular a *accuracy*.

Estas duas novas funções apresentam-se da seguinte forma:

```
def accuracy_calc(test):
    correct = 0
    total = test.shape[0]

    for _,row in test.iterrows():
        if row['Class_label'] == row['Prediction']:
            correct += 1

    accuracy = correct/total
    return accuracy

def get_accuracy(test,training_rumour,training_valid,training_uncertain,vocabular
↪  y,n_rumour,n_uncertain,n_vocabulary,n_valid,pRumour,pValid,pUncertain):
    accuracy_val = dict()
    #Iterate through the values of alpha
    for alpha in alphaL:
        #Calculate the conditional probability for each class
        p_rumour_dict,p_valid_dict,p_uncertain_dict =
        ↪  calc_conditional_probability(alpha,training_rumour,training_valid,voc
        ↪  abulary,n_rumour,n_vocabulary,n_valid,training_uncertain,n_uncertain)

        #Iterate through the test daset and proceed to the classification
        for index,message in enumerate(test['Complete_Message']):
            #Classify each tweet from the test dataset
            test.loc[index,'Prediction'] = classify_messages(message,p_rumour_dic
            ↪  t,p_valid_dict,p_uncertain_dict,pRumour,pValid,pUncertain)
        #Store the accuracy from each value of alpha
        accuracy_val[alpha] = accuracy_calc(test)

    return accuracy_val
```

5.3 Tunning do Modelo

Os hyperparâmetros são variáveis de configuração que são externas ao modelo, ou seja, são especificados antes do treino. Estes definem os conceitos de maior nível em relação ao modelo. O objetivo do *tunning* é descobrir os melhores valores para estes hyperparâmetros. Tendo em conta o algoritmo que estamos a utilizar, um dos hyperparâmetros sobre o qual poderemos otimizar será o valor de *alpha*. Assim sendo, e como já referido anteriormente, foi criada uma lista com diversos valores para este parâmetro e, por sua vez, calculado para cada um destes a *accuracy* do nosso modelo. Assim, conseguimos gerar o seguinte gráfico para três combinações de treino/teste e diferentes valores de α.

