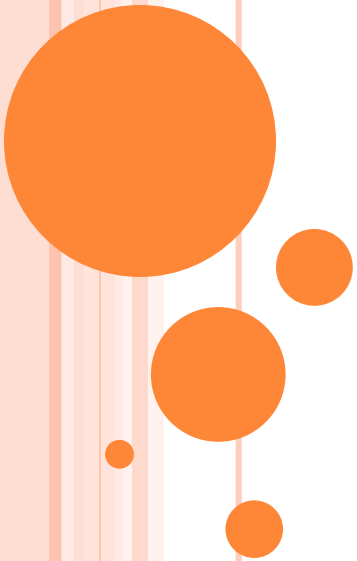
# Python para PLN

**NLTK e spaCy**

**Roney Lira de Sales Santos** roneysantos@usp.br
Prof. Thiago A. S. Pardo



# NLTK

# NLTK

- Biblioteca de ferramentas úteis para a utilização dos princípios de PLN
- Linguagem Python
- Funcionalidades para manipulação de strings
- Interfaces padrões para realizar tarefas como etiquetar textos, frequência de palavras, lematização e stemização de palavras, entre vários outros.
- LIVRO ONLINE GRATUITO! http://www.nltk.org/book/

# NLTK - Instalação

- Requer pelo menos a versão 3.5 do Python

- Linux, MacOS e Windows 32-bit
  - Para instalação no Windows 64-bit, seguir esse tutorial.

- Deu certo? Testar com o comando **`import nltk`**.

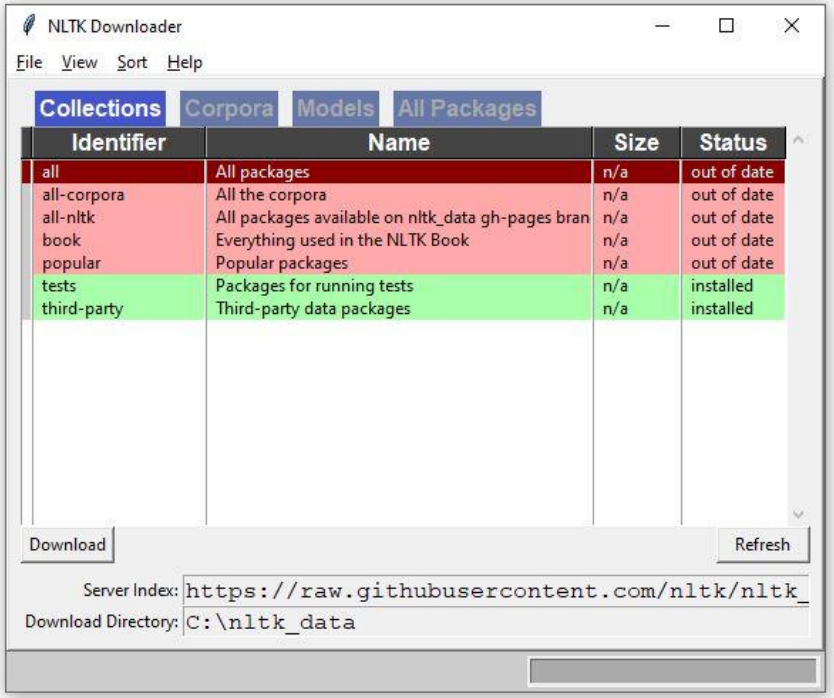
```
>>> import nltk
>>>
```

- Após a instalação do *toolkit*, instalar os dados necessários para o funcionamento.
  - NLTK data

# NLTK - Instalação

- Verificar quais pacotes estão desatualizados e clicar em **Download**.
  - Selecionar o identificador **all** e clicar em **Download**.

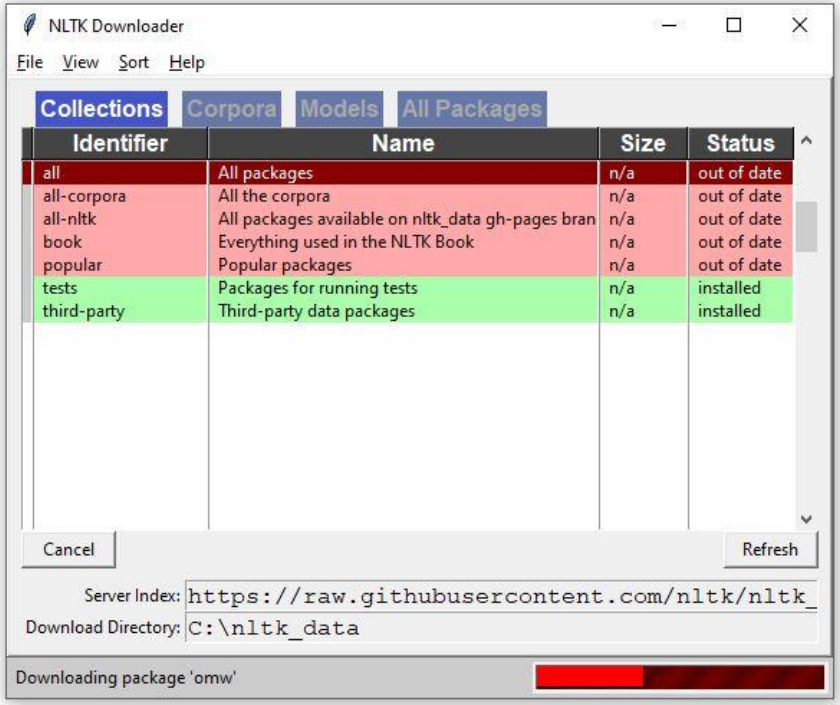
```
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```



# NLTK - Instalação

- A barra de progresso abaixo da janela mostra o andamento do processo.

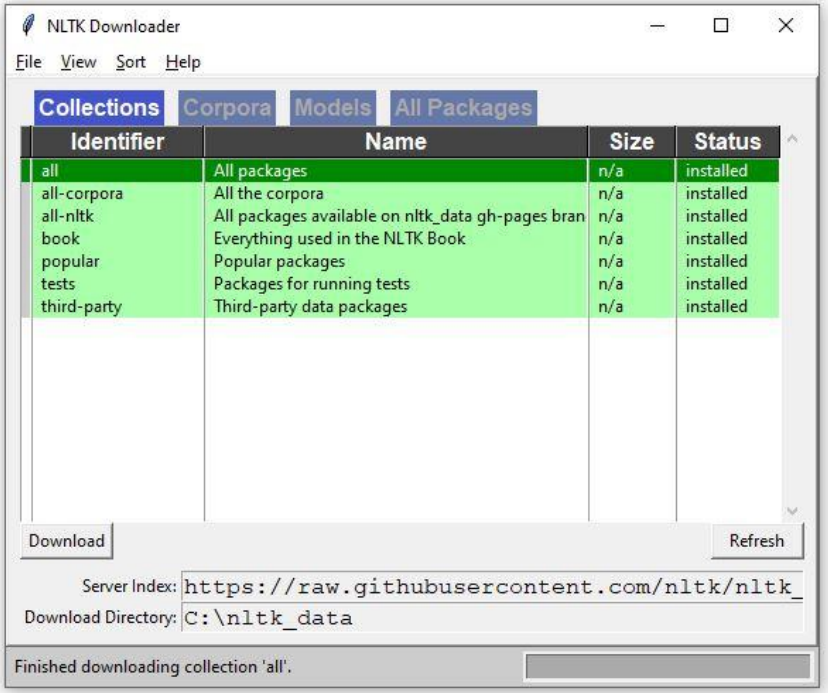
```
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```



