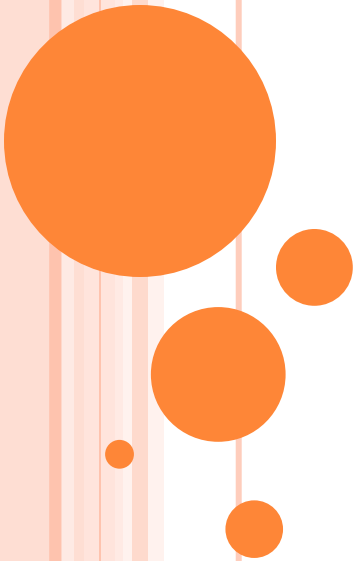

Python para PLN

Introdução ao spaCy

Rogério Figueredo de Sousa rogerfig@usp.br
Roney Lira de Sales Santos roneysantos@usp.br
Prof. Thiago A. S. Pardo

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!

SPACY – ETIQUETADOR

- O spaCy tem um atributo que contém informações mais detalhadas: o **`morph`**

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> texto = "Os Giants começaram com a posse de bola, e mostraram logo que iria
m alongar ao máximo suas posses de bola."
>>> doc = nlp(texto)
>>> etiquetas = [(token.orth_, token.morph) for token in doc]
>>> etiquetas
[('Os', 'DET__Definite=Def|Gender=Masc|Number=Plur|PronType=Art'), ('Giants', '
PROPN__Gender=Masc|Number=Plur'), ('começaram', 'VERB__Mood=Ind|Number=Plur|Per
son=3|Tense=Past|VerbForm=Fin'), ('com', 'ADP'), ('a', 'DET__Definite=Def|Gende
r=Fem|Number=Sing|PronType=Art'), ('posse', 'NOUN__Gender=Fem|Number=Sing'), ('
de', 'ADP'), ('bola', 'NOUN__Gender=Fem|Number=Sing'), (',', 'PUNCT'), ('e', 'C
CONJ'), ('mostraram', 'VERB__Mood=Ind|Number=Plur|Person=3|Tense=Past|VerbForm=
Fin'), ('logo', 'ADV'), ('que', 'SCONJ'), ('iriam', 'AUX__Mood=Cnd|Number=Plur|
Person=3|VerbForm=Fin'), ('alongar', 'VERB__VerbForm=Inf'), ('ao', 'ADP_DET__De
finite=Def|Gender=Masc|Number=Sing|PronType=Art'), ('máximo', 'NOUN__Gender=Mas
c|Number=Sing'), ('suas', 'DET__Gender=Fem|Number=Plur|PronType=Prs'), ('posses
', 'NOUN__Gender=Fem|Number=Plur'), ('de', 'ADP'), ('bola', 'NOUN__Gender=Fem|N
umber=Sing'), ('.', 'PUNCT')]
```

- Características mais morfológicas dos tokens!

spaCy – Etiquetador

- O modelo de linguagem para o Português usado no Spacy tem como fonte o Bosque
  - Acurácia de **95.02%** na etiquetagem quando utilizado o modelo de linguagem **large**

- Existem outros etiquetadores para o Português que alcançam uma acurácia maior
  - NLPNet – 97,33% (free)
  - PALAVRAS – 98% (pago)
    - É importante frisar que estes etiquetadores tem um foco total no Português.

Aqui e aqui todas as informações sobre o NLPNet, como detalhes e utilização.

spaCy – Entidades Nomeadas

- Vamos colocar a mão na massa agora em coisas mais robustas que o spaCy nos proporciona
- Vimos no NLTK uma dificuldade inicial de identificar as entidades nomeadas de uma sentença
- Será que o spaCy facilita esse trabalho?

spaCy – Entidades Nomeadas

- SIM!! Basta usar a propriedade **`ents`** na variável **`doc`!**

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> texto = "Com um passe de Eli Manning para Plaxico Burress a 39 segundos do f
im, o New York Giants anotou o touchdown decisivo e derrubou o favorito New Engl
and Patriots por 17 a 14 neste domingo, em Glendale, no Super Bowl XLII. O resul
tado, uma das maiores zebras da história do Super Bowl, acabou com a temporada p
erfeita de Tom Brady e companhia, que esperavam fazer história ao levantar o tro
féu da NFL sem sofrer uma derrota no ano."
>>> doc = nlp(texto)
>>> entidades = list(doc.ents)
>>> entidades
[Eli Manning, Plaxico, Burress, New York Giants, New England Patriots, Glendale,
 Super Bowl XLII, Super Bowl, Tom Brady, NFL]
```

