PROCESSAMENTO DE LINGUAGEM NATURAL

Introdução ao PLN usando Python

TÓPICOS

1. PLN utilizando NLTK
2. PLN utilizando Spacy

NLTK

- O NTLK (*Natural Language ToolKit*) é uma biblioteca escrita em Python que possui uma série de funcionalidades, como tokenização, lista de *stopwords*, *part-of-speech tagging*, simplificação de termos, etc.
- É uma das mais famosas na área de processamento de linguagem natural
- Livro sobre PLN utilizando a biblioteca NLTK: http://www.nltk.org/book/

NLTK

- A biblioteca NLTK é munida com vários *corpora*, os quais podem ser utilizados para testar as técnicas de processamento de linguagem natural
- O *corpora* está disponível no pacote `nltk.corpus`
- De cada *corpus*, pode-se obter o identificador, o texto puro, as palavras que compõem o texto, dentre outras informações

NLTK – *Corpora*

```
import nltk # importando a biblioteca NLTK
nltk.download('machado') #Fazendo o download do corpus caso esse não tenha sido previamente baixado
nltk.download('punkt') #Biblioteca utilizada para tokenizar as sentenças
from nltk.corpus import machado #Importanto o corpus machado
```

```
[nltk_data] Downloading package machado to /root/nltk_data...
[nltk_data]   Package machado is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
machado.fileids()[:3] # A função fileids() retorna os ids de todos os documentos do corpus
```

```
['contos/macn001.txt', 'contos/macn002.txt', 'contos/macn003.txt']
```

```
print(machado.raw('contos/macn001.txt')) # Retornando o conteúdo completo de um texto
```

```
Conto, Contos Fluminenses, 1870

Contos Fluminenses

Texto-fonte:

Obra Completa, Machado de Assis, vol. II,
```

NLTK – *Stopwords*

- A ferramenta NLTK já possui uma lista de *stopwords* para diferentes linguagens
- As listas de *stopwords* estão disponíveis no pacote `nltk.corpus`
- Caso ainda não tenha feito o *download* da lista de *stopwords* para a máquina local, é necessário fazer o *download* utilizando a função `nltk.download('stopwords')`
- Para retornar a lista de *stopwords*, deve-se utilizar o método `words([língua])` do objeto *stopwords*

NLTK – *Stopwords*

```
import nltk #Importando a biblioteca NLTK
nltk.download('stopwords') #Fazendo o download das listas de stopwords
from nltk.corpus import stopwords #Importando o objeto de stopwords
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
"""
Para retornar as stopwords de uma língua, basta utilizar a função
words e informa a string da língua
"""
stopwords.words('portuguese')
```

```
['a',         'com',       'e',                  'tivermos',
 'à',         'como',      'é',                  'tivesse',
 'ao',        'da',        'ela',                'tivessem',
 'aos',       'das',       'elas',               'tivéssemos',
 'aquela',    'de',        'ele',                'tu',
 'aquelas',   'dela',      'eles',       ...     'tua',
 'aquele',    'delas',     'em',                 'tuas',
 'aqueles',   'dele',      'entre',              'um',
 'aquilo',    'deles',     'era',                'uma',
 'as',        'depois',    'eram',               'você',
 'às',        'do',        'éramos',             'vocês',
 'até',       'dos',       'essa',               'vos']
```

NLTK – Tokenização

- A tokenização é um processo muito importante no processamento de linguagem natural
- É responsabilidade da tokenização extrair as sequências de caracteres que serão candidatas a algum tipo de processamento, ex., análise sintática, ou para serem utilizadas como atributos na representação de textos
- A biblioteca NLTK possui funcionalidades para extrair palavras individuais e sinais de pontuação como *tokens*, além de uma série de tokenizadores específicos, inclusive para o Twitter, e tokenizadores personalizados por meio de expressões regulares

