PMR3201 - Computação para Automação
Exercício Programa 1 - 2020

Manipulação de strings: uma aplicação na análise de sentimentos

Prof. Dr. Thiago de Castro Martins
Prof. Dr. Marcos de Sales Guerra Tsuzuki
Prof. Dr. Newton Maruyama
Prof. Dr. Rafael Traldi Moura

Deadline: 13/04/2020 - 23h59min (SISTEMA MOODLE)

1 Introdução

Nos últimos anos a área denominada Processamento de Linguagem Natural (Natural language processing) sofreu vários desdobramentos e convergências com outras áreas como a área de Aprendizado de Máquina (Machine Learning). Várias terminologias estão sendo utilizadas nessa área que causam uma certa confusão: text analysis, text analytics, text mining, sentiment analysis, etc.

A análise de sentimento, por exemplo, foca na análise de texto para extrair o seu significado, contexto, e intenção. Sua maior utilização está por exemplo na análise de opiniões de consumidores em sites de compra on-line e monitoramento de redes sociais.

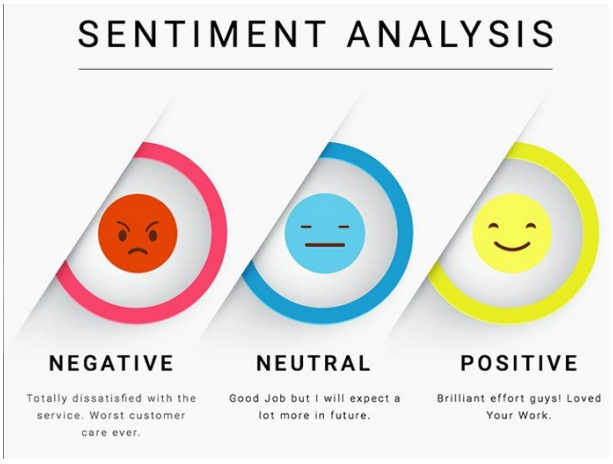

Figura 1: Análise de sentimento de consumidores. Fonte: https://towardsdatascience.com/statistical-sentiment-analysis-for-survey-data-using-python-9c824ef0c9b0

Uma análise de sentimento simples pode ser feito através da análise estatística de texto aonde analisa-se quais palavras tem o maior número de ocorrências num texto.

Por exemplo, podemos analisar qual o sentimento dos autores das letras do álbum PARANOID da banda Black Sabbath. A Figura 2 ilustra um tipo de gráfico denominado wordcloud contendo as 20 palavras mais frequentes nas letras de Paranoid. As palavras mais frequentes são colocadas com tamanhos de fonte maiores. A palavra mais frequente é 'yeah', depois temos palavras como 'time', 'war', 'funeral', 'death', etc.

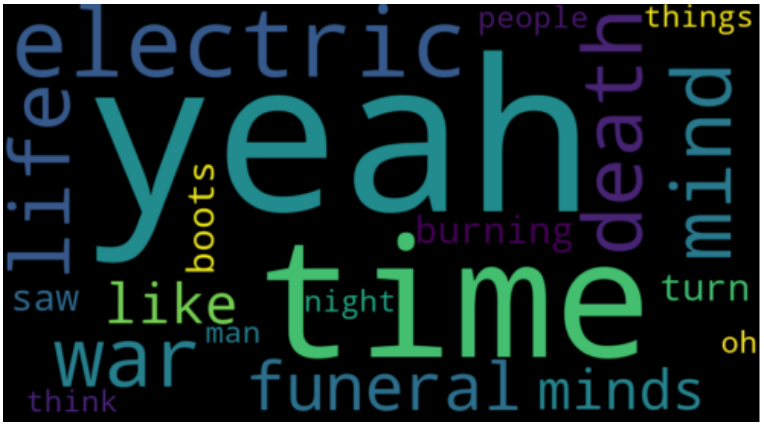

Figura 2: Gráfico wordcloud com as vinte palavras mais frequentes do álbum PARANOID. Fonte: geração própria.

Na análise estatística de texto a frequência de ocorrência de cada palavra do texto é calculada.

O texto deve ser lido de um arquivo do tipo `.txt` e depois transformado numa lista de strings (*Word tokenization*) contendo as palavras relevantes em letras minúsculas.

O algoritmo para a contagem das palavras atua sobre essa lista de strings.