# NLTK - Instalação

- A atualização demora por volta de **1 minuto e meio**. Quando estiver tudo certo, todos os pacotes estarão com a cor verde.

```
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```



# NLTK - Uso

- Dentro do NLTK contém **vários corpora**
  - Úteis para os etiquetadores, entidades nomeadas, estruturas sintáticas e várias outras funcionalidades.

| Corpus | Compiler | Contents |
|---|---|---|
| Brown Corpus | Francis, Kucera | 15 genres, 1.15M words, tagged, categorized |
| CESS Treebanks | CLiC-UB | 1M words, tagged and parsed (Catalan, Spanish) |
| Chat-80 Data Files | Pereira & Warren | World Geographic Database |
| CMU Pronouncing Dictionary | CMU | 127k entries |
| CoNLL 2000 Chunking Data | CoNLL | 270k words, tagged and chunked |
| CoNLL 2002 Named Entity | CoNLL | 700k words, pos- and named-entity-tagged (Dutch, Spanish) |
| CoNLL 2007 Dependency Treebanks (sel) | CoNLL | 150k words, dependency parsed (Basque, Catalan) |
| Dependency Treebank | Narad | Dependency parsed version of Penn Treebank sample |
| FrameNet | Fillmore, Baker et al | 10k word senses, 170k manually annotated sentences |
| Floresta Treebank | Diana Santos et al | 9k sentences, tagged and parsed (Portuguese) |
| Gazetteer Lists | Various | Lists of cities and countries |
| Genesis Corpus | Misc web sources | 6 texts, 200k words, 6 languages |
| Gutenberg (selections) | Hart, Newby, et al | 18 texts, 2M words |
| Inaugural Address Corpus | CSpan | US Presidential Inaugural Addresses (1789-present) |
| Indian POS-Tagged Corpus | Kumaran et al | 60k words, tagged (Bangla, Hindi, Marathi, Telugu) |
| MacMorpho Corpus | NILC, USP, Brazil | 1M words, tagged (Brazilian Portuguese) |
| Movie Reviews | Pang, Lee | 2k movie reviews with sentiment polarity classification |
| Names Corpus | Kantrowitz, Ross | 8k male and female names |
| NIST 1999 Info Extr (selections) | Garofolo | 63k words, newswire and named-entity SGML markup |
| Nombank | Meyers | 115k propositions, 1400 noun frames |
| NPS Chat Corpus | Forsyth, Martell | 10k IM chat posts, POS-tagged and dialogue-act tagged |



Disponível em https://www.nltk.org/book/ch02.html

# NLTK - Uso

- Dentro do NLTK contém **vários corpora**
  - Úteis para os etiquetadores, entidades nomeadas, estruturas sintáticas e várias outras funcionalidades.

```
>>> nltk.corpus.mac_morpho.words()
['Jersei', 'atinge', 'média', 'de', 'Cr$', '1,4', ...]
>>> nltk.corpus.mac_morpho.sents()
[['Jersei', 'atinge', 'média', 'de', 'Cr$', '1,4', 'milhão', 'em', 'a', 'venda', '
de', 'a', 'Pinhal', 'em', 'São', 'Paulo'], ['Programe', 'sua', 'viagem', 'a', 'a',
'Exposição', 'Nacional', 'do', 'Zebu', ',', 'que', 'começa', 'dia', '25'], ...]
>>> nltk.corpus.mac_morpho.tagged_words()
[('Jersei', 'N'), ('atinge', 'V'), ('média', 'N'), ...]
>>> nltk.corpus.mac_morpho.tagged_sents()
[[('Jersei', 'N'), ('atinge', 'V'), ('média', 'N'), ('de', 'PREP'), ('Cr$', 'CUR')
, ('1,4', 'NUM'), ('milhão', 'N'), ('em', 'PREP|+'), ('a', 'ART'), ('venda', 'N'),
('de', 'PREP|+'), ('a', 'ART'), ('Pinhal', 'NPROP'), ('em', 'PREP'), ('São', 'NPRO
P'), ('Paulo', 'NPROP')], [('Programe', 'V'), ('sua', 'PROADJ'), ('viagem', 'N'),
('a', 'PREP|+'), ('a', 'ART'), ('Exposição', 'NPROP'), ('Nacional', 'NPROP'), ('do
', 'NPROP'), ('Zebu', 'NPROP'), (',', ','), ('que', 'PRO-KS-REL'), ('começa', 'V')
, ('dia', 'N'), ('25', 'N|AP')], ...]
```

- Veremos como fazer isso para um texto qualquer!

# NLTK - Uso

- Nessa aula, vamos ver detalhadamente algumas funções importantes no tratamento de textos com NLTK.

  - Tokenização
  - Frequência/Contagem de palavras
  - *Stopwords*
  - N-gramas
  - Stemmer e Lemma
  - Etiquetadores

- Para testar as funções, utilizaremos esse corpus!

# NLTK - Tokenização

- **Tokenizar = separar** as palavras do texto
  - Tipo um **`split`**?

- Nível linguístico lexical: uma palavra, número ou pontuação agora é um **token**.

- Dado o texto que vai ser tokenizado, basta usar a função **`nltk.word_tokenize(texto)`**:

```
>>> import nltk
>>> texto = "O jogador, que está de camisa verde, marcou o gol da vitória!"
>>> nltk.word_tokenize(texto)
['O', 'jogador', ',', 'que', 'está', 'de', 'camisa', 'verde', ',', 'marcou'
, 'o', 'gol', 'da', 'vitória', '!']
```

# NLTK - Tokenização

- O tokenizador do NLTK pode ter algumas variações, como por exemplo, retornar apenas os tokens sem as pontuações:
  - Entramos em um novo mundo chamado **expressões regulares**.

```
>>> from nltk.tokenize import RegexpTokenizer
>>> tokenizer = RegexpTokenizer(r'\w+')
>>> tokens = tokenizer.tokenize(texto)
>>> tokens
['O', 'jogador', 'que', 'está', 'de', 'camisa', 'verde', 'marcou', 'o', 'go
l', 'da', 'vitória']
```

  - E se quiséssemos os tokens sem pontuações e numerais?

