**ESPECIALIZAÇÃO DATA SCIENCE E BIG DATA**

**ANÁLISE DE TEXTOS COM R E PYTHON**

**Professor Leandro ESCOBAR**

**PARTE 1 – Pré processamento e análise de frequência**

A análise de textos é realizado mediante a aplicação de métodos NLP – Natural Language Processing, que consiste em tratar textos ou fala de maneira a obter informações sobre o texto e sobre o significado e intenções nele expressas.

# NLP = Natural Language Processing

* Uso de recursos computacionais para processar e classificar o que ocorre dentro de textos escritos ou falados;
* Orientado ao idioma;
* Requer grandes volumes de dados, considerada a riqueza da linguagem escrita ou falada.

## Conceitos iniciais

* **Corpora** = coleção de corpus
* **Corpus** = coleção de documento
* **Documento** = coleção de sentenças
* **Sentença** = frase que encerra um pensamento
* **Token** = menor partícula de uma sentença (palavra, numeral ou pontuação)

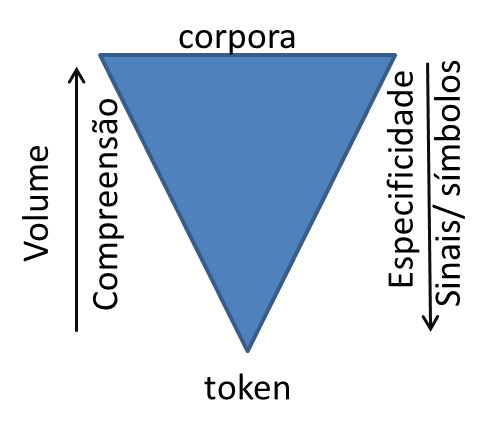


Figura 1 Volume e especificidade em textos

## Operações clássicas NLP

* **Análise quantitativa**
* **Marcação** (tagging)
  + NER: Named Entity Recognition: Identifica as coisas expressas no texto;
  + POS: Part of Speach: Identificar os elementos sintáticos;
  + Polarity: Análise de sentimentos;
* **Relationship**: relação entre os termos
* **Sumarização**: Resumir textos
* **Busca**: Localizar documentos conforme chave e relevância
* **Compreensão**: Assistentes virtuais

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

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\_NLP1

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 nlp1 (a IDE atribuirá a extensão .py ao arquivo)

*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.*

**from** nltk.corpus **import** reuters *#reuters é um corpus previamente montado  
#Carrega os documento para dentro do objeto files*files = reuters.fileids()  
print(files) *#imprime o conteúdo do objeto files no terminal  
  
#Carregar um dos documentos*doc16097 = reuters.words([**'test/16097'**])  
print(doc16097)  
  
*#Carregar um número específico de palavras*doc\_20\_palavras = reuters.words([**'test/16097'**])[:20]  
*#Também é possivel acessar as palavras no documento extraído*doc\_20\_palavras =doc16097[:20]  
print(doc\_20\_palavras)  
  
*#O corpus reuters está organizado em 90 tópicos. Cada tópico possui vários arquivos relacionados*topicos\_reuters = reuters.categories()  
print(topicos\_reuters)  
  
*#É possível acessar palavras associadas a um determinado tópico*

*#Copie o resultado impresso no terminal ou no notebook e cole em um processador de textos (Word ou similar, por exemplo)***for** palavra **in** reuters.words(categories = **'coffee'**):  
 print(palavra + **' '**, end=**''**)  
  
*#É possível acessar palavras associadas a vários tópicos***for** palavra **in** reuters.words(categories = [**'coffee'**,**'cocoa'**]):  
 print(palavra + **' '**, end=**''**)

## 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*.

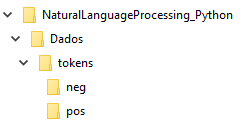


Figura 2 Estrutura de pastas categorizando documentos

## 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 10 minutos**

Imprima as palavras dos documentos **neg/cv002\_tok-3321.txt** e **pos/cv003\_tok-8338.txt**

# CONTANDO PALAVRAS

## Distribuição de frequência

Trata-se da contagem da frequência de termos em documentos ou corpus

O corpus Brown, dentro do NLTK, foi organizado pela *Brown University* e contem 500 documentos distribuídos em 15 categorias.