Deve ser inicialmente realizado um pré-processamento que seleciona somente palavras de relevância abandonando as pontuações, preposições, pronomes, etc. e utilizando somente letras minúsculas. Esse processamento de texto não é trivial. Utilizaremos aqui funções da biblioteca denominada *Natural Language Toolkit* (NLTK).

```
import pandas as pd
import nltk as nk
import matplotlib.pyplot as plt
import wordcloud as wd
from nltk.corpus import stopwords
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

#   Abre o arquivo
f = open('AliceInWonderland.txt','r')
raw = f.read() # Le o conteudo do arquivo
#    Cria um tokenizer que mantem somente palavras
tokenizer = nk.tokenize.RegexpTokenizer('\w+')
tokens = tokenizer.tokenize(raw)
#    A variavel tokens e' uma lista de tokens
#    transformando todos os tokens em letras minusculas
lwords = []
for word in tokens:
    lwords.append(word.lower())
#   lwords agora so contem tokens com letras minusculas
#
#   carrega as stopwords da lingua Inglesa em sw
sw = nk.corpus.stopwords.words('english')
words_ns = []
for word in lwords:
    if word not in sw:
      words_ns.append(word)
#   word_ns contem tokens sem stopwords
```

Programa 1: Código para pré-processamento do texto.

Vamos tomar como exemplo uma análise da música do conjunto Iron Maiden, 'Seventh son of a seventh son', ilustrada a seguir:

```
Seventh son of a seventh son

Here they stand, brothers them all
All the sons, divided they'd fall
Here await the birth of the son
The seventh, the heavenly, the chosen one

Here the birth from an unbroken line
Born the healer, the seventh, his time
Unknowingly blessed and as his life unfolds
Slowly unveiling the power he holds

Seventh son of a seventh son
Seventh son of a seventh son
Seventh son of a seventh son
Seventh son of a seventh son

Then they watch the progress he makes
The Good and Evil which path will he take
Both of them trying to manipulate
The use of his powers, before it's too late

Seventh son of a seventh son
Seventh son of a seventh son
Seventh son of a seventh son
Seventh son of a seventh son

Today is born the seventh one
Born of woman the seventh son
And he in turn of a seventh son
He has the power to heal
He has the gift of the second sight
He is the chosen one
So it shall be written
So it shall be done
```

O processo de tokenização gera a seguinte lista de strings `tokens`:

['Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Here', 'they', 'stand', 'brothers', 'them', 'all', 'All', 'the', 'sons', 'divided', 'they', 'd', 'fall', 'Here', 'await', 'the', 'birth', 'of', 'the', 'son', 'The', 'seventh', 'the', 'heavenly', 'the', 'chosen', 'one', 'Here', 'the', 'birth', 'from', 'an', 'unbroken', 'line', 'Born', 'the', 'healer', 'the', 'seventh', 'his', 'time', 'Unknowingly', 'blessed', 'and', 'as', 'his', 'life', 'unfolds', 'Slowly', 'unveiling', 'the', 'power', 'he', 'holds', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Then', 'they', 'watch', 'the', 'progress', 'he', 'makes', 'The', 'Good', 'and', 'Evil', 'which', 'path', 'will', 'he', 'take', 'Both', 'of', 'them', 'trying', 'to', 'manipulate', 'The', 'use', 'of', 'his', 'powers', 'before', 'it', 's', 'too', 'late', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Seventh', 'son', 'of', 'a', 'seventh', 'son', 'Today', 'is', 'born', 'the', 'seventh', 'one', 'Born', 'of', 'woman', 'the', 'seventh', 'son', 'And', 'he', 'in', 'turn', 'of', 'a', 'seventh', 'son', 'He', 'has', 'the', 'power', 'to', 'heal', 'He', 'has', 'the', 'gift', 'of', 'the', 'second', 'sight', 'He', 'is', 'the', 'chosen', 'one', 'So', 'it', 'shall', 'be', 'written', 'So', 'it', 'shall', 'be', 'done']

Após a transformação de todas as strings em letras minúsculas e a retirada das *Non-stop words* a lista de strings `words_ns` se torna:

['seventh', 'son', 'seventh', 'son', 'stand', 'brothers', 'sons', 'divided', 'fall', 'await', 'birth', 'son', 'seventh', 'heavenly', 'chosen', 'one', 'birth', 'unbroken', 'line', 'born', 'healer', 'seventh', 'time', 'unknowingly', 'blessed', 'life', 'unfolds', 'slowly', 'unveiling', 'power', 'holds', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'watch', 'progress', 'makes', 'good', 'evil', 'path', 'take', 'trying', 'manipulate', 'use', 'powers', 'late', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'seventh', 'son', 'today', 'born', 'seventh', 'one', 'born', 'woman', 'seventh', 'son', 'turn', 'seventh', 'son', 'power', 'heal', 'gift', 'second', 'sight', 'chosen', 'one', 'shall', 'written', 'shall', 'done']

