Processamento de Linguagem Natural

Aula 02

Normalização de Textos

PUC Minas

Agenda

- Biblioteca NLTK
- Préprocessamento de texto
- Tokenização
- Stemming
- Lemmatização

Python e PLN

- Até o presente momento, trabalhamos com ideias para o pré-processamento de textos utilizando apenas conceitos e estruturas básicas de Python.
- Problemas práticos é adequado utilizar bibliotecas que ofereçam recursos avançados de PLN.
- Uma das vantagens do Python é a existência de diversas dessas bibliotecas

Python e PLN

- SpaCy
  - Processamento avançado de linguagem natural
- Gensim
  - Modelagem de tópico não supervisionada e processamento de linguagem natural, usando o aprendizado de máquina estatístico
- NLPNET
  - Biblioteca para tarefas de PLN baseadas em redes neurais
- NLTK
  - Conjunto de bibliotecas e programas para PLN simbólica e estatística para o inglês escrito na linguagem de programação Python

Utilizaremos o NLTK, graças à facilidade de uso e a grande quantidade de ferramentas disponíveis

NLTK

- **Natural Language Toolkit**
  - Criada originalmente em 2001 como parte de um curso de linguística computacional do Departamento de Ciência da Computação e Informação da Universidade da Pensilvânia.
  - Tinha 3 aplicações pedagógicas:
    - Experimentos, demonstrações e projetos.
  - NLTK é uma plataforma usada para construir programas Python que trabalham com dados de linguagem humana para aplicação em PLN.

NLTK

- **Natural Language Toolkit**
  - O NLTK define uma infraestrutura que pode ser usada para construir programas de PLN em Python
    - Classes básicas para representar dados relevantes para o processamento da linguagem natural;
    - Interfaces padrão para executar tarefas como tokenização, Part-Of-Speech, análise sintática e classificação de texto
    - Implementações padrões para cada tarefa que podem ser combinadas para resolver problemas complexos.

Córpus NLTK

- A utilização de corpus sempre foi um recurso empregado em pesquisas lingüísticas
- É considerado o conjunto de enunciados a partir do qual se estabelece a gramática descritiva de uma língua.
- NLTK inclui uma pequena seleção de textos do Projeto Gutenberg
  - Arquivo eletrônico textual que contém por volta de 25 mil livros eletrônicos gratuitos
  - http://www.gutenberg.org/

```
>>> emma = nltk.corpus.gutenberg.words('austen-emma.txt')
>>> len(emma)
192427
```

```
>>> from nltk.corpus import gutenberg
>>> gutenberg.fileids()
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', ...]
>>> emma = gutenberg.words('austen-emma.txt')
```

PUC Minas

Córpus NLTK Português

- A biblioteca tem uma série de textos em português
  - http://www.nltk.org/howto/portuguese_en.html
- Um exemplo é o *córpus* **machado** que inclui uma série de textos do autor Machado de Assis

```
>>> from nltk.corpus import machado
>>> machado.fileids() ['contos/macn001.txt', 'contos/macn002.txt',
'contos/macn003.txt', 'contos/macn004.txt', ...]
```

A lista completa do *córpus* pode ser lida em *machado.readme()*

PUC Minas

Análise Textual e PLN

- Comunicação textual é a forma mais popular de conversa atualmente
  - Mensagens, Whatsapp, Tweet, Emails, Blogs
- Grande quantidade de texto gerado -> Não estruturado
- PLN
  - Identificação de Sentimentos
  - Busca de entidades de uma frase
  - Categorizar um blog ou artigo

Importância do Pré Processamento

- O PLN e toda a sua “mágica” não acontece sem trabalho nos bastidores
- Transformação do texto em algo que um algoritmo pode tratar é um processo complicado
- O processo pode ser dividido em quatro partes
  - Limpeza do texto
  - Anotação
  - Normalização da informação
  - Análise

Pré Processamento - Limpeza

- A limpeza do texto consiste no processo de remoção de partes textuais que não serão úteis no processo como um todo
  - Remoção de *StopWords*, palavras especiais
  - Tratamento de letras maiúsculas/minúsculas
  - Tratamento de caracteres especiais (acentos, cedilhas)
  - Remoção de espaços em branco
  - Remoção de pontuações

