

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Tatiana Novaes Carvalho

SAÚDE EM *TWEETS* -
UMA TAREFA DE CLUSTERIZAÇÃO

Belo Horizonte

2020

Tatiana Novaes Carvalho

**SAÚDE EM *TWEETS* -
UMA TAREFA DE CLUSTERIZAÇÃO**

Trabalho de Conclusão de Curso apresentado
ao Curso de Especialização em Ciência de
Dados e Big Data como requisito parcial à
obtenção do título de especialista.

Belo Horizonte

2020

SUMÁRIO

1. Introdução.....	4
1.1. Contextualização	4
1.2. O problema proposto	5
2. Coleta de Dados	6
3. Processamento/Tratamento de Dados	7
4. Análise e Exploração dos Dados	14
5. Criação de Modelos de Machine Learning	23
6. Apresentação dos Resultados	50
7. Links	59
REFERÊNCIAS.....	61

1. Introdução

O trabalho visa à análise de textos de *tweets* relacionados à saúde, no intuito de buscar *insights* que possam auxiliar os profissionais da área no exercício de suas atividades. Para isso, o estudo foi estruturado em seis partes. Na introdução, será apresentada a contextualização do assunto tratado e o problema proposto. Na sequência, serão apresentadas as informações acerca da coleta dos dados (Capítulo 2) e os passos realizados para o tratamento inicial, consistente em sua limpeza e estruturação (Capítulo 3). Passo seguinte, serão efetuadas a análise e a exploração desses dados, com vistas à obtenção de informações estatísticas relevantes (Capítulo 4). Como o trabalho gira em torno de análise textual, o Capítulo 5 – Aplicação de Modelos de *Machine Learning* – abrangerá processamento de linguagem natural, bem como aplicação de algoritmos para agrupamento dos dados. Por fim, no Capítulo 6 serão apresentados os resultados obtidos.

1.1. Contextualização

Milhares de mensagens são escritas diariamente acerca de temas relevantes para a humanidade, entre os quais a saúde. A Revista Exame publicou, já em 2014, artigo em que se destaca que o conteúdo digital dobra a cada dois anos no mundo [1]. Extrair informações relevantes desses dados com vistas à obtenção de conhecimento para a tomada de decisões de forma eficiente e otimizada é uma tarefa árdua para o ser humano, vez que tais dados são, regra geral, não estruturados. Eles se apresentam em linguagem natural, na qual há variação de estilo, grau de formalidade, formato, entre outros, o que dificulta sua análise de forma automatizada.

De outro lado, a variedade de interesses e necessidades por parte da sociedade – principalmente na área da saúde – é enorme, o que demanda o desenvolvimento e o aprimoramento de ferramentas que proporcionem a minimização do esforço humano no consumo desses dados.

Oportuna, por conseguinte, a realização de projetos de pesquisa que apliquem a tecnologia de informação – e, nesse contexto, a utilização de algoritmos de aprendizado de máquina (*machine learning*) – no tratamento de textos vinculados

a temas ligados à saúde, com vistas à otimização da ação humana e a consequente melhoria do bem-estar da população.

1.2. O problema proposto

Uma enorme quantidade de mensagens é escrita e postada em redes sociais diariamente acerca de temas relevantes para a humanidade, entre os quais a saúde. Nesse contexto encontram-se os *tweets* de agências de notícias como BBC, CNN e New York Times.

Ocorre que os *tweets* são escritos em linguagem natural com pouca ou nenhuma estruturação e são vários os assuntos abordados. Ou seja, extrair informações desses dados não é tarefa fácil.

Um desafio que se põe, dessa forma, é encontrar meios de analisar esses dados e obter informações que possam direcionar a atenção dos responsáveis pela área de saúde, seja na seara da prevenção, do diagnóstico ou até mesmo da pesquisa.

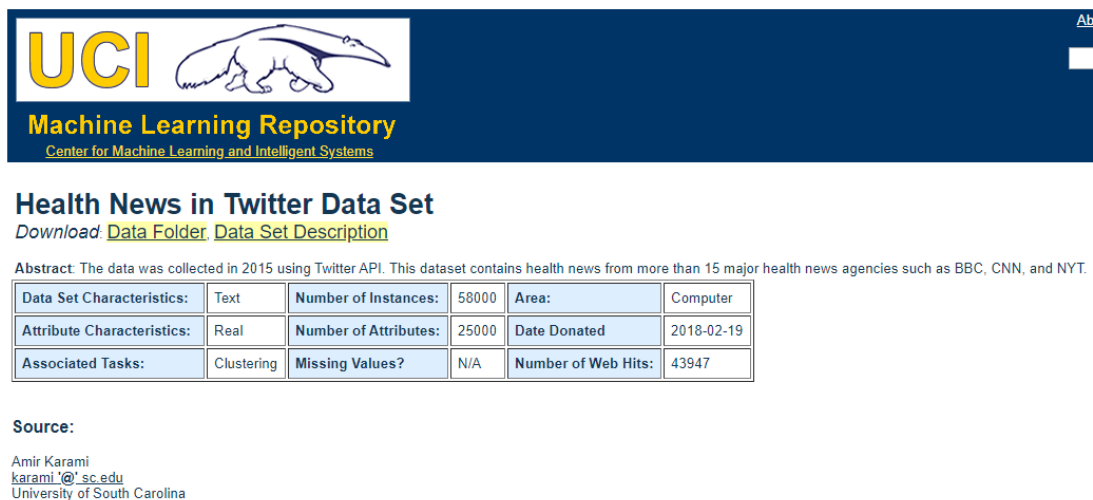
No intuito de contribuir com tal tarefa, foram acessados dados do sítio da *UCI Machine Learning Repository* em 29 de janeiro de 2020. O conjunto de dados – *Health News in Twitter Data Set* – consiste em 16 (dezesesseis) *datasets* (arquivos texto), cada um relacionado a uma conta Twitter de uma agência de notícias, totalizando 63.326 mensagens, relativas ao período de 2011 a 2015 [2]. Como o escopo principal do trabalho é a análise textual, decidiu-se pela busca de *datasets* já disponíveis em repositório conhecido, no caso da UCI, em vez da coleta de dados diretamente no Twitter.

O objetivo do projeto é, portanto, analisar uma coleção de *tweets* de agências de notícias, postados no período de 2011 a 2015, no intuito de descobrir novos padrões nos dados e identificar temas ou tópicos de interesse ligados à saúde, que possam ser utilizados por especialistas da área. Para tanto, os dados serão pré-processados por meio de aplicação dos modelos *Bag of Words* e TF-IDF e, em seguida, agrupados em clusters pelo modelo K-Means de aprendizado de máquina não supervisionado, os quais serão detalhados mais à frente.

2. Coleta de Dados

O conjunto de dados *Health News in Twitter Data Set*, objeto deste projeto, foi obtido no sítio da *UCI Machine Learning Repository* em 29 de janeiro de 2020. Os dados, por seu turno, consistentes em 16 (dezesesseis) *datasets* com mesma estrutura, foram coletados em 2015 por meio de Twitter API por Amir Karami, da Universidade da Carolina do Sul [2].

Como exposto, trata-se de 16 arquivos texto, cada um relacionado a uma conta Twitter de uma agência de notícias, totalizando, segundo o sítio, 58.000 instâncias. A referência a essa quantidade pode ser observada na Figura 2.1. Cada linha dos arquivos contém um *tweet id|date and time|tweet*. O separador é “|” e todos os campos são do tipo *string*.



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

Health News in Twitter Data Set

Download: [Data Folder](#) [Data Set Description](#)

Abstract: The data was collected in 2015 using Twitter API. This dataset contains health news from more than 15 major health news agencies such as BBC, CNN, and NYT.

Data Set Characteristics:	Text	Number of Instances:	58000	Area:	Computer
Attribute Characteristics:	Real	Number of Attributes:	25000	Date Donated	2018-02-19
Associated Tasks:	Clustering	Missing Values?	N/A	Number of Web Hits:	43947

Source:
Amir Karami
karami@sc.edu
University of South Carolina

Figura 2.1 – Informações sobre os datasets utilizados. Disponível em: <
<https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter>>

Os nomes dos *datasets* utilizados e as respectivas propriedades podem ser verificados na Figura 2.2. Exemplos do formato dos arquivos (primeiras linhas de *bbchealth.txt*), por seu turno, podem ser observados na Figura 2.3.

















Nome	Data de modificação	Tipo	Tamanho
 bbchealth	09/04/2015 03:39	Documento de Texto	423 KB
 cbchealth	09/04/2015 19:15	Documento de Texto	664 KB
 cnnhealth	09/04/2015 19:20	Documento de Texto	638 KB
 everydayhealth	09/04/2015 18:58	Documento de Texto	458 KB
 foxnewshealth	09/04/2015 03:44	Documento de Texto	257 KB
 gdnhealthcare	09/04/2015 03:47	Documento de Texto	535 KB
 goodhealth	09/04/2015 18:56	Documento de Texto	1.227 KB
 KaiserHealthNews	09/04/2015 18:29	Documento de Texto	542 KB
 latimeshealth	09/04/2015 19:10	Documento de Texto	626 KB
 msnhealthnews	09/04/2015 19:19	Documento de Texto	410 KB
 NBChealth	09/04/2015 18:37	Documento de Texto	544 KB
 nprhealth	09/04/2015 19:11	Documento de Texto	606 KB
 nytimeshealth	09/04/2015 19:13	Documento de Texto	881 KB
 reuters_health	09/04/2015 19:16	Documento de Texto	634 KB
 usnewshealth	09/04/2015 19:00	Documento de Texto	215 KB
 wsjhealth	09/04/2015 18:34	Documento de Texto	558 KB

Figura 2.2 – Nomes e propriedades dos datasets utilizados.

```

585978391360221184|Thu Apr 09 01:31:50 +0000 2015|Breast cancer risk test devised http://bbc.in/1CimpJF
585947808772960257|Wed Apr 08 23:30:18 +0000 2015|GP workload harming care - BMA poll http://bbc.in/1ChTBRv
585947807816650752|Wed Apr 08 23:30:18 +0000 2015|Short people's 'heart risk greater' http://bbc.in/1ChTANp
585866060991078401|Wed Apr 08 18:05:28 +0000 2015|New approach against HIV 'promising' http://bbc.in/1E6jAjt
585794106170839041|Wed Apr 08 13:19:33 +0000 2015|Coalition 'undermined NHS' - doctors http://bbc.in/1CnLwK7
585733482413891584|Wed Apr 08 09:18:39 +0000 2015|Review of case against NHS manager http://bbc.in/1Ffj6ci
585733481608646657|Wed Apr 08 09:18:39 +0000 2015|VIDEO: 'All day is empty, what am I going to do?' http://bbc.in/1N7wSSz
585701601131765761|Wed Apr 08 07:11:58 +0000 2015|VIDEO: 'Overhaul needed' for end-of-life care http://bbc.in/1CmrRu3

```

Figura 2.3 – Primeiras linhas do arquivo bbchealth.txt.

3. Tratamento de Dados

O projeto foi executado por meio da linguagem de programação interpretada Python [3] no Jupyter Notebook— ambiente de desenvolvimento interativo baseado na Web [4] – tendo sido utilizadas, primordialmente, as bibliotecas listadas na Figura 3.1. O estilo definido para a elaboração dos gráficos por meio da biblioteca Seaborn [5] foi *whitegrid*.

```
# IMPORTAÇÃO DAS BIBLIOTECAS
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
```

Figura 3.1 – Principais bibliotecas utilizadas no projeto.

Para a leitura dos dados, os arquivos (documentos de texto) foram buscados no diretório especificado e incluídos em uma *lista* da linguagem de programação Python, conforme a Figura 3.2. A leitura posterior das linhas de cada um dos 16 arquivos foi efetuada por meio da função *open*. O parâmetro *encoding* padrão dessa função é UTF-8, porém, como houve erro (*UnicodeDecodeError*) em seu processamento (vide Figura 3.3), foi necessário ler parte dos arquivos com *encoding* CP 1252 (Figura 3.4). Essa solução foi obtida por meio da consulta ao website Stack OverFlow [6].

```
# COLETA DE DADOS
```

```
# Definindo a pasta de trabalho
dir_name = 'D:\\TCC\\DATASETS\\'
```

```
# Criando lista com nomes de arquivos txt
def ler_diretorio():
    lista_txt = []
    for txt in os.listdir(dir_name):
        if txt.endswith('.txt'):
            lista_txt.append(txt)
    return lista_txt
lista_txt = ler_diretorio()
print('Os datasets utilizados no projeto são: \n      {}'.format(lista_txt))
print('O total de datasets utilizados é: {}'.format(len(lista_txt)))
print('O tipo da variável lista_txt é: {}'.format(type(lista_txt)))
print('O tipo dos elementos da lista_txt é: {}'.format(type(lista_txt[0])))
```

```
Os datasets utilizados no projeto são:
```

```
['bbchealth.txt', 'cbchealth.txt', 'cnnhealth.txt', 'everydayhealth.txt', 'foxnewshealth.txt', 'gdhealthcare.txt', 'goodhealth.txt', 'KaiserHealthNews.txt', 'latimeshealth.txt', 'msnhealthnews.txt', 'NBChealth.txt', 'nprhealth.txt', 'nytimeshealth.txt', 'reuters_health.txt', 'usnewshealth.txt', 'wsjhealth.txt']
```

```
O total de datasets utilizados é: 16
O tipo da variável lista_txt é: <class 'list'>
O tipo dos elementos da lista_txt é: <class 'str'>
```

Figura 3.2 – Código em Python para criação de lista com nomes dos arquivos a serem lidos.


```
# Lendo os arquivos txt com encoding 'utf-8' #explicar o erro
mensagens = []
for txt in lista_txt:
    with open(dir_name+txt, 'r', encoding='utf-8') as f:
        for linha in f.readlines():
            atributo = linha.split('|')
            mensagens.append([atributo[0], atributo[1], atributo[2]])
```

```
-----
UnicodeDecodeError                                Traceback (most recent call last)
<ipython-input-4-5d84c0bcb896> in <module>()
      3 for txt in lista_txt:
      4     with open(dir_name+txt, 'r', encoding='utf-8') as f:
----> 5         for linha in f.readlines():
      6             atributo = linha.split('|')
      7             mensagens.append([atributo[0], atributo[1], atributo[2]])

~\Anaconda3\lib\codecs.py in decode(self, input, final)
    320     # decode input (taking the buffer into account)
    321     data = self.buffer + input
--> 322     (result, consumed) = self._buffer_decode(data, self.errors, final)
    323     # keep undecoded input until the next call
    324     self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode byte 0x92 in position 2648: invalid start byte
```

Figura 3.3 – Erro encontrado na leitura dos arquivos.

```
# Lendo os arquivos txt - encoding 'utf-8' e 'cp1252' (para correção do erro 'UnicodeDecodeError')
mensagens = []
msg_1252 = []
for txt in lista_txt:
    try:
        with open(dir_name+txt, 'r', encoding='utf-8') as f1:
            for linha in f1.readlines():
                col = linha.split('|')
                mensagens.append(col)
    except:
        msg_1252.append(txt)

for txt in msg_1252:
    with open(dir_name+txt, 'r', encoding='cp1252') as f2:
        for linha in f2.readlines():
            atributo = linha.split('|')
            mensagens.append(atributo)
print('O total de mensagens lidas é: {}'.format(len(mensagens)))
print('Os datasets com erro na leitura com utf-8 são:\n{}\n'.format(msg_1252))
```

O total de mensagens lidas é: 63327

Os datasets com erro na leitura com utf-8 são:

['foxnewshealth.txt', 'KaiserHealthNews.txt', 'msnhealthnews.txt', 'NBCHealth.txt', 'wsjhealth.txt']

Figura 3.4 – Código em Python para leitura dos arquivos.

Foram lidas, portanto, 63.327 linhas. Porém, após a realização de testes (Figura 3.5), verificou-se que alguns *tweets* haviam sido indevidamente divididos porque continham o símbolo “|” utilizado para separar as colunas. Observe-se que o

tamanho máximo da variável *mensagens* - do tipo lista - foi 5, quando deveria ser 3 (id, data e hora, e *tweet*).

```
# Verificando tamanho máximo de cada mensagem
col = 3
for mensagem in mensagens:
    tam = len(mensagem)
    if tam > col:
        col = tam
print(col)
```

5

Figura 3.5 – Código em Python para verificação do conteúdo das mensagens.

Para a necessária correção, a lista *mensagens* foi convertida em um *dataframe* do Pandas com as colunas “Id”, “Dia_Hora”, “Mensagem”, “X” e “Y” (Figura 3.6). As duas últimas colunas foram assim intituladas para permitir a investigação do motivo do erro na geração de cinco colunas e sua correção. Confirmado que se tratava de divisão indevida do teor da mensagem pela investigação de algumas linhas selecionadas aleatoriamente, as três últimas colunas foram concatenadas e atribuídas a uma nova coluna, intitulada “Mensagem_Concatenada”. Para tanto, a informação “NaN” das colunas “X” e “Y” foram substituídas por espaço (Figura 3.7).

```
data = pd.DataFrame(mensagens, columns=['Id', 'Dia_Hora', 'Mensagem', 'X', 'Y'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63327 entries, 0 to 63326
Data columns (total 5 columns):
Id          63327 non-null object
Dia_Hora    63326 non-null object
Mensagem    63326 non-null object
X           214 non-null object
Y           2 non-null object
dtypes: object(5)
memory usage: 2.4+ MB
```

Figura 3.6 – Código em Python para geração do dataframe a partir da lista ‘mensagens’.

```
data['X'].fillna('', inplace=True)
data['Y'].fillna('', inplace=True)
data['Mensagem_Concatenada'] = (data['Mensagem'] + data['X'] + data['Y'])
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63327 entries, 0 to 63326
Data columns (total 6 columns):
Id                63327 non-null object
Dia_Hora          63326 non-null object
Mensagem          63326 non-null object
X                 63327 non-null object
Y                 63327 non-null object
Mensagem_Concatenada 63326 non-null object
dtypes: object(6)
memory usage: 2.9+ MB
```

Figura 3.7 – Código em Python para criação da coluna 'Mensagem_Concatenada'.

