**ESPECIALIZAÇÃO DATA SCIENCE E BIG DATA**

**ANÁLISE DE TEXTOS COM R E PYTHON**

**Professor Leandro ESCOBAR**

**PARTE 3 – DEPENDÊNCIAS**

O Python NLTK pode ser usado para fazer o reconhecimento de classes gramaticais (POS) em um determinado trecho de texto. Adicionalmente, obter mais detalhes sobre o texto pode descrever como os termos dependem entre si, apoiando a compreensão automática do texto. Por exemplo, encontrar nomes de personalidades, lugares ou produtos serve como entrada para interpretação do texto. Podemos manter um dicionário com todos esses nomes. Mas de forma mais simples, pode-se aplicar uma análise de PDV para identificar tais padrões

**Chunking:** é o processo de extrair frases curtas do texto.Os algoritmos de marcação de POS fazem essa fragmentação. Lembrando de que os tokens (palavras) produzidos pela fragmentação não se sobrepõem.

1. **Stemming**

Processo de redução de uma palavra ao seu radical que se afixa a sufixos e prefixos. O stemming é importante na compreensão da linguagem natural (NLU) e no processamento da linguagem natural (PNL).

Stemming é uma estratégia frequente para indexadores e buscadores, porque reduz diferentes termos com a mesma ideia a um só termo, seu radical.

Por exemplo, os termos Esperar ; Esperava; Esperando serão convertidos em "espera".

**Demonstração: Stemming**

Abra o Pycharm ou a IDE se sua preferência e siga as orientações do professor.

Crie um novo projeto: File / New Project com o nome Aula\_NLP2

Será necessário instalar o pacote de métodos para NPL chamado NLTK (siga as orientações do professor)

Crie um novo arquivo do tipo Python (será o seu programa para esta lição)

File / New / Python File, com o nome stemm1 (a IDE atribuirá a extensão .py ao arquivo)

Você precisa baixar o rslp, que é um processador para stemming em português.

*Atenção, o código abaixo é o exemplo do que será realizado. Evite copiá-lo, mas consulte-o caso tenha dúvida. Criar seu próprio código aprofunda o aprendizado e desenvolve sua habilidade com a programação.*

**import** nltk

**from** nltk **import** PorterStemmer, LancasterStemmer, word\_tokenize

nltk.download(**'rslp'**)

text1 = **'My name is Maximus Decimus Meridius, commander of the armies of the north, General of the Felix legions and loyal servant to the true emperor, Marcus Aurelius. Father to a murdered son, husband to a murdered wife. And I will have my vengeance, in this life or the next (Gladiator, the movie).'**

text2 = **''**

tokens = word\_tokenize(text1)

porter = PorterStemmer()

print(porter.stem(**'commanded'**)) *#Tentar com outras derivações da palavra commander*

*#PORTER STEMMER*

*#Stemmatizar a frase*

*#Criar uma lista que receberá o stemm de cada token*

stems\_porter = []

*#percorrer os tokens, stemmatizar e adicionar a lista de destino*

**for** t **in** tokens:

stems\_porter.append(porter.stem(t))

print(stems\_porter)

*#LANCASTER STEMMER*

lancaster = LancasterStemmer()

stems\_lancaster=[]

*#percorrer os tokens, stemmatizar e adicionar a lista de destino*

**for** t **in** tokens:

stems\_lancaster.append(lancaster.stem(t))

print(stems\_lancaster)

*#Usando o idioma pprtuguês*

text2 = **'Meu nome é Maximus Decimus Meridius, comandante dos exércitos do norte, general das legiões de Félix e servo leal ao verdadeiro imperador, Marcus Aurelius. Pai de um filho assassinado, marido de uma esposa assassinada. E eu terei minha vingança, nesta vida ou na próxima (Gladiador, o filme).'**

tokens\_pt = word\_tokenize(text2)

stems\_porter = []

*#percorrer os tokens, stemmatizar e adicionar a lista de destino*

**for** t **in** tokens\_pt:

stems\_porter.append(porter.stem(t))