NLTK – Tokenização

```
from nltk.tokenize import word_tokenize # Importanto o tokenizador de palavras
from nltk.tokenize import sent_tokenize # Importanto o tokenizador de sentenças
```

```
#Criando um texto de exemplo
texto = 'Goku is a hero in the Dragon Ball since 1989! Goku saved the earth so many times.'
```

```
nltk.sent_tokenize(texto) # Tokenizando um texto considerando sentenças como unidades
```

```
['Goku is a hero in the Dragon Ball since 1989!',
 'Goku saved the earth so many times.']
```

```
word_tokenize(texto) #Tokenizando um texto considerando palavras como unidades
```

```
['Goku',
 'is',
 'a',
 'hero',
 'in',
 'the',
 'Dragon',
 'Ball',
 'since',
 '1989',
 '!',
 'Goku',
 'saved',
 'the',
 'earth',
 'so',
 'many',
 'times',
 '.']
```

NLTK – Tokenização

```
from nltk.tokenize import TweetTokenizer
texto2 = "I'm very veryyyy happyyyyyyyyy #betterlife @barneys :P :D"
```

```
twitterTokenizer = TweetTokenizer()
twitterTokenizer.tokenize(texto2)
```

```
"""
-strip_handles=True: remove os nomes de usuário do texto
-reduce_len=True: substitui sequencias de 3 ou mais caracteres
                  repetidos por sequência de 3 caracteres
"""
```

```
["I'm",
 'very',
 'veryyyy',
 'happyyyyyyyyy',
 '#betterlife',
 '@barneys',
 ':P',
 ':D']
```

```
twitterTokenizer2 = TweetTokenizer(strip_handles=True, reduce_len=True)
twitterTokenizer2.tokenize(texto2)
```

```
["I'm", 'very', 'veryyy', 'happyyy', '#betterlife', ':P', ':D']
```

NLTK – Simplificação das palavras

- A NLTK possui um série de diferentes algoritmos para radicalização de palavras
- Para algumas linguagens possui uma única opção
- Também possui algoritmo para fazer a lematização

NLTK – Simplificação das palavras

```
from nltk.stem.snowball import SnowballStemmer
```

```
SnowballStemmer.languages # Listando as linguas suportadas pelo SnowBallStemmer
snowballStemmer = SnowballStemmer("portuguese") # Criando um objeto stemizador
```

```
snowballStemmer.stem('computação') # Radicalizando as palavras
```

```
'comput'
```

```
snowballStemmer.stem('computador')
```

```
'comput'
```

```
snowballStemmer.stem('computando')
```

```
'comput'
```

NLTK – Simplificação das palavras

```
from nltk.stem import WordNetLemmatizer # Importando o WordNetLemmatizer
                        # e fazendo download do recurso caso não exista
"""Também é possível informar a função sintática da palavra no parâmetro 'pos'.\
        No exemplo abaixo, 'a' denota adjetivo"""
wordNetLemmatizer.lemmatize("better", pos ="a")
```

```
'good'
```

```
"""O uso da função sintárica pode alterar/melhorar o resultado da lematização \
"""
wordNetLemmatizer.lemmatize("clustering")
```

```
'clustering'
```

```
wordNetLemmatizer.lemmatize("clustering", pos='v')
```

```
'cluster'
```

NLTK – *POS Tagging*

- **Part-of-Speech *tagging* corresponde à identificação de classe gramatical de uma unidade do texto**

- **Veja a lista dos significados das *tags* em**
  - https: //www.guru99.com/pos-tagging-chunking-nltk.html

- **É necessário fazer o *download* de alguns recursos para o funcionamento:**
  - `nltk.download('averaged_perceptron_tagger')`
  - `nltk.download('maxent_ne_chunker')`
  - `nltk.download('words')`

NLTK – *POS Tagging*

```
import nltk
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')

text2 = 'Rafael is working at Google in the South America'
```

```
print(nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(text2))))
```

```
(S
  (GPE Rafael/NNP)
  is/VBZ
  working/VBG
  at/IN
  (ORGANIZATION Google/NNP)
  in/IN
  the/DT
  (LOCATION South/NNP America/NNP))
```