Dada a divergência entre o número de mensagens concatenadas (63.326) e de entradas (63.327), procedeu-se à identificação do registro divergente, tendo sido verificado que se tratava de uma linha com "Id" igual a "\n". Sendo assim, excluiu-se esse registro, o que pode ser verificado na Figura 3.8.

```
data[data['Mensagem'].notnull() == False]
```

	Id	Dia_Hora	Mensagem	X	Y	Mensagem_Concatenada
25831	\n	None	None			NaN

```
df = data[data['Mensagem'].notnull()]
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 63326 entries, 0 to 63326
Data columns (total 6 columns):
Id                63326 non-null object
Dia_Hora          63326 non-null object
Mensagem          63326 non-null object
X                 63326 non-null object
Y                 63326 non-null object
Mensagem_Concatenada 63326 non-null object
dtypes: object(6)
memory usage: 3.4+ MB
```

Figura 3.8 – Código em Python para exclusão de registro indevido.

Por fim, considerando que o sítio da UCI informa que o total de instâncias é 58.000 e tendo em vista que o total de entradas após a leitura dos dados foi 63.326, procedeu-se à verificação quanto à existência de registros duplicados. Estes, porém, inexistem (Figura 3.9). Com efeito, o somatório dos totais de linhas de cada um dos arquivos importa em 63.326 (cálculo efetuado manualmente para conferência das quantidades). Tentou-se uma verificação no sítio da UCI do motivo de eventual equívoco na informação, mas nada foi encontrado nesse sentido. Assumiu-se, assim, que a leitura dos dados aqui efetuada está correta.

```
df.duplicated().value_counts()  
  
False    63326  
dtype: int64
```

Figura 3.9 – Código em Python para verificação de existência de registros duplicados.

Com vistas a uma exploração de dados mais acurada, a coluna “Dia_Hora” foi convertida em *datetime* para extração de informações relativas a “Ano” e foram criadas as colunas “Dia_Semana”, “Mensagem Final” (com exclusão do http), “http”, “Tamanho_Mensagem”, “Tamanho_http” e “Video” (para indicar se a mensagem indica existência de vídeo no *post*). Por não serem necessárias para a continuidade da análise, as seguintes colunas foram excluídas do *dataframe*: “Id”, “Dia_Hora”, “Mensagem”, “X”, “Y” e “Mensagem_Concatenada”. O código em Python utilizado para esse tratamento dos dados pode ser observado na Figura 3.10. As alterações foram efetuadas em uma cópia do *dataframe*, com vistas a evitar mensagens de aviso *SettingWithCopyWarning error* do Python [7].

```
# TRATAMENTO DOS DADOS
```

```
# https://stackoverflow.com/questions/45037907/python-astypestr-gives-settingwithcopywarning-and-requests-i-use-loc
```

```
# Atribuindo cópia a df para evitar mensagem "SettingWithCopyWarning"
df = df.copy()
# Criando novas colunas
df.loc[:, 'Data'] = df.Dia_Hora.astype('datetime64[D]')
df.loc[:, 'Dia_Semana'] = df.Dia_Hora.apply(lambda x: (x.split())[0])
df.loc[:, 'Ano'] = df.Data.dt.year
df.loc[:, 'Mensagem_Final'] = df.Mensagem_Concatenada.apply(lambda x: ' '.join(list([w for w in x.split(' ') if w[0:4] != 'http'])))
df.loc[:, 'http'] = df.Mensagem_Concatenada.apply(lambda x: ' '.join(list([w for w in x.split(' ') if w[0:4] == 'http'])))
df.loc[:, 'Tamanho_Mensagem'] = df.Mensagem_Final.apply(lambda x: len(x))
df.loc[:, 'Tamanho_http'] = df.http.apply(lambda x: len(x))
df.loc[:, 'Video'] = df.Mensagem_Final.apply(lambda x: x[0:6] == 'VIDEO:')

# Excluindo colunas desnecessárias
df.drop(columns=['Id', 'Dia_Hora', 'Mensagem', 'X', 'Y', 'Mensagem_Concatenada'], inplace=True)

df.head(3)
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
0	2015-04-09	Thu	2015	Breast cancer risk test devised	http://bbc.in/1CimpJFn	31	22	False
1	2015-04-08	Wed	2015	GP workload harming care - BMA poll	http://bbc.in/1ChTBRvIn	35	22	False
2	2015-04-08	Wed	2015	Short people's 'heart risk greater'	http://bbc.in/1ChTANpIn	35	22	False

Figura 3.10 – Código em Python para tratamento dos dados.

No intuito de possibilitar uma análise específica dos *tweets* com indicação de “vídeo”, foi elaborado um *dataframe* apenas com os registros correspondentes. Conforme se verifica na Figura 3.11, esse *dataframe* possui 816 linhas.

```
# Construindo df apenas com mensagens de video
```

```
df_video = df[df['Video'] == True]
df_video.info()
df_video.head(3)
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 816 entries, 6 to 50003
Data columns (total 8 columns):
Data                816 non-null datetime64[ns]
Dia_Semana          816 non-null object
Ano                 816 non-null int64
Mensagem_Final      816 non-null object
http                816 non-null object
Tamanho_Mensagem    816 non-null int64
Tamanho_http        816 non-null int64
Video               816 non-null bool
dtypes: bool(1), datetime64[ns](1), int64(3), object(3)
memory usage: 51.8+ KB
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
6	2015-04-08	Wed	2015	VIDEO: 'All day is empty, what am I going to do?'	http://bbc.in/1N7wSSzIn	49	22	True
7	2015-04-08	Wed	2015	VIDEO: 'Overhaul needed' for end-of-life care	http://bbc.in/1CmrRu3In	45	22	True
9	2015-04-07	Tue	2015	VIDEO: NHS: Labour and Tory key policies	http://bbc.in/1Ci5eqDIn	40	22	True

Figura 3.11 – Código em Python para criação de dataframe com tweets de vídeo.

Por fim, foi criado um último *dataframe* com as *hashtags* encontradas nas mensagens postadas, por meio do código apresentado na Figura 3.12.

```
# Construindo df apenas com hashtags

hash = df['Mensagem_Final'].tolist() # converte series em lista de strings
hashtags_list=[]
for msg in hash:
    for w in msg.split():
        if w[0]=='#':
            hashtags_list.append(w)
            ' '.join(hashtags_list)

hashtags = ' '.join(hashtags_list)

df_hash = pd.DataFrame(hashtags_list, columns=['hash'])
df_hash.head()

#print(type(hash)) # lista de mensagens
#print(type(hashtags_list)) # lista de hashtags
#print(type(hashtags)) # string de hashtags
#print(hashtags)
#print(hashtags_list)
```

	hash
0	#naturalhealth
1	#essentialoils
2	#HealthCanada
3	#drugsafety
4	#Physicians:

Figura 3.12 – Código em Python para criação de dataframe com hashtags encontradas.

4. Análise e Exploração dos Dados

Em termos de estatística descritiva, não há informações relevantes a serem analisadas nos *dataframes* construídos, haja vista que os dados primordiais consistem em textos, os quais ensejam cuidados específicos. Entretanto, algumas abordagens, principalmente para fins de análise da consistência dos dados, mostram-se oportunas neste momento.

Uma verificação inicial revela que o número de mensagens relativas a saúde foi mais elevado em 2014, o que poderia decorrer de algum acontecimento específico na saúde pública (Figuras 4.1 e 4.2). Entretanto, como a coleta dos dados se deu em abril de 2015 (data de modificação dos arquivos coletados observada na Figura 2.2), a redução no número de mensagens decorre claramente do fato de que o ano ainda não estava concluído à época.

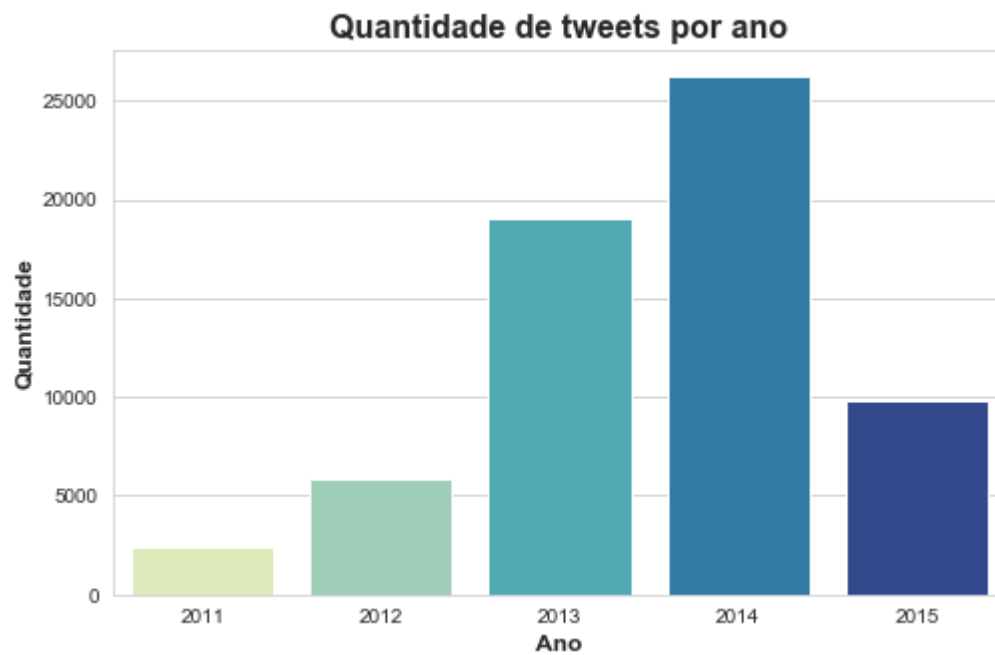


Figura 4.1 – Quantidade de tweets por ano.

```
#Countplot do df total
fig, ax = plt.subplots(figsize=(8,5))
sns.countplot(df['Ano'], palette='YlGnBu')
plt.title('Quantidade de tweets por ano', fontsize=16, fontweight='bold')
ax.set_xlabel('Ano', fontsize=12, fontweight='bold')
ax.set_ylabel('Quantidade', fontsize=12, fontweight='bold')
plt.show()
```

Figura 4.2 – Código em Python para elaboração do gráfico da Figura 4.1.

A quantidade por ano de *tweets* relativos a vídeos também reduziu de 2014 a 2015 (da mesma forma, deduz-se que o motivo seja a data de extração dos dados), e nada foi verificado nos anos de 2011 e 2012 (Figuras 4.3 e 4.4).

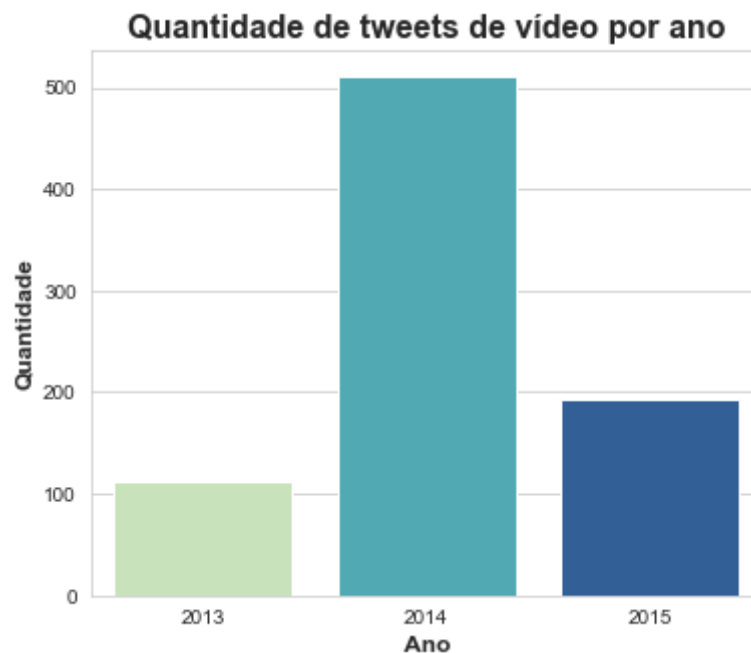


Figura 4.3 – Quantidade de tweets de vídeo por ano.

```
#Countplot do df video
fig, ax = plt.subplots(figsize=(6,5))
sns.countplot(df_video['Ano'], palette='YlGnBu')
plt.title('Quantidade de tweets de vídeo por ano', fontsize=16, fontweight='bold')
ax.set_xlabel('Ano', fontsize=12, fontweight='bold')
ax.set_ylabel('Quantidade', fontsize=12, fontweight='bold')
plt.show()
```

Figura 4.4 – Código em Python para elaboração do gráfico da Figura 4.3.

Quanto ao tamanho das mensagens, observa-se que a distribuição do tamanho relativo aos *tweets* com vídeo é mais uniforme que a distribuição do total de *tweets* (Figuras 4.5, 4.6, 4.7 e 4.8). Esse fato é esperado, uma vez que, regra geral, as mensagens com vídeos fazem apenas referência a eles, como forma de despertar a atenção para o que ali está contido. Já os demais *tweets* abarcam uma gama maior de possibilidades quanto a seu teor.

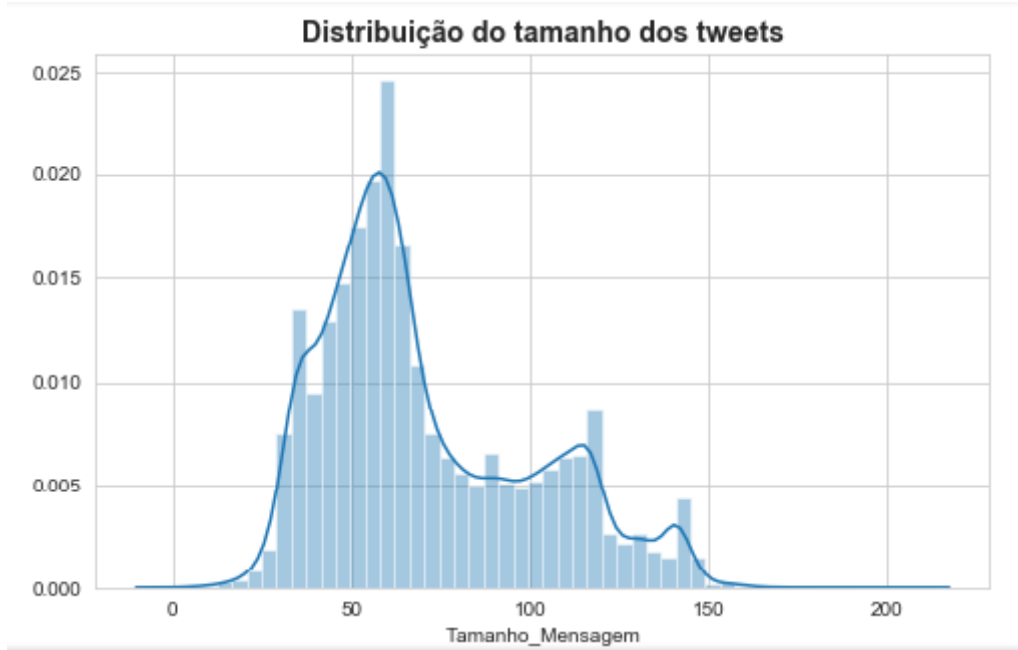


Figura 4.5 – Distribuição do tamanho dos tweets.

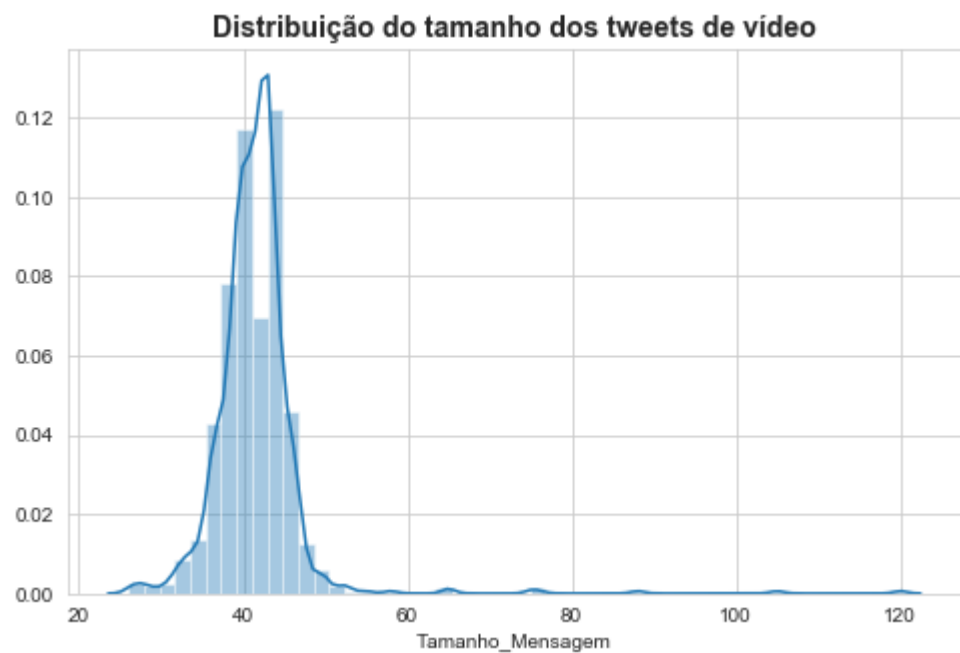


Figura 4.6 – Distribuição do tamanho dos tweets de vídeo.

```
#Distplot do df total
fig, ax = plt.subplots(figsize=(8,5))
sns.distplot(df['Tamanho_Mensagem'])
plt.title('Distribuição do tamanho dos tweets', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.7 – Código em Python para elaboração do gráfico da Figura 4.5.

```
#Distplot do df video
fig, ax = plt.subplots(figsize=(8,5))
sns.distplot(df_video['Tamanho_Mensagem'])
plt.title('Distribuição do tamanho dos tweets de vídeo', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.8 – Código em Python para elaboração do gráfico da Figura 4.6

Com objetivo de verificar a existência de *outliers* e possíveis erros na coleta dos dados, foram criados *boxplots* dessas duas variáveis, os quais podem ser observados nas figuras 4.9 a 4.11.