Crie um novo programa Py com o nome contar\_brown.py

1. Importe a biblioteca NLTK
2. Importe e corpus brown

**import** nltk  
**from** nltk.corpus **import** brown *#Importa o corpus Brown*print(brown.categories())*#Mostra todas as categorias no corpus  
  
#Selecionar três categorias livremente*categorias = [**'fiction'**,**'humor'**,**'romance'**]  
  
*#montar uma lista de palavras para a contagem.  
#neste caso, serão palavras da lingua inglesa que possuam 'wh'*wh\_words = [**'what'**,**'whitch'**,**'how'**, **'why'**,**'when'**,**'where'**,**'who'**]  
  
*#percorre a lista de categorias***for** i **in** range(0, len(categorias)):  
 categoria = categorias[i]*#captura a categoria relativa ao índice i* print()  
 print(**'Analisando palavras na categoria'**, categoria, **''**)  
 cat\_texto = brown.words(categories = categoria)*#Coleta as palavras da categoria capturada: categoria[i]* print(cat\_texto)  
 fdist = nltk.FreqDist(cat\_texto) *#o objeto fdist tem a frequencia de cada palavra dentro da lista cat\_texto  
  
 #agora, basta percorrer o objeto fdist e imprimir o resultado relatico a cada palavra na lista wh\_words* **for** p **in** wh\_words:  
 print(**'Freq palavra: '**, p, **':'**, fdist[p], **'; '**, end=**''**)  
 print(**'Total de palavras:'**, fdist.N())  
 print()

## Análise de frequência no corpus WebText

Analisar a distribuição de frequência no arquivo de publicidade pessoal dentro de nltk.corpus.webtext.

Explorar as funcionalidades do objeto nltk.FreqDist, como a contagem de palavras distintas, 10 palavras mais comuns, palavras de frequência máxima, gráfico de distribuição de frequência e tabulação.

Crie um novo programa Py com o nome analisar\_webtext.py

1. Importe a biblioteca NLTK
2. Importe e corpus webtext

**import** nltk  
**from** nltk.corpus **import** webtext  
  
print(webtext.fileids())  
arquivo = **'singles.txt'**palavras = webtext.words(arquivo) *#coleta as palavras do arquivo 'singles.txt'*fdist = nltk.FreqDist(palavras) *#cria o objeto com a distribuição de frequencia*print(**'Maior frequencia no arquivo'**,fdist.max(),**': '**,fdist[fdist.max()])

print(**'Quantidade de palavras distintas no arquivo:'**, fdist.N())  
print(**'Top 10:'**, fdist.most\_common(10)) *#10 palavras mais frequentes*print(**'Distribuição de frequencia no arquivo Singles.txt'**)  
print(fdist.tabulate())*#Gera uma tabela de dados. Palavras na primeira linha, frequencias na segunda linha  
  
#Plotar gráfico*fdist.plot(cumulative = **True**)*#Gera um gráfico de frequencia*

## n-Gramas: Conjuntos de palavras que ocorrem juntas

Os n-gramas permitem análises fundamentais, demonstrando pares, trios, quadras (ou outros números) de palavras vizinhas.

**from** nltk **import** ngrams  
  
*#extrair o trigramas (3 termos) do texto***for** ng **in** ngrams(palavras,3):  
 print(ng)  
  
*#extrair os n-gramas que contenham uma determinada palavra*palavra\_alvo =**'lady'**n\_gramas\_filtrado =[]  
**for** ng **in** ngrams(palavras,3):  
 **if** palavra\_alvo **in** ng:  
 n\_gramas\_filtrado.append(ng)  
  
print(**'Lista filtrada'**)  
print(n\_gramas\_filtrado)

*#Calcula a frequência dos trigramas*fd\_ngramas = nltk.FreqDist(n\_gramas\_filtrado)  
print(**'10 Trigramas mais frequentes'**)  
print(fd\_ngramas.most\_common(10))

## EXERCITANDO 2

**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 liste os 20 bigramas e trigramas mais frequentes obtidos do texto.