Pré Processamento - Anotação

- Consiste da aplicação de esquemas de texto
- Podem incluir marcações estruturais e de *tagueamento* de partes da fala

Pré Processamento - Normalização

- Consiste na tradução e mapeamento de termos e reduções linguísticas
  - *Stemming*
  - *Lemmatization*
  - Outras formas de padronização

Pré Processamento - Análise

- Consiste na sondagem estatística, manipulação e generalização de informações a partir do dataset gerado para análise de dados

Pré Processamento

- Grande variedade de métodos de pré-processamento
- As citadas anteriormente não são exclusivas, mas são um ponto de partida
- O pré-processamento e a transformação dos dados para um formato melhor para o PLN reduz e generaliza parte dos dados, gerando a consequência de perder alguma fidelidade com os dados reais ao longo do caminho
- Sempre escolher bem os métodos a serem utilizados em cada caso levando em conta o seu **pró** e o seu **contra**

Capitalização

- Texto geralmente tem uma variação de letras maiúsculas/minúsculas refletindo como começo de sentença, nomes próprios, siglas.
- Comum reduzir tudo para letras minúsculas para simplificar a tarefa do processamento
- Vocabulário é reduzido, mas alguns significados podem ser perdidos
    - USA ≠ usa, Apple ≠ apple, siglas, nomes próprios

Capitalização

- Texto geralmente tem uma variação de letras maiúsculas/minúsculas refletindo como começo de sentença, nomes próprios, siglas.
- Comum reduzir tudo para letras minúsculas para simplificar a tarefa do processamento
- Vocabulário é reduzido, mas alguns significados podem ser perdidos
  - USA ≠ usa, Apple ≠ apple, siglas, nomes próprios

```
In [1]: input_str = "Os 5 maiores países em população em 2017 eram China, Índia, Estados Unidos, Indonésia, e Brasil."
        input_str = input_str.lower()
        print(input_str)

        os 5 maiores países em população em 2017 eram china, índia, estados unidos, indonésia, e brasil.
```

Remoção de Números

- Se números não são relevantes para sua análise, remôva-os
- Geralmente são utilizadas expressões regulares para a remoção

```
In [5]: import re
        input_str = "Caixa A contém 3 bolas vermelhas e 5 bolas brancas, enquanto a Caixa B contém 4 bolas vermelhas e 2 azuis."
        result = re.sub(r"\d+", "", input_str)
        print(result)

Caixa A contém  bolas vermelhas e  bolas brancas, enquanto a Caixa B contém  bolas vermelhas e  azuis.
```

PUC Minas

Remoção de Pontuação

- A biblioteca abaixo remove os caracteres
  - [!”#$%&’()*+,-./:;<=>?@[\]^_`{|}~]:

In [10]:
```
import string
input_str = "Esse é& um exemplo [de] texto {com} várias?.. pontuações!!!!"
result = input_str.translate(str.maketrans('', '', string.punctuation))
print(result)
```

```
Esse é um exemplo de texto com várias pontuações
```

PUC Minas

Remoção de Espaços em Branco

- Para remover espaços em branco no início ou no final, utilize a função *strip()*

```
In [12]: input_str = " \t a string example\t "
         input_str = input_str.strip()
         input_str

Out[12]: 'a string example'
```

PUC Minas

Tokenização

- É o primeiro passo na análise de texto
- O processo de quebra de um texto em partes menores (*chunks)*, que podem ser palavras ou frases é chamado de ***Tokenização***
- *Token* é uma entidade única que é parte construtora de sentenças ou parágrafos
- Tokenizador de Sentenças
  - Quebra o texto em frases
- Tokenizador de Palavras
  - Quebra parágrafos em palavras