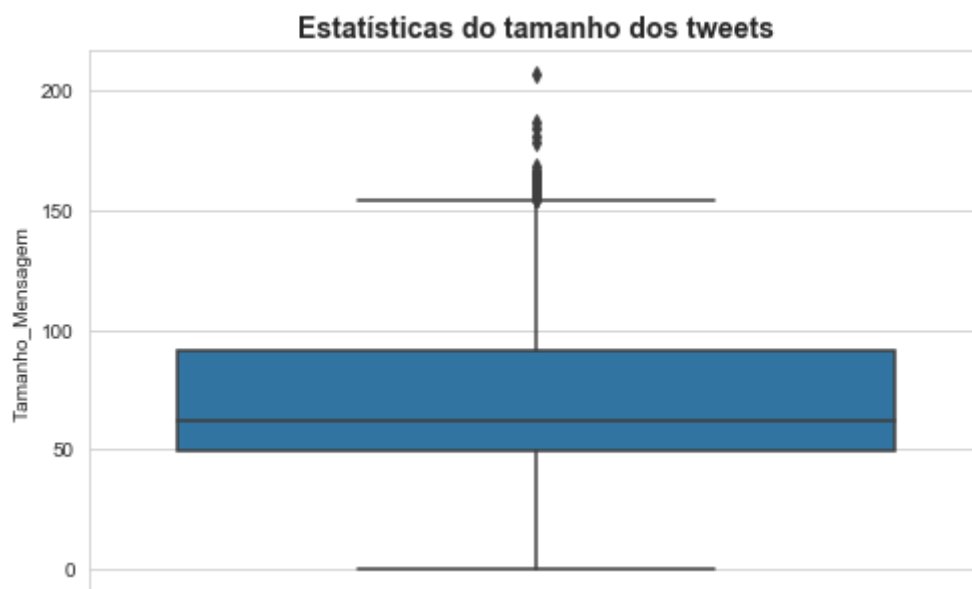


Figura 4.9 – Boxplot do tamanho dos tweets.

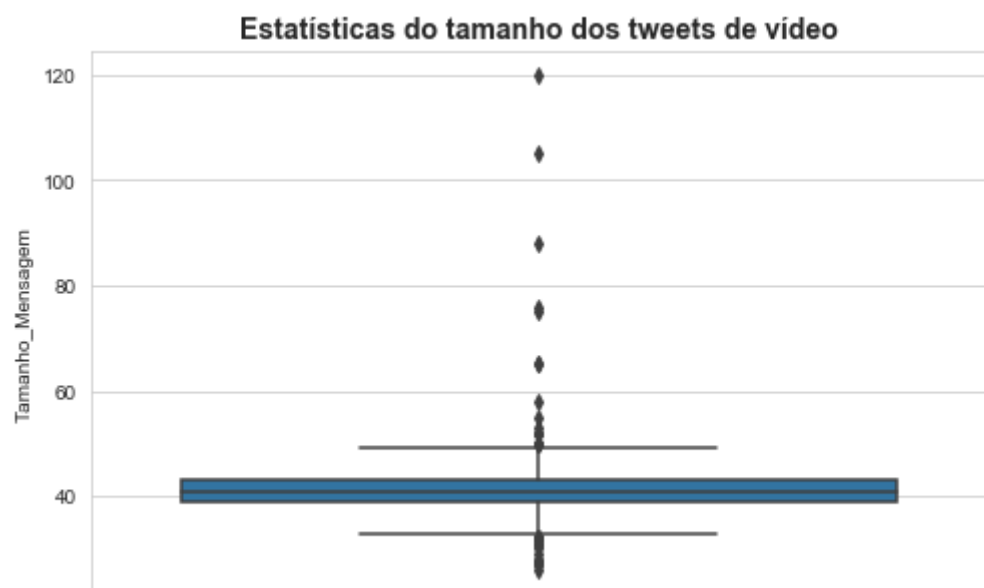


Figura 4.10 – Boxplot do tamanho dos tweets de vídeo.

```
#Boxplot do df total
fig, ax = plt.subplots(figsize=(8,5))
sns.boxplot(y=df['Tamanho_Mensagem'])
plt.title('Estatísticas do tamanho dos tweets', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.11 – Código em Python para elaboração do gráfico da Figura 4.9.

```
#Boxplot do df video
fig, ax = plt.subplots(figsize=(8,5))
sns.boxplot(y=df_video['Tamanho_Mensagem'])
plt.title('Estatísticas do tamanho dos tweets de vídeo', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.12 – Código em Python para elaboração do gráfico da Figura 4.10.

Especificamente quanto ao tamanho dos *tweets* gerais, a média girou em torno de 70 caracteres, com um mínimo de 0 e um máximo de 207 caracteres. De outro lado, com relação ao tamanho dos *tweets* de vídeo, a média foi 41,50 e os mínimo e máximo foram 26 e 120 caracteres, respectivamente. Essas estatísticas podem ser verificadas nas Figuras 4.13 e 4.14. Os *boxplot* apresentados evidenciam ainda a presença de vários *outliers*.

```
df['Tamanho_Mensagem'].describe()
```

```
count    63326.000000
mean      70.645896
std       29.590005
min        0.000000
25%       49.000000
50%       62.000000
75%       91.000000
max      207.000000
Name: Tamanho_Mensagem, dtype: float64
```

Figura 4.13 – Estatísticas relativas ao tamanho dos tweets.

```
df_video['Tamanho_Mensagem'].describe()
```

```
count      816.000000
mean       41.504902
std         5.698346
min        26.000000
25%        39.000000
50%        41.000000
75%        43.000000
max       120.000000
Name: Tamanho_Mensagem, dtype: float64
```

Figura 4.14 – Estatísticas relativas ao tamanho dos tweets de vídeo.

No intuito de analisar possíveis inconsistências nos dados, foram verificadas as mensagens referentes a cada um dos mínimos e máximos indicados, por meio do código Python apresentado na Figura 4.15. Entretanto, a única informação não esperada (de tamanho 0) foi confirmada como correta, pois o *tweet* original apenas mencionava um *http*, o qual foi destacado em coluna própria do *dataframe*, como anteriormente exposto.

```
df[df['Tamanho_Mensagem']==0]
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
17581	2014-11-13	Thu	2014	http://gu.com/p/438n2/tw/n		0	25	False

```
df[df['Tamanho_Mensagem']==207]
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
39927	2013-07-17	Wed	2013	RT @ronicaryn: The cost of insurance in New Yo...		207	0	False

```
df_video[df_video['Tamanho_Mensagem']==26]
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
136	2015-03-14	Sat	2015	VIDEO: The art of microbes	http://bbc.in/1BD9KTrn	26	22	True

```
df_video[df_video['Tamanho_Mensagem']==120]
```

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
13440	2013-01-30	Wed	2013	VIDEO: BRAND NEW! @JillianMichaels answers a 1...	http://bit.ly/l20lay	120	20	True

```
print('Tamanho min df: {}'.format(df.iloc[17581]['Mensagem_Final']))
print('Tamanho max df: {}'.format(df.iloc[39927]['Mensagem_Final']))
print('Tamanho min df_video: {}'.format(df.iloc[136]['Mensagem_Final']))
print('Tamanho max df_video: {}'.format(df.iloc[13440]['Mensagem_Final']))
```

```
Tamanho min df:
Tamanho max df: RT @paula_span: Is 90 really the new 80? Danish study shows very old doing better, mentally and functionally,
but how much better?
Tamanho min df_video: VIDEO: The art of microbes
Tamanho max df_video: VIDEO: BRAND NEW! @JillianMichaels answers a 13 year-old's questions about body image, watch now: #DDJil
lian #EHYOUTUBE
```

Figura 4.15 – Análise do conteúdo dos tweets de tamanhos máximo e mínimo.

Relativamente aos *http* mencionados nos *tweets*, observou-se que o terceiro quartil, de valor 25, revela que 75% dos dados estão bem próximos da média de 26,46 (Figura 4.16). Quanto àqueles que superam esse tamanho de 25 caracteres, verificou-se que totalizam 11.995 registros. Um exame amostral desses itens revelou que tais *http*, diferentemente dos menores – os quais apenas remetem para a página das agências de notícias do dia corrente – direcionam para páginas com notícias específicas, relacionadas ao teor do *tweet* correspondente. Como o escopo do trabalho se restringe ao estudo dos textos dos *posts*, o conteúdo dessas páginas da

Internet não foi aqui tratado; porém pode ser objeto de projeto posterior, para fins de aprofundamento do estudo (Figura 4.17).

```
df['Tamanho_http'].describe()

count    63326.000000
mean      26.426334
std       21.821225
min        0.000000
25%       20.000000
50%       22.000000
75%       25.000000
max       386.000000
Name: Tamanho_http, dtype: float64
```

Figura 4.16 – Estatísticas relativas ao tamanho dos http citados nos tweets.

```
df[df['Tamanho_http']>25] # total de 11995 http
```

ID	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
63316	2013-12-26	Thu	2013	China Tightens Formula Standards	http://on.wsj.com/19ikqtw/n	32	26	False
63317	2013-12-26	Thu	2013	Reactors on Slow Road to Demolition	http://on.wsj.com/19ikqdh/n	35	26	False
63318	2013-12-26	Thu	2013	Health-Insurance Deadlines Keep Slipping	http://on.wsj.com/1cSuJvV/n	40	26	False
63319	2013-12-25	Wed	2013	Fake Knee Surgery as Good as Real Thing, Study...	http://on.wsj.com/19hQKga/n	52	26	False
63324	2013-12-24	Tue	2013	RT @stefaniei: Health-Insurance Deadline Exten...	http://on.wsj.com/1cOF1BT/n	111	26	False
63325	2013-12-24	Tue	2013	Boston Scientific Eyes China Expansion	http://on.wsj.com/1kBRC4a/n	38	26	False
63326	2013-12-24	Tue	2013	For Desperate Family in India, a Ray of Hope F...	http://on.wsj.com/1kBFGsB/n	57	26	False

11995 rows x 8 columns

```
df.iloc[3929]['http'] #remete a uma página de notícia específica (não é a data corrente, como os comuns=>possível aprofundamento)
```

'http://www.cbc.ca/news/health/drugs-need-careful-monitoring-for-expiry-dates-pharmacists-say-1.3026749?cmp=rss\n'

```
df.iloc[3958]['http'] #também remete a página específica
```

'http://www.cbc.ca/news/world/ghana-how-canada-is-scaling-up-pediatric-nursing-to-save-little-lives-1.3020564?cmp=rss\n'

```
df[df['Tamanho_http']==386]
```

ID	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
49766	2014-10-10	Fri	2014	RT @charlesornstein: @SecBurwell acknowledged...	http://capsules.kaiserhealthnews.org/index.php...	110	386	False

```
df.iloc[49766]['http'] #referência a mais de um http
```

'http://healthcare.gov http://khne.ws/1s1BCIs\n'

Figura 4.17 – Análise do conteúdo dos tamanhos dos http citados nos tweets.

Poder-se-ia pensar também em alguma correlação entre o tamanho dos tweets e o dia da semana, ou entre o tamanho e o ano, mas não se observou nada nesse sentido. Os gráficos de dispersão constantes das Figuras 4.18 e 4.19

demonstram a inexistência de qualquer curva de correlação entre tais dados. Os trechos dos códigos em Python correspondentes constam das Figuras 4.20 e 4.21.

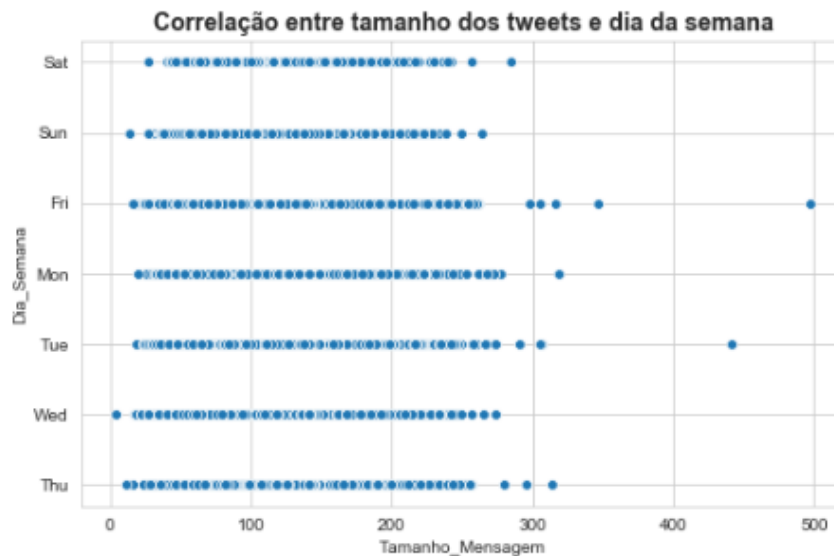


Figura 4.18 – Correlação entre tamanho dos tweets e dia da semana.

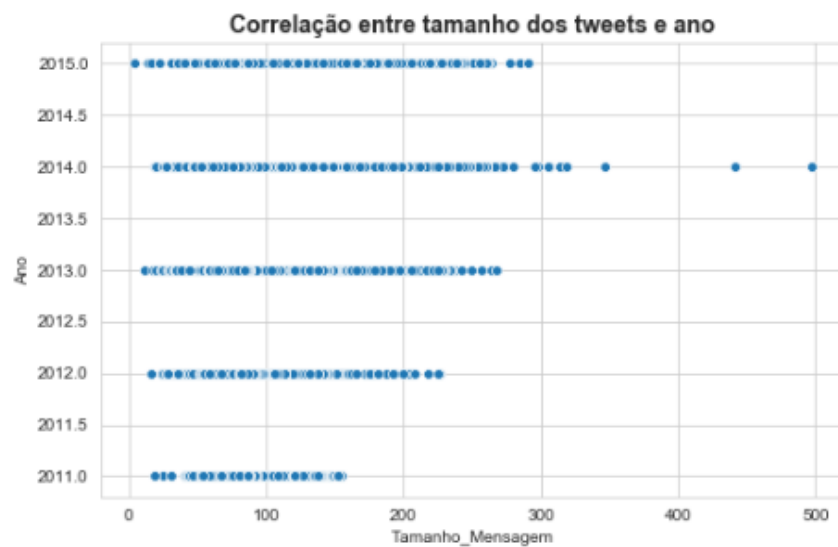


Figura 4.19 – Correlação entre tamanho dos tweets e ano.

```
#Scatterplot tamanho x dia semana df total
fig, ax = plt.subplots(figsize=(8,5))
sns.scatterplot(df['Tamanho_Mensagem'], df['Dia_Semana'])
plt.title('Correlação entre tamanho dos tweets e dia da semana', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.20 – Código em Python para elaboração do gráfico da Figura 4.18.

```
#Scatterplot tamanho x ano df total
fig, ax = plt.subplots(figsize=(8,5))
sns.scatterplot(df['Tamanho_Mensagem'], df['Ano'])
plt.title('Correlação entre tamanho dos tweets e ano', fontsize=14, fontweight='bold')
plt.show()
```

Figura 4.21 – Código em Python para elaboração do gráfico da Figura 4.19.

Em resumo, a quantidade de mensagens seguiu uma curva ascendente até 2014 e sofreu uma queda significativa no ano de 2015, o que claramente está relacionado à data de extração dos dados. Relativamente ao tamanho dos *tweets* em geral e dos *tweets* com vídeos, a análise dos *outliers* não evidenciou nenhum problema de consistência nos dados. No tocante aos *https* citados, como mencionado, embora inexista erro, alguns deles revelam a possibilidade de aprofundamento posterior das análises, haja vista remeterem o usuário a páginas específicas relacionadas a saúde. Além disso, pode-se constatar a inexistência de qualquer correlação entre o tamanho da mensagem e as variáveis “Ano” e “Dia_Semana”.

5. Criação de Modelos de *Machine Learning*

O objetivo do projeto, como mencionado, é identificar tópicos de interesse referentes à saúde, a partir de uma coleção de *tweets* de agências de notícias postados no período de 2011 a 2015.

A proposta, por conseguinte, consiste na identificação de subconjuntos de dados, dentro da coleção de *tweets*, que tenham algum padrão em comum e que possam gerar insumo para tomada de decisões no âmbito da saúde. Essa tarefa se insere no escopo do aprendizado de máquina, o qual, estando relacionado a processos estatísticos, busca o desenvolvimento de algoritmos que proporcionem o aprimoramento de tarefas a partir da experiência passada por meio do processo de indução.

O aprendizado de máquina (*machine learning*) pode ser classificado em: a) aprendizado supervisionado, em que os dados estão rotulados e o algoritmo busca encontrar uma função a partir de dados conhecidos de entradas e de saída, com vistas à predição de uma variável desejada; b) aprendizado não supervisionado, em

que o objetivo é explorar ou descrever um conjunto de dados sem o uso de atributos de saída, com vistas à identificação de padrões não conhecidos; e c) aprendizado por reforço, em que a interação e o *feedback* são necessários ao desenvolvimento do processo [8].

O presente estudo está diretamente ligado ao aprendizado não supervisionado, visto que os dados não estão rotulados e que inexiste uma variável-alvo a ser quantificada ou determinada. O modelo desejado, portanto, deverá buscar estruturas possíveis a partir dos dados de entrada, com o objetivo de identificar um padrão não conhecido previamente.

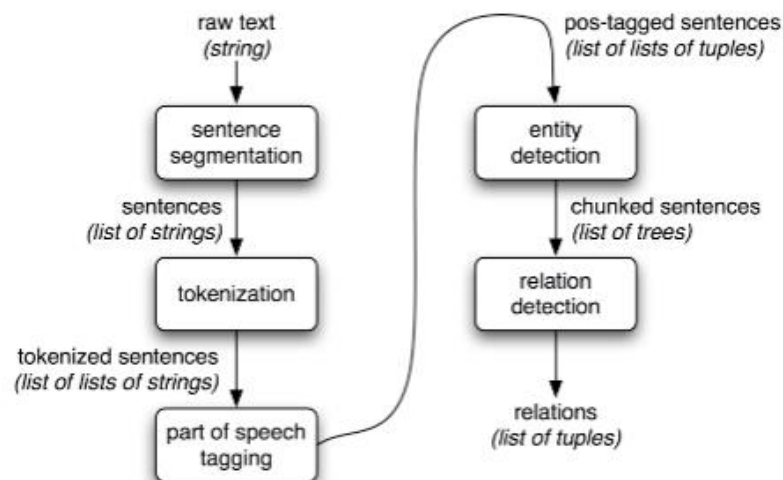
Considerando, ainda, que o objeto do trabalho é identificar tópicos de interesse, ou subconjuntos da coleção de *tweets*, o modelo indicado é de agrupamento ou *clustering*, que busca a descrição do conjunto de dados mediante o seu agrupamento segundo medidas de semelhança. Dois aspectos, no entanto, devem ser registrados neste ponto: o primeiro é que os dados agrupados podem apresentar estruturas heterogêneas, uma vez que cada cluster pode estar em conformidade com um critério de agrupamento diferente; o segundo ponto é que a etapa de validação do modelo geralmente requer um conhecimento profundo de especialistas no domínio dos dados, o que nem sempre está ao alcance do cientista de dados [9].