VBZ: verb, present tense with 3rd person singular (bases)

IN: preposition/subordinating conjunction

NLTK – *POS Tagging*

POS *tagger* para o português

https://github.com/inoueMashuu/POS-tagger-portuguese-nltk

SPACY

- O Spacy é uma biblioteca Python para PLN também bastante conhecida e mais recente que o NLTK
- NTLK: variedade de soluções para uma determinada tarefa
- Spacy: basicamente uma única solução para cada tarefa
- Permite que diversos resultados de técnicas de PLN sejam acessados a partir de um próprio objeto de um *token*, tornando a programação mais simples e clara
- Para utilizar o Spacy em diferentes linguagens:

https://spacy.io/models

SPACY – Tokenização

```
import spacy
nlp = spacy.load("en_core_web_sm")
"""Gerando um objeto o qual gerará um objeto iteravel e cada item representa um token, e
   cada token contém além da palavra em si, sua versão simplicada, função sintática, etc."""
sentenca = nlp('Rafael is working at Google in the South America. He works very hard :D')
for token in sentenca: # o atributo .text retorna o texto do respectivo token
  print(token.text)
```

```
Rafael
is
working
at
```

```
#Caso queira considerar primeiramente cada sentença como token também é possível.
#Para isso basta utilizar a propriedade sents do objeto gerado
doc = nlp('Lucas is working at Google in the South America. He works very hard!')
```

```
for sent in doc.sents:
  print(sent.text)
```

```
Lucas is working at Google in the South America.
He works very hard!
```

SPACY – *Stopwords*

```
sentenca = nlp('Lucas is working at Google in the South America.\
                He works very hard!')
```

```
for token in sentenca: #A propriedade .is_stop retorna True se o token está
  print(f'{token.text} - {token.is_stop}')          # na lista de stopwords
```

```
Rafael - False
is - True
working - False
at - True
Google - False
in - True
the - True
South - False
America - False
. - False
He - True
works - False
very - True
hard - False
! - False
```

SPACY – Composição dos *tokens*

```
sentenca = nlp('Lucas is working at Google in the South America since 1999.\
                He works very hard! This product costs €90.89')
#A propriedade .is_alpha retorna True se o token é composto apenas por
                                                # caracteres alfabéticos
for token in sentenca:
  print(f'{token.text} - {token.is_alpha}')
```

```
Rafael - True       . - False
is - True           He - True
working - True      works - True
at - True           very - True
Google - True       hard - True
in - True           ! - False
the - True          This - True
South - True        product - True
America - True      costs - True
since - True        € - False
1999 - False        90.89 - False
```

SPACY – Lematização

```
sentenca = nlp('Lucas has drinked two coffees while the\
     computer is computing the values of the matrices.')
```

```
for token in sentenca:
  print(f'{token.text} - {token.lemma_}')
```

```
Rafael - Rafael
has - have
drinked - drink
two - two
coffees - coffee
while - while
the - the
computer - computer
is - be
computing - compute
the - the
values - value
of - of
the - the
matrices - matrix
. - .
```