*#LANCASTER STEMMER*

lancaster = LancasterStemmer()

stems\_lancaster=[]

*#percorrer os tokens, stemmatizar e adicionar a lista de destino*

**for** t **in** tokens\_pt:

stems\_lancaster.append(lancaster.stem(t))

*#Imprimir as duas listas de stemms para comaparação*

print(**'Em pt'**)

print(stems\_porter)

print(stems\_lancaster)

*#UTILIZANDO UM STEMMER PARA O IDIOMA PORTUGUÊS*

rslp = nltk.RSLPStemmer()

stems\_rslp=[]

**for** t **in** tokens\_pt:

stems\_rslp.append(rslp.stem(t))

*#*

print(**'Em português, com RSLP'**)

print(**'Porter: '**,stems\_porter)

print(**'Lancaster: '**,stems\_lancaster)

print(**'Rslp: '**,stems\_rslp)

**Também é possível acessar corpus externos**

Baixe o arquivo mix20\_rand700\_tokens\_cleaned.zip da pasta de dados da turma e descompacte-o em uma pasta segura do seu computador.

O corpus está organizado em duas pastas Pos e Neg, que são as categorias dos dados.

1. Baixe o arquivo do link <http://www.cs.cornell.edu/people/pabo/movie-review-data/mix20_rand700_tokens_cleaned.zip>
2. Ou da pasta de dados do módulo (veja o link com o professor)
3. Descompacte o arquivo e analise a estrutura de pastas criada.

As pastas **neg** e **pos** indicam as categorias dos documentos nesse *corpus*.

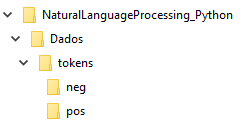


Figura 2 Estrutura de pastas categorizando documentos

**Demonstração: Extrair documentos e palavras de corpus previamente preparados**

*Atenção, o código abaixo é o exemplo do que será realizado. Evite copiá-lo, mas consulte-o caso tenha dúvida. Criar seu próprio código aprofunda o aprendizado e desenvolve sua habilidade com a programação.*

Inicie um novo arquivo (arquivo\_externo.py)

**from** nltk.corpus **import** CategorizedPlaintextCorpusReader  
  
*#Abrir os documentos dentro do caminho específico  
#Argumentos  
#1. Caminho absoluto para os documentos  
#2. tipo / extensão dos documentos (\*.txt)  
#3. indicativo das pastas que formarão as categorias  
# todos os argumentos são expressões regulares*leitor = CategorizedPlaintextCorpusReader(  
 **r'C:/Users/Convexa/Documents/Aulas/NaturalLanguageProcessing\_Python/Dados/tokens'**,  
 **r'.\*.txt'**, cat\_pattern = **r'(\w+)/\*'**)  
  
*#Verificar o que foi carregado*print(leitor.categories())  
print(leitor.fileids())  
  
*#Separar o corpus de acordo com as categorias*posFiles = leitor.fileids(categories=**'pos'**)  
negFiles= leitor.fileids(categories=**'neg'**)  
print(**'Arquivos pos:'**, posFiles)  
print(**'Arquivos neg:'**, negFiles)  
  
*#Carregar os primeiros arquivos das categorias*arqP = posFiles[0]  
arqN = negFiles[1]  
  
print(**"ArqP: "**, arqP)  
print(**"ArqN: "**, arqN)  
  
*#Imprimir as sentenças dos arquivos*print(**'Palavras nos arquivos selecionados'**)  
**for** p **in** leitor.words(arqP):  
 print(p + **' '**, end = **''**)  
  
print(**'---'**)  
  
**for** p **in** leitor.words(arqN):  
 print(p+ **' '**, end = **''**)

**EXERCITANDO 1**

**Execute o que se pede. Logo após, cole a solução do exercício neste documento.**

**Lembre-se de salvar seu programa, para estudos posteriores**

**Tempo estimado 20 minutos**

Utilize o arquivo com Noticias\_2 (pasta Dados) e compare os *stemms* obtidos a partir do 2º e 3º parágrafos do texto. Utilize Porter, Lancaster e RSPL.