```
>>> texto = "O jogador, que está com a camisa 10, marcou o gol da vitória!"
>>> from nltk.tokenize import RegexpTokenizer
>>> tokenizer = RegexpTokenizer(r'[A-z]\w*')
>>> tokens = tokenizer.tokenize(texto)
>>> tokens
['O', 'jogador', 'que', 'está', 'com', 'a', 'camisa', 'marcou', 'o', 'gol',
'da', 'vitória']
```



Mais de expressões regulares aqui nesse link.

# NLTK – Frequência/Contagem

- Com a lista de tokens, é possível fazer a contagem de ocorrência de tokens pelo NLTK.
- Uso da classe **`FreqDist()`**
  - A função **`most_common()`** ordena a frequência dos tokens. Pode ser usado um argumento para informar a quantidade de tokens mais comuns.

```
>>> frequencia = nltk.FreqDist(tokens)
>>> frequencia.most_common()
[(',', 6), ('o', 4), ('a', 3), ('de', 2), ('do', 2), ('New', 2), ('e', 2),
('no', 2), ('Super', 2), ('Bowl', 2), ('.', 2), ('uma', 2), ('da', 2), ('hi
stória', 2), ('Com', 1), ('um', 1), ('passe', 1), ('Eli', 1), ('Manning', 1
), ('para', 1), ('Plaxico', 1), ('Burress', 1), ('39', 1), ('segundos', 1),
('fim', 1), ('York', 1), ('Giants', 1), ('anotou', 1), ('touchdown', 1), ('
decisivo', 1), ('derrubou', 1), ('favorito', 1), ('England', 1), ('Patriots
', 1), ('por', 1), ('17', 1), ('14', 1), ('neste', 1), ('domingo', 1), ('em
', 1), ('Glendale', 1), ('XLII', 1), ('O', 1), ('resultado', 1), ('das', 1)
, ('maiores', 1), ('zebras', 1), ('acabou', 1), ('com', 1), ('temporada', 1
), ('perfeita', 1), ('Tom', 1), ('Brady', 1), ('companhia', 1), ('que', 1),
('esperavam', 1), ('fazer', 1), ('ao', 1), ('levantar', 1), ('troféu', 1),
('NFL', 1), ('sem', 1), ('sofrer', 1), ('derrota', 1), ('ano', 1)]
>>> frequencia.most_common(5)
[(',', 6), ('o', 4), ('a', 3), ('de', 2), ('do', 2)]
```

# NLTK – Frequência/Contagem

- É importante notar que os tokens em maiúsculo e minúsculo são considerados diferentes.
- Portanto, caso o objetivo da contagem seja de palavras iguais, por exemplo, é necessário usar as funções **`lower()`** (ou **`upper()`**) para normalizar os tokens.
  - Um novo princípio: **list comprehesion**

```
>>> frequencia = nltk.FreqDist(w.lower() for w in tokens)
>>> frequencia.most_common()
[(',', 6), ('o', 5), ('a', 3), ('com', 2), ('de', 2), ('do', 2), ('new', 2)
, ('e', 2), ('no', 2), ('super', 2), ('bowl', 2), ('.', 2), ('uma', 2), ('d
a', 2), ('história', 2), ('um', 1), ('passe', 1), ('eli', 1), ('manning', 1
), ('para', 1), ('plaxico', 1), ('burress', 1), ('39', 1), ('segundos', 1),
('fim', 1), ('york', 1), ('giants', 1), ('anotou', 1), ('touchdown', 1), ('
decisivo', 1), ('derrubou', 1), ('favorito', 1), ('england', 1), ('patriots
', 1), ('por', 1), ('17', 1), ('14', 1), ('neste', 1), ('domingo', 1), ('em
', 1), ('glendale', 1), ('xlii', 1), ('resultado', 1), ('das', 1), ('maiore
s', 1), ('zebras', 1), ('acabou', 1), ('temporada', 1), ('perfeita', 1), ('
tom', 1), ('brady', 1), ('companhia', 1), ('que', 1), ('esperavam', 1), ('f
azer', 1), ('ao', 1), ('levantar', 1), ('troféu', 1), ('nfl', 1), ('sem', 1
), ('sofrer', 1), ('derrota', 1), ('ano', 1)]
```

Mais de list comprehension aqui e aqui!

# NLTK - STOPWORDS

- **Stopwords** são palavras que podem ser consideradas irrelevantes para um certo resultado buscado.
  - Artigos, preposições, conjunções, por exemplo...

```
>>> import nltk
>>> nltk.corpus.stopwords.words('portuguese')
['de', 'a', 'o', 'que', 'e', 'é', 'do', 'da', 'em', 'um', 'para', 'com', 'não', 'u
ma', 'os', 'no', 'se', 'na', 'por', 'mais', 'as', 'dos', 'como', 'mas', 'ao', 'ele
', 'das', 'à', 'seu', 'sua', 'ou', 'quando', 'muito', 'nos', 'já', 'eu', 'também',
'só', 'pelo', 'pela', 'até', 'isso', 'ela', 'entre', 'depois', 'sem', 'mesmo', 'ao
s', 'seus', 'quem', 'nas', 'me', 'esse', 'eles', 'você', 'essa', 'num', 'nem', 'su
as', 'meu', 'às', 'minha', 'numa', 'pelos', 'elas', 'qual', 'nós', 'lhe', 'deles',
'essas', 'esses', 'pelas', 'este', 'dele', 'tu', 'te', 'vocês', 'vos', 'lhes', 'me
us', 'minhas', 'teu', 'tua', 'teus', 'tuas', 'nosso', 'nossa', 'nossos', 'nossas',
'dela', 'delas', 'esta', 'estes', 'estas', 'aquele', 'aquela', 'aqueles', 'aquelas
', 'isto', 'aquilo', 'estou', 'está', 'estamos', 'estão', 'estive', 'esteve', 'est
ivemos', 'estiveram', 'estava', 'estávamos', 'estavam', 'estivera', 'estivéramos',
'esteja', 'estejamos', 'estejam', 'estivesse', 'estivéssemos', 'estivessem', 'esti
ver', 'estivermos', 'estiverem', 'hei', 'há', 'havemos', 'hão', 'houve', 'houvemos
', 'houveram', 'houvera', 'houvéramos', 'haja', 'hajamos', 'hajam', 'houvesse', 'h
ouvéssemos', 'houvessem', 'houver', 'houvermos', 'houverem', 'houverei', 'houverá'
, 'houveremos', 'houverão', 'houveria', 'houveríamos', 'houveriam', 'sou', 'somos'
, 'são', 'era', 'éramos', 'eram', 'fui', 'foi', 'fomos', 'foram', 'fora', 'fôramos
', 'seja', 'sejamos', 'sejam', 'fosse', 'fôssemos', 'fossem', 'for', 'formos', 'fo
rem', 'serei', 'será', 'seremos', 'serão', 'seria', 'seríamos', 'seriam', 'tenho',
'tem', 'temos', 'tém', 'tinha', 'tínhamos', 'tinham', 'tive', 'teve', 'tivemos', '
tiveram', 'tivera', 'tivéramos', 'tenha', 'tenhamos', 'tenham', 'tivesse', 'tivéss
emos', 'tivessem', 'tiver', 'tivermos', 'tiverem', 'terei', 'terá', 'teremos', 'te
rão', 'teria', 'teríamos', 'teriam']
```

# NLTK - STOPWORDS

- É possível, então, fazer vários tipos de pré-processamento.
  - Exemplo: Frequência dos tokens sem stopwords

