



Processamento de Linguagem Natural Engenharia Biomédica

Luís Filipe Cunha

lfc@di.uminho.pt

José João Almeida

jj@di.uminho.pt





Processamento de Linguagem Natural

“Natural language processing (NLP) is concerned with giving computer program the ability to understand human language as it is spoken and written -- referred to as natural language.”



Plano

- Introdução ao Python
- Unix Filters
- Expressões Regulares
- Corpora
- Gramática do Português
- Terminologia Dicionários e enciclopédias
- Word Embeddings
- Web Scraping
- Domain Specific Language
- NER / PoS / Q&A ...

Ferramentas

- spaCy
- Stanza
- Flask
- BS4
- Selenium
- Gensim
- FastText
- HuggingFace
- NLTK
- Terminal

Ficheiros

- html
- xml
- json
- yaml
- latex
- tmx
- pdf
- texto



Aulas

Ambiente de trabalho:

- Unix
- Python > 3.8
- Editor de texto
 - VS CODE
 - Ctrl + B
 - Theme
 - Ctrl + (+/-)
 - Alt + Z

GitHub:

- <https://github.com/lfcc1/pln2224>
- Bibliografia
- Aulas
- Data
- Slides
- Tpcs



Avaliação

- Testes
- Trabalhos práticos
- TPCs

- Teste - 40%
- Trabalhos práticos - 40%
- Trabalhos de casa - 20%
- Nota mínima 8

Trabalho obrigatório!

Datas: ...



TPC

- Criação de repositório Github: pln-2224
- Criação de diretoria para cada TPC
 - TPC1
 - TPC2
 - ...
- Descrição para cada TPC em Markdown (Readme.md)
 - [Basic Syntax | Markdown Guide](#)



Github

Installation:

ubuntu:

```
sudo apt update  
sudo apt upgrade  
sudo apt install git
```

windows:

<https://gitforwindows.org/>

Setup:

```
echo "# Repositório PLNEB-2224" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git remote add origin https://github.com/lfcc1/plneb-2224.git  
git push -u origin master
```



Introdução ao Python

```
1  #!/usr/bin/env python3
2
3  print("Hello World")
4
5  editor = input("What is your favorite text editor? ")
6
7  a = [3, 6, 7, 2, 5, 9, 4, 0, 10]
8
9  for i in a:
10     if i < 5:
11         print(str(i) + " is lower than 5")
12     elif i > 5:
13         print(str(i) + " is higher than 5")
14     else:
15         print("FIVE!")
16
17  # comment
```

- Listas
- Dicionários
- Ficheiros
- Exercícios



Strings

Creating a String: "... " ou '...';
multiline String: """...""" ou '''...''';

```
string1 = "hello!"  
string2 = 'hello'  
multilineString1 = """Hello everyone!  
I'm a multiline string.  
"""
```

```
multilineString2 = '''Hello guys!  
I'm also a multiline string!'''
```

```
string2 = 'I'm also a multiline string!'
```

Strings



- * Comprimento duma string:
``...len(frase)...`;`
- * Verificar se a frase contem a palavra "dia" uma parte:
``if "dia" in frase...`;`
- * Verificar se não contém a palavra "dia":
``if "dia" not in frase...`;`
- * Conversão para maiúsculas:
``frase.upper()`;`
- * Conversão para minúsculas:
``frase.lower()`;`
- * Remover espaço branco do início e do fim:
``frase.strip()`;`
`frase.lstrip() frase.rstrip()`
- * Substituir a palavra "Hoje" por "Today":
``frase.replace("Hoje", "Today")`;`
- * Partir uma string em bocados usando um separador:
``frase.split("#")`;`
- * Concatenação de 2 strings:
``s1 + s2`.`

String Slicing



- * Slicing a string: ``frase[start:stop:step]``; # start through not past stop, by step
 - from the start: ``frase[:stop]``;
 - until the end: ``frase[start:]``;
 - negative indexes.
 - amount by which the index increases ``frase[::step]``

```
text = "hello world"
slice = text[1:5]
slice = text[:5]
slice = text[5:]
slice = text[5:-1]
slice = text[-5:]
slice = text[:-2]
slice = text[:2]
slice = text[::2]
```

Listas

```
listaA = [1,2,3]
listaB = [1,2,3,"Joana","Bruno","Filipe"]
lista = ["Joana","Bruno","Filipe"]

lista[2]
# "Filipe"
"Susana" in lista
# False
lista.append("Maria")
# ['Joana', 'Bruno', 'Filipe', 'Maria']
lista.insert(2,"Pedro")
# ['Joana', 'Bruno', 'Pedro', 'Filipe', 'Maria']
lista[3] = "João"
# ['Joana', 'Bruno', 'Pedro', 'João', 'Maria']
concat = [1,2,3] + [4,5,6]
# [1, 2, 3, 4, 5, 6]
```