1. **POS (Part of Speach) e NER (Named Entitity Recognition)**

POS: Trata-se de, a partir de um modelo pré elaborado, determinar quais são as classes das palavras no texto.

NER: Trata-se de reconhecer os substantivos (que representam entidades - coisas ou seres) nomeados previamente. Por exemplo, Curitiba = Local; João = Nome próprio

A operação de classificação de palavras tem como saída a lista de palavras do texto marcadas com suas classes. Esta aplicação é chamada *tagging*.

**Demonstração: POS - Part of speach**

*Atenção, o código abaixo é o exemplo do que será realizado. Evite copiá-lo, mas consulte-o caso tenha dúvida. Criar seu próprio código aprofunda o aprendizado e desenvolve sua habilidade com a programação.*

**import** nltk

**import** pandas **as** pd

**import** numpy **as** np

text = **'Apple aquires Zoom in China on wednesday 6th may 2020.'** \

**'This news has made Apple and Google stock jump by 5% in the United States of America.'**

textp=**'Apple adquire Zoom na China na quinta-feira 6 de maio de 2020. '** \

**'Essa notícia fez as ações da Apple e da Google subirem 5% nos Estados Unidos.'**

*#COM O TEXTO EM INGLES*

*#Tokenizar*

palavras = nltk.word\_tokenize(text)

print(palavras)

*#Part of Speach tagging*

pos\_tags = nltk.pos\_tag(palavras)

print(pos\_tags)

*# Para descrever as tags*

print(nltk.help.upenn\_tagset(**'IN'**))

**Demonstração NER - Named Entity Recognition**

*#Atenção: Continue escrevendo no mesmo programa para POS, pois NER é uma tarefa derivada de POS*

*#NER - NAMED ENTITY RECOGNITION*

*#Requer Numpy*

chuncks = nltk.ne\_chunk(pos\_tags, binary=**True**)

**for** chunck **in** chuncks:

print(chunck)

*#Recuperar as Entidades dentro do texto*

entitidades = []

rotulos = []

**for** chunck **in** chuncks:

**if** hasattr(chunck,**'label'**):

entitidades.append(**' '**.join(c[0] **for** c **in** chunck))

rotulos.append(chunck.label())

print(**'###'**)

print(entitidades)

print(rotulos)

entitidades\_com\_rotulos = list(set(zip(entitidades,rotulos)))

print(entitidades\_com\_rotulos)

entitidades\_df = pd.DataFrame(entitidades\_com\_rotulos)

entitidades\_df.columns=[**'Entidades'**,**'Rotulos'**]

print(entitidades\_df)

**EXERCITANDO 2 (para casa)**

**Execute o que se pede. Logo após, cole a solução do exercício neste documento.**

**Lembre-se de salvar seu programa, para estudos posteriores**

**Tempo estimado 20 minutos**

Utilize o arquivo Noticia\_1 disponível na pasta de dados da turma e realize as tarefas de POS e NER.

Salve o resultado de POS em um arquivo, com <seu\_nome>\_POS\_Noticia1

Salve o resultado de NER em um arquivo, com <seu\_nome>\_NER\_Noticia1

**Extrair o vocabulário comum entre dois textos**

O vocabulário comum tem várias aplicações, podendo ser útil para a indicação de plágios, determinação do vocabulário e identificação da estratégia de comunicação escrita.