## Limpando o texto – Stopwords

Uma estratégia simples porém importante é a remoção de termos indesejados (Stopwords), uma vez que podem poluir a análise ou são irrelevantes.

É comum removerem-se pontuações, artigos e preposições.

Atenção: stopwords não são um recurso para uso indiscriminado, pois há análises que requerem o texto completo, por exemplo, análise de relacionamentos, em que o “e” e o “ou” alteram o sentido das frases.

O NLTK possui um corpus de Stopwords.

**from** nltk.corpus **import** stopwords *#importa o corpus de stopwords*print(stopwords.fileids())*#mostra as categorias do corpus*stopwords\_pt = stopwords.words(**'portuguese'**)*#carrega as stopwords da categoria 'portuguese'*print(**'Stopwords em português:'**)  
**for** p **in** stopwords\_pt:  
 print(p)  
  
*#limpar um texto*texto = **'Tem um armário no meio da sala, mas deveria ser um sofá.'***#aceitam-se as palavras do texto que não estão nas stopwords.  
#montando a lista de palavras filtradas*texto\_limpo = []  
**for** p **in** texto:  
 **if** p **not in** stopwords\_pt:  
 texto\_limpo.append(p)  
print(**'Texto limpo:'**, texto\_limpo)  
*#Surpreza!! O python processou letra a letra  
#Porque o texto não é uma lista de palavras, mas de caracteres.  
#É necessário segmentar o texto original em palavras = tokens  
# texto\_limpo = [w for w in word\_tokens if not w in stop\_words]***from** nltk **import** tokenize  
tokens = tokenize.word\_tokenize(texto)  
print(**'Após tokenizar: '**)  
print(tokens)  
*#para percorrer a lista de tokens***for** p **in** tokens:  
 print(p)  
  
*#Agora, a limpeza pode ser realizada*print(**'Limpando o texto'**)  
**for** p **in** tokens:  
 **if** p **not in** stopwords\_pt:  
 texto\_limpo.append(p)  
**for** p **in** texto\_limpo:  
 print(p)  
  
*#Há uma maneira mais curta para esse código, porém, menos legível*

*#(não recomendo para programadores inciantes*filtrado = [p **for** p **in** tokens **if not** p **in** stopwords\_pt]  
**for** p **in** filtrado:  
 print(p)  
  
*#Mas ainda há caracteres indesejados [',','.'] => vírgula e ponto final  
#Stopwords personalizadas*minha\_stopwords = [**','**,**'.'**]  
*#o procedimento é o mesmo***for** p **in** tokens:  
 **if** (p **not in** stopwords\_pt) **and** (p **not in** minha\_stopwords):  
 texto\_limpo.append(p)  
  
print(**'Retirando stopwords personalizadas'**)  
**for** p **in** texto\_limpo:  
 print(p)

## EXERCITANDO 3

**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 10 minutos**

Analise a frequência das palavras [‘the’,’that’] no arquivo singles.txt e,depois, no arquivo pirates.txt.

Inclua a geração do gráfico de frequência

Gere a lista dos 15 bigramas mais frequentes do texto

Gere a lista dos 20 quadrigramas gramas mais frequentes que possuam a palavra ‘life’

## EXERCITANDO 4

**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 25 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 o corpus machado, elabore um programa que atenda aos requisitos:
3. Quais são as categorias presentes no corpus?
4. Quais são os documentos dentro desse corpus?
5. Imprima o conteúdo do arquivo do documento que contem a obra Memórias Postumas de Braz Cubas
6. Analise a frequência das palavras [‘olhos’,’estado’] em Memórias Postumas de Bras Cubas
7. Quantas palavras há no texto? Use len(texto)
8. Quantas palavras distintas há na obra?
9. Qual é o vocabulário (palavras) presentes na obra?
10. Quais são os 15 termos mais repetidos no texto de Machado de Assis?
11. Tabular a frequência de palavras
12. Gerar um gráfico com os 15 termos mais repetidos
13. Remova os termos indesejados e repita as questões ‘h’ a ‘j’
14. Obter a lista de todos os trigramas do texto
15. Obter a lista dos 15 bigramas que contenham a palavra ‘olhos’
16. Gerar o gráfico dos bigramas com a palavra ‘olhos’

Inclua a geração do gráfico de frequência