- Olha só! Nossa lista contém praticamente todas as entidades nomeadas da sentença!

spaCy – Entidades Nomeadas

- Detalhadamente, veja como o spaCy classifica cada entidade:

```
>>> entidades_detalhes = [(entidade, entidade.label_) for entidade in doc.ents]
>>> entidades_detalhes
[(Eli Manning, 'PER'), (Plaxico, 'PER'), (Burress, 'PER'), (New York Giants, 'O
RG'), (New England Patriots, 'ORG'), (Glendale, 'LOC'), (Super Bowl XLII, 'ORG'
), (Super Bowl, 'ORG'), (Tom Brady, 'PER'), (NFL, 'ORG')]
```

  - A acurácia alta permite que as entidades sejam classificadas corretamente.
    - Por mais que tenhamos no nosso resultado uma entidade "separada"
      - A medida de acerto do modelo de linguagem treinado é de **91.24%** (F-Score)

spaCy – Entidades Nomeadas

- Um exemplo com menos entidades em inglês:

```
>>> texto = "\'A gente sabe que, quando uma pessoa está mentindo, inconscientem
ente, isso afeta a produção do texto. Mudam as palavras que ela usa e as estrut
uras do texto. Além disso, a pessoa costuma ser mais assertiva e emotiva. Então
, uma das formas de detectar textos enganosos é medir essas características\',
explica o professor Thiago Pardo, do Instituto de Ciências Matemáticas e de Com
putação (ICMC) da USP, em São Carlos. Pesquisador do Núcleo Interinstitucional
de Linguística Computacional (NILC), Thiago é o coordenador do projeto que resu
ltou na criação da plataforma e na publicação do artigo Contributions to the St
udy of Fake News in Portuguese: New Corpus and Automatic Detection Results, apr
esentado no final de setembro na 13ª Conferência Internacional de Processamento
Computacional do Português."
>>> doc = nlp(texto)
>>> entidades = list(doc.ents)
>>> entidades
[Thiago Pardo, Instituto de Ciências Matemáticas e de Computação, ICMC, USP, Sã
o Carlos, Núcleo Interinstitucional de Linguística Computacional, NILC, Thiago,
Contributions to the Study of Fake News in Portuguese: New Corpus and Automatic
Detection Results, Conferência Internacional de Processamento Computacional do
Português]
>>> entidades_detalhes = [(entidade, entidade.label_) for entidade in doc.ents]
>>> entidades_detalhes
[(Thiago Pardo, 'PER'), (Instituto de Ciências Matemáticas e de Computação, 'OR
G'), (ICMC, 'ORG'), (USP, 'LOC'), (São Carlos, 'LOC'), (Núcleo Interinstitucion
al de Linguística Computacional, 'ORG'), (NILC, 'MISC'), (Thiago, 'ORG'), (Cont
ributions to the Study of Fake News in Portuguese: New Corpus and Automatic Det
ection Results, 'MISC'), (Conferência Internacional de Processamento Computacio
nal do Português, 'ORG')]
```

spaCy – Entidades Nomeadas

- É possível visualizar essas entidades nomeadas de forma gráfica, por meio do **displaCy**.
  - Usando o nosso primeiro exemplo, com um trecho do córpus, é possível destacar todas as entidades nomeadas:

```
15 import spacy
16 from pathlib import Path
17
18 nlp = spacy.load("pt_core_news_lg")
19 texto = "Com um passe de Eli Manning para Plaxico Burress a 39
20 doc = nlp(texto)
21
22 html = spacy.displacy.render(doc, style="ent")
23 output_path = Path("entidades_nomeadas.html")
24 output_path.open("w", encoding="utf-8").write(html)
```

spaCy – Entidades Nomeadas

- Resultado:

Com um passe de Eli Manning **PER** para Plaxico **PER** Burress **PER** a 39 segundos do fim, o New York Giants **ORG** anotou o touchdown decisivo e derrubou o favorito New England Patriots **ORG** por 17 a 14 neste domingo, em Glendale **LOC** , no Super Bowl XLII **ORG** . O resultado, uma das maiores zebras da história do Super Bowl **ORG** , acabou com a temporada perfeita de Tom Brady **PER** e companhia, que esperavam fazer história ao levantar o troféu da NFL **ORG** sem sofrer uma derrota no ano.

- Obs: o layout pode ser modificado da forma que você prefira. Veja aqui nesse link como fazer!

spaCy – Análise Sintática

- Uma outra função importante do spaCy é a representação sintática do texto
  - Quais as relações entre os tokens

- O atributo **dep_** retorna a dependência sintática do token em questão

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> texto = "Os Giants começaram com a posse de bola, e mostraram logo que iriam alo
ngar ao máximo suas posses de bola."
>>> doc = nlp(texto)
>>> sintatica = [(token.orth_, token.dep_) for token in doc]
>>> sintatica
[('Os', 'det'), ('Giants', 'nsubj'), ('começaram', 'ROOT'), ('com', 'case'), ('a', '
det'), ('posse', 'obl'), ('de', 'case'), ('bola', 'nmod'), (',', 'punct'), ('e', 'cc
'), ('mostraram', 'conj'), ('logo', 'advmod'), ('que', 'mark'), ('iriam', 'aux'), ('
alongar', 'ccomp'), ('ao', 'case'), ('máximo', 'obl'), ('suas', 'det'), ('posses', '
obj'), ('de', 'case'), ('bola', 'nmod'), ('.', 'punct')]
```

  - O conjunto de etiquetas tá nesse link.

spaCy – Análise Sintática

- O spaCy também permite a visualização das dependências de forma gráfica pelo **displaCy**:

```
15  import spacy
16  from pathlib import Path
17
18  nlp = spacy.load("pt_core_news_lg")
19  texto = "Os Giants começaram com a posse de bola, e m
20  doc = nlp(texto)
21
22  svg = spacy.displacy.render(doc, style="dep")
23  output_path = Path("analise_dependencia.svg")
24  output_path.open("w", encoding="utf-8").write(svg)
```

- Aqui o resultado!

SPACY – DISPLACY

- O spaCy contém dois sites onde podem ser feitas as análises de entidades nomeadas e de dependências de forma bem simples:

- Visualizador de Entidades Nomeadas
  - https://explosion.ai/demos/displacy-ent

- Visualizador de Dependências
  - https://explosion.ai/demos/displacy

- Basta selecionar o modelo para português (ou qualquer outra linguagem) e brincar!

spaCy – Similaridade entre Palavras

- Por ter um bom e grande modelo de linguagem para o Português, o spaCy permite avaliar similaridade entre palavras!

- E continua sendo simples: só usar o método **`similarity()`**!

```
>>> import spacy
>>> nlp = spacy.load("pt_core_news_lg")
>>> palavras = "conversar falar correr"
>>> doc = nlp(palavras)
>>> tokens = [token for token in doc]
>>> tokens[0].similarity(tokens[1])
0.73501545
>>> tokens[0].similarity(tokens[2])
0.44497716
>>> tokens[1].similarity(tokens[2])
0.4326754
```

spaCy – Similaridade entre Palavras

- Então, podemos fazer várias análises de similaridade entre palavras no texto!
- Exemplo 1: homem e mulher

```
>>> tokens[0].similarity(tokens[1])
0.6595782
```

- Exemplo 2: Roma e Itália

```
>