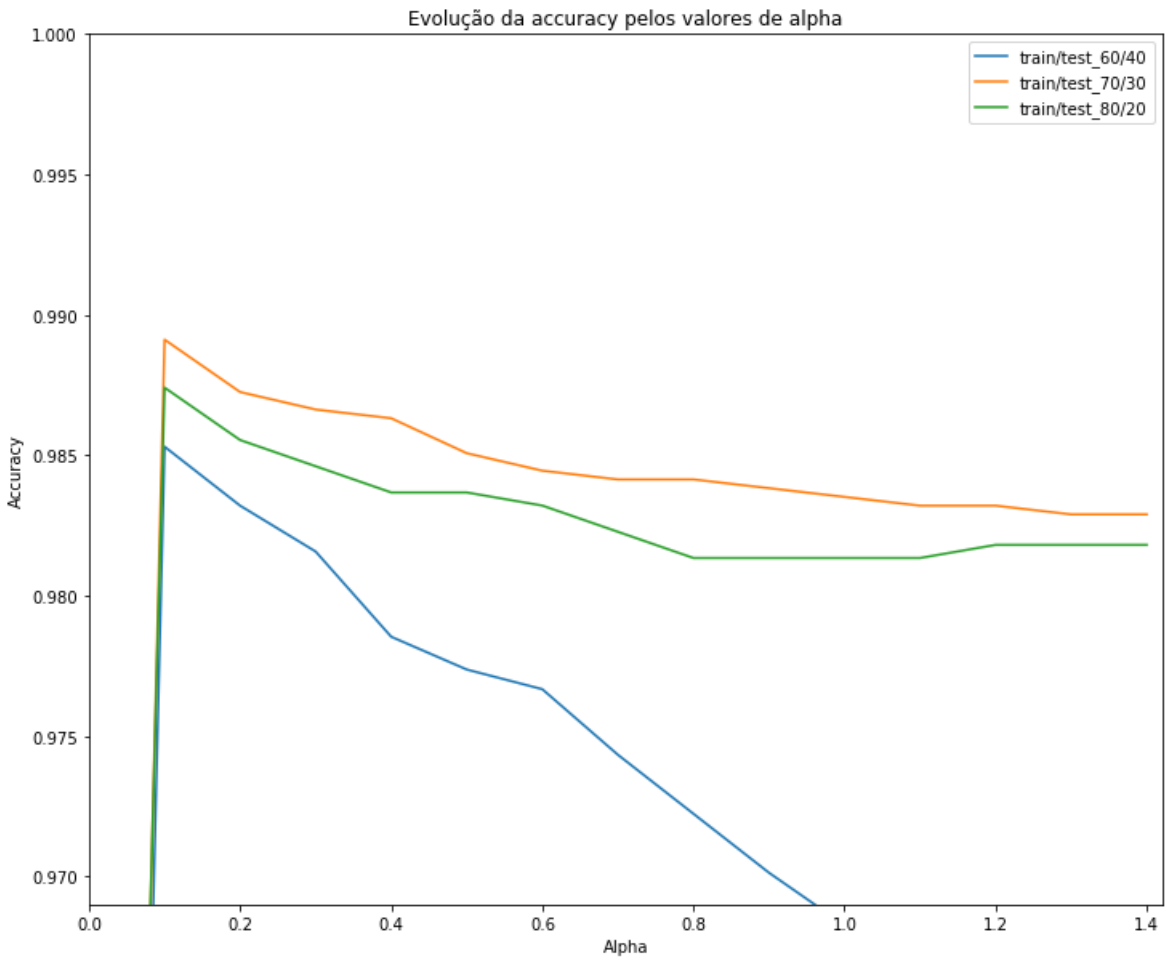
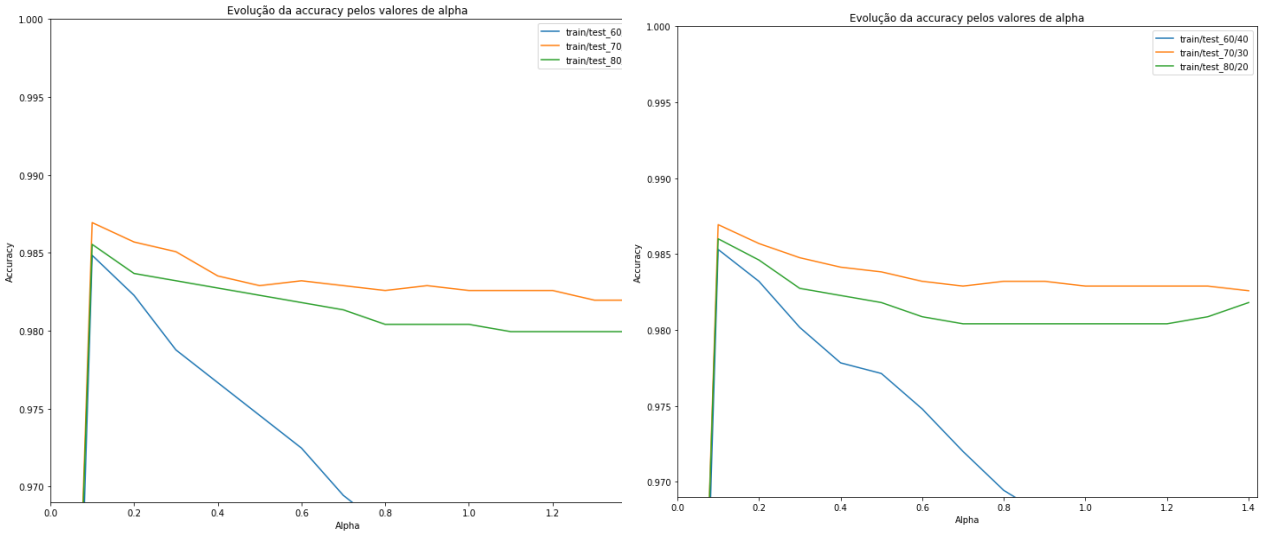


Figura 7: Evolução da accuracy pelos valores de alpha

Como podemos verificar, cada uma das combinações apresenta o seu pico de *accuracy*, para um valor α igual a 0.1, em valores muito próximos de 0.99 . Contudo, podemos destacar a combinação de treino com 70% de treino e 30% de teste, que apresenta para todos os valores de α calculados, a melhor *accuracy*.