Os principais métodos utilizados pelos algoritmos de agrupamento são o hierárquico, em que, como o próprio nome diz, os dados são organizados em uma estrutura hierárquica; e o particional, em que os dados são apresentados em clusters ou grupos sem qualquer nível de hierarquia. Neste último, apropriado à presente tarefa, uma partição inicial aleatória baseada em um número k de clusters escolhido previamente é criada e, a partir de um processo iterativo, os elementos são realocados para o aprimoramento do particionamento.

Para a tarefa de agrupamento proposta neste projeto, optou-se pela utilização do algoritmo de particionamento K-Means. Esse é um modelo clássico e muito utilizado, em que cada cluster é representado por um ponto central e a similaridade é calculada com base em medidas de distância. Ou seja, o k-Means é baseado em centroides: cada elemento do conjunto de dados pertence ao cluster cujo centro é mais próximo. O número k de clusters, por outro lado, deve ser escolhido previamente, o que demonstra uma limitação do modelo.

Sob outra perspectiva, tendo em vista que os dados a serem agrupados são textos, é necessário realizar um pré-processamento, de forma que esses dados sejam representados em modelos numéricos que possam ser consumidos por algoritmos de *machine learning*, no caso, o K-Means.

A arquitetura típica de um sistema de extração de informações de textos pode ser representada conforme a Figura 5.1 [10].



Fonte: <http://www.nltk.org/book/ch07.html>

Figura 5.1 – Arquitetura de um sistema de extração de informações de textos.

A sequência apresentada é, portanto: obtenção do dado bruto; segmentação do texto em sentenças; tokenização das sentenças em unidades linguísticas identificáveis (geralmente palavras); marcação dos *tokens* com a parte do discurso a que pertencem; identificação de entidades (segundo a classe gramatical, por exemplo); e extração de relação entre as entidades mapeadas.

Devido a limitações de tempo e de recursos de *hardware*, o presente projeto restringiu-se à tokenização dos *tweets* por meio da construção de um modelo de *Bag of Words*, em que são identificadas as palavras existentes nos diversos documentos e a sua frequência. Esse modelo cria uma matriz em que cada linha representa um documento e cada coluna, uma palavra (a dimensão da matriz é dada pelo número de palavras distintas).

Entretanto, a frequência de uma palavra não implica necessariamente sua relevância, uma vez que algumas delas podem se repetir em todos os documentos sem agregar significativo valor para o cálculo da similaridade, necessário ao

agrupamento desejado. Devido a isso, optou-se pelo uso do modelo TF-IDF, o qual efetua uma ponderação da frequência por meio da equação constante da Figura 5.2 [11], na tentativa de medir a “relevância” de uma palavra no contexto do conjunto de dados trabalhados.

$$\text{tfidf}_{i,j} = \text{tf}_{i,j} \times \log\left(\frac{N}{\text{df}_i}\right)$$

$\text{tf}_{i,j}$ = total number of occurrences of i in j
 df_i = total number of documents (speeches) containing i
 N = total number of documents (speeches)

Figura 5.2 – Equação do TF-IDF.

Estabelecidas as premissas do estudo, importa recuperar a estrutura do *dataframe* em análise, apresentado na Figura 5.3, para a continuidade da análise.

df.head(10)

	Data	Dia_Semana	Ano	Mensagem_Final	http	Tamanho_Mensagem	Tamanho_http	Video
0	2015-04-09	Thu	2015	Breast cancer risk test devised	http://bbc.in/1CimpJF\n	31	22	False
1	2015-04-08	Wed	2015	GP workload harming care - BMA poll	http://bbc.in/1ChTBRv\n	35	22	False
2	2015-04-08	Wed	2015	Short people's 'heart risk greater'	http://bbc.in/1ChTANp\n	35	22	False
3	2015-04-08	Wed	2015	New approach against HIV 'promising'	http://bbc.in/1E6jAjt\n	36	22	False
4	2015-04-08	Wed	2015	Coalition 'undermined NHS' - doctors	http://bbc.in/1CnLwK7\n	36	22	False
5	2015-04-08	Wed	2015	Review of case against NHS manager	http://bbc.in/1Ffj6ci\n	34	22	False
6	2015-04-08	Wed	2015	VIDEO: 'All day is empty, what am I going to do?'	http://bbc.in/1N7wSSz\n	49	22	True
7	2015-04-08	Wed	2015	VIDEO: 'Overhaul needed' for end-of-life care	http://bbc.in/1CmrRu3\n	45	22	True
8	2015-04-08	Wed	2015	Care for dying 'needs overhaul'	http://bbc.in/1FdSGri\n	31	22	False
9	2015-04-07	Tue	2015	VIDEO: NHS: Labour and Tory key policies	http://bbc.in/1Ci5eqD\n	40	22	True

Figura 5.3 – Primeiras linhas do dataframe construído a partir da coleção de tweets.

Do exame do conjunto de dados, observa-se que a matéria-prima essencial para a tarefa proposta reside na coluna “Mensagem_Final”. Ocorre que, no processamento de linguagem natural, é recomendada a normalização do texto para a redução da dimensionalidade da matriz gerada pelos modelos *Bag of Words* e TF-IDF e a consequente minimização do tamanho do modelo numérico.

Para tanto, procedeu-se à construção de uma função em Python – para utilização no parâmetro *analyzer* do modelo *Bag of Words* – para a realização dos seguintes procedimentos (Figuras 5.4 e 5.5): a) exclusão de palavras iniciadas por @, visto não conterem qualquer conteúdo semântico; b) padronização das palavras em minúsculo; c) exclusão de pontuação, dígitos e *stopwords* (palavras tão comuns que pouco significado – ou poder de discriminação – acrescentam ao modelo); d) redução das palavras a seu radical (*stemming*), haja vista a preservação de sua semântica; e e) exclusão das palavras com duas ou menos letras e das palavras com 15 ou mais letras (pois provavelmente revelam erros na obtenção dos dados). Quanto ao corte de 15 letras, registre-se que, neste ponto do processamento, os tokens já estão reduzidos a seu radical, os quais, por natureza, possuem tamanho mais reduzido. As bibliotecas do Python utilizadas nesse processo constam da Figura 5.4.

#PRÉ-PROCESSAMENTO DOS DADOS

```
# Importando bibliotecas
import nltk
import re
import string
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

Figura 5.4 – Bibliotecas utilizadas no pré-processamento dos dados.

```

# Definindo função para limpeza e processamento do texto
def processa_texto(msg):
    #excluindo palavras com @
    sem_arr = [palavra for palavra in msg.split() if palavra[0] != '@']
    sem_arr = ' '.join(sem_arr)

    #retirando as pontuações e transformando caracteres em minúsculo (lower)
    sem_pontuacao = ' '.join([caracter.lower() for caracter in sem_arr.split()
                              if caracter not in string.punctuation])

    #removendo as stopwords
    msg_sem_stopword = [palavra for palavra in sem_pontuacao.split()
                        if palavra.lower() not in stopwords.words('english')]

    #reduzindo palavras ao seu radical (stemming)
    ps = PorterStemmer() #criando objeto para Stemming
    msg_stem = [ps.stem(palavra) for palavra in msg_sem_stopword]

    #excluindo caracteres diferentes de letras e palavras com 2 ou menos letras
    msg_processada = re.sub(r'^[a-zA-Z]', ' ', ' '.join(msg_stem))
    msg_processada = msg_processada.split()
    for palavra in msg_processada:
        if len(palavra) <= 2 or len(palavra) >= 15:
            msg_processada.remove(palavra)

    return msg_processada

```

Figura 5.5 – Função Python para limpeza e pré-processamento dos dados textuais.

Para a criação do modelo *Bag of Words* e sua transformação em TF-IDF foi utilizado o código Python constante da Figura 5.6, tendo sido utilizados os seguintes parâmetros:

- *analyzer*: função “processa_texto” em Python definida para pré-processamento dos dados;
- *min_df*: mínimo de presença em dois *dataframes* para que o token seja considerado no modelo;
- *max_df*: máximo de presença dos tokens em 85% dos *dataframes*, visto que percentual superior revela pouco poder discriminante para o modelo;
- *max_features*: máximo estabelecido de 10.000 *features*, para fins de redução da dimensionalidade da matriz gerada.

```
# Criando um modelo Bag of Words (bow) Total
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer=processa_texto, min_df=2, max_df=.85, max_features=10000)
bow_vector = vectorizer.fit_transform(df['Mensagem_Final']) # cria vocabulário e retorna matriz termo-doc
feature_names=vectorizer.get_feature_names()

# Ponderando e normalizando os dados com TF-IDF usando TfidfTransformer do Scikit-Learn
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer = TfidfTransformer().fit(bow_vector) # transforma contagem da matriz esparsa em idf

# Transformando todo o bag of words em corpus TF-IDF
tfidf_vector = tfidf_transformer.transform(bow_vector)
```

Figura 5.6 – Código em Python para aplicação dos modelos Bag of Words e TF-IDF aos dados de tweets.

A aplicação do modelo sobre o conjunto de dados resultou na matriz esparsa “tfidf_vector”, gerada com 63.326 linhas (registros de *tweets*), 10.000 colunas (radicais de palavras consideradas importantes com base no limite estabelecido no parâmetro “max_features”, para a aplicação de modelos de aprendizado de máquina) e 444.373 ocorrências diferentes de zero (Figura 5.7).

```
tfidf_vector
<63326x10000 sparse matrix of type '<class 'numpy.float64'>'
  with 444373 stored elements in Compressed Sparse Row format>
```

Figura 5.7 – Matriz esparsa gerada pelo modelo TF-IDF.

No intuito de propiciar uma visualização direta dos termos mais frequentes, foram elaboradas as seguintes nuvens de *tags*: a) nuvem gerada a partir dos radicais dos tokens encontrados nos *tweets* (Figura 5.8); b) nuvem gerada a partir dos radicais dos tokens encontrados nos *posts* com vídeos (Figura 5.9); e c) nuvem gerada a partir das *hashtags* identificadas nos *tweets* (Figura 5.10). Os códigos de programação correspondentes encontram-se nas Figuras 5.11, 5.12 e 5.13.

[illegible][illegible]

Figura 5.10 – Nuvem de palavras para hashtags encontradas nos tweets.

```
# Nuvem de palavras com feature_names do df total
from wordcloud import WordCloud

text = ' '.join(feature_names)
wordcloud = WordCloud(max_font_size=100,width = 1520, height = 535, colormap='viridis', background_color='white').generate(text)
plt.figure(figsize=(16,9))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Figura 5.11 – Código em Python para gerar nuvem de tags dos tweets.

```
corpus_video=df_video['Mensagem_Final']

# Criando um modelo Bag of Words (bow) VIDEO
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_video = CountVectorizer(analyzer=processa_texto, min_df=2, max_df=.85, max_features=None)
bow_vector_video = vectorizer_video.fit_transform(corpus_video) # cria vocabulário e retorna matriz termo-doc
feature_names_video=vectorizer_video.get_feature_names()
print(feature_names_video)
print(list(vectorizer_video.vocabulary_.keys())[:10])
print(bow_vector_video.toarray())

# Ponderando e normalizando os dados com TF-IDF usando TfidfTransformer do Scikit-Learn
from sklearn.feature_extraction.text import TfidfTransformer
tfidf_transformer_video = TfidfTransformer().fit(bow_vector_video) # transforma contagem da matriz esparsa em idf

# Transformando todo o bag of words em corpus TF-IDF
tfidf_vector_video = tfidf_transformer_video.transform(bow_vector_video)

# Nuvem de palavras com feature_names do df VIDEO
from wordcloud import WordCloud

text_video = ' '.join(feature_names_video)
wordcloud = WordCloud(max_font_size=100,width = 1520, height = 535, colormap='viridis', background_color='white')
wordcloud.generate(text_video)
plt.figure(figsize=(16,9))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Figura 5.12 – Código em Python para gerar nuvem de tags dos tweets de vídeo.

```
# Nuvem de palavras com hashtags
from wordcloud import WordCloud

text_hash = hashtags

wordcloud = WordCloud(max_font_size=100,width = 1520, height = 535, max_words=200, background_color='white', collocations=False)
wordcloud.generate(text_hash)
plt.figure(figsize=(16,9))
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

Figura 5.13 – Código em Python para gerar nuvem de hashtags encontradas nos tweets.

account, respectivamente. Os códigos em Python correspondentes utilizados nessa extração constam das Figuras 5.17 e 5.18.

```
Cumin recalled for unreported traces of almonds
RT @kimbrunhuber: Study in CMAJ: Kids who only drink non-cow's milk (like almond & soy) have > 2x chance of VitD deficiency as kids who only...

Peanut and almond butters recalled over salmonella risk
Peanut & almond butters sold @TraderJoesList @kroger @Safeway @WholeFoods recalled over salmonella fears
RT @namibycandy: Quinoa flour pancakes topped w/ almond butter & strawberries! ☺ RT @cnnhealth What's in ur breakfast? Show us for RT!
RT @goodhealth: #BREAKFAST: Yummy banana and almond butter toast is just 280 calories!
RT @ENERGYbits: @EverydayHealth A5 I will never let myself run out of coconut oil or almond milk! #healthtalk

RT @IntegHealthRev: @EverydayHealth A3: We love almond milk, or any nut milk for that matter. #healthtalk

Almond milk, almond butter, chia seeds, and more trendy health foods with serious benefits:
RT @CynthiaSass: @goodhealth Q2 Add good-4-u fat to smoothies (avocado, almond butter, coconut oil)-it helps boost antioxidant absorption ...

Make these honey-and-chili-glazed almonds for smart snacking this weekend:
RT @cynthiasass: @goodhealth Q3 Rather than green bean casserole whip up fresh green beans sautéed in olive oil topped with sliced almonds ...
```

Figura 5.15 – Tweets com radical almond.

```
RT @NightShiftMD: Beth: Beth's full & astonishing account of why baby Michael died a preventable death.

Readers share 1st-person Yaz, Yasmin accounts
24 @twitter accounts to follow to understand #EbolaOutbreak. @drsanjaygupta is No. 3 via @voxdotcom

Hey there! @DrSanjayGupta here. I'm taking over this account until 3p ET to answer your questions about brain health. Let's chat! #AskSanjay

The importance of making 'deposits' into our health bank accounts from @AgapiSays:
RT @Sprouttk2001: #healthtalk:Q6 Thanks to my family holding me accountable I have been achieving my goals in 2013

RT @LHawke45: A co-worker and I txt each other every morning before we work out. A little accountability to get me going. #HealthTalk

RT @sarahstanley: Share your goals with others so you can be held accountable. #healthtalk

RT @theRCN: Did you know, nursing staff account for around 1,800 voters in each constituency? Pledge to vote on 7 May > Some people think all we do in NHS payroll is press a button and cash appears in people's bank accounts on payday. Read one person's account of what it's like living with Parkinson's
Mark Porter responds that #NHS can't be depoliticised when it accounts for so much public spending #GuardianLiveNHS
```

Figura 5.16 – Tweets com radical account.

```
var = df['Mensagem_Final'].tolist()

for msg in var:
    for w in msg.split():
        if w == 'almond' or w == 'almonds':
            print(msg)
```

Figura 5.17 – Código em Python utilizado para extrair tweets com tag almond.

```
#var = df['Mensagem_Final'].tolist()

for msg in var:
    for w in msg.split():
        if w[:7] == 'account':
            print(msg)
```

Figura 5.18 – Código em Python utilizado para extrair tweets com tag account.

outra questão que se põe é a discussão sobre planos de saúde, revelada pela presença das *tags* *Obamacare* e *healthcare* na Figura 5.20.

Efetuada a exploração visual das nuvens de palavras e a extração de informações pelo tamanho dos termos e pelas diferenças entre os dados apresentados nas três nuvens, passa-se a uma questão sempre presente nos casos de processamento de linguagem natural: a necessidade de redução da dimensionalidade das matrizes esparsas geradas pelos modelos de processamento de linguagem natural, tais como o TF-IDF utilizado neste projeto, antes da realização do agrupamento dos dados pelo modelo de clusterização. Essa necessidade decorre da dificuldade conhecida como “maldição da dimensionalidade”, em que o desempenho dos algoritmos de reconhecimento de padrões deteriora-se em demasia quando aplicados a matrizes esparsas [13].

Uma das técnicas de redução de dimensionalidade mais conhecidas, consistentes na extração e combinação de alguns atributos para geração de novos, é *Principal Component Analysis* (PCA). Porém, como esse algoritmo não pode ser utilizado com matrizes esparsas [14], que é o presente caso, tentou-se efetuar a redução da dimensionalidade com o algoritmo *SVDTruncated*, que utiliza uma técnica de álgebra linear de fatoração de matrizes, conhecida como *singular value decomposition* (SVD) [13]. Para tanto, foi utilizado o código em Python elaborado por Sawant [15] para a realização de testes com número de componentes variando de 100 a 3.500. Entretanto, não foi possível executar o código citado por limitações de *hardware*, que ocasionaram erro de memória (Figura 5.21). Embora tenha sido possível obter um cálculo até o número de 1.000 componentes, como a razão da variância explicada foi insatisfatória, inferior a 60%, decidiu-se não realizar a mencionada redução da dimensionalidade da matriz esparsa.

```

from sklearn.decomposition import TruncatedSVD

data = tfidf_vector
n_comp = [100,150,200,500,800,900,1000,1500,2000,2500,3000,3500] # List containing different values of components
explained = [] # explained variance ratio for each component of Truncated SVD
for x in n_comp:
    svd = TruncatedSVD(n_components=x)
    svd.fit(data)
    explained.append(svd.explained_variance_ratio_.sum())
    print("Number of components = %r and explained variance = %r"%(x,svd.explained_variance_ratio_.sum()))
plt.plot(n_comp, explained)
plt.xlabel('Number of components')
plt.ylabel("Explained Variance")
plt.title("Plot of Number of components v/s explained variance")
plt.show()

Number of components = 100 and explained variance = 0.15846202146998145
Number of components = 150 and explained variance = 0.20635209524319253
Number of components = 200 and explained variance = 0.24631432603302272
Number of components = 500 and explained variance = 0.40679572531538155
Number of components = 800 and explained variance = 0.5090672130407304
Number of components = 900 and explained variance = 0.5362219823004944
Number of components = 1000 and explained variance = 0.5609105649288224

-----
MemoryError                                Traceback (most recent call last)
<ipython-input-96-19c45cab0af8> in <module>()
      7 for x in n_comp:
      8     svd = TruncatedSVD(n_components=x)
----> 9     svd.fit(data)

```

Figura 5.21 – Código em Python para gerar análise de explicação de variância pelo modelo SVD.

