projeto2

December 3, 2020

[1]:
```
import pyspark
import math
import pandas as pd
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
import matplotlib.pyplot as plt
import time
```

[2]:
```
def conta_palavras(item):
    url, conteudo = item
    palavras = conteudo.strip().split()
    return [(palavra.lower(), 1) for palavra in set(palavras)]

def junta_contagens(nova_contagem, contagem_atual):
    return nova_contagem + contagem_atual

def computa_relevancia(item):
    palavra, contagem = item
    return (palavra, math.log10(1 + contagem))

def cria_data_frame(data = []):
    palavras = [item[0] for item in data]
    relevancias = [item[1] for item in data]

    dataFrameData = {
        'PALAVRA': palavras,
        'RELEVANCIA': relevancias
    }

    df = pd.DataFrame(data=dataFrameData, columns=['PALAVRA', 'RELEVANCIA'])
    pd.set_option('display.max_rows', df.shape[0]+1)
    return df


def read_stopwords(fileName):
    stopwords = set(STOPWORDS)

    stopwords_pt = []
```

```
    with open(fileName, "r") as file:
        for line in file.readlines():
            stopwords_pt.append(line.strip().replace('\n', ''))
    stopwords.update(stopwords_pt)

    return stopwords

def save_word_cloud(word_cloud, fileName):
    word_cloud.to_file(fileName)

def plot_word_cloud(word_cloud):
    fig, ax = plt.subplots(figsize=(16,8))
    ax.imshow(word_cloud, interpolation='bilinear')
    ax.set_axis_off()
    plt.imshow(word_cloud)

def create_word_cloud(data, fileName, stopwords, size = (1600, 800),␣
↪background_color = 'black'):
    words = ' '.join(data)
    width, height = size
    word_cloud = WordCloud(
        stopwords=stopwords,
        background_color=background_color,
        width=width,
        height=height
    ).generate(words)

    save_word_cloud(word_cloud, fileName)
    plot_word_cloud(word_cloud)

    return word_cloud
```

[3]:
```
sc = pyspark.SparkContext(appName="CalculaRelevancia")
sc
```

[3]:
```
<SparkContext master=local[*] appName=CalculaRelevancia>
```

[4]:
```
rdd =  sc.sequenceFile('part-00000')
```

[5]:
```
palavra1 = 'intel'
palavra2 = 'amd'
stopwords = read_stopwords('stopwords_pt.txt')
```

[6]:
```
# Filtra por paginas que possuem a palavra1 e não possuem a palavra2
def filtra_palavra1(item):
    url, conteudo = item
    conteudo = conteudo.strip().split()
```

```
    return palavra1 in conteudo and palavra2 not in conteudo

# Filtra por paginas que possuem a palavra2 e não possuem a palavra1
def filtra_palavra2(item):
    url, conteudo = item
    conteudo = conteudo.strip().split()
    return palavra2 in conteudo and palavra1 not in conteudo

# Filtra por paginas que possuem a palavra1 e a palavra2
def filtra_palavras_juntas(item):
    url, conteudo = item
    conteudo = conteudo.strip().split()
    return palavra1 in conteudo and palavra2 in conteudo
```

[7]:
```
rdd_palavra1 = rdd.filter(filtra_palavra1)
rdd_palavra2 = rdd.filter(filtra_palavra2)
rdd_palavras = rdd.filter(filtra_palavras_juntas)
```