**Demonstração Vocabulário em comum**

**Atenção:** Utilize o fragmento de programa disponível no arquivo Vocabulário\_em\_comum para inicar o seu programa Py.

*#Cinco primeiros parágrafos da Noticia\_1*

*texto1 =* ***'Lily Ebert, de 97 anos, é uma das últimas sobreviventes do Holocausto do Reino Unido. E mais de 75 anos após vencer os horrores nazistas, ela superou outra ameaça: a covid-19. '*** *\*

***'Eu estive em Auschwitz, passei pela pior situação que um ser humano pode viver. E graças a Deus eu sobrevivi", contou ao repórter Guy Lynn, da BBC, em sua casa no norte de Londres. Então eu deveria sobreviver a isso (covid-19) também. E então eu sobrevivi.'*** *\*

***'Inferno em Auschwitz '*** *\*

***'Lily tinha apenas 21 anos quando foi salva por tropas americanas em 1945, no fim da Segunda Guerra Mundial. Ela e duas irmãs estavam em uma Marcha da Morte, uma transferência forçada de prisioneiros para outro campo de concentração. Muitos não chegavam vivos ao fim do trajeto. '*** *\*

***'Um ano antes, ela tinha sido deportada de Bonyhád, sua cidade natal na Hungria, e levada a Auschwitz, campo de concentração na Polônia onde cerca de 1 milhão de judeus e milhares de outras pessoas foram assassinadas pelos nazistas.'***

*#Três primeiros parágrafos da Noticia\_2*

*texto2 =* ***'De saída do comando do maior banco da América Latina, o presidente do Itaú Unibanco, Candido Bracher, considera “pobres” os resultados da atuação brasileira na proteção da Amazônia. '*** *\*

***'A falta de progresso nessa área faz com que o mundo, na sua visão, mantenha uma postura “muito crítica” em relação ao Brasil, e ainda afasta potenciais investidores. '*** *\*

***'A percepção internacional em relação à atuação brasileira na proteção da Amazônia não é boa. Independentemente da maneira como se julguem os esforços realizados, o fato é que os resultados têm sido muito pobres”, avalia Bracher, em comentário ao Estadão/Broadcast, no âmbito do Fórum Econômico Mundial de Davos, que este ano está sendo realizado virtualmente por conta da pandemia. '*** *\*

***'O executivo não quis responder questões sobre outros temas como macroeconomia, política e quanto ao fim do seu mandato à frente do Itaú. Concordou, porém, em comentar sobre às relacionadas ao meio ambiente e proteção da Amazônia, temas centrais do fórum de Davos.'***

*#1 - Tokenizar os textos*

***from*** *nltk* ***import*** *word\_tokenize*

*tokens\_1 = word\_tokenize(texto1)*

*tokens\_2 = word\_tokenize(texto2)*

*#2 - Obter o vocabilário (palávras únicas) dos textos*

*vocabulario\_1 = set(tokens\_1)*

*vocabulario\_2 = set(tokens\_2)*

*#3 - Obter o vocabulário comum*

*vocabulario\_comum = vocabulario\_1 & vocabulario\_2*

*print(vocabulario\_comum)*

**EXERCITANDO 3 (para casa)**

**Execute o que se pede. Logo após, cole a solução do exercício neste documento.**

**Lembre-se de salvar seu programa, para estudos posteriores**

**Tempo estimado 45 minutos**

O NLTK possui um corpus com as obras de Machado de Assis (from nltk.corpus import machado)

1. Execute print(machado.readme()) para conhecer melhor o corpus
2. Utilizando oo documentos relativos a Dom Casmurro e O alienista, faça o que se pede
3. Classifique as palavras de acordo com suas classes gramaticais de cada documento. Salve o corpus POS Tagged em uma planilha ou texto para uso posterior. É importante manter a informação sobre o documento origem dos novos documentos;
4. Obtenha a lista de entidades em cada documento, salvando para uso posterior;
5. Analisando os documentos marcados (tagged) tanto POS quanto NER, quais são as classes mais utilizadas?
6. Observe que há uma tendência de que termos menos relevantes para a análise sejam mais frequentes. Então, repita os procedimentos anteriores, mas com os termos que são relevantes para uma análise do que está sendo falado (trata-se de uma análise preliminar e ainda superficial do discurso)
7. Determine o vocabulário comum entre os textos
8. Determine a frequência de termos comuns nos dois textos, separadamente
9. Determine a frequência de termos comuns utilizados pelo autor considerando os dois textos.
10. Desafio: Quais são as entidades mais citadas pelo autor?