Pois bem. Obtida a matriz esparsa com o valor ponderado TF-IDF de cada uma das *tags* consideradas, e tendo em vista a inviabilidade de se reduzir a dimensionalidade dessa matriz, passou-se à especificação dos parâmetros necessários ao modelo de agrupamento para clusterização dos *tweets*, o já mencionado K-Means.

Para fins de determinação do número *k* ideal de clusters, parâmetro necessário ao algoritmo, buscou-se encontrar, pelo Método do Cotovelo, o ponto em que a soma dos quadrados das distâncias dentro dos *clusters* (ou somatório da variância dos dados) é otimizada. O resultado da utilização do código constante da Figura 5.22, extraído de artigo publicado na Internet [16], revelou a inexistência de um ponto ótimo (de inflexão) relativamente aos dados aqui trabalhados – veja-se que a curva apresentada na Figura 5.23 se aproxima de uma reta. De fato, esse problema ocorre muito quando se trabalha com textos, em que a dimensionalidade da matriz esparsa é muito elevada.

```
#PROCURANDO O K IDEAL PARA K-MEANS
```

```
#https://minerandodados.com.br/algorithm-k-means-python-passo-passo/  
#Método do Cotovelo
```

```
from sklearn.cluster import KMeans  
wcss = []  
for i in range(1,30):  
    kmeans = KMeans(n_clusters=i, random_state=0)  
    kmeans = kmeans.fit(tfidf_vector)  
    print (i, kmeans.inertia_)  
    wcss.append(kmeans.inertia_)  
plt.plot(range(1,30), wcss)  
plt.title('Método do Cotovelo', fontweight= 'bold')  
plt.xlabel('Número de Clusters')  
plt.ylabel('WSS - within cluster sum of squares')  
plt.show()
```

Figura 5.22 – Código em Python para aplicação do Método do Cotovelo.

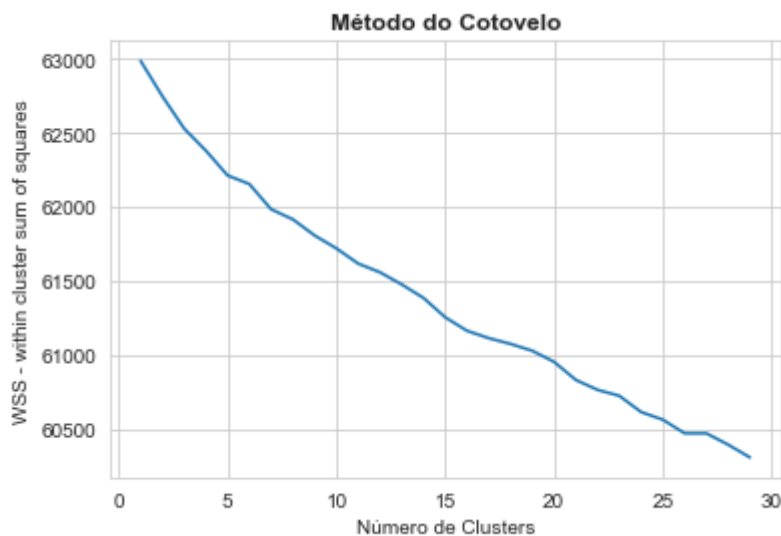


Figura 5.23 – Método do Cotovelo aplicado aos dados de tweets.

Outra opção para escolha do k ideal seria a utilização do método *Silhouette*, que fornece uma representação gráfica de quão bem cada objeto foi classificado. A silhueta varia de -1 a +1, sendo que, quanto mais alto o valor, maior a correspondência do objeto ao próprio cluster [17]. No entanto, esse cálculo também esbarraria na “maldição da dimensionalidade” anteriormente mencionada, motivo pelo qual não foi realizado. Uma solução seria a consulta a especialistas na área, o que, no presente caso, é inviável, dado o caráter acadêmico e restrito do projeto.

Diante da impossibilidade de se encontrar um *k* ótimo por meio do Método do Cotovelo, decidiu-se utilizar o modelo K-Means para clusterização dos *tweets* em 10 (dez) grupos, uma vez que esse número de clusters torna viável a análise não técnica do agrupamento por não ser excessivo. O parâmetro *random_state* foi indicado como 0 (zero) para permitir a reprodução do modelo construído (Figura 5.24). Treinado o modelo, criou-se uma outra coluna no *dataframe*, intitulada “Cluster”, com o rótulo gerado pelo algoritmo como resultado de sua aplicação ao conjunto de dados sob análise. O resultado do agrupamento consta da Figura 5.25.

#TREINAMENTO DE MODELOS

```
from sklearn.cluster import KMeans
num_clusters = 10
kmeans = KMeans(n_clusters=num_clusters, random_state=0)
kmeans = kmeans.fit(tfidf_vector)
centroides = kmeans.cluster_centers_
labels = kmeans.labels_
df["Cluster"] = kmeans.fit_predict(tfidf_vector)
```

kmeans

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=10, n_init=10, n_jobs=1, precompute_distances='auto',
       random_state=0, tol=0.0001, verbose=0)
```

centroides

```
array([[2.34154707e-03, 1.56516258e-04, 5.25885875e-05, ...,
        1.43123949e-05, 0.00000000e+00, 6.89697412e-06],
       [1.60246356e-04, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.50831960e-03, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [1.92981092e-03, 1.60462766e-03, 0.00000000e+00, ...,
        1.18235656e-04, 0.00000000e+00, 0.00000000e+00],
       [1.63256867e-03, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.21882331e-03, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00]])
```

labels

```
array([5, 8, 5, ..., 7, 0, 0])
```

Figura 5.24 – Código em Python para aplicação do modelo KMeans aos dados de tweets.

```
df_clusters = df["Cluster"].value_counts()
print(df_clusters)
print(type(df_clusters))
```

0	41015
3	3967
7	3912
9	2856
2	2517
4	2326
6	1860
5	1753
8	1651
1	1469

```
Name: Cluster, dtype: int64
<class 'pandas.core.series.Series'>
```

Figura 5.25 – Código em Python para demonstração dos clusters.

Foi assim atribuído um cluster para cada uma das mensagens existentes. Conforme se observa na Figura 5.26, o grupo com maior número de *tweets* foi o 0, seguido, com grande diferença, pelos clusters 3, 7 e 9. Essa informação pode ser facilmente extraída da análise dos gráficos relativos às quantidades de *posts* por cluster, constantes das Figuras 5.26, 5.27 e 5.28. O código em Python utilizado para geração desses gráficos está apresentado na Figura 5.29.

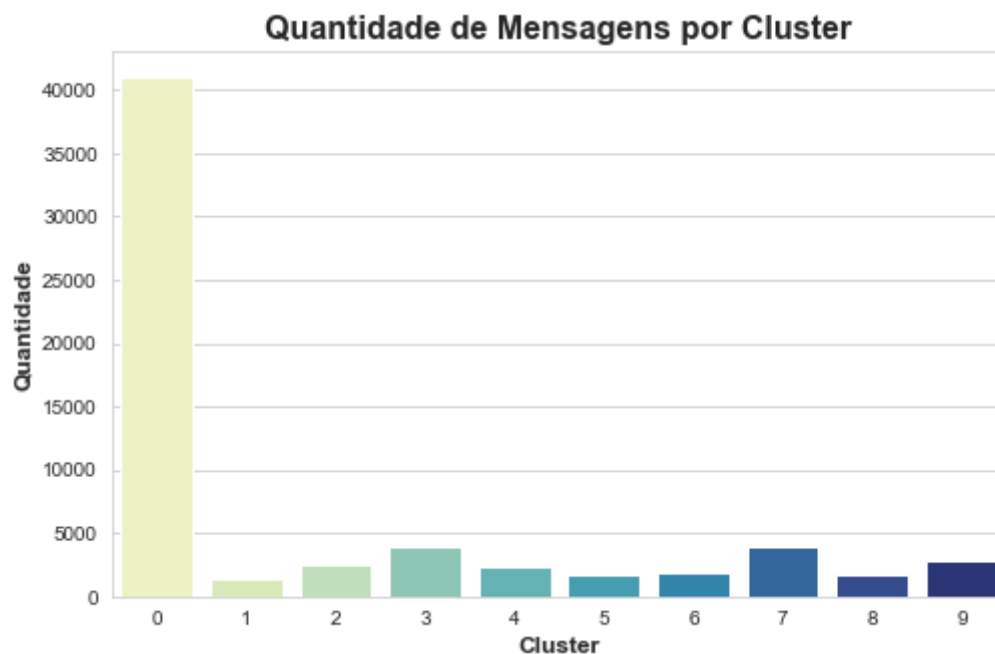


Figura 5.26 – Quantidade de tweets por cluster (KMeans).

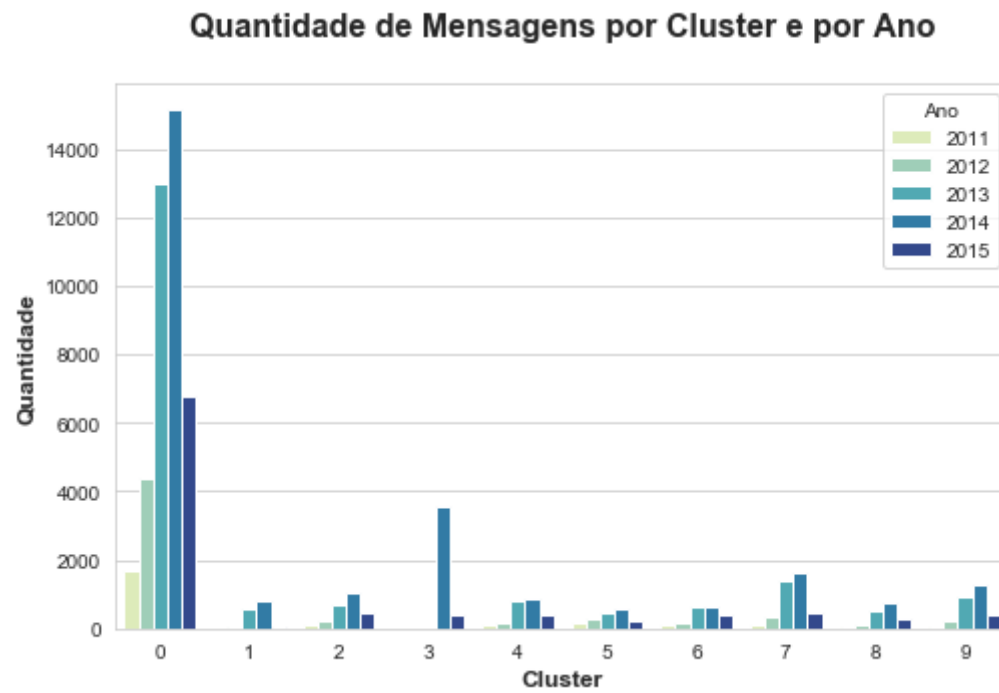


Figura 5.27 – Quantidade de tweets por cluster e por ano (KMeans).

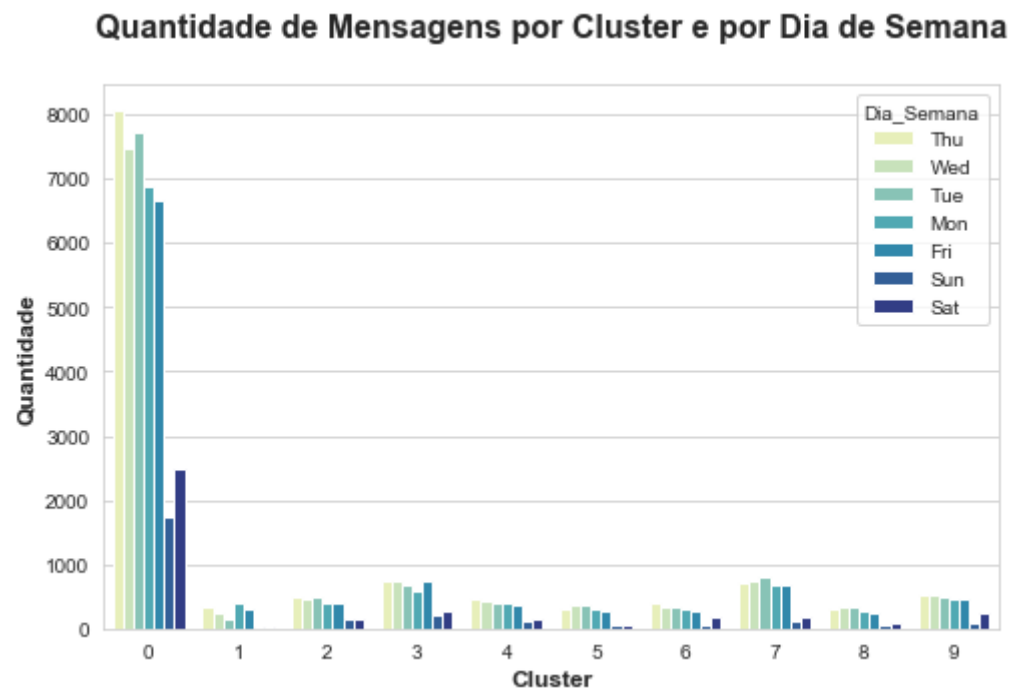


Figura 5.28 – Quantidade de tweets por cluster e por dia da semana (KMeans)


```
fig, ax = plt.subplots(figsize=(8,5))
sns.countplot(df['Cluster'], palette='YlGnBu')
plt.title('Quantidade de Mensagens por Cluster', fontsize=16, fontweight='bold')
ax.set_xlabel('Cluster', fontsize=12, fontweight='bold')
ax.set_ylabel('Quantidade', fontsize=12, fontweight='bold')
plt.show()
```

```
fig, ax = plt.subplots(figsize=(8,5))
sns.countplot(df['Cluster'], hue=df['Ano'], palette='YlGnBu')
plt.title('Quantidade de Mensagens por Cluster e por Ano\n', fontsize=16, fontweight='bold')
ax.set_xlabel('Cluster', fontsize=12, fontweight='bold')
ax.set_ylabel('Quantidade', fontsize=12, fontweight='bold')
plt.show()
```

```
fig, ax = plt.subplots(figsize=(8,5))
sns.countplot(df['Cluster'], hue=df['Dia_Semana'], palette='YlGnBu')
plt.title('Quantidade de Mensagens por Cluster e por Dia de Semana\n', fontsize=16, fontweight='bold')
ax.set_xlabel('Cluster', fontsize=12, fontweight='bold')
ax.set_ylabel('Quantidade', fontsize=12, fontweight='bold')
plt.show()
```

Figura 5.29 – Código em Python para elaboração dos gráficos das Figuras 5.26, 5.27 e 5.28.

Na sequência, foi extraída uma lista com os termos mais comuns de cada cluster, para fins de comparação. A referida lista consta da Figura 5.30 e o código em Python correspondente na Figura 5.31.

```
Termos mais frequentes por cluster:
Cluster 0:
may cancer say studi drug food make doctor eat amp
Cluster 1:
well ask doctor cancer live exercis think may like new
Cluster 2:
help video may weight could cancer lose food studi patient
Cluster 3:
ebola liberia s africa outbreak patient sierra case say leon
Cluster 4:
get way health help fit here new kid like free
Cluster 5:
risk cancer may heart rais increas studi higher lower link
Cluster 6:
like work look opinion sell still smoothli november kidney end
Cluster 7:
health law insur mental exchang plan say state may today
Cluster 8:
care health new afford act hospit patient social cost home
Cluster 9:
new old blog age york studi drug year find pay
```

Figura 5.30 – Termos mais frequentes por cluster.

```

print('Termos mais frequentes por cluster:')

order_centroids = centroides.argsort()[ :, ::-1]
terms = vectorizer.get_feature_names()
for i in range(10):
    print('Cluster {}'.format(i))
    for ind in order_centroids[i, :10]:
        print(' %s' % terms[ind], end='')
    print()

```

Figura 5.31 – Código em Python para elaboração da lista constante da Figura 30.

Por fim, foi construído um *dataframe* com as palavras mais comuns de cada um dos clusters, para fins de aprofundamento da análise iniciada com a elaboração dos gráficos de quantidade. Esse *dataframe* foi elaborado por meio do código em Python constante da Figura 5.32 e o gráfico correspondente, por meio do código da Figura 5.33.

```

#feature_names=vectorizer.get_feature_names()
#centroids = kmeans.cluster_centers_
#labels = kmeans.labels_

bow_transformer = vectorizer.fit(df['Mensagem_Final'])

print('Palavras mais frequentes por cluster:')
order_centroids = centroides.argsort()[ :, ::-1]
lista_terms=[]

for i in range(0, 10):
    print('\n Cluster {}'.format(i))
    for x in order_centroids[i, :10]:
        cluster = i
        term = feature_names[x]
        valor = tfidf_transformer.idf_[bow_transformer.vocabulary_[term]]
        print(term, ': ', valor)
        lista_terms.append([i, term, valor])

df_terms = pd.DataFrame(lista_terms, columns=[ 'Cluster','Termo', 'Tf-Idf'])

```

Figura 5.32 – Código em Python para construção de dataframe com termos mais comuns por cluster.