```
>>> tokenizer = RegexpTokenizer(r'[A-z]\w*')
>>> tokens = tokenizer.tokenize(texto)
>>> stopwords = nltk.corpus.stopwords.words('portuguese')
>>> tokens_sem_stopwords = [w.lower() for w in tokens if w not in stopwords]
>>> frequencia = nltk.FreqDist(tokens_sem_stopwords)
>>> print(frequencia.most_common())
[('new', 2), ('super', 2), ('bowl', 2), ('história', 2), ('com', 1), ('passe',
1), ('eli', 1), ('manning', 1), ('plaxico', 1), ('burress', 1), ('segundos', 1)
, ('fim', 1), ('york', 1), ('giants', 1), ('anotou', 1), ('touchdown', 1), ('de
cisivo', 1), ('derrubou', 1), ('favorito', 1), ('england', 1), ('patriots', 1),
('neste', 1), ('domingo', 1), ('glendale', 1), ('xlii', 1), ('o', 1), ('resulta
do', 1), ('maiores', 1), ('zebras', 1), ('acabou', 1), ('temporada', 1), ('perf
eita', 1), ('tom', 1), ('brady', 1), ('companhia', 1), ('esperavam', 1), ('faze
r', 1), ('levantar', 1), ('troféu', 1), ('nfl', 1), ('sofrer', 1), ('derrota',
1), ('ano', 1)]
```

# NLTK – N-Gramas

- Com a lista de tokens, é possível ter os n-gramas necessários para qualquer análise.
  - Por exemplo: predição da **próxima palavra** de uma sentença em smartphones.

- Bigramas: **`from nltk import bigrams`**
- Trigramas: **`from nltk import trigrams`**
- 4-gram ou mais: **`from nltk import ngrams`**

  - Um 'novo' conceito: importando módulos.

# NLTK – N-GRAMAS

- Bigramas

```
>>> texto = "Com um passe de Eli Manning para Plaxico Burress a 39 segundos do
fim, o New York Giants anotou o touchdown decisivo e derrubou o favorito New En
gland Patriots por 17 a 14 neste domingo, em Glendale, no Super Bowl XLII. O re
sultado, uma das maiores zebras da história do Super Bowl, acabou com a tempora
da perfeita de Tom Brady e companhia, que esperavam fazer história ao levantar
o troféu da NFL sem sofrer uma derrota no ano."
>>> from nltk import bigrams
>>> list(bigrams(tokens))#já com o texto tokenizado
[('Com', 'um'), ('um', 'passe'), ('passe', 'de'), ('de', 'Eli'), ('Eli', 'Manni
ng'), ('Manning', 'para'), ('para', 'Plaxico'), ('Plaxico', 'Burress'), ('Burre
ss', 'a'), ('a', 'segundos'), ('segundos', 'do'), ('do', 'fim'), ('fim', 'o'),
('o', 'New'), ('New', 'York'), ('York', 'Giants'), ('Giants', 'anotou'), ('anot
ou', 'o'), ('o', 'touchdown'), ('touchdown', 'decisivo'), ('decisivo', 'e'), ('
e', 'derrubou'), ('derrubou', 'o'), ('o', 'favorito'), ('favorito', 'New'), ('N
ew', 'England'), ('England', 'Patriots'), ('Patriots', 'por'), ('por', 'a'), ('
a', 'neste'), ('neste', 'domingo'), ('domingo', 'em'), ('em', 'Glendale'), ('Gl
endale', 'no'), ('no', 'Super'), ('Super', 'Bowl'), ('Bowl', 'XLII'), ('XLII',
'O'), ('O', 'resultado'), ('resultado', 'uma'), ('uma', 'das'), ('das', 'maiore
s'), ('maiores', 'zebras'), ('zebras', 'da'), ('da', 'história'), ('história',
'do'), ('do', 'Super'), ('Super', 'Bowl'), ('Bowl', 'acabou'), ('acabou', 'com'
), ('com', 'a'), ('a', 'temporada'), ('temporada', 'perfeita'), ('perfeita', 'd
e'), ('de', 'Tom'), ('Tom', 'Brady'), ('Brady', 'e'), ('e', 'companhia'), ('com
panhia', 'que'), ('que', 'esperavam'), ('esperavam', 'fazer'), ('fazer', 'histó
ria'), ('história', 'ao'), ('ao', 'levantar'), ('levantar', 'o'), ('o', 'troféu
'), ('troféu', 'da'), ('da', 'NFL'), ('NFL', 'sem'), ('sem', 'sofrer'), ('sofre
r', 'uma'), ('uma', 'derrota'), ('derrota', 'no'), ('no', 'ano')]
```

# NLTK – N-GRAMAS

- Trigramas

```
>>> from nltk import trigrams
>>> list(trigrams(tokens))
[('Com', 'um', 'passe'), ('um', 'passe', 'de'), ('passe', 'de', 'Eli'), ('de',
'Eli', 'Manning'), ('Eli', 'Manning', 'para'), ('Manning', 'para', 'Plaxico'),
('para', 'Plaxico', 'Burress'), ('Plaxico', 'Burress', 'a'), ('Burress', 'a', '
segundos'), ('a', 'segundos', 'do'), ('segundos', 'do', 'fim'), ('do', 'fim', '
o'), ('fim', 'o', 'New'), ('o', 'New', 'York'), ('New', 'York', 'Giants'), ('Yo
rk', 'Giants', 'anotou'), ('Giants', 'anotou', 'o'), ('anotou', 'o', 'touchdown
'), ('o', 'touchdown', 'decisivo'), ('touchdown', 'decisivo', 'e'), ('decisivo'
, 'e', 'derrubou'), ('e', 'derrubou', 'o'), ('derrubou', 'o', 'favorito'), ('o'
, 'favorito', 'New'), ('favorito', 'New', 'England'), ('New', 'England', 'Patri
ots'), ('England', 'Patriots', 'por'), ('Patriots', 'por', 'a'), ('por', 'a', '
neste'), ('a', 'neste', 'domingo'), ('neste', 'domingo', 'em'), ('domingo', 'em
', 'Glendale'), ('em', 'Glendale', 'no'), ('Glendale', 'no', 'Super'), ('no', '
Super', 'Bowl'), ('Super', 'Bowl', 'XLII'), ('Bowl', 'XLII', 'O'), ('XLII', 'O'
, 'resultado'), ('O', 'resultado', 'uma'), ('resultado', 'uma', 'das'), ('uma',
'das', 'maiores'), ('das', 'maiores', 'zebras'), ('maiores', 'zebras', 'da'), (
'zebras', 'da', 'história'), ('da', 'história', 'do'), ('história', 'do', 'Supe
r'), ('do', 'Super', 'Bowl'), ('Super', 'Bowl', 'acabou'), ('Bowl', 'acabou', '
com'), ('acabou', 'com', 'a'), ('com', 'a', 'temporada'), ('a', 'temporada', 'p
erfeita'), ('temporada', 'perfeita', 'de'), ('perfeita', 'de', 'Tom'), ('de', '
Tom', 'Brady'), ('Tom', 'Brady', 'e'), ('Brady', 'e', 'companhia'), ('e', 'comp
anhia', 'que'), ('companhia', 'que', 'esperavam'), ('que', 'esperavam', 'fazer'
), ('esperavam', 'fazer', 'história'), ('fazer', 'história', 'ao'), ('história'
, 'ao', 'levantar'), ('ao', 'levantar', 'o'), ('levantar', 'o', 'troféu'), ('o'
, 'troféu', 'da'), ('troféu', 'da', 'NFL'), ('da', 'NFL', 'sem'), ('NFL', 'sem'
, 'sofrer'), ('sem', 'sofrer', 'uma'), ('sofrer', 'uma', 'derrota'), ('uma', 'd
errota', 'no'), ('derrota', 'no', 'ano')]
```

# NLTK – N-GRAMAS

- N-gramas: testando com 4-gram