Tokenização

In [15]:
```
from nltk.tokenize import sent_tokenize
text="O livro, redigido em italiano, foi publicado em três partes. A primeira delas foi divulgada em 1317, a segunda em 1319 e a t
tokenized_text=sent_tokenize(text)
print(tokenized_text)
```

```
['O livro, redigido em italiano, foi publicado em três partes.', 'A primeira delas foi divulgada em 1317, a segunda em 1319 e
a terceira após a morte do autor.', 'Estima-se que Dante tenha dedicado catorze anos da sua vida a composição do livro (inicio
u em 1307 e concluiu o trabalho pouco antes de sua morte, em 1321).']
```

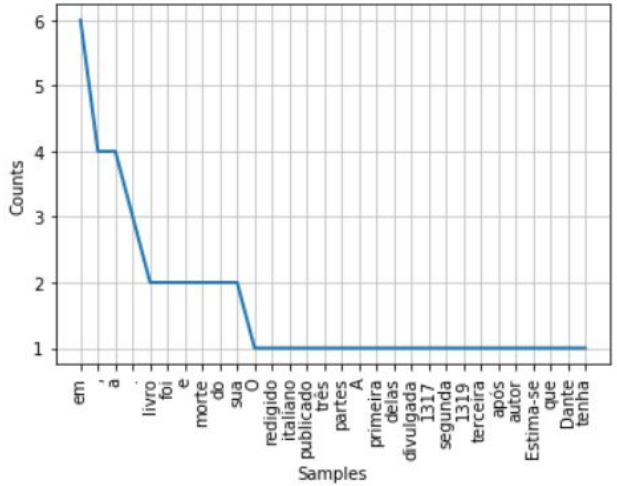
In [16]:
```
from nltk.tokenize import word_tokenize
text="O livro, redigido em italiano, foi publicado em três partes. A primeira delas foi divulgada em 1317, a segunda em 1319 e a t
tokenized_word=word_tokenize(text)
print(tokenized_word)
```

```
['O', 'livro', ',', 'redigido', 'em', 'italiano', ',', 'foi', 'publicado', 'em', 'três', 'partes', '.', 'A', 'primeira', 'dela
s', 'foi', 'divulgada', 'em', '1317', ',', 'a', 'segunda', 'em', '1319', 'e', 'a', 'terceira', 'após', 'a', 'morte', 'do', 'au
tor', '.', 'Estima-se', 'que', 'Dante', 'tenha', 'dedicado', 'catorze', 'anos', 'da', 'sua', 'vida', 'a', 'composição', 'do',
'livro', '(', 'iniciou', 'em', '1307', 'e', 'concluiu', 'o', 'trabalho', 'pouco', 'antes', 'de', 'sua', 'morte', ',', 'em', '1
321', ')', '.']
```

In [22]:
```
fdist = FreqDist(tokenized_word)
print(fdist)
```

```
<FreqDist with 47 samples and 66 outcomes>
```

In [23]:
```
import matplotlib.pyplot as plt
fdist.plot(30,cumulative=False)
plt.show()
```



PUC Minas

Stopwords

- *Stopwords* são consideradas ruídos no texto. Os textos vão conter palavras como **em**, **o, a**.
- A remoção de *stopwords* com a biblioteca NLTK é feita a partir de uma lista de *tokens* com essas palavras

