Imports e downloads

In [1]:

```
import nltk
import pandas as pd
import csv
from collections import OrderedDict, defaultdict, Counter
from urllib import request
from nltk import ngrams, FreqDist
from nltk.corpus import floresta
```

In [2]:

```
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('floresta')
```

```
[nltk_data] Downloading package punkt to /home/ellen/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /home/ellen/nltk_dat
a...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package floresta to /home/ellen/nltk_data...
[nltk_data]   Package floresta is already up-to-date!
```

Out[2]:

```
True
```

Coleta de documento

In [3]:

```
def open_file(filepath):
    file = open(filepath, 'r')

    return file.read()
```

Pré-processamento

In [4]:

```
def pre_process(raw):
    stopwords = nltk.corpus.stopwords.words('portuguese')

    tokens = nltk.word_tokenize(raw.lower())
    filtered = [t for t in tokens if t not in stopwords and t.isalpha() and len(
t) > 1]
    text = nltk.Text(tokens)
    dist = FreqDist(filtered)

    return tokens, filtered, text, dist
```

Informações básicas sobre o texto

In [5]:

```
def info(tokens, filtered_tokens):
    print(f"quantidade de palavras: {len(tokens)}")
    print(f"quantidade de palavras após o filtro: {len(filtered_tokens)}")
    print(f"quantidade de palavras únicas: {len(set(tokens))}")
    print(f"quantidade de palavras únicas após o filtro: {len(set(filtered_token
s))}")
```

Frequência e distribuição de palavras

In [6]:

```
def most_frequent(tokens, dist):
    sortedToken = sorted(list(set(tokens)), key=lambda token: dist[token], rever
se=True)
    frequent_tokens = [(token, dist[token]) for token in sortedToken[:20]]

    return frequent_tokens
```

Análise de contexto

In [7]:

```
def n_grams(text, dist):
    target_word = dist.max()
    fd = FreqDist(ng for ng in ngrams(text, 5) if target_word in ng)
    for hit in fd:
        print(' '.join(hit))
```

In [8]:

```
# O método concordance permite ver palavras em um contexto
def context(text, dist):
    target_word = dist.max()

    return text.concordance(target_word)
```

In [9]:

```
def collocation(text):
    return text.collocations()
```

In [10]:

```
def cont_index(text):
    return text.ContextIndex
```

Análise de Emoção

In [11]:

```
def emolex():
    wordList = defaultdict(list)
    emotionList = defaultdict(list)

    with open('lexico/teste.csv', 'r') as f:
        reader = csv.DictReader(f)
        for row in reader:
            if int(row['present']) == 1:
                wordList[row['word']].append(row['emotion'])
                emotionList[row['emotion']].append(row['word'])

    return wordList, emotionList
```

In [12]:

```
def generate_count(word_list, filtered_tokens):
    emoCount = Counter()
    for t in filtered_tokens:
        if len(word_list[t]) > 0:
            emoCount += Counter(word_list[t])
    return emoCount
```

In [13]:

```
def newList(emotionList, filtered_tokens):
    emoList = defaultdict(list)
    for t in filtered_tokens:
        for e in emotionList:
            for w in emotionList[e]:
                if w == t:
                    emoList[e].append(w)

    return emoList
```

In [14]:

```
def analysis(filepath):
    raw = open_file(filepath)

    tokens, filtered, text, dist = pre_process(raw)

    info(tokens, filtered)

    print("20 palavras mais frequentes:")
    print(most_frequent(filtered, dist))
    print("\n")
    print(f"Palavra mais frequente: {dist.max()} - {dist[dist.max()]} vezes")
    print("\n")
    print("N-grams da palavra de maior frequência \n")
    print(n_grams(text, dist))
    print("\n")
    print(context(text, dist))
    print("\n")
    print("Colocações significantes: \n")
    print(collocation(text))
    print("\n")

    wordList, emotionList = emolex()
    emoList = newList(emotionList, filtered)

    emotionCounts = generate_count(wordList, filtered)
    print(emotionCounts.most_common())
    print("\n")

    wordCounts = generate_count(emoList, filtered)
    print(wordCounts.most_common(30))
    print("\n")

    for w in wordCounts.most_common(30):
        print(w[0])
        print(wordList[w[0]])
        print('-------------------------------------')
```

In [ ]:

```

```