MINERAÇÃO DE TEXTOS

Parte 2 - WORD PAIRS

Instalação dos pacotes necessários:

Para trabalhar com mineração de texto, alguns pacotes precisam ser instalados no computador: o **NLTK** (Natural Language Toolkit) e, nesta etapa, o **NETWORKX** (para criação dos pares de palavras). Os comandos são

- pip install nltk
- pip install networkx

Obs: eles só precisaram ser instalados uma única vez por máquina. Uma vez feito isso, pode-se importar as bibliotecas necessárias.

Importação de bibliotecas:

In [1]:
```
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[1]:
```
True
```

Importação do texto "The Amateur Emigrant":

In [2]:
```
df = pd.read_csv('data/TheAmateurEmigrant.txt',sep='\t')\
    .dropna()

df.head(6)
```

Out[2]:

	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...

Preparação do texto (limpeza e tokenização):

In [3]:
```
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[3]:

	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:

In [4]:
```
df['wordpairs'] = df['text'].map(lambda x: list(ngrams(x, 2)))
df = df.explode('wordpairs')

df.head(10)
```

Out[4]:

	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 [5]:
```
df['wordpairs'].value_counts().head(6)
```

Out[5]:
```
(of, the)     181
(in, the)     150
```

```
(to, the)      84
(he, was)      81
(and, the)     80
(it, was)      72
Name: wordpairs, dtype: int64
```

Separando as Word Pairs para remover as Stop Words:

In [6]:
```
df = pd.DataFrame(df.wordpairs.values.tolist(), columns=['word1', 'word2']).dropna()

df.head(8)
```

Out[6]:

	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

In [7]:
```
df.shape
```

Out[7]:
```
(24030, 2)
```

In [8]:
```
en_stopwords = set(stopwords.words('english'))

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

df.head(8)
```

Out[8]:

	word1	word2
1	amateur	emigrant
3	second	cabin
5	first	encountered
8	fellow	passengers
21	familiar	spirit
24	looking	askance
30	possible	enemies
36	already	grown

In [9]:
```
df.shape
```

Out[9]:
```
(4005, 2)
```

Nota: Com a remoção das Stop Words, o texto foi reduzido em 83% das palavras (de 24030 para 4005), mantendo apenas as relevantes para o entendimento da história.

Classificando as Word Pairs por frequência:

In [10]:
```
df = df.groupby(['word1', 'word2'])\
    .size()\
    .to_frame('n')\
    .reset_index()\
    .sort_values('n', ascending=False)

df.head(12)
```

Out[10]:

	word1	word2	n
2872	second	cabin	18
2207	new	york	12
1035	fellow	passengers	11
2970	sick	man	6
2213	next	morning	5
657	dare	say	5
611	could	see	5
1034	fellow	passenger	4
2106	mr	jones	4
2321	one	day	4
1513	hurricane	deck	4
3166	steerage	passenger	4

Visualização dos dados

- Restrigindo as word pairs para apenas as que aparecem mais de 3 vezes.

In [11]:
```
df[df.n > 3].head(10)
```

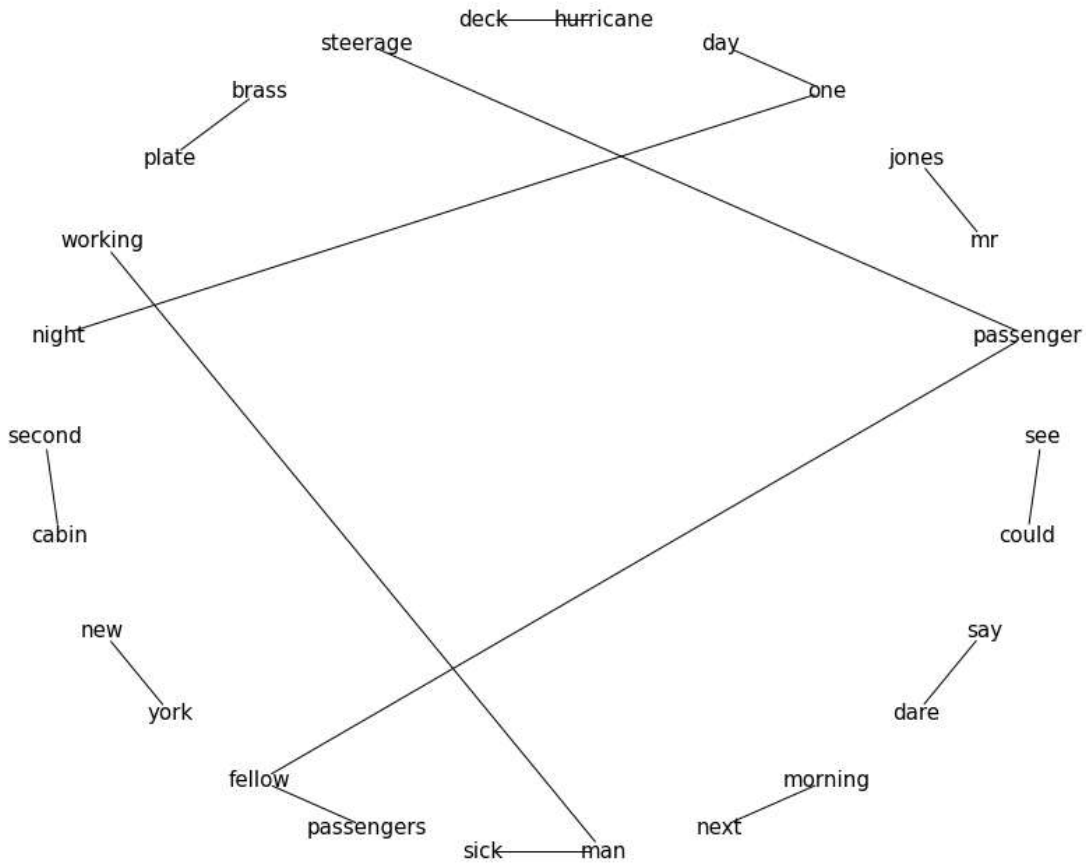
Out[11]:

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

	word1	word2	n
2321	one	day	4

Gráfico de conexões entre os pares de palavras:

In [12]:
```
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)
```



Pela associação de palavras acima, temos uma ideia daquelas que foram combinadas por Stevenson em mais de 3 vezes ao longo de sua narrativa. Alguns destaques são "steerage + passenger" (passageiro da terceira classe), "second + cabin" (cabine da segunda classe), "working + man" (trabalhador), "new + york" (New York), "next + morning" (manhã seguinte), dentre outras. Essa ação ajuda-nos a descobrir mais detalhes sobre o texto que queremos entender.