Stopwords

In [33]:
```
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
from nltk.corpus import stopwords

stop_words=set(stopwords.words("portuguese"))

text="O livro, redigido em italiano, foi publicado em três partes. A primeira delas foi divulgada em 1317, a segunda em 1319 e a t
tokenized_word=word_tokenize(text)
print(tokenized_word)

filtered_sent=[]
for w in tokenized_word:
    if w not in stop_words:
        filtered_sent.append(w)
print("Tokenized Sentence:",tokenized_word)
print("Filterd Sentence:",filtered_sent)
```

```
['O', 'livro', ',', 'redigido', 'em', 'italiano', ',', 'foi', 'publicado', 'em', 'três', 'partes', '.', 'A', 'primeira', 'dela
s', 'foi', 'divulgada', 'em', '1317', ',', 'a', 'segunda', 'em', '1319', 'e', 'a', 'terceira', 'após', 'a', 'morte', 'do', 'au
tor', '.', 'Estima-se', 'que', 'Dante', 'tenha', 'dedicado', 'catorze', 'anos', 'da', 'sua', 'vida', 'a', 'composição', 'do',
'livro', '(', 'iniciou', 'em', '1307', 'e', 'concluiu', 'o', 'trabalho', 'pouco', 'antes', 'de', 'sua', 'morte', ',', 'em', '1
321', ')', '.']
Tokenized Sentence: ['O', 'livro', ',', 'redigido', 'em', 'italiano', ',', 'foi', 'publicado', 'em', 'três', 'partes', '.',
'A', 'primeira', 'delas', 'foi', 'divulgada', 'em', '1317', ',', 'a', 'segunda', 'em', '1319', 'e', 'a', 'terceira', 'após',
'a', 'morte', 'do', 'autor', '.', 'Estima-se', 'que', 'Dante', 'tenha', 'dedicado', 'catorze', 'anos', 'da', 'sua', 'vida',
'a', 'composição', 'do', 'livro', '(', 'iniciou', 'em', '1307', 'e', 'concluiu', 'o', 'trabalho', 'pouco', 'antes', 'de', 'su
a', 'morte', ',', 'em', '1321', ')', '.']
Filterd Sentence: ['O', 'livro', ',', 'redigido', 'italiano', ',', 'publicado', 'três', 'partes', '.', 'A', 'primeira', 'divul
gada', '1317', ',', 'segunda', '1319', 'terceira', 'após', 'morte', 'autor', '.', 'Estima-se', 'Dante', 'dedicado', 'catorze',
'anos', 'vida', 'composição', 'livro', '(', 'iniciou', '1307', 'concluiu', 'trabalho', 'pouco', 'antes', 'morte', ',', '1321',
')', '.']
```

PUC Minas

Normalização Léxica

- A normalização lexical considera outro tipo de ruído no texto
- Por exemplo, as palavras conectado, conectados seriam reduzidas a uma palavra comum “conect”
- Assim reduzimos as formas derivadas de uma palavra para uma palavra raiz comum
- Temos dois processos para fazer essa normalização
  - Stemming
  - Lematização

Stemming

- É um processo de normalização linguística que reduz as palavras para uma palavra raiz, cortando os afixos de derivação.

```
In [40]: from nltk.stem import RSLPStemmer
         stemmer = nltk.stem.RSLPStemmer()

         print(stemmer.stem("conectado"))
         print(stemmer.stem("conectar"))
         print(stemmer.stem("conectando"))

            conect
            conect
            conect
```

- O RSLP Stemmer faz parte da biblioteca NLTK e faz o algoritmo para a língua portuguesa.

Lematização

- A lematização reduz as palavras para sua forma básica, que é o seu *lemma* linguisticamente correto. Transforma a palavra raiz com o uso de vocabulário e análise morfológica
- Lematização é geralmente um processo mais sofisticado que o *stemming*. Stemmer trabalha de forma individual nas palavras sem conhecimento do contexto

Stemmers vs. Lemmatizers

- Ambos tiram as palavras de suas formas derivadas para uma mesma forma base
- Stemmers usam uma abordagem algorítmica para remover os prefixos e sufixos. **O resultado pode não ser uma palavra real**
- Stemmers são mais rápidos do que lemmatizers.

Stemmers vs. Lemmatizers

- Lemmatizers usam uma base de palavras. **O resultado é sempre uma palavra real encontrada no dicionário.**
- Lemmatizers precisam de informação extra sobre a parte do texto que estão processando. “Para” pode ser um verbo ou uma preposição

Quando usar Stemmer x Lemmatizer

- Quando velocidade for importante, use Stemmers - Lemmatizers fazem uma busca em todo o seu grupo de palavras, enquanto os stemmers realizam simples operações de texto.
- Se você só quer garantir que o seu sistema é tolerante a variações de palavras, use *Stemmers.*
- Se você precisa de palavras existentes num dicionário, use um *Lemmatizer.* Por exemplo, se você estiver construindo um sistema de geração de linguagem natural)