No entanto, o *tunning* de um modelo não passa apenas pela busca em encontrar a melhor combinação de hyperparâmetros pode passar, por exemplo, no tratamento de dados. Assim sendo foram testadas duas possíveis otimizações, através de *Stemming* e *Lemmatization*, porém estas duas novas implementações não se apresentaram como frutíferas, descartando-se o seu uso.



(a) Stemming

(b) Lemmanization

Figura 8: Resultados do modelo para cada uma das implementações

Um possível motivo para este decréscimo nos valores da *accuracy*, tem por base que as duas técnicas modificam as palavras. Mais concretamente, *Lemmatization* reduz um verbo conjugado para a sua forma defetiva, enquanto o *Stemming* remove o prefixo e afixo com base nalgumas regras.[14] Na imagem seguinte tentámos ilustrar os princípios de cada uma das técnicas.

Word	***Lemmatization***	***Stemming***
was	be	wa
studies	study	studi
studying	study	study

Figura 9: Quadro introdutório para cada uma das técnicas

Ou seja, ao reduzir estes tempos verbais, torna-se muito mais difícil para o nosso modelo, através da probabilidade condicionada prever a real classe ao qual um *tweet* se encontra.

6 Resultados

Com a obtenção do melhor valor de α, e melhor combinação de % treino e teste, falta uma análise sobre as suas métricas. Assim sendo, foram calculadas diversas métricas para uma melhor compreensão sobre o comportamento do nosso modelo, fazendo-se uso dessas melhores combinações.

	precision	recall	f1-score	support
0	0.804688	0.962617	0.876596	107.00000
1	0.996166	0.988587	0.992362	1840.00000
2	0.997619	0.992107	0.994856	1267.00000
accuracy	0.989110	0.989110	0.989110	0.98911
macro avg	0.932824	0.981104	0.954605	3214.00000
weighted avg	0.990364	0.989110	0.989491	3214.00000

Figura 10: Métricas sobre a performance do modelo

Pela tabela podemos verificar que estamos perante métricas como **Precision**, **Recall**, **F1-Score**, **Accuracy** e **Macro Avg**. Cada uma desta apresenta a seguinte funcionalidade:[12][13]

- **Precision**: Valor que demonstra a relação entre as previsões que são de facto os valores corretos;
- **Recall**: Indica-nos o quão bem o nosso modelo está preparado para identificar uma classe;
- **F1-Score**: É uma métrica que nos apresenta uma relação entre a *Precision* e a *Recall*;
- **Accuracy**: Valor que nos indica a quantidade de classes identificadas corretamente;
- **Macro Avg**: Apresenta-se como sendo a média de cada uma das métricas mencionadas acima;

Tendo em conta cada uma dessas métricas podemos verificar que o nosso modelo apresenta resultados muito bons, e com boa capacidade para reconhecer diferentes tipos de *label*. Estas métricas podem ser obtidas através da biblioteca **sklearn**, e pelos seguintes comandos:

```
report_confusion_matrix7030 =
↪  metrics.classification_report(y_pred7030,y_real7030,output_dict=True)
report_confusion_matrix_df7030 =
↪  pd.DataFrame(report_confusion_matrix7030).transpose()
```

7 Conclusões

Com a realização deste trabalho prático o grupo ficou a compreender melhor a dinâmica de deteção de rumores em plataformas de *Microblogging*, nomeadamente o Twitter. De modo a aprofundar esta pesquisa, ficou do interesse do grupo a implementação de outras abordagens como, por exemplo, o uso de análise sentimental, outros algoritmos de *machine learning* como *Random Forest*, etc.

Para além destas, poderia também ser apresentada uma análise sobre os utilizadores que publicaram rumores previstos, ou seja, analisar as principais pessoas que seguem, e como essas se correlacionam com a publicação de notícias falaciosas.