[8]:
```
DOC_COUNT_MIN = 3
DOC_COUNT_MAX = 0.7
N = rdd.count()
def palavras_mais_relevantes(_rdd, n):
    tempo_inicial = time.time()

    # Função para remover palavras muico comuns como 'a', 'o', 'seus', etc...
    def _filtra_stopword(palavra):
        return (len(palavra) and palavra.isalpha() and (palavra not in␣
 ↪stopwords))

    # Função para remover palavras com contagem menor que 5 e menor que 70% do␣
 ↪numero total de paginas
    def _filtra_contagem(contagem):
        return (contagem >= DOC_COUNT_MIN and (contagem < DOC_COUNT_MAX * N))

    # Função para agregar os filtros relacionados ao item
    def filtra_item(item):
        palavra, contagem = item
        return (_filtra_stopword(palavra) and _filtra_contagem(contagem))


    # Conta quantas vezes cada palavra aparece agr o item passa a ser (palavra,␣
 ↪contagem)
    rdd_contagens = _rdd.flatMap(conta_palavras).reduceByKey(junta_contagens)

    # Faz a filtragem dos itens pela contagem e pela lista de palavras␣
 ↪proibidas (stopwords)
    rdd_contagens_filtrado = rdd_contagens.filter(filtra_item)
```

```
    # Calcula a Relevância da palavra usando a contagem
    rdd_relevancia = rdd_contagens_filtrado.map(computa_relevancia)

    # Funções para ordernar os n primeiros items em ordem decrescente
    ordena_freq = lambda item: -item[1]
    top_n = rdd_relevancia.takeOrdered(n, ordena_freq)

    tempo_final = time.time()

    print(f'Tempo: {tempo_final-tempo_inicial}')
    return top_n



top100_palavra1 = palavras_mais_relevantes(rdd_palavra1, 100)
top100_palavra2 = palavras_mais_relevantes(rdd_palavra2, 100)
top100_palavras = palavras_mais_relevantes(rdd_palavras, 100)
```

```
Tempo: 5.036628723144531
Tempo: 4.206704139709473
Tempo: 4.114330053329468
```

[9]:
```
palavra1_df = cria_data_frame(top100_palavra1)
palavra2_df = cria_data_frame(top100_palavra2)
palavras_df = cria_data_frame(top100_palavras)
```

0.1 Relevância de palavras para as paginas que contem a palavra1

[10]:
```
palavra1_df
```

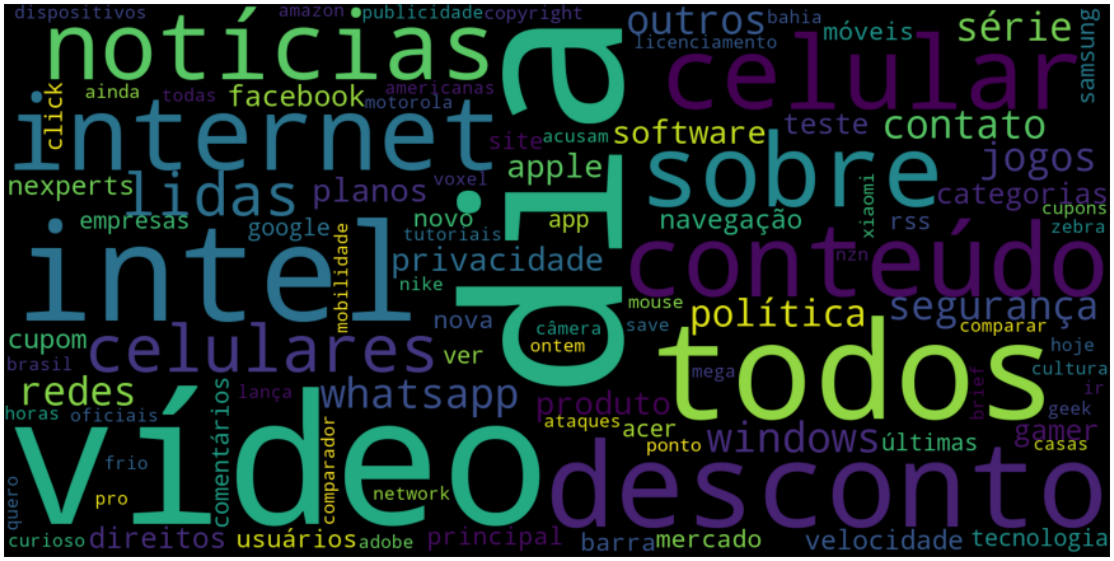
[10]:

	PALAVRA	RELEVANCIA
0	todos	1.716003
1	intel	1.672098
2	desconto	1.591065
3	internet	1.568202
4	conteúdo	1.544068
..	…	…
95	quero	1.204120
96	zebra	1.204120
97	ontem	1.204120
98	nzn	1.204120
99	cultura	1.204120

[100 rows x 2 columns]

[11]:
```
create_word_cloud(palavra1_df['PALAVRA'], 'palavra1.png', stopwords)
```

[11]: <wordcloud.wordcloud.WordCloud at 0x7f688ec528d0>



0.2 Relevância de palavras para as paginas que contem a palavra2

[12]:
```
palavra2_df
```

[12]:

	PALAVRA	RELEVANCIA
0	novo	1.000000
1	series	0.954243
2	us	0.954243
3	youtube	0.954243
4	contato	0.954243
..	…	…
95	vai	0.778151
96	pesquisa	0.778151
97	parabéns	0.778151
98	ribeiro	0.778151
99	novos	0.778151

[100 rows x 2 columns]

[13]:
```
create_word_cloud(palavra2_df['PALAVRA'], 'palavra2.png', stopwords)
```

[13]: <wordcloud.wordcloud.WordCloud at 0x7f688e8cb5c0>

0.3 Relevância de palavras para as paginas que contem a palavra1 e a palavra2

[14]:
```
palavras_df
```

[14]:

	PALAVRA	RELEVANCIA
0	iphone	0.778151
1	samsung	0.778151
2	todos	0.778151
3	dell	0.778151
4	todas	0.698970
5	sony	0.698970
6	hd	0.698970
7	jogos	0.698970
8	ipad	0.698970
9	ram	0.698970
10	amd	0.698970
11	smartphone	0.698970
12	whatsapp	0.698970
13	motorola	0.698970
14	intel	0.698970
15	produto	0.698970
16	pro	0.698970
17	tela	0.602060
18	melhor	0.602060
19	precisa	0.602060
20	micro	0.602060
21	led	0.602060

22	tempo	0.602060
23	note	0.602060
24	leitor	0.602060
25	moto	0.602060
26	apple	0.602060
27	entrar	0.602060
28	u	0.602060
29	core	0.602060
30	smartphones	0.602060
31	hp	0.602060
32	ajuda	0.602060
33	flex	0.602060
34	quero	0.602060
35	celulares	0.602060
36	top	0.602060
37	entrega	0.602060
38	f	0.602060
39	condições	0.602060
40	original	0.602060
41	pode	0.602060
42	snapdragon	0.602060
43	tudo	0.602060
44	acer	0.602060
45	monitor	0.602060
46	geração	0.602060
47	câmeras	0.602060
48	vai	0.602060
49	aparelho	0.602060
50	apenas	0.602060
51	hoje	0.602060
52	ultra	0.602060
53	vista	0.602060
54	x	0.602060
55	tecnologia	0.602060
56	fabricantes	0.602060
57	lg	0.602060
58	bateria	0.602060
59	vc	0.602060
60	computador	0.602060
61	evo	0.602060
62	games	0.602060
63	agora	0.602060
64	celular	0.602060
65	quase	0.602060
66	pc	0.602060
67	memória	0.602060
68	xperia	0.602060

```
69              g    0.602060
70          então    0.602060
71        próximo    0.602060
72            tim    0.602060
73             sd    0.602060
74           todo    0.602060
```

[15]:
```
create_word_cloud(palavras_df['PALAVRA'], 'palavras.png', stopwords)
```

[15]:
```
<wordcloud.wordcloud.WordCloud at 0x7f688e042e80>
```

[16]:
```
sc.stop()
```