Após o cálculo das ocorrências das palavras o resultado está armazenado na lista `freqdist`:

[('seventh', 23), ('son', 21), ('one', 3), ('born', 3), ('birth', 2), ('chosen', 2), ('power', 2), ('shall', 2), ('stand', 1), ('brothers', 1), ('sons', 1), ('divided', 1), ('fall', 1), ('await', 1), ('heavenly', 1), ('unbroken', 1), ('line', 1), ('healer', 1), ('time', 1), ('unknowingly', 1)]

Um gráfico de barras com as 20 palavras de maior ocorrência está ilustrado na Figura 3. O código para o gráfico de barras está ilustrado abaixo. OBS: o código depende da biblioteca Pandas e Matplotlib como declarados no Programa 1.

```
freqword=pd.DataFrame.from_records(freqdist,columns=['word','count'])
freqword.plot(kind='bar',x='word')
```

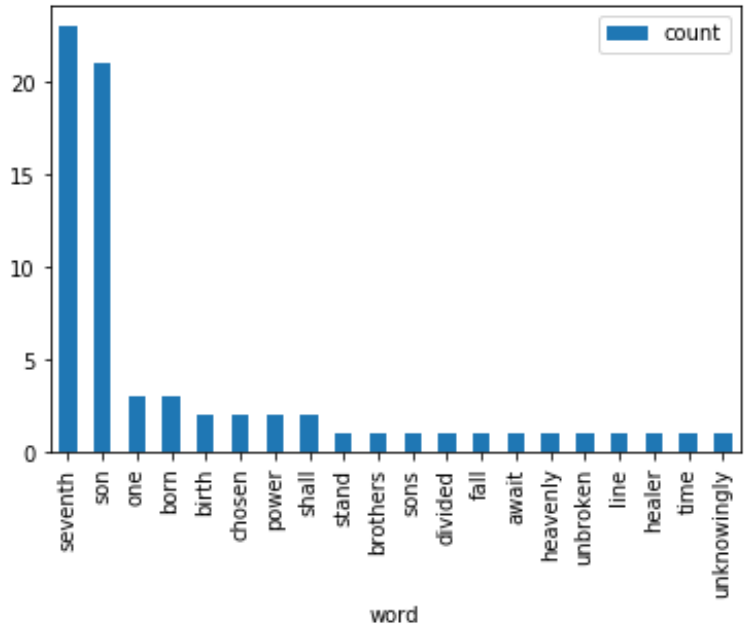

Figura 3: Gráfico de barras com as 20 palavras com o maior número de ocorrências. Fonte: geração própria.

Uma outro tipo de gráfico utilizado para ilustrar o número de ocorrências de palavras é denominado wordcloud (Figura 4). Utilizando a lista `freqdist` o gráfico wordcloud pode ser obtido através do código ilustrado a seguir. OBS: o código depende da biblioteca Pandas e Matplotlib como declarados no Programa 1.

```
wordc = wd.WordCloud(width=900,height=500, max_words=20,\
relative_scaling=1,normalize_plurals=False).generate_from_frequencies(freqdist)
plt.figure();
plt.imshow(wordc, interpolation='bilinear');
plt.axis("off")
plt.show()
```

Figura 4: Gráfico wordcloud com as 20 palavras com o maior número de ocorrências. Fonte: geração própria.

2 Especificações do programa

Uma especificação mais abstrata do programa para análise estatística de texto está ilustrada no Algoritmo 1.

```
- Leitura do arquivo *.txt
- Realiza pré-processamento que resulta na lista de strings <words_ns>
- Inicialize o dicionario <dicfreq> que contem pares (<x>,<nocorrer>)
            # (<x>,<nocorrer>) = (string,número de ocorrências)
- Para cada string <x> na lista <words_ns> faça:
       Se <x> se encontra em <dicfreq> Então
          <n_ocorrer> = <nocorrer> + 1
          Atualizar o par (<x>,<n_ocorrer>) em <dicfreq>
       Senão
          nocorrer = 1
          inserir (<x>,<nocorrer>) em <dicfreq>
            # O dicionário <dicfreq> deve conter todas as strings do texto e o número de
            #  vezes que a string aparece no texto
- Para cada par (<x>,<nocorrer>) de <dicfreq> inserir em uma lista ordenada <lwocorrencias>
            # Nessa lista os elementos devem estar ordenados em ordem decrescente do número
            # de ocorrências <nocorrer>
- Gerar o gráfico de barras indicando as 20 palavras de maior ocorrência
- Gerar o gráfico wordcloud com as 20 palavras de maior ocorrência
```

Algoritmo 1: Especificação da análise estatística de texto.