```
>>> from nltk import ngrams
>>> list(ngrams(tokens, 4))
[('Com', 'um', 'passe', 'de'), ('um', 'passe', 'de', 'Eli'), ('passe', 'de', 'E
li', 'Manning'), ('de', 'Eli', 'Manning', 'para'), ('Eli', 'Manning', 'para', '
Plaxico'), ('Manning', 'para', 'Plaxico', 'Burress'), ('para', 'Plaxico', 'Burr
ess', 'a'), ('Plaxico', 'Burress', 'a', 'segundos'), ('Burress', 'a', 'segundos
', 'do'), ('a', 'segundos', 'do', 'fim'), ('segundos', 'do', 'fim', 'o'), ('do'
, 'fim', 'o', 'New'), ('fim', 'o', 'New', 'York'), ('o', 'New', 'York', 'Giants
'), ('New', 'York', 'Giants', 'anotou'), ('York', 'Giants', 'anotou', 'o'), ('G
iants', 'anotou', 'o', 'touchdown'), ('anotou', 'o', 'touchdown', 'decisivo'),
('o', 'touchdown', 'decisivo', 'e'), ('touchdown', 'decisivo', 'e', 'derrubou')
, ('decisivo', 'e', 'derrubou', 'o'), ('e', 'derrubou', 'o', 'favorito'), ('der
rubou', 'o', 'favorito', 'New'), ('o', 'favorito', 'New', 'England'), ('favorit
o', 'New', 'England', 'Patriots'), ('New', 'England', 'Patriots', 'por'), ('Eng
land', 'Patriots', 'por', 'a'), ('Patriots', 'por', 'a', 'neste'), ('por', 'a',
'neste', 'domingo'), ('a', 'neste', 'domingo', 'em'), ('neste', 'domingo', 'em'
, 'Glendale'), ('domingo', 'em', 'Glendale', 'no'), ('em', 'Glendale', 'no', 'S
uper'), ('Glendale', 'no', 'Super', 'Bowl'), ('no', 'Super', 'Bowl', 'XLII'), (
```

# NLTK – N-GRAMAS

- Os n-gramas são importantes para várias análises. Um exemplo, no nosso caso, seria conseguir as entidades nomeadas do nosso trecho.

```
35  from nltk import bigrams
36  from nltk import trigrams
37
38  bigramas_lista = list(bigrams(tokens))
39  trigramas_lista = list(trigrams(tokens))
40
41  for info in bigramas_lista:
42      if (info[0][0].isupper() and info[1][0].isupper()):
43          print(info)
44
45  for info in trigramas_lista:
46      if (info[0][0].isupper() and info[1][0].isupper() and info[2][0].isupper()):
47          print(info)
```

```
('Eli', 'Manning')
('Plaxico', 'Burress')
('New', 'York')
('York', 'Giants')
('New', 'England')
('England', 'Patriots')
('Super', 'Bowl')
('Bowl', 'XLII')
('XLII', 'O')
('Super', 'Bowl')
('Tom', 'Brady')
('New', 'York', 'Giants')
('New', 'England', 'Patriots')
('Super', 'Bowl', 'XLII')
('Bowl', 'XLII', 'O')
```

# NLTK – Stemmer e Lemmatizer

- STEMMING: consiste em reduzir a palavra ao seu **radical**.
  - amig: amigo, amiga, amigão
  - gat: gato, gata, gatos
  - **prop: propõem, propuseram, propondo**

- LEMATIZAÇÃO: consiste em reduzir a palavra à sua **forma canônica**, levando em conta sua **classe gramatical**.
  - **propor: propõem, propuseram, propondo**
  - estudar: estudando, estudioso, estudei

# NLTK – Stemmer e Lemmatizer

- O NLTK tem implementado vários algoritmos de para stemmer:
  - RSLP
  - Porter
  - ISRI
  - Lancaster
  - Snowball

# NLTK – Stemmer e Lemmatizer

- O NLTK tem implementado várias variantes de stemmers:
  - **RSLP – Removedor de Sufixos da Língua Portuguesa**
    - Porter
    - ISRI
    - Lancaster
    - Snowball

```
>>> import nltk
>>> stemmer = nltk.RSLPStemmer()
>>> stemmer.stem('amigão')
'amig'
>>> stemmer.stem('amigo')
'amig'
>>> stemmer.stem('propuseram')
'propus'
>>> stemmer.stem('propõem')
'propõ'
>>> stemmer.stem('propondo')
'prop'
>>>
```

Mais informações em http://www.inf.ufrgs.br/~viviane/rslp/index.htm

# NLTK – Stemmer e Lemmatizer

- Infelizmente o NLTK ainda **não tem** um lematizador para o Português bom o bastante.
- Tentativa: **WordNet Lemmatizer**
  - Funciona somente para o inglês...
    - Mas fiquem tranquilos, no spaCy tem para o português... =)

```
>>> lemmatizer = nltk.stem.WordNetLemmatizer()
>>> lemmatizer.lemmatize('propõem', pos='v')
'propõem'
>>> lemmatizer.lemmatize('estudei', pos='v')
'estudei'
>>> lemmatizer.lemmatize('propõem', pos='n')
'propõem'
>>> lemmatizer.lemmatize('studied', pos='v')
'study'
>>> lemmatizer.lemmatize('studying', pos='v')
'study'
>>> lemmatizer.lemmatize('sings', pos='v')
'sing'
>>>
```

# NLTK – Etiquetadores

- O NLTK possui **dois corpus** que servem como base para o etiquetador em português: o **Floresta** e o **Mac Morpho**.
  - Para o inglês já existe um etiquetador padrão treinado: o **`nltk.pos_tag()`**.
- Os etiquetadores passam primeiramente por uma fase de treinamento com as sentenças presentes.
  - Floresta: 9.266 sentenças etiquetadas
  - Mac Morpho: 51.397 sentenças etiquetadas
- Como resultado, os etiquetadores retornam uma tupla (‘palavra’, ‘classe gramatical’)
  - Na qual a classe gramatical depende do treinamento que é realizado.

# NLTK – Etiquetadores: Mac Morpho