```

# Plotando gráficos de palavras mais frequentes por cluster (continuação)
for i in range(0,10):
    fig, ax = plt.subplots(figsize=(8,4))
    plt.title('Palavras mais frequentes no Cluster {}'.format(i), fontsize=14, fontweight='bold')
    sns.barplot(data=df_terms[df_terms['Cluster']==i], x='Tf-Idf', y='Termo', orient='h', palette='YlGnBu')
    ax.set_xlabel('TF-IDF', fontsize=12, fontweight='bold')
    ax.set_ylabel('Termo', fontsize=12, fontweight='bold')
    plt.show()

```

Figura 5.33 – Código em Python para elaboração de gráficos de palavras mais frequentes por cluster.

Os gráficos gerados, relativos aos termos mais frequentes para cada um dos grupos, constam das Figuras 5.34 a 5.43.

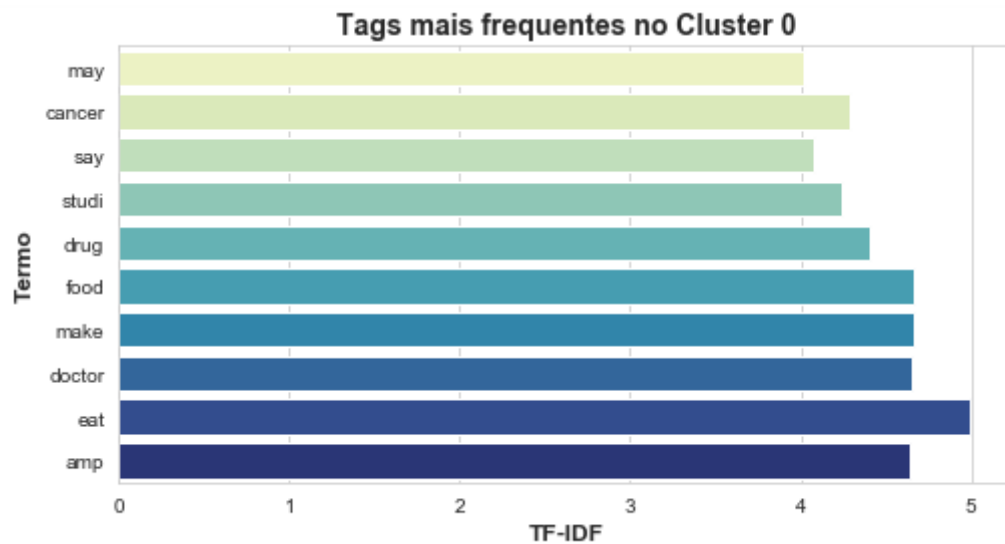


Figura 5.34 – Tags mais frequentes no Cluster 0.

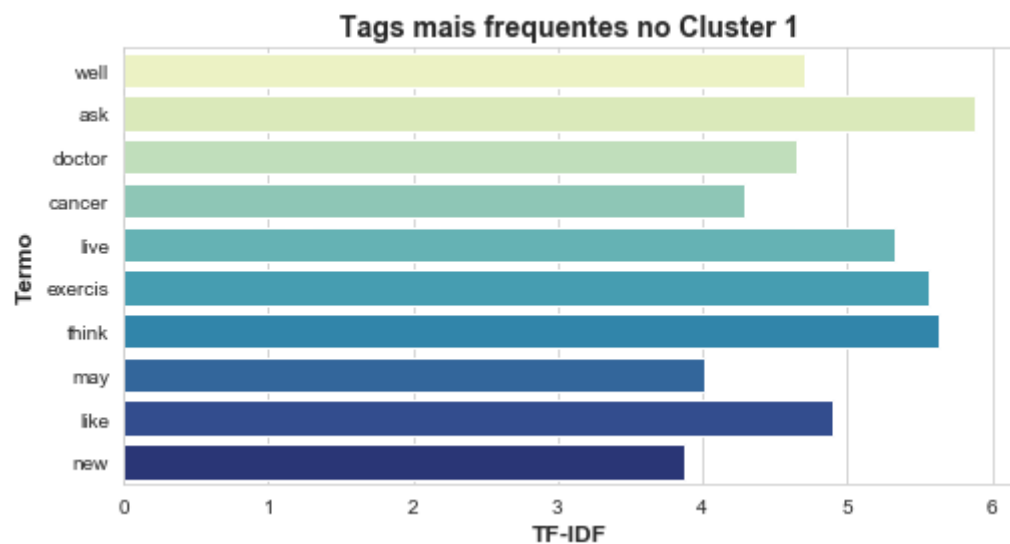


Figura 5.35 – Tags mais frequentes no Cluster 1.

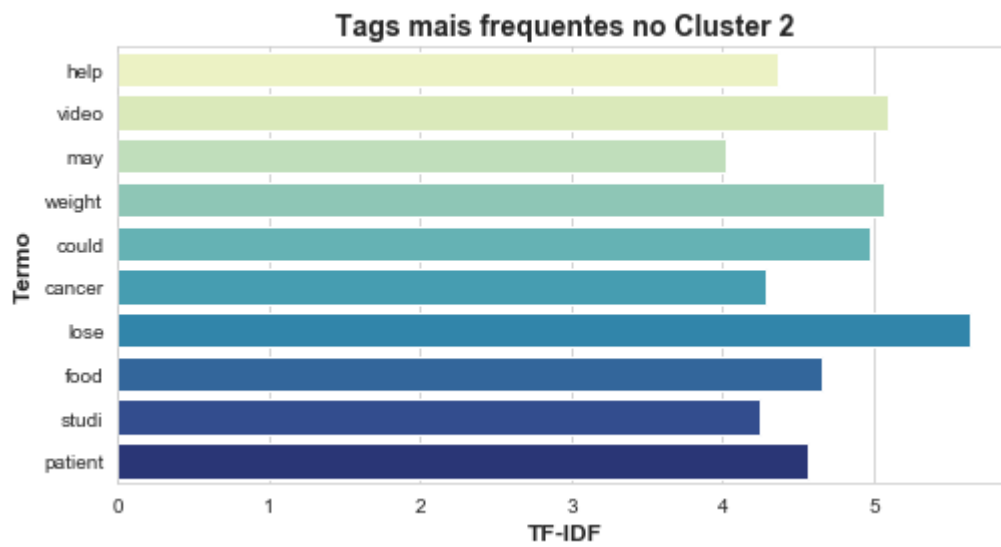


Figura 5.36 – Tags mais frequentes no Cluster 2.

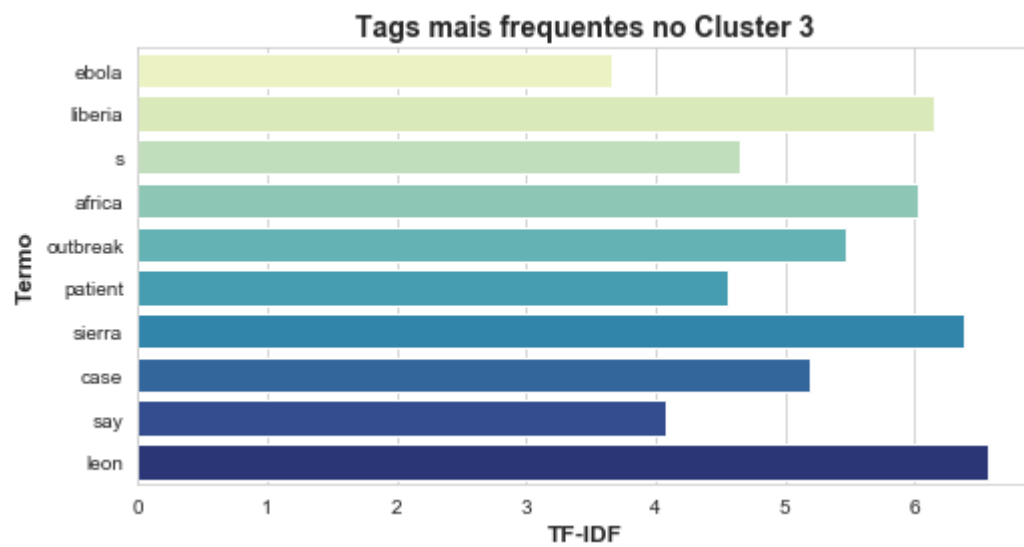


Figura 5.37 – Tags mais frequentes no Cluster 3.

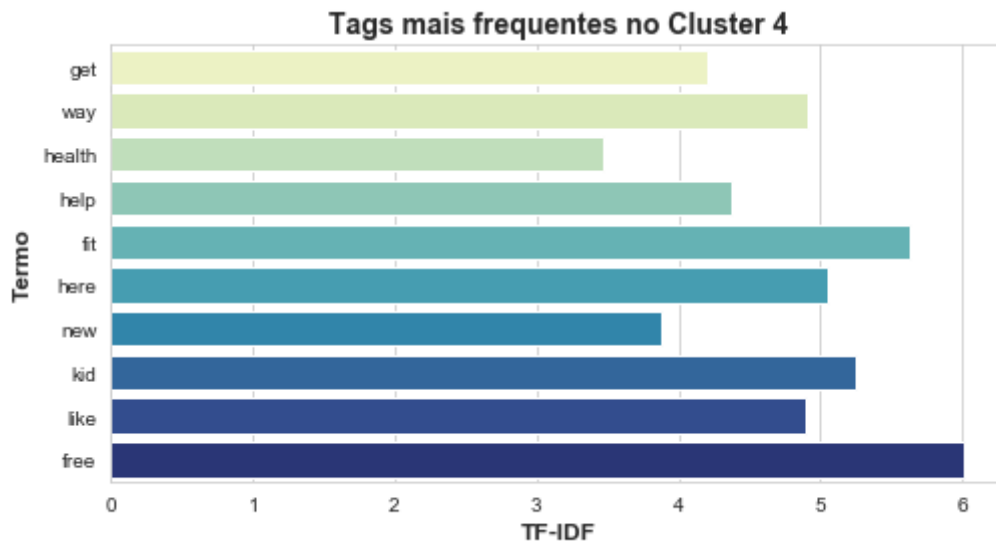


Figura 5.38 – Tags mais frequentes no Cluster 4.

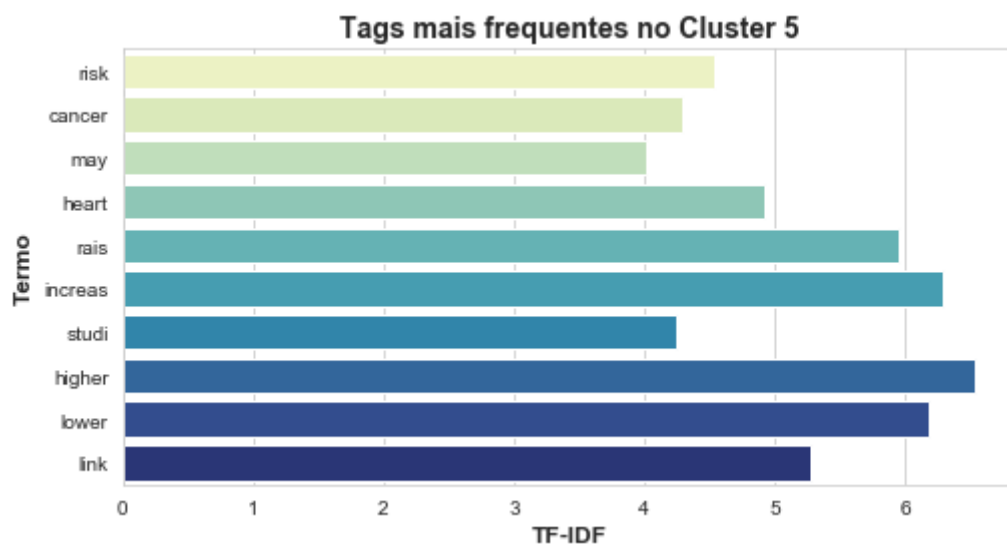


Figura 5.39 – Tags mais frequentes no Cluster 5.

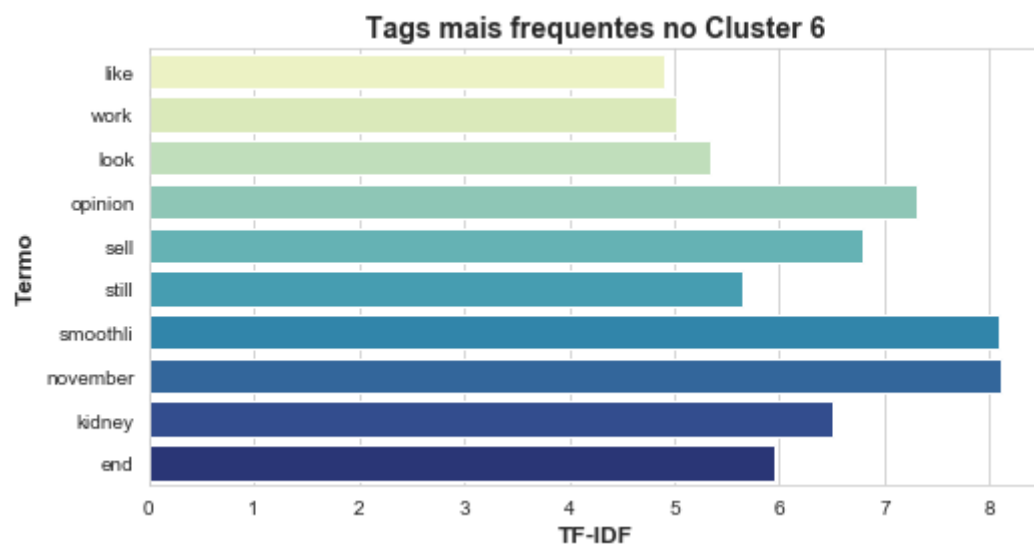


Figura 5.40 – Tags mais frequentes no Cluster 6.

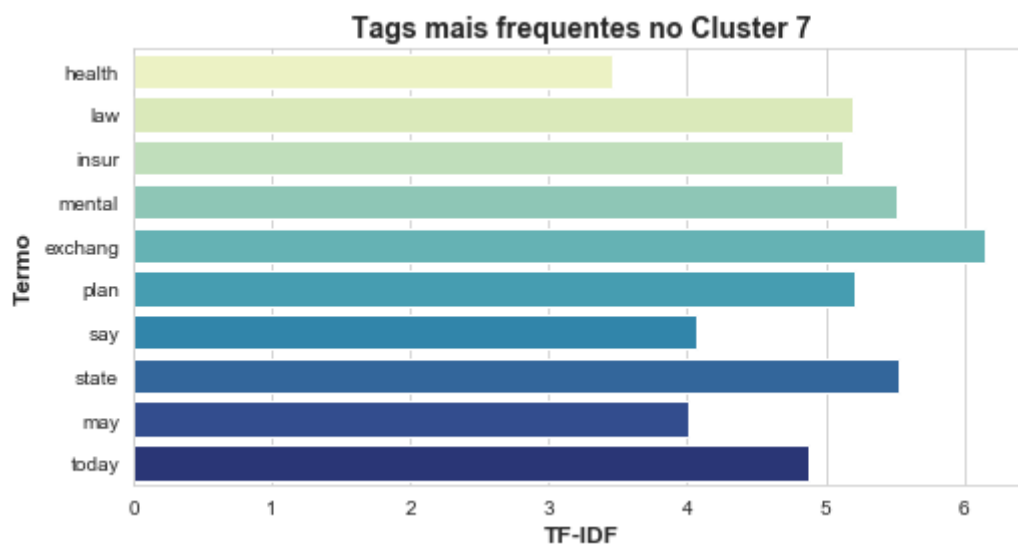


Figura 5.41 – Tags mais frequentes no Cluster 7.

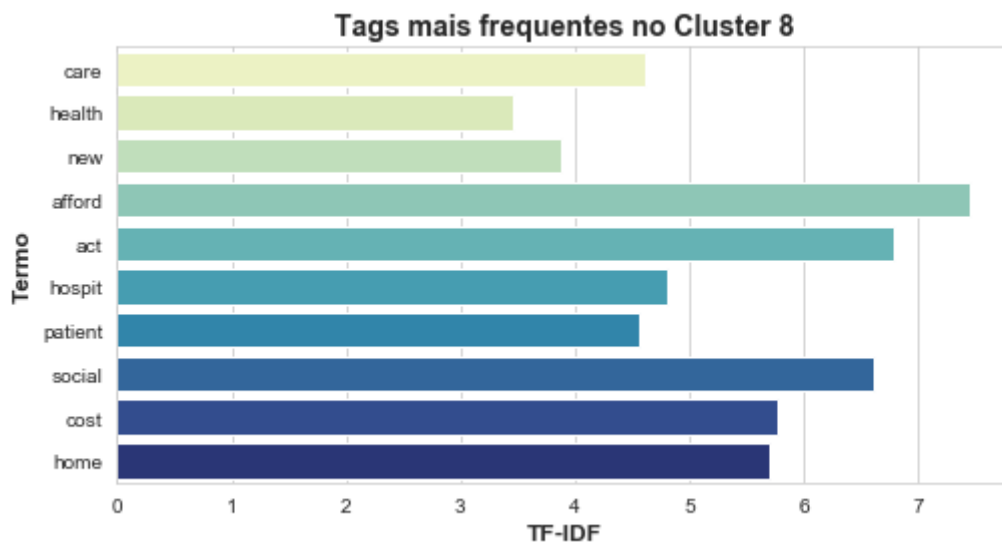


Figura 5.42 – Tags mais frequentes no Cluster 8.

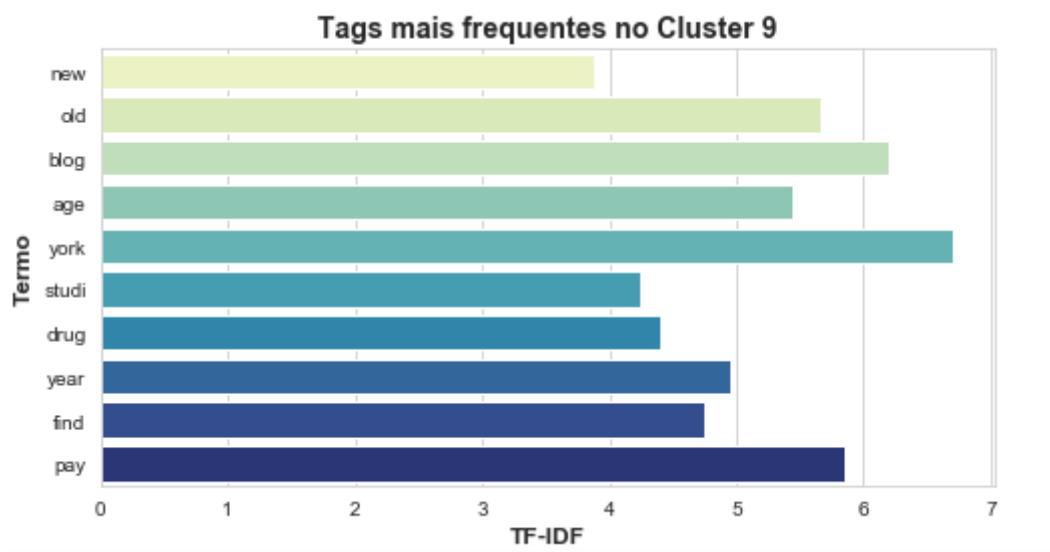


Figura 5.43 – Tags mais frequentes no Cluster 9.

O cluster com maior quantidade de *tweets* foi o de rótulo 0; e as dez *tags* mais frequentes desse rótulo foram: *may*, *cancer*, *say*, *studi*, *drug*, *food*, *make*, *doctor*, *eat*, *amp*. Uma verificação inicial poderia sugerir o agrupamento de *tweets* em torno da doença câncer, abrangendo notícias de estudos, opiniões médicas, medicações e alimentação. Porém, a simples análise dessas *tags* não permite interpretar com segurança o critério do agrupamento, inclusive porque a *tag cancer* também aparece como uma das palavras mais frequentes nos clusters 1, 2 e 5. Por outro lado, a quantidade excessiva de *tweets* classificados nesse cluster torna mais difícil

ainda sua interpretação. Interessante observar, ainda, que a nuvem de *tags* dos *tweets*, anteriormente examinada, não indicou a *tag cancer* como uma das mais frequentes, o que sugere a necessidade de validação do critério de agrupamento por especialista da área.

Exemplos de mensagens agrupadas no cluster 0 constam da Figura 5.44.