A principal estrutura de dados é identificada como `<dicfreq>` que deve ser implementada como um dicionário de dicionários. Cada sub-dicionário deve conter palavras que começam com o mesmo caracter. Essa estrutura de dados deve ser inicializada com um sub-dicionário vazio para cada uma das letras do alfabeto como ilustrado a seguir:

```
      dicfreq = {a:{}, b:{}, c:{}, .... w:{}, x:{}, y:{}, z:{}}
```

Um exemplo de dicionários de dicionários aonde apenas os caracteres `'a'` e `'b'` é ilustrado no código a seguir:

```
# inicialmente lista e' um dicionario de dicionarios
# cujo indexador primario sao letras do alfabeto
lista = {'a': {},'b':{}};
print(lista)
# Se for encontrado a palavra abelha devemos adicionar
# no dicionario da letra 'a'
lista['a'].update([('abelha',1)])
# o mesmo para 'bola' no dicionario da letra 'b'
lista['b'].update([('bola',1)])
lista['a'].update([('amora',1)])
lista['b'].update([('beterraba',1)])
print(lista)
# Para cada palavra devemos saber se ela ja existe ou nao
# A insercao deve ser condicional
# como mostrado abaixo
if 'beterraba' in lista['b']: # se ja existe o conteudo deve ser
# incrementado
count=lista['b']['beterraba']
count=count+1
lista['b']['beterraba']=count
else: # palavra ainda nao existe
lista['b'].update([('beterraba',1)])
print(lista)
```

O resultado da execução desse código é ilustrado a seguir:

```
{'a': {}, 'b': {}}
{'a': {'abelha': 1, 'amora': 1}, 'b': {'bola': 1, 'beterraba': 1}}
{'a': {'abelha': 1, 'amora': 1}, 'b': {'bola': 1, 'beterraba': 2}}
```

O resultado final do algoritmo deve ser a construção da estrutura de dados identificada como `<lwocorrencias>`. Cada elemento de `<dicfreq>` é representado por um par `(<x>,<nocorr>)` onde `<x>` é uma string e `<nocorr>` representa o número de

ocorrências da string no texto. Os elementos devem ser lidos de <dicfreq> e inseridos na lista <lwocorrencias> numa ordem decrescente quando considerando <nocorr>.

Como retirar os elementos de <dicfreq> ?

Continuando com o exemplo acima os elementos do dicionário lista podem ser retirados como um par (<x>,<nocorr>) como no exemplo abaixo:

```
x = lista['a'].popitem()
print('x = ',x)
print('x[0] = ',x[0])
print('x[1] = ',x[1])
print('lista = ',lista)
```

O resultado desse programa é apresentado abaixo:

```
x = ('amora', 1)
x[0] = amora
x[1] = 1
lista = {'a': {'abelha': 1}, 'b': {'bola': 1, 'beterraba': 2}}
```

Note que a operação popitem() retira o par ('amora',1) da variável lista.

3 Para você fazer

Projetar o programa principal único de forma a analisar os três romances listados a seguir:

- Alice in Wonderland, Lewis Caroll,
- Through the looking glass, Lewis Caroll,
- War and Peace, Liev Tolstói.

Os arquivos *.txt se encontram em anexo. São obras de domínio publico que podem ser acessadas no site do Projeto Gutenberg.

SOMENTE as funções dos pacotes NLTK, PANDAS e Matplotlib descritas acima podem ser utilizadas.

4 Deadline

- Você deve submeter o seu arquivo fonte *.py no sistema MOODLE.
- Prazo final: 13/04/2020 - 23h59min

5 Bibliografia

1. Natural Language Processing. Acesso: https://en.wikipedia.org/wiki/Natural_language_processing
2. Sentiment Analysis. Acesso: https://en.wikipedia.org/wiki/Sentiment_analysis
3. Natural language Toolkit. Acesso: https://www.nltk.org/
4. Pandas. Acesso: https://pandas.pydata.org/
5. Matplotlib. Acesso https://matplotlib.org/
6. Projeto Gutenberg. Acesso: https://www.gutenberg.org/wiki/PT_Principal