SPACY – *POS Tagging*

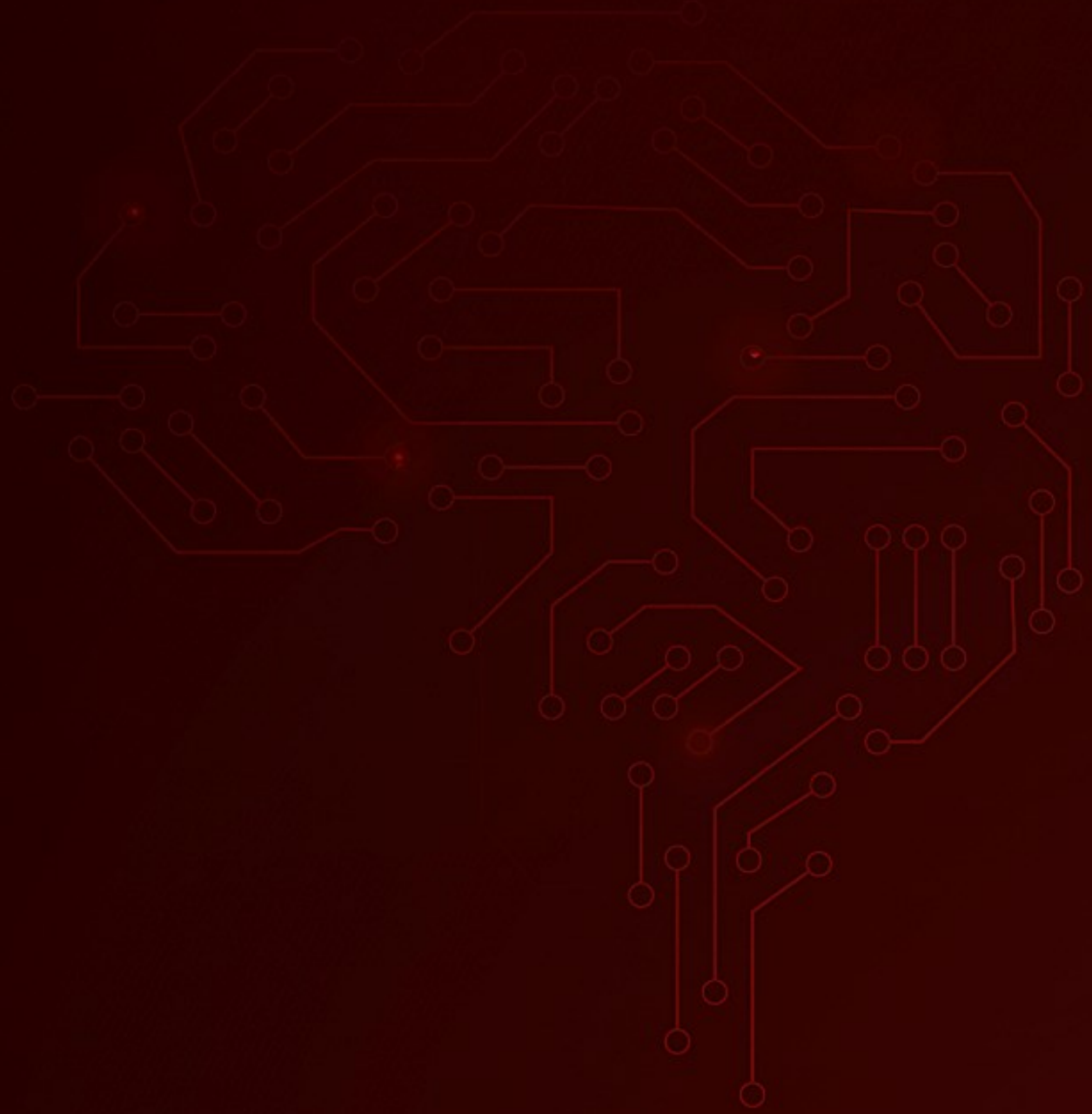
```
sentenca = nlp('Rafael is working at Google in the South America. He works very hard!')
```

```
for token in sentenca:
  print(f'{token.text} - {token.pos_} - {token.tag_}')
```

```
Rafael - PROPN - NNP
is - AUX - VBZ
working - VERB - VBG
at - ADP - IN
Google - PROPN - NNP
in - ADP - IN
the - DET - DT
South - PROPN - NNP
America - PROPN - NNP
. - PUNCT - .
He - PRON - PRP
works - VERB - VBZ
very - ADV - RB
hard - ADV - RB
! - PUNCT - .
```

O QUE VIMOS?

- PLN utilizando NLTK
- PLN utilizando Spacy



PRÓXIMA VIDEOAULA

- *Corpus*

REFERÊNCIAS

- Curso de Tópicos em Inteligência Artificial
  - Prof. Rafael G. Rossi (UFMS)
  - https://www.youtube.com/@RafaelRossiTech/playlists