

# MINERAÇÃO DE TEXTOS (Parte 2) - WORD PAIRS

## Livro: The Amateur Emigrant de Robert Louis Stevenson

Usando o algoritmo de WordPairs, do pacote NETWORKX, para descobrir as associações entre as palavras do texto.

### Instalando o pacote 'NETWORKX' e as bibliotecas necessárias

```
In [1]: pip install networkx
```

Requirement already satisfied: networkx in c:\users\mmateus\anaconda3\lib\site-packages (2.6.3)  
Note: you may need to restart the kernel to use updated packages.

```
In [2]: import pandas as pd
import matplotlib.pyplot as plt
import re
import nltk
import networkx as nx

from nltk import ngrams
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.corpus import opinion_lexicon

nltk.download('stopwords', quiet=True)
nltk.download('opinion_lexicon', quiet=True)
nltk.download('punkt', quiet=True)
```

```
Out[2]: True
```

### Importação do texto:

```
In [3]: df = pd.read_csv('data/TheAmateurEmigrant.txt', sep='\t')\
        .dropna()

df.head(10)
```

```
Out[3]:
```

	text
0	THE AMATEUR EMIGRANT
1	THE SECOND CABIN
2	I first encountered my fellow-passengers on th...
3	Thence we descended the Clyde in no familiar s...
4	on each other as on possible enemies. A few S...
5	already grown acquainted on the North Sea, wer...
6	their long pipes; but among English speakers d...

	text
7	reigned supreme. The sun was soon overclouded...
8	grew sharp as we continued to descend the wide...
9	falling temperature the gloom among the passen...

## Preparação do texto

```
In [4]: def clean_text(text):
        text = text.lower()
        text = text.replace("'", '')
        text = re.sub(r'^\w', ' ', text)
        text = re.sub(r'\s+', ' ', text)
        text = text.strip()
        return text

df['text'] = df['text'].map(clean_text)
df['text'] = df['text'].map(word_tokenize)
df.head()
```

```
Out[4]:
```

	text
0	[the, amateur, emigrant]
1	[the, second, cabin]
2	[i, first, encountered, my, fellow, passengers...
3	[thence, we, descended, the, clyde, in, no, fa...
4	[on, each, other, as, on, possible, enemies, a...

## Criando tokens de pares de palavras (WORD PAIRS)

```
In [5]: df['wordpairs'] = df['text'].map(lambda x: list(ngrams(x, 2)))
df = df.explode('wordpairs')

df.head(10)
```

```
Out[5]:
```

	text	wordpairs
0	[the, amateur, emigrant]	(the, amateur)
0	[the, amateur, emigrant]	(amateur, emigrant)
1	[the, second, cabin]	(the, second)
1	[the, second, cabin]	(second, cabin)
2	[i, first, encountered, my, fellow, passengers...	(i, first)
2	[i, first, encountered, my, fellow, passengers...	(first, encountered)
2	[i, first, encountered, my, fellow, passengers...	(encountered, my)
2	[i, first, encountered, my, fellow, passengers...	(my, fellow)
2	[i, first, encountered, my, fellow, passengers...	(fellow, passengers)
2	[i, first, encountered, my, fellow, passengers...	(passengers, on)

## Classificando os tokens por frequência

```
In [6]: df['wordpairs'].value_counts().head(10)
```

```
Out[6]: (of, the)      181
        (in, the)     150
        (to, the)      84
        (he, was)       81
        (and, the)      80
        (it, was)       72
        (on, the)       63
        (he, had)       63
        (of, a)         62
        (with, a)       53
        Name: wordpairs, dtype: int64
```

## Separando as Word Pairs para remover as Stop Words

```
In [7]: df = pd.DataFrame(df.wordpairs.values.tolist(), columns=['word1', 'word2']).dropna()
        df.head(10)
```

```
Out[7]:
```

	word1	word2
0	the	amateur
1	amateur	emigrant
2	the	second
3	second	cabin
4	i	first
5	first	encountered
6	encountered	my
7	my	fellow
8	fellow	passengers
9	passengers	on

```
In [8]: df.shape
```

```
Out[8]: (24030, 2)
```

## Removing Stop Words

```
In [9]: en_stopwords = set(stopwords.words('english'))

        df = df[~(df.word1.isin(en_stopwords) | df.word2.isin(en_stopwords))]

        df.head()
```

```
Out[9]:
```

	word1	word2
--	-------	-------

	<b>word1</b>	<b>word2</b>
<b>1</b>	amateur	emigrant
<b>3</b>	second	cabin
<b>5</b>	first	encountered
<b>8</b>	fellow	passengers
<b>21</b>	familiar	spirit

In [10]: `df.shape`

Out[10]: (4005, 2)

Com a remoção das Stop Words, o texto foi reduzido em 84% de palavras, mantendo apenas as que são relevantes.

## Classificando as Word Pairs por frequência

```
In [11]: df = df.groupby(['word1', 'word2'])\
        .size()\
        .to_frame('n')\
        .reset_index()\
        .sort_values('n', ascending=False)

df.head(20)
```

Out[11]:

	<b>word1</b>	<b>word2</b>	<b>n</b>
<b>2872</b>	second	cabin	18
<b>2207</b>	new	york	12
<b>1035</b>	fellow	passengers	11
<b>2970</b>	sick	man	6
<b>2213</b>	next	morning	5
<b>657</b>	dare	say	5
<b>611</b>	could	see	5
<b>1034</b>	fellow	passenger	4
<b>2106</b>	mr	jones	4
<b>2321</b>	one	day	4
<b>1513</b>	hurricane	deck	4
<b>3166</b>	steerage	passenger	4
<b>317</b>	brass	plate	4
<b>3681</b>	working	man	4
<b>2337</b>	one	night	4
<b>841</b>	emigrant	ship	3
<b>3717</b>	would	say	3

	word1	word2	n
<b>3589</b>	west	street	3
<b>1837</b>	long	ago	3
<b>335</b>	broken	meat	3

## Visualizando os dados através de uma tabela.

- Restrição das word pairs para apenas aquelas que aparecem mais de 3 vezes.

In [12]: `df[df.n > 3].head(10)`

Out[12]:

	word1	word2	n
<b>2872</b>	second	cabin	18
<b>2207</b>	new	york	12
<b>1035</b>	fellow	passengers	11
<b>2970</b>	sick	man	6
<b>2213</b>	next	morning	5
<b>657</b>	dare	say	5
<b>611</b>	could	see	5
<b>1034</b>	fellow	passenger	4
<b>2106</b>	mr	jones	4
<b>2321</b>	one	day	4

## Visualizando o gráfico de conexões entre os pares de palavras

In [13]:

```
G = nx.from_pandas_edgelist(df[df.n > 3], 'word1', 'word2')
plt.figure(figsize=(12, 10))
nx.draw_shell(G, with_labels=True, node_color='white', font_size=15)
```