```
df[df['Cluster']==0]['Mensagem_Final']
```

4	Coalition 'undermined NHS' - doctors
5	Review of case against NHS manager
10	Have GP services got worse?
12	Parties row over GP opening hours
13	Why strenuous runs may not be so bad after all
16	80,000 'might die' in future outbreak
17	Skin cancer 'linked to holiday boom'
18	Public 'back tax rises to fund NHS'
21	Five ideas to transform the NHS
22	Personal cancer vaccines 'exciting'
23	Child heart surgery deaths 'halved'
25	Unsafe food 'growing global threat'
27	Ambulance progress 'not fast enough'
28	Children's hospital builds sleep app
29	Drug giant 'blocks' eye treatment
30	Blood test for Down's syndrome hailed
32	Paracetamol 'no good for back pain'
35	MS drug 'may already be out there'
38	Uganda circumcision truck fights HIV
40	E-cigarette use 'high among teens'
41	Medieval eye remedy 'kills MRSA'
42	Parents rarely spot child obesity
44	Chikungunya revives herbal remedies in Antigua
45	The Bolivian women who knit parts for hearts
46	Meningitis B vaccine deal agreed
51	Tories to pledge 'seven-day NHS'
53	NHS: Labour's private profits cap
55	Labour to cap private profits in NHS
59	Office workers 'too sedentary'
60	NHS medical accidents unit 'needed'

Figura 5.44 – Exemplos de tweets classificados no cluster 0.

O cluster 3, segundo grupo em quantidade de *tweet*, apresentou as seguintes *tags* como mais frequentes: *ebola*, *liberia*, *s*, *africa*, *outbreak*, *patient*, *sierra*, *case*, *say*, *leon*. A Figura 5.27, por seu turno, indica que os *posts* se concentraram nos anos de 2014 e 2015. De fato, o tema sugerido pelas palavras mais frequentes – relacionados ao Ebola - esteve mais fortemente presente nesse período. Observe-se

que, segundo a Organização Mundial de Saúde, o pico da transmissão ocorreu em agosto e setembro de 2014 e, em maio de 2015, a Libéria estava livre da transmissão do vírus Ebola [18].

O grupo seguinte, em termos de quantidade de *tweets*, foi o de rótulo 7, com as seguintes *tags* mais frequentes: *health, law, insur, mental, exchang, plan, say, state, may, today*. O tópico correspondente parece girar em torno de discussões referentes a custos da saúde, abarcando questões relativas a planos de saúde, leis, seguros e similares. Saliente-se que esse foi um dos temas relevantes observados por meio da visualização da nuvem de *hashtags*, apresentada na Figura 5.13.

Por fim, o cluster 9, quarto lugar em termos de expressividade em sua quantidade, apresentou as seguintes *tags* como mais frequentes: *new, old, blog, age, york, studi, drug, year, find, e pay*. Tais *tags* sugerem o possível interesse em notícias relacionadas a estudos e descobertas referentes a saúde.

Dada a quantidade menos expressiva dos demais clusters, eles não foram considerados relevantes para fins de identificação de possíveis tópicos de interesse na área da saúde.

Efetuada, assim, a análise das variáveis, por meio da visualização de nuvens de *tags* e da geração de gráficos de contagens e de frequências, passa-se à apresentação das conclusões obtidas.

6. Apresentação dos Resultados

Milhares de mensagens são escritas diariamente acerca de temas relevantes para a humanidade, entre os quais a saúde. Como exemplo, citam-se *tweets* de agências de notícias como BBC, CNN e New York Times. Entretanto, extrair informações desses dados não é tarefa fácil: não há rótulos para essas mensagens, os *tweets* são escritos em linguagem natural com pouca ou nenhuma estruturação, e vários são os assuntos abordados. Vejam-se os exemplos da Figura 6.1.

[An abundance of online info can turn us into e-hypochondriacs. Or, worse, lead us to neglect getting the care we need <http://cnn.it/1l1t1Fv>
 [A plant-based diet that incorporates fish may be the key to preventing colorectal cancers: <http://cnn.it/1xdpsjT> <http://pbs.twimg.com/media/CAARHEGNEAAJGz6.jpg>
 [It doesn't take much to damage your hearing at a sports bar or nightclub. That's why a billion people are at risk. <http://cnn.it/1B0phBk>
 [RT @CNN: Forever young? Discover this island's secrets to longevity on #TheWonderList w/ @BillWeirCNN <http://cnn.it/1Gldmqc> <https://t.co/...>
 [RT @CNN: Is post-traumatic stress disorder in your genes? A simple blood test may one day help tell you <http://cnn.it/1x1s8v5> <https://t.co/...>
 [Maysoon Zayid, a touring standup comic with Cerebral Palsy, has a message to share. <http://cnn.it/1GNiH0L> http://pbs.twimg.com/media/B_ubV_UQAAUN00.jpg
 [How women can wipe out Alzheimer's, from @mariashriver. <http://cnn.it/1Ak1XJQ> http://pbs.twimg.com/media/B_sQM1UUAAGMre.jpg
 [RT @CNNOpinion: Women can defeat #Alzheimers, says @mariashriver. #WipeOutAlz challenge will make it happen. <http://cnn.it/1Ak1XJQ> <http://...>
 [Is it time to raise the legal smoking age? <http://cnn.it/1GNhBC8> http://pbs.twimg.com/media/B_rngYFuWAAALomW.jpg
 [CDC: Misuse of garments may have led to release of bioterror bacteria at Tulane monkey lab. <http://cnn.it/18HDKDg>
 [Losing a brain tumor, gaining perspective: CNN's Jessica Moskowitz's #FirstPerson experience. <http://cnn.it/1Eh1rjJ> http://pbs.twimg.com/media/B_-tiaAUgAArA_6.jpg
 [You may be your germs: Microbe genes slipped into human DNA, study says. <http://cnn.it/1AukT5d> http://pbs.twimg.com/media/B_-rUKAUUAAXUyO.jpg
 [RT @CNN: A plant-based diet that incorporates fish may be the key to preventing colorectal cancers: <http://cnn.it/1x1ffMU> <https://t.co/O1u6...>
 [#FitNation: Finding the right life balance between family, work and getting fit. <http://cnn.it/18GEfgh> http://pbs.twimg.com/media/B_-p3vAU8AAKb-F.jpg
 [Robert Downey Jr. presents a child with his own 'Iron Man' robotic arm. <http://cnn.it/1Gxrm3R> http://pbs.twimg.com/media/B_-o-RyU1AAHCKo.jpg
 [RT @cnnntech: Tim Cook tried to give Steve Jobs his liver <http://cnnmon.ie/1Eh7Usn> via @DavidGoldmanCNN http://pbs.twimg.com/media/B_-Sd8DVAADtY7.jpg
 [RT @drsanjaygupta: what are you having for dinner? a lot more #sugar thank you think.. <http://cnn.it/1MuzOTx> http://pbs.twimg.com/media/B_6NZFnUkAAebfM.jpg
 [RT @cnnbrk: U.S. Ebola patient headed to National Institutes of Health. <http://cnn.it/18xKGIa>
 [RT @cnn1: Eat yourself healthy with these amazing superfoods: <http://cnn.it/1Gvs95n> #CNNAfrica http://pbs.twimg.com/media/B_-S1cnuMAEuIFv.jpg
 [RT @CNNMoney: This computer can diagnose cancer. Watch: <http://cnnmon.ie/1GKz7qP> CC @enlitic @jeremyphoward By @jillianeugenios <https://t.co/V...>
 [RT @CNN: .@RobertDowneyJr presented a child with his own 'Iron Man' robotic arm: <http://cnn.it/1CbnIPC> via @cnnireport
 [RT @TIME: The hidden dangers of being 'skinny fat' <http://ti.me/1Gvz26A>
 [Kids = constant ear infections. Save yourself a trip to the ER w/ this: <http://cnn.it/1xg6t8x> #cnninstantstartups

Figura 6.1 – Exemplos do formato e do teor de tweets.

Um desafio que se põe, dessa forma, é encontrar meios de analisar esse tipo de dado e obter informações que possam direcionar a atenção dos responsáveis pela área de saúde, seja na seara da prevenção, do diagnóstico ou até mesmo da pesquisa.

No intuito de contribuir com tal tarefa, foram acessados dados do sítio da *UCI Machine Learning Repository* - - *Health News in Twitter Data Set* - em janeiro de 2020, que consistem em 16 arquivos texto, cada um relacionado a uma conta Twitter de uma agência de notícias, totalizando 63.326 mensagens relativas ao período de 2011 a 2015.

O objetivo do projeto é, portanto, analisar uma coleção de *tweets* de agências de notícias, postados no período de 2011 a 2015, no intuito de descobrir novos padrões nos dados e identificar temas ou tópicos de interesse ligados à saúde, que possam ser utilizados por especialistas da área no direcionamento de suas atividades.

Para tanto, os dados foram pré-processados por meio de aplicação dos modelos *Bag of Words* e TF-IDF e, em seguida, agrupados em 10 clusters pelo modelo K-Means de aprendizado de máquina não supervisionado. Conforme se verifica na Figura 6.2, o grupo com maior número de *tweets* foi o 0, seguido pelos clusters 3, 7 e 9. Um aspecto importante consiste na diferença bastante significativa entre a quantidade de *tweets* agrupados no cluster 0 e nos demais.

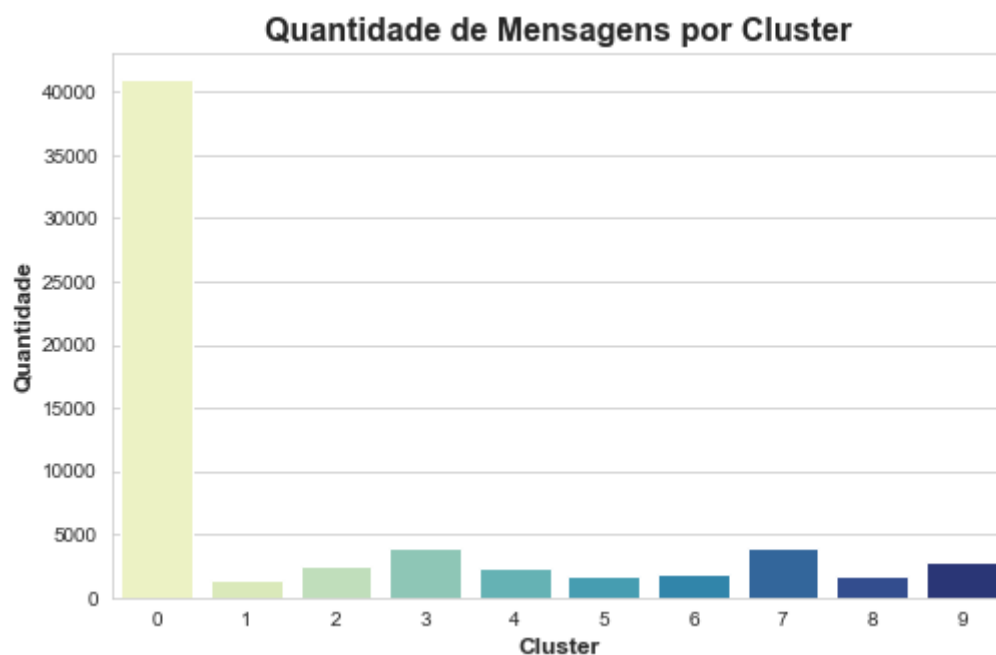


Figura 6.2 – Quantidade de tweets por cluster (KMeans).

Ainda com relação à quantidade de mensagens por cluster, um ponto a se destacar é a inexistência de *tweets* nos anos anteriores a 2014 relativamente ao cluster 3, conforme se observa na Figura 6.3. Já o cluster 1 parece ser bastante específico, visto que apenas apresenta *tweets* nos anos de 2013 e 2014.

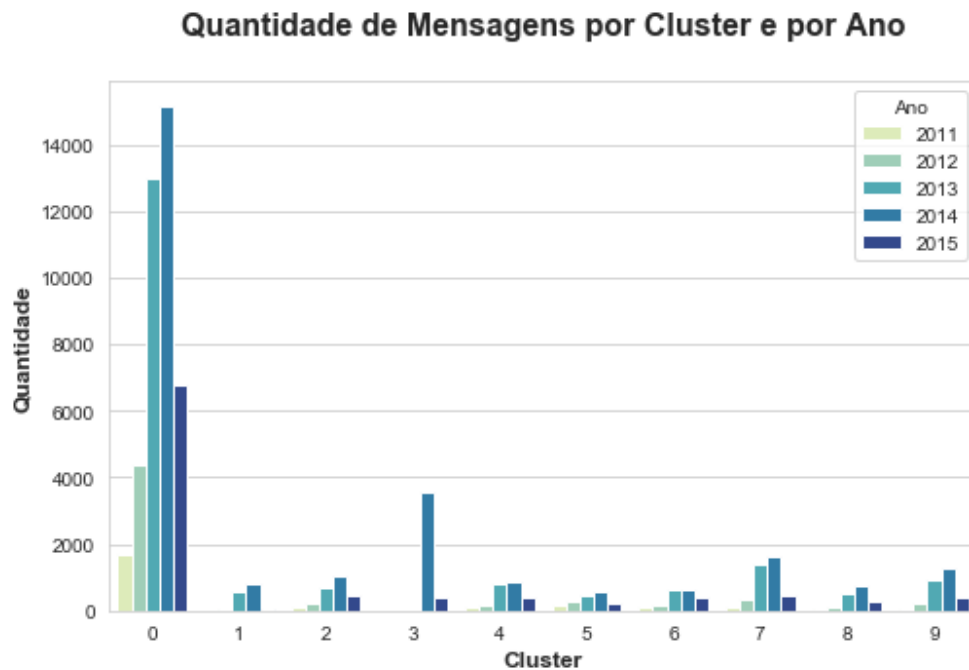


Figura 6.3 – Quantidade de tweets por cluster e por ano (KMeans).

No tocante aos gráficos de quantidades, um último aspecto a se considerar diz respeito à distribuição das mensagens por dia da semana. Da análise do gráfico apresentado na Figura 6.4, verifica-se que os *tweets* relativos a assuntos de saúde são mais frequentes nos dias úteis, cabendo destacar que, no que tange ao cluster 1, praticamente inexistem *tweets* em finais de semana.

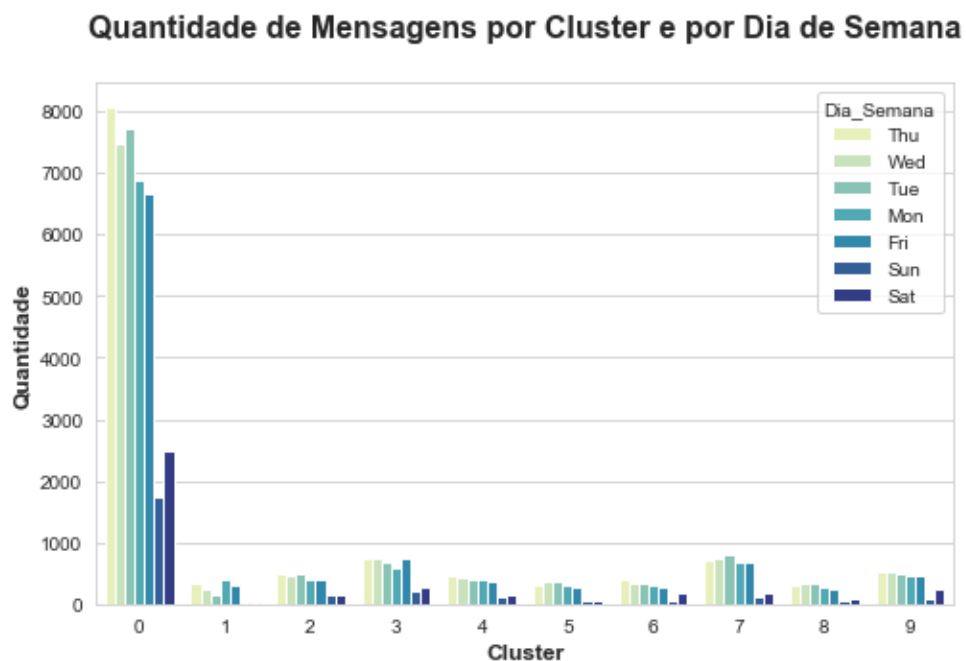


Figura 6.4 – Quantidade de tweets por cluster e por dia da semana (KMeans).

Os termos mais frequentes registrados em cada cluster estão apresentadas na Figura 6.5.

Termos mais frequentes por cluster:

Cluster 0:
may cancer say studi drug food make doctor eat amp

Cluster 1:
well ask doctor cancer live exercis think may like new

Cluster 2:
help video may weight could cancer lose food studi patient

Cluster 3:
ebola liberia s africa outbreak patient sierra case say leon

Cluster 4:
get way health help fit here new kid like free

Cluster 5:
risk cancer may heart rais increas studi higher lower link

Cluster 6:
like work look opinion sell still smoothli november kidney end

Cluster 7:
health law insur mental exchang plan say state may today

Cluster 8:
care health new afford act hospit patient social cost home

Cluster 9:
new old blog age york studi drug year find pay

Figura 6.5 – Tags mais frequentes de cada cluster.

Sob o enfoque dos termos mais frequentes, destacam-se os gráficos relativos aos *clusters* com maior quantidade de *tweets*: os de rótulo 0, 3, 7 e 9 (Figuras 6.6 a 6.9).

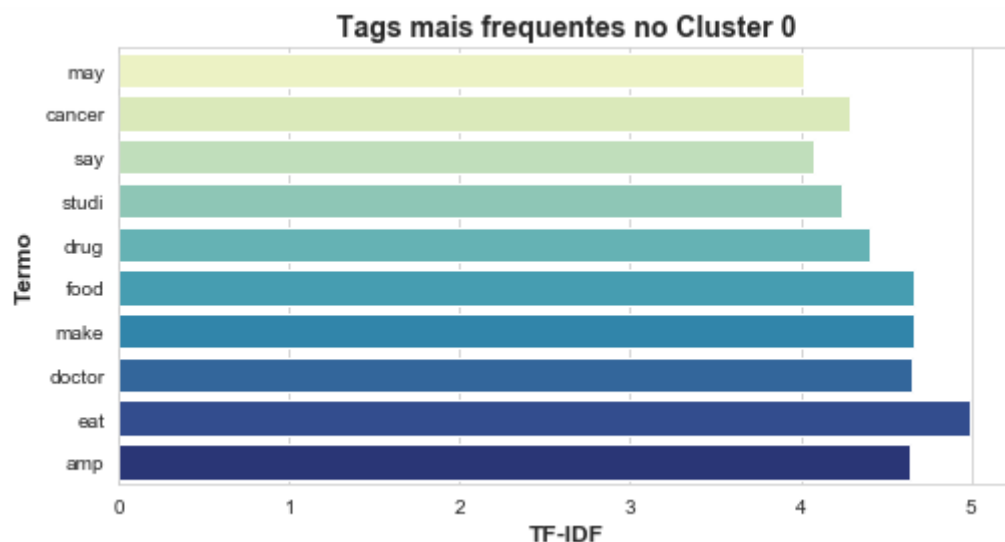


Figura 6.6 – Tags mais frequentes no Cluster 0.

O gráfico de figura 6.6 evidencia as dez *tags* mais frequentes no cluster 0. Observa-se que o termo com maior peso TF-IDF é *eat*, e que três das *tags* seguintes com maior TF-IDF são: *food*, *make*, *doctor*. Verifica-se, também a presença de *tags* de importante significado semântico, que são *cancer*, *studi* e *drug*. A análise superficial desses termos sugere a conclusão de que muito se noticia a respeito de pesquisas sobre efeito de alimentos na saúde e, principalmente, na aquisição do câncer. Porém, a simples análise dessas *tags* não permite interpretar com segurança o critério do agrupamento porque a *tag cancer* também aparece como uma das palavras mais frequentes nos clusters 1, 2 e 5.

Por outro lado, causa certa estranheza a quantidade de *tweets* tão superior aos demais grupos, o que também pode sugerir a existência de outro critério de agrupamento, não identificável por leigos na área.

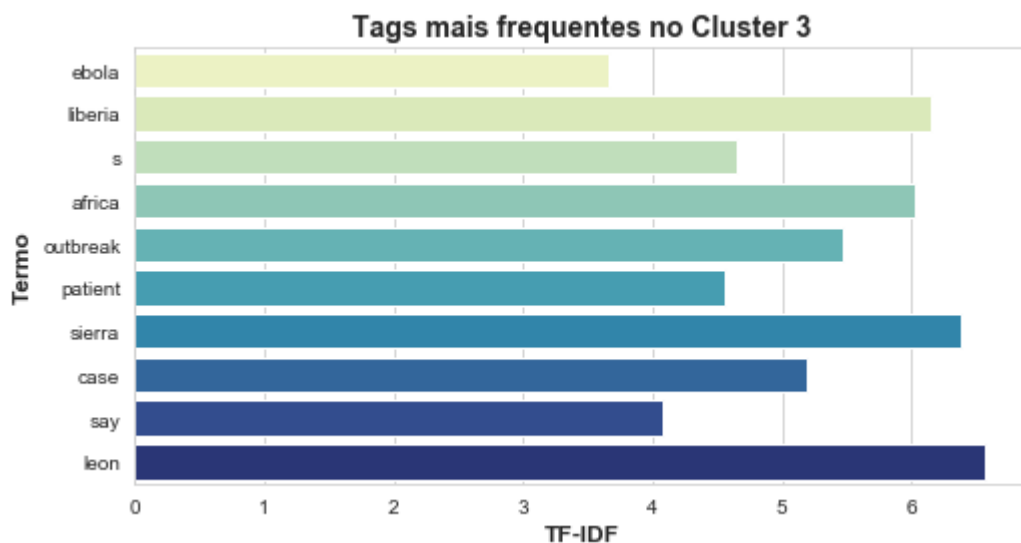


Figura 6.7 – Tags mais frequentes no Cluster 3.

O cluster 3, segundo grupo em quantidade de *tweets*, apresentou as seguintes *tags* como mais frequentes: *ebola*, *liberia*, *s*, *africa*, *outbreak*, *patient*, *sierra*, *case*, *say*, *leon*. A Figura 6.7 torna bem claro que os termos com maior peso TF-IDF têm características geográficas: *leon*, *sierra*, *liberia*, e *africa*. Além dessas *tags*, a presença dos termos *ebola* e *outbreak* deixa bem claro critério adotado nesse agrupamento.

A Figura 6.3, por seu turno, indica que os *posts* se concentraram nos anos de 2014 e 2015. De fato, o tema sugerido pelas palavras mais frequentes – relacionados ao ebola - esteve mais fortemente presente nesse período. Observe-se que, segundo a Organização Mundial de Saúde, o pico da transmissão ocorreu em agosto e setembro de 2014 e, em maio de 2015, a Libéria estava livre da transmissão do vírus Ebola [18].

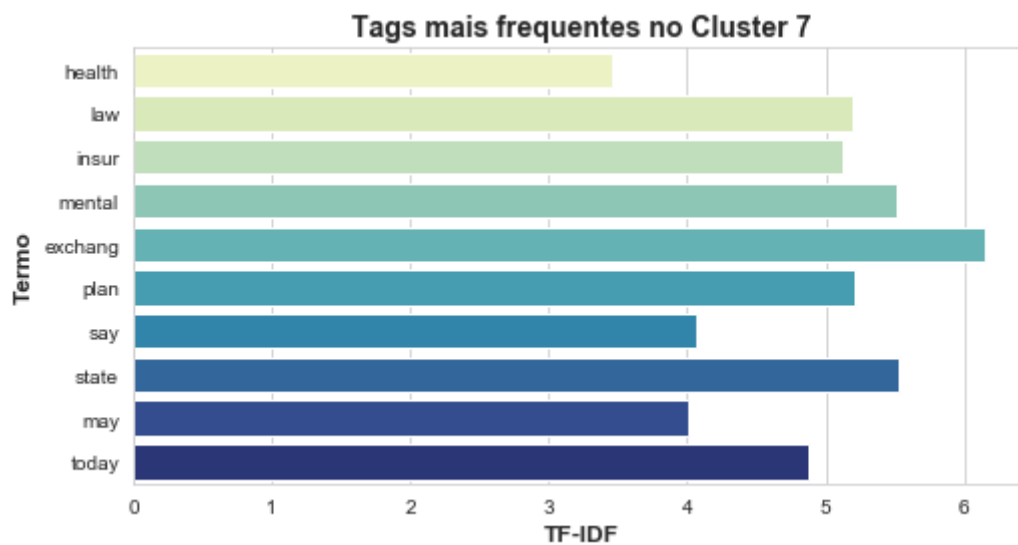


Figura 6.8 – Tags mais frequentes no Cluster 7.

O grupo seguinte, em termos de quantidade de *tweets*, foi o de rótulo 7, com as seguintes *tags* mais frequentes: *health*, *law*, *insur*, *mental*, *exchang*, *plan*, *say*, *state*, *may*, *today*. O tópico correspondente parece girar em torno de discussões referentes a custos da saúde, abarcando questões relativas a planos de saúde, leis, seguros e similares. Um ponto interessante, no entanto, é o alto peso TF-IDF da *tag mental*, que parece sugerir uma abordagem mais específica, o que pode ser verificado na Figura 6.8.

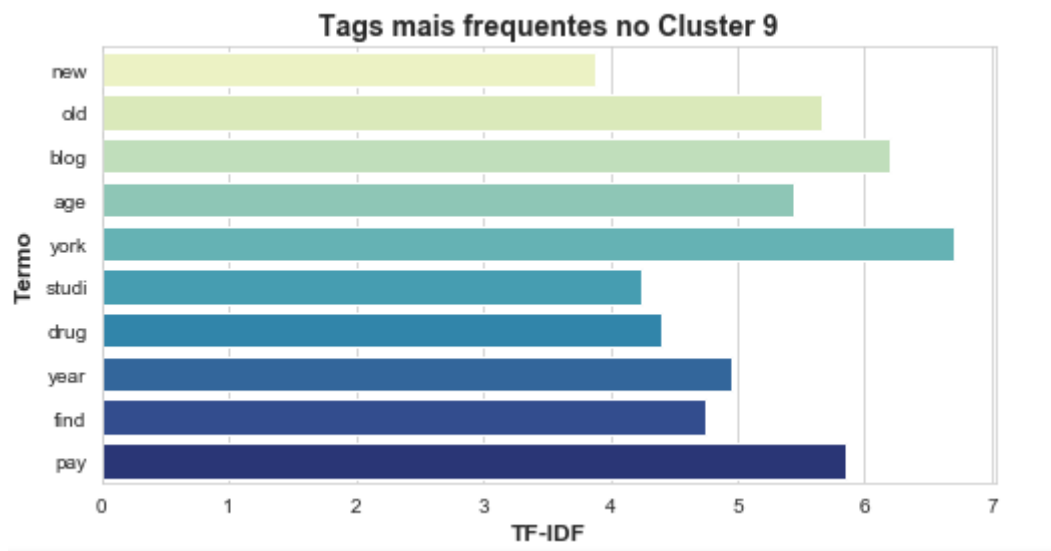


Figura 6.9 – Tags mais frequentes no Cluster 9.

Por fim, o cluster 9, quarto lugar em termos de expressividade em sua quantidade, apresentou as seguintes *tags* como mais frequentes: *new*, *old*, *blog*, *age*, *york*, *studi*, *drug*, *year*, *find*, e *pay* (Figura 6.9). Tais *tags* sugerem o possível interesse em notícias relacionadas a estudos e descobertas referentes a saúde.

Sob outra perspectiva, as de nuvens de palavras proporcionam uma visualização mais direta dos termos considerados de maior relevância, conforme o peso obtido no modelo TF-IDF. Para isso, foram elaboradas as seguintes nuvens de *tags*: a) nuvem gerada a partir dos radicais dos tokens encontrados nos *tweets* (Figura 6.10); b) nuvem gerada a partir dos radicais dos tokens encontrados nos *posts* com vídeos (Figura 6.11); e c) nuvem gerada a partir das *hashtags* identificadas nos *tweets* (Figura 6.12).



Figura 6.11 – Nuvem de tags para dados de tweets com vídeos.

Algumas das *tags* já referidas se repetiram nos *tweets* com *posts* de vídeos, como *alzheimers*, *abort* e *addict* (Figura 6.11). Outras, como *alcohol*, *anorexia*, *anxiety* e *africa* – esta última provavelmente ligada à questão da doença Ebola - foram mais frequentes nesse tipo de mensagem. Um ponto a se destacar é que prevalecem nos *tweets* com vídeos mensagens com viés negativo: além das doenças mencionadas, aparecem também as *tags* *death*, *crisis*, *risk*, em contraposição apenas a *live*, mais positiva.

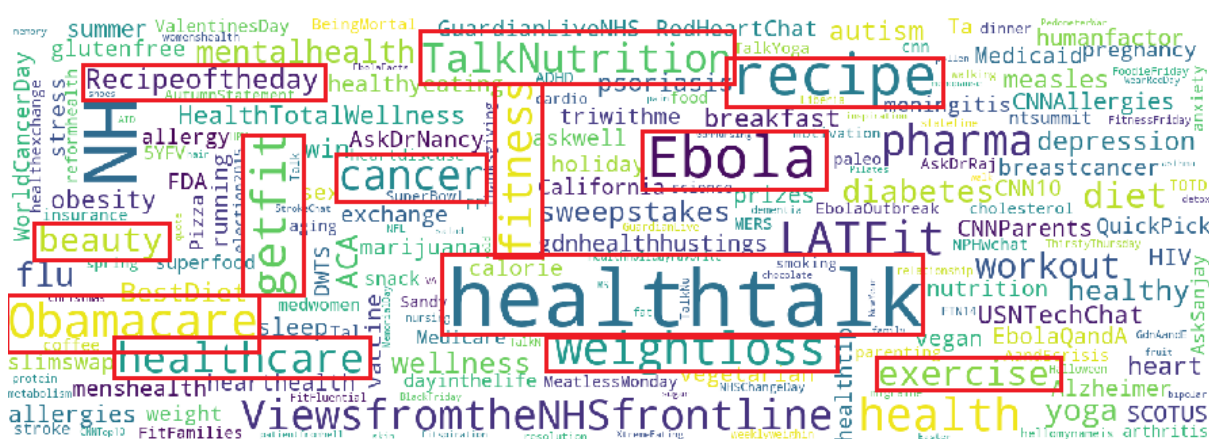


Figura 6.12 – Nuvem de palavras para hashtags encontradas nos tweets.

Já as *hashtags* observadas nos *tweets* revelam a presença de mensagens com viés mais positivo, relacionadas a receitas, nutrição e perda de peso. Relativamente a doenças, câncer e ebola aparecem com maior frequência. E uma outra questão que se põe é a discussão sobre planos de saúde, revelada pela presença das *tags* *Obamacare* e *healthcare*.

Do exposto, conclui-se que a aplicação dos algoritmos de *machine learning* e a visualização por nuvem de palavras permitiram a identificação de alguns temas cujos *tweets* podem ser explorados de maneira mais detalhada por especialistas da área de saúde. O consumo de amêndoas, doenças como Alzheimer, Ebola, câncer e anorexia, bem como questões ligadas a ansiedade, vício e planos de saúde destacam-se entre os demais tópicos. Sob outro prisma, as *hashtags* se mostraram importante instrumento de incentivo ao cuidado da saúde, podendo ser utilizada como ferramenta de prevenção de forma geral.

É importante registrar que esses resultados ainda podem ser aprimorados pela otimização na escolha do número *k* de clusters utilizado no modelo K-Means de agrupamento, bem como pelo aperfeiçoamento da função de pré-processamento dos textos de *tweets* mediante a inclusão de tratamento de sintaxe e morfologia. A necessidade de redução da dimensionalidade da matriz esparsa gerada pelo modelo TF-IDF também deve ser enfrentada.

Quanto aos dados, a divergência entre o número de instâncias informado pela UCI e o número de entradas computado neste projeto ainda merece mais investigação.

Por fim, como o presente projeto tratou de algoritmo de aprendizado de máquina não supervisionado, em que a interpretação dos resultados é dificultada pela falta de transparência nos critérios de agrupamento utilizados pelo algoritmo, é relevante lembrar que a validação de seus resultados por especialistas da área é de fundamental importância.

7. Links

Vídeo de apresentação do projeto:

https://www.dropbox.com/s/yjc878nf77wbzzc/Saude_em_Tweets_Tarefa_Clusteriza%C3%A7%C3%A3o_Apresenta%C3%A7%C3%A3o_video.mp4?dl=0

Scripts criados em Python:

https://www.dropbox.com/s/kdjzwjev1bmp1hj/Saude_em_Tweets_Tarefa_Clusteriza%C3%A7%C3%A3o.ipynb?dl=0

Repositório dos dados utilizados no projeto:

<https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter>

REFERÊNCIAS

- [1] GREGO, Maurício. *Conteúdo digital dobra a cada dois anos no mundo*. 2014. Disponível em: <<https://exame.abril.com.br/tecnologia/conteudo-digital-dobra-a-cada-dois-anos-no-mundo/>>. Acesso em: 08/03/2020.
- [2] KARAMI, A., Gangopadhyay, A., ZHOU, B., & KHARRAZI, H. *Fuzzy approach topic discovery in health and medical corpora*. *International Journal of Fuzzy Systems*, 1-12., 2017. Disponível em: <<https://archive.ics.uci.edu/ml/datasets/Health+News+in+Twitter>>. Acesso em: 29/01/2020.
- [3] PYTHON. Disponível em: <<https://www.python.org/>>.
- [4] JUPYTER. Disponível em: <<https://jupyter.org/>>.
- [5] SEABORN. Disponível em: <<https://seaborn.pydata.org/>>.
- [6] STACK OVERFLOW. Disponível em: <<https://stackoverflow.com/questions/46000191/utf-8-codec-cant-decode-byte-0x92-in-position-18-invalid-start-byte>>. Acesso em: 11/02/2020.
- [7] STACK OVERFLOW. Disponível em: <<https://stackoverflow.com/questions/45037907/python-astypestr-gives-settingwithcopywarning-and-requests-i-use-loc>>.
- [8] WIKIPEDIA. *Aprendizado de Máquina*. Disponível em: <https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina>. Acesso em: 10/03/2020.
- [9] NASSIF, Luís Filipe da Cruz. *Técnicas de Agrupamento de Textos Aplicadas à Computação Forense*. Dissertação de Mestrado – Universidade de Brasília - UnB. 2011, p. 15.
- [10] BIRD, Steven, Edward Loper e Ewan Klein, *Natural Language Processing with Python*. O'Reilly Media Inc., 2009.
- [11] GOMES, M. Lucas. *Clusterização de texto de reclamação não supervisionada usando K-means com python*. Disponível em: < https://lamfo-unb.github.io/2019/09/02/cluster_texto/>. Acesso em: 17/02/2020.

- [12] INCA – Instituto Nacional de Câncer. Ministério da Saúde. *Duas décadas de Dia Mundial do Câncer e “Estimativa 2020” marcam o 4 de Fevereiro no INCA*. Disponível em: < <https://www.inca.gov.br/noticias/duas-decadas-de-dia-mundial-do-cancer-e-estimativa-2020-marcam-o-4-de-fevereiro-no-inca> >. Acesso em: 11/03/2020.
- [13] _op. Cit. NASSIF, 2011, p. 11 e 12.
- [14] SCIKIT LEARN. *Sklearn.decomposition.TruncatedSVD*. Disponível em: < <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html> > . Acesso em 10/03/2020.
- [15] SAWANT, Mrunal. *Truncated Singular Value Decomposition (SVD) using Amazon Food Review*. Disponível em: <<https://medium.com/swlh/truncated-singular-value-decomposition-svd-using-amazon-food-reviews-891d97af5d8d>>. Acesso em: 05/03/2020.
- [16] SANTANA, Felipe. *Algoritmo K-means: Aprenda essa Técnica Essencial através de Exemplos Passo a Passo com Python*. Disponível em: <<https://minerandodados.com.br/algoritmo-k-means-python-passo-passo/>>. Acesso em 04/03/2020.
- [17] WIKIPEDIA. *Silhueta (agrupamento)*. Disponível em: <[https://en.wikipedia.org/wiki/Silhouette_\(clustering\)](https://en.wikipedia.org/wiki/Silhouette_(clustering))>. Acesso em: 11/03/2020.
- [18] WORLD HEALTH ORGANIZATION. The Ebola outbreak in Liberia is over. Disponível em: <<https://www.afro.who.int/news/ebola-outbreak-liberia-over>>. Acesso em: 02/03/2020.