```
50  from nltk.corpus import mac_morpho
51  from nltk.tag import UnigramTagger
52
53  tokens = nltk.word_tokenize(corpus_teste.read())
54
55  sentencas_treinadoras = mac_morpho.tagged_sents()
56  etiq = UnigramTagger(sentencas_treinadoras)
57  tags = etiq.tag(tokens)
58  print(tags)
59
```

```
[('Giants', 'NPROP'), ('batem', 'V'), ('os', 'ART'), ('Patriots', None),
('no', 'KC'), ('Super', 'NPROP'), ('Bowl', 'NPROP'), ('XLII', None),
('Azarões', None), ('acabam', 'VAUX'), ('com', 'PREP'), ('a', 'ART'),
('invencibilidade', 'N'), ('de', 'PREP'), ('New', 'NPROP'), ('England',
'NPROP'), ('e', 'KC'), ('ficam', 'V'), ('com', 'PREP'), ('o', 'ART'),
('título', 'N'), ('da', 'NPROP'), ('temporada', 'N'), ('04/02/2008', None),
('-', '-'), ('01h07m', None), ('-', '-'), ('Atualizado', None), ('em',
'PREP|+'), ('04/02/2008', None), ('-', '-'), ('09h49m', None), ('Com',
'PREP'), ('um', 'ART'), ('passe', 'N'), ('de', 'PREP'), ('Eli', 'NPROP'),
('Manning', 'NPROP'), ('para', 'PREP'), ('Plaxico', None), ('Burress',
None), ('a', 'ART'), ('39', 'NUM'), ('segundos', 'N'), ('do', 'NPROP'),
('fim', 'N'), (',', ','), ('o', 'ART'), ('New', 'NPROP'), ('York', 'NPROP'),
('Giants', 'NPROP'), ('anotou', 'V'), ('o', 'ART'), ('touchdown', 'N|EST'),
```

# NLTK – Etiquetadores: Mac Morpho

```
50  from nltk.corpus import mac_morpho
51  from nltk.tag import UnigramTagger
52
53  tokens = nltk.word_tokenize(corpus_teste.read())
54
55  sentencas_treinadoras = mac_morpho.tagged_sents()
56  etiq = UnigramTagger(sentencas_treinadoras)
57  tags = etiq.tag(tokens)
58  print(tags)
59
```

```
[('Giants', 'NPROP'), ('batem', 'V'), ('os', 'ART'), ('Patriots', None),
('no', 'KC'), ('Super', 'NPROP'), ('Bowl', 'NPROP'), ('XLII', None),
('Azarões', None), ('acabam', 'VAUX'), ('com', 'PREP'), ('a', 'ART'),
('invencibilidade', 'N'), ('de', 'PREP'), ('New', 'NPROP'), ('England',
'NPROP'), ('e', 'KC'), ('ficam', 'V'), ('com', 'PREP'), ('o', 'ART'),
('título', 'N'), ('da', 'NPROP'), ('temporada', 'N'), ('04/02/2008', None),
('-', '-'), ('01h07m', None), ('-', '-'), ('Atualizado', None), ('em',
'PREP|+'), ('04/02/2008', None), ('-', '-'), ('09h49m', None), ('Com',
'PREP'), ('um', 'ART'), ('passe', 'N'), ('de', 'PREP'), ('Eli', 'NPROP'),
('Manning', 'NPROP'), ('para', 'PREP'), ('Plaxico', None), ('Burress',
None), ('a', 'ART'), ('39', 'NUM'), ('segundos', 'N'), ('do', 'NPROP'),
('fim', 'N'), (',', ','), ('o', 'ART'), ('New', 'NPROP'), ('York', 'NPROP'),
('Giants', 'NPROP'), ('anotou', 'V'), ('o', 'ART'), ('touchdown', 'N|EST'),
```

- Mas e aqueles None alí? O que significam?

# NLTK – Etiquetadores: Mac Morpho

- Por ter de passar por uma **fase de treinamento**, tinham palavras que o etiquetador não conseguiu identificar e fazer a classificação.

- Uma solução é pré-classificar todas as palavras do texto como substantivos (**`N`**) e depois treinar o etiquetador normalmente.
  - Usa-se o pacote **`DefaultTagger`**

# NLTK – Etiquetadores: Mac Morpho

```
50  from nltk.corpus import mac_morpho
51  from nltk.tag import DefaultTagger
52  from nltk.tag import UnigramTagger
53
54  tokens = nltk.word_tokenize(corpus_teste.read())
55
56  etiq_padrao = DefaultTagger('N')
57  sentencas_treinadoras = mac_morpho.tagged_sents()
58  etiq = UnigramTagger(sentencas_treinadoras, backoff=etiq_padrao)
59  tags = etiq.tag(tokens)
60  print(tags)
61
```

```
[('Giants', 'NPROP'), ('batem', 'V'), ('os', 'ART'), ('Patriots', 'N'),
('no', 'KC'), ('Super', 'NPROP'), ('Bowl', 'NPROP'), ('XLII', 'N'),
('Azarões', 'N'), ('acabam', 'VAUX'), ('com', 'PREP'), ('a', 'ART'),
('invencibilidade', 'N'), ('de', 'PREP'), ('New', 'NPROP'), ('England',
'NPROP'), ('e', 'KC'), ('ficam', 'V'), ('com', 'PREP'), ('o', 'ART'),
('título', 'N'), ('da', 'NPROP'), ('temporada', 'N'), ('04/02/2008', 'N'),
('-', '-'), ('01h07m', 'N'), ('-', '-'), ('Atualizado', 'N'), ('em',
'PREP|+'), ('04/02/2008', 'N'), ('-', '-'), ('09h49m', 'N'), ('Com',
'PREP'), ('um', 'ART'), ('passe', 'N'), ('de', 'PREP'), ('Eli', 'NPROP'),
('Manning', 'NPROP'), ('para', 'PREP'), ('Plaxico', 'N'), ('Burress', 'N'),
('a', 'ART'), ('39', 'NUM'), ('segundos', 'N'), ('do', 'NPROP'), ('fim',
'N'), (',', ','), ('o', 'ART'), ('New', 'NPROP'), ('York', 'NPROP'),
('Giants', 'NPROP'), ('anotou', 'V'), ('o', 'ART'), ('touchdown', 'N|EST'),
```

# NLTK – Etiquetadores: Mac Morpho

| Classe Gramatical | Etiqueta | Exemplos |
|---|---|---|
| Adjetivo | ADJ | bom - ruim - ótimo - péssimo |
| Advérbio | ADV | muito - pouco - normalmente |
| Advérbio Conectivo Subordinativo | ADV-KS | Sei onde mora |
| Advérbio Relativo Subordinativo | ADV-KS-REL | onde - quando - como |
| Artigo | ART | o - a - os - as |
| Conjunção Coordenativa | KC | e - nem - mas - ou - pois |
| Conjunção Subordinativa | KS | que - porque - assim |
| Interjeição | IN | ufa! - viva! - ai! - oi! |
| Numeral | NUM | três - quatro - 3 - 4 |
| Palavra Denotativa | PDEN | até - apenas - eis - cá |
| Particípio | PCP | dormido - espalhado - tido |
| Pronome Adjetivo | PROADJ | meu - nosso - este - algum |
| Pronome Conectivo Subordinativo | PRO-KS | Sei quem chegou |
| Pronome Conectivo Subord. Relativo | PRO-KS-REL | o qual - cujo |
| Pronome Pessoal | PROPESS | eu - me - Vossa Alteza |
| Pronome Substantivo | PROSUB | isto - isso - aquilo - alguém |
| Símbolo de Moeda Corrente | CUR | R$ - US$ |
| Substantivo | N | hotel - quarto - atendimento |
| Substantivo Próprio | NPROP | Maria - Vinícius - Globo |
| Verbo | V | é - foi - gostar - ir |
| Verbo Auxiliar | VAUX | ter - haver |

# NLTK – Etiquetadores: Mac Morpho

- É possível, então, fazer várias manipulações com a lista de tuplas resultante:
  - Análises descritivas
  - Análises sintáticas
  - **Chunking**
    - Reconhecimento de Entidades Nomeadas
      - Nosso problema antigo!!
  - E várias outras...!

# NLTK – Etiquetadores: Mac Morpho

```
71  from nltk.chunk import RegexpParser
72  pattern = 'NP:{<NPROP><NPROP>|<N><N>}'
73  analiseGramatical = RegexpParser(pattern)
74  arvore = analiseGramatical.parse(tags)
75  print(arvore)
76  arvore.draw()
77
```

```
(S
  Com/PREP
  um/ART
  passe/N
  de/PREP
  (NP Eli/NPROP Manning/NPROP)
  para/PREP
  (NP Plaxico/N Burress/N)
  a/ART
  39/NUM
  segundos/N
  do/NPROP
  fim/N
```

# NLTK – Trabalhando com Corpus

- Faça uma **análise descritiva** completa do nosso corpus de teste, utilizando as funções do NLTK.

- Exemplos de atributos:
  - Quantidade de tokens
  - Quantidade de sentenças / média do tamanho das sentenças
  - Quantidade de substantivos, adjetivos, advérbios...
  - Quantidade de palavras com o mesmo radical
  - Quantidade de símbolos de pontuação
  - Palavras mais frequentes do corpus
  - ...

- Use sua imaginação!!! =D

# spaCy

# spaCy

- Biblioteca Python para processamento de textos
  - Escala industrial
- Feito para uso em produção
  - Criação de aplicações que conseguem processar um grande volume de dados
- Versão ~~2.1~~ 3.0!
  - O parser sintático mais rápido do mundo (!!)
  - Acurácia de 92.6%
    - 1% a mais que o melhor parser disponível
- Suporte para mais de 61 linguagens



Aqui o artigo de 2015 que faz a comparação entre os parsers.

# spaCy - Instalação

- Guia de instalação completo aqui

- Compatível com versões ~~2.7~~/3.6+ do Python
  - ~~Uma das poucas bibliotecas que ainda possuem suporte para o Python 2.x~~

- Linux, MacOS e Windows 64-bit
  - Instalação por linha de comando

```
pip install -U spacy
```

- Necessário instalar dados adicionais
  - Parecido com o que fizemos no NLTK

# spaCy - Instalação

- Dados adicionais para lematização

```
pip install –U spacy-lookups-data
```

- Modelo de linguagem
  - Para o spaCy conseguir realizar suas funções, é necessário que um modelo de linguagem esteja presente.
  - Modelos pré-treinados
    - Entidades Nomeadas
    - Classes gramaticais
    - Dependências sintáticas
  - Parecido com os córpus que utilizamos como treinamento no NLTK

# spaCy - Instalação

- Modelos de linguagem para o português

```
python -m spacy download pt_core_news_sm
python -m spacy download pt_core_news_md
python -m spacy download pt_core_news_lg
```

- Praticamente todas as atribuições que os modelos mais robustos possuem (exemplo: inglês)
  - Baseado no corpus WikiNER
    - Vetores dos tokens e classes gramaticais
    - Análise de dependência
    - Entidades nomeadas

- Mais detalhes sobre os modelos aqui.

# spaCy - Instalação

- Além do modelo de linguagem padrão do spaCy, é possível criar o seu próprio modelo!
  - Ou usar um pronto, já treinado para algum fim

- Um ótimo guia pode ser encontrado aqui!
  - Spoiler: inserção de alguns exemplos para treinamento em um código próprio do spaCy

# spaCy - Uso

- Para o uso das funções poderosas do spaCy, é preciso entender dois objetos importantes:
  - O objeto **`Doc`**
  - O objeto **`Token`**

- Um **`Doc`** é uma sequência de objetos **`Token`**
  - Ou seja, um documento com vários tokens manipuláveis
  - Métodos da classe **`Doc`** levam em consideração a manipulação desses tokens
    - Exemplo: quantidade de tokens no documento

- Um **`Token`** é o token que aprendemos na aula de NLTK: pode ser uma palavra, uma pontuação, numeral, espaços...

# spaCy - Uso

- Assim, antes de qualquer utilização das funções do spaCy, deve-se criar a variável que vai guardar o modelo de linguagem

```
1  import spacy
2
3  nlp = spacy.load("pt_core_news_lg")
4  doc = nlp(palavras) #o texto, não os tokens!
```

  - IMPORTANTE: no NLTK era utilizado sempre a lista de tokens, mas aqui no spaCy, o parâmetro é sempre a string do texto!

- Portanto, a partir de agora, todas as funções serão provenientes da variável **doc**!

# spaCy - Uso

- Bom, aqui vamos começar a usar as funções mais interessantes do spaCy:
  - Tokenização
  - Stemming e Lematizador
  - Etiquetador
  - Entidades Nomeadas
- Utilizaremos o mesmo córpus das aulas anteriores
  - Tá aqui, para quem ainda não tem.
- Claro, o spaCy contém várias outras funções!

# spaCy - Tokenização

- Para recuperar os tokens, basta usar o conceito de *list comprehension*

```
12  tokens = [token for token in doc]
13  print(tokens)
14
```

```
[Giants, batem, os, Patriots, no, Super, Bowl, XLII,
, Azarões, acabam, com, a, invencibilidade, de, New, England, e,
ficam, com, o, título, da, temporada,
, 04/02/2008, -, 01h07, m, -, Atualizado, em, 04/02/2008, -,
09h49, m,

, Com, um, passe, de, Eli, Manning, para, Plaxico, Burress, a,
39, segundos, do, fim, ,, o, New, York, Giants, anotou, o,
touchdown, decisivo, e, derrubou, o, favorito, New, England,
Patriots, por, 17, a, 14, n, este, domingo, ,, em, Glendale, ,,
```

# spaCy - Tokenização

- Para recuperar os tokens, basta usar o conceito de *list comprehension*

```
12  tokens = [token for token in doc]
13  print(tokens)
14
```

```
[Giants, batem, os, Patriots, no, Super, Bowl, XLII,
, Azarões, acabam, com, a, invencibilidade, de, New, England, e,
ficam, com, o, título, da, temporada,
, 04/02/2008, -, 01h07, m, -, Atualizado, em, 04/02/2008, -,
09h49, m,

, Com, um, passe, de, Eli, Manning, para, Plaxico, Burress, a,
39, segundos, do, fim, ,, o, New, York, Giants, anotou, o,
touchdown, decisivo, e, derrubou, o, favorito, New, England,
Patriots, por, 17, a, 14, n, este, domingo, ,, em, Glendale, ,,
```

- Dá pra perceber algumas coisas aqui, concordam?
  - Uma delas: não parece ser uma lista de strings...

# spaCy - Tokenização

- Para recuperar os tokens, basta usar o conceito de *list comprehension*

```
12 tokens = [token.orth_ for token in doc]
13 print(tokens)
14
```

```
['Giants', 'batem', 'os', 'Patriots', 'no', 'Super', 'Bowl',
'XLII', '\n', 'Azarões', 'acabam', 'com', 'a', 'invencibilidade',
'de', 'New', 'England', 'e', 'ficam', 'com', 'o', 'título', 'da',
'temporada', '\n', '04/02/2008', '-', '01h07', 'm', '-',
'Atualizado', 'em', '04/02/2008', '-', '09h49', 'm', '\n\n',
'Com', 'um', 'passe', 'de', 'Eli', 'Manning', 'para', 'Plaxico',
'Burress', 'a', '39', 'segundos', 'do', 'fim', ',', 'o', 'New',
'York', 'Giants', 'anotou', 'o', 'touchdown', 'decisivo', 'e',
'derrubou', 'o', 'favorito', 'New', 'England', 'Patriots', 'por',
'17', 'a', '14', 'n', 'este', 'domingo', ',', 'em', 'Glendale',
',', 'no', 'Super', 'Bowl', 'XLII', '.', 'O', 'resultado', ',',
'uma', 'das', 'maiores', 'zebras', 'da', 'história', 'do',
'Super', 'Bowl', ',', 'acabou', 'com', 'a', 'temporada',
'perfeita', 'de', 'Tom', 'Brady', 'e', 'companhia', ',', 'que',
```

- Agora sim! Só usar o atributo **`orth_`**

# spaCy - Tokenização

- Retorno com tipos de tokens diferentes:
  - Somente as palavras: **`is_alpha`**

```
>>> texto = "Com um passe de Eli Manning para Plaxico Burress a 39 segun
dos do fim, o New York Giants anotou o touchdown decisivo e derrubou o f
avorito New England Patriots por 17 a 14 neste domingo."
>>> doc = nlp(texto)
>>> tokens_palavras = [token .orth_ for token in doc if token.is_alpha]
>>> tokens_palavras
['Com', 'um', 'passe', 'de', 'Eli', 'Manning', 'para', 'Plaxico', 'Burre
ss', 'a', 'segundos', 'do', 'fim', 'o', 'New', 'York', 'Giants', 'anotou
', 'o', 'touchdown', 'decisivo', 'e', 'derrubou', 'o', 'favorito', 'New'
, 'England', 'Patriots', 'por', 'a', 'n', 'este', 'domingo']
```

  - Somente os números: **`is_digit`**
  - Somente pontuações: **`is_punct`**

```
>>> tokens_numeros = [token .orth_ for token in doc if token.is_digit]
>>> tokens_numeros
['39', '17', '14']
>>> tokens_pontuacoes = [token .orth_ for token in doc if token.is_punct]
>>> tokens_pontuacoes
[',', '.']
```

# spaCy - Tokenização

- Retorno com tipos de tokens diferentes:
  - Pontuação esquerda ou direita
    - Parênteses e colchetes
  - Espaços
  - Símbolos financeiros
  - Números (10.9, 10, “dez”)
  - E-mail
  - Stopwords...
  - Lista completa com os atributos aqui.

# spaCy – Stemming e Lematização

- Olha que interessante (e surpreendente): o spaCy não tem um stemmer padrão...

- Porém, o spaCy tem um lematizador!
  - O inverso do NLTK, pelo menos para o Português!

- Lematizar é simples: só utilizar o atributo **`lemma_`**:

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> texto = "Os Giants começaram com a posse de bola, e mostraram logo que iriam
 alongar ao máximo suas posses de bola."
>>> doc = nlp(texto)
>>> lemmas = [token.lemma_ for token in doc if token.pos_ == 'VERB']
>>> lemmas
['começar', 'mostrar', 'alongar']
```

# spaCy – Lematização

- É importante observar que foi utilizado um outro atributo que ainda não falamos: o **`pos_`**

- Esse atributo é referente ao ***P**art-**O**f-**S**peech*, ou simplesmente, a classe gramatical do token.

- Vale ressaltar que a lematização geralmente remete à forma canônica da palavra para os verbos, então é necessária essa condição.

- Ok, mas como obtida essa classe gramatical? É simples assim, só com um atributo?

# spaCy – Etiquetador

- Sim. Basta chamar o atributo **pos_** no token e assim é retornada a classe gramatical referente!

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> texto = "Os Giants começaram com a posse de bola, e mostraram logo que iriam
 alongar ao máximo suas posses de bola."
>>> doc = nlp(texto)
>>> etiquetas = [(token.orth_, token.pos_) for token in doc]
>>> etiquetas
[('Os', 'DET'), ('Giants', 'PROPN'), ('começaram', 'VERB'), ('com', 'ADP'), ('a'
, 'DET'), ('posse', 'NOUN'), ('de', 'ADP'), ('bola', 'NOUN'), (',', 'PUNCT'), ('
e', 'CCONJ'), ('mostraram', 'VERB'), ('logo', 'ADV'), ('que', 'SCONJ'), ('iriam'
, 'AUX'), ('alongar', 'VERB'), ('ao', 'DET'), ('máximo', 'NOUN'), ('suas', 'DET'
), ('posses', 'NOUN'), ('de', 'ADP'), ('bola', 'NOUN'), ('.', 'PUNCT')]
```

- O conjunto de etiquetas para o português está aqui!