Contudo, acreditamos que os objetivos foram concluídos com sucesso apesar dos obstáculos que enfrentámos.

Referências

[1] Hamidian, S. and Diab, M., 2019. Rumor Detection And Classification For Twitter Data. The George Washington University.

[2] Covid19.min-saude.pt. 2020. Perguntas Frequentes - COVID-19. [online] Available at: https://covid19.min-saude.pt/perguntas-frequentes/ [Accessed 18 March 2020].

[3] Wisconsin Department of Health Services. 2020. COVID-19 (Coronavirus Disease). [online] Available at: https://www.dhs.wisconsin.gov/covid-19/index.htm [Accessed 18 March 2020].

[4] Esrc.ukri.org. 2020. What Is Twitter And Why Should You Use It? - Economic And Social Research Council. [online] Available at: https://esrc.ukri.org/research/impact-toolkit/social-media/twitter/what-is-twitter/ [Accessed 17 March 2020].

[5] Koumchatzky, N. and Andryeyev, A., 2017. Using Deep Learning At Scale In Twitter'S Timelines. [online] Blog.twitter.com. Available at: https://blog.twitter.com/engineering/en_us/topics/insights/2017/using-deep-learning-at-scale-in-twitters-timelines.html [Accessed 27 March 2020].

[6] Sproutsocial.com How the Twitter algorithm works in 2020 by Cole Nemeth. [online] Available at: https://sproutsocial.com/insights/twitter-algorithm/ [Acessed 26 March 2020]

[7] Oxfordlearnersdictionaries.com. 2020. Rumour_1 Noun - Definition, Pictures, Pronunciation And Usage Notes | Oxford Advanced Learner's Dictionary At Oxfordlearnersdictionaries.Com. [online] Available at: https://www.oxfordlearnersdictionaries.com/definition/english/rumour_1 [Accessed 7 July 2020].

[8] Freitas, A., 2020. Taxa Estimada De Mortalidade Global Da Covid-19 É De 2%. [online] PÚBLICO. Available at: https://www.publico.pt/2020/02/28/ciencia/noticia/taxa-estimada-mortalidade-global-covid19-2-1905844 [Accessed 8 July 2020].

[9] MonkeyLearn. 2020. Text Classification Using Naive Bayes. [online] Available at: https://monkeylearn.com/text-classification-naive-bayes/ [Accessed 9 July 2020].

[10] Docs.oracle.com. 2020. Naive Bayes. [online] Available at: https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nb.htm#i1005770 [Accessed 9 July 2020].

[11] Scikit-learn.org. 2020. 1.9. Naive Bayes — Scikit-Learn 0.23.1 Documentation. [online] Available at: https://scikit-learn.org/stable/modules/naive_bayes.html [Accessed 18 July 2020].

[12] Web.stanford.edu. 2020. Text Classification And Naïve Bayes - The Task Of Text Classification. [online] Available at: https://web.stanford.edu/ jurafsky/slp3/slides/7_NB.pdf [Accessed 19 July 2020].

[13] Gabrielschade.github.io. 2020. Machine Learning: Métricas Para Modelos De Classificação. [online] Available at: https://gabrielschade.github.io/2019/03/12/ml-classificacao-metricas.html [Accessed 19 July 2020].

[14] Medium. 2020. TF-IDF For Document Ranking From Scratch In Python On Real World Dataset [online] Available at: https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089 [Accessed 19 July 2020].

[15] WikiHow. Como Avaliar a Credibilidade de uma Fonte [online] Available at: https://pt.wikihow.com/Avaliar-a-Credibilidade-de-uma-Fonte [Accessed 19 July 2020].

[16] Cornell University Library. 2020. Critically Analyzing Information Sources: Critical Appraisal and Analysis [online] Available at: https://guides.library.cornell.edu/critically_analyzing [Accessed 19 July 2020].

[17] Jin, Z., Cao, J., Zhang, Y., Wang, Y. and Luo, J., 2020. Detection And Analysis Of 2016 US Presidential Electionrelated Rumors On Twitter. [online] Arxiv.org. Available at: https://arxiv.org/pdf/1701.06250.pdf [Accessed 19 July 2020].