Listas

```
lista = list(range(0,10))  
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]  
lista.pop(3)  
# [0, 1, 2, 4, 5, 6, 7, 8, 9]  
lista.pop(-3)  
# [0, 1, 2, 4, 5, 6, 8, 9]  
lista.remove(5)  
# [0, 1, 2, 4, 6, 8, 9]
```

```
dict = {}  
colors = {"fcp": "blue", "slb": "red", "scp": "green"}
```



Dicionários

```
len(colors)  
# 3  
"slb" in colors  
# True  
"fcb" in colors  
# False  
colors["fcp"]  
# blue  
colors["aca"]  
#KeyError: 'aca'  
colors.get("aca")  
# None
```

```
colors.keys()  
# ['fcp', 'slb', 'scp']  
colors.values()  
# ['blue', 'red', 'green']  
colors.items()  
#[('fcp', 'blue'), ('slb', 'red'), ('scp', 'green')]  
  
colors | {"aca": "black", "fcp": "blue and white"}
```

Listas por compreensão



```
newlist = [expression for item in iterable if condition == True]
```

```
lista = list(range(1,10))
```

```
# [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
quadrados = [x * x for x in lista]
```

```
# [1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Gera a lista [2, 4, 6, 8] usando listas em compreensão:

```
pares = [x for x in lista if x % 2 == 0 ]
```

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
```

Gera a lista das palavras com a letra "a" usando listas em compreensão:

```
frutas_a = [fruta for fruta in fruits if "a" in fruta ]
```

Ordenação



```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
```

Ordena a lista alfabeticamente: #['banana', 'kiwi', 'mango', 'orange', 'pineapple']

```
thislist.sort()
```

```
res = sorted(thislist)
```

Ordena a lista alfabeticamente por ordem inversa: #['pineapple', 'orange', 'mango', 'kiwi', 'banana']

```
thislist.sort(reverse = True)
```

```
res = sorted(thislist, reverse = True)
```

```
frutas = {"orange":12, "mango":10, "kiwi":43, "pineapple":11, "banana":20}
```

Qual o resultado de: res = sorted(frutas)

```
#['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Ordena os elementos do dicionário de frutas por quantidade

```
#[( 'mango', 10), ( 'pineapple', 11), ( 'orange', 12), ( 'banana', 20), ( 'kiwi', 43)]
```

```
res = sorted(frutas.items(), key=ordena)
```

```
def ordena(e):  
    return e[1]
```




Ficheiros

```
file = open('data/exemplo', "r")  
lines = file.readlines()
```

```
file1 = open('data/exemplo', "r")  
string = file1.read()
```

```
file2 = open('data/exemplo', "r")  
line = file2.readline(10)
```

```
['Ola! Bem vindo a Scripting no Processamento de Linguagem Natural \n', 'Este  
ficheiro é apenas um exemplo.\n', 'Boa sorte!']
```

```
---
```

```
Ola! Bem vindo a Scripting no Processamento de Linguagem Natural  
Este ficheiro é apenas um exemplo.
```

```
Boa sorte!
```

```
---
```

```
Ola! Bem v
```



Exercícios

1. Programa que pergunta ao utilizador o nome e imprime em maiúsculas.
2. Função que recebe array de números e imprime números pares.
3. Função que recebe nome de ficheiro e imprime linhas do ficheiro em ordem inversa.
4. Função que recebe nome de ficheiro e imprime número de ocorrências das 10 palavras mais frequentes no ficheiro.
5. Função que recebe um texto como argumento e o "limpa": separa palavras e pontuação com espaços, converte para minúsculas, remove acentuação de caracteres, etc.



Exercícios

Create a function that:

1. given a string "s", reverse it.
2. given a string "s", returns how many "a" and "A" characters are present in it.
3. given a string "s", returns the number of vowels there are present in it.
4. given a string "s", convert it into lowercase.
5. given a string "s", convert it into uppercase.
6. Verifica se uma string é capicua
7. Verifica se duas strings estão balanceadas (Duas strings, s1 e s2, estão balanceadas se todos os caracteres de s1 estão presentes em s2.)
8. Calcula o número de ocorrências de s1 em s2
9. Verifica se s1 é anagrama de s2.
 - o "listen" e "silent": Deve imprimir True
 - o "hello", "world": Deve imprimir False



Processamento de Linguagem Natural Engenharia Biomédica

Luís Filipe Cunha

lfc@di.uminho.pt

José João Almeida

jj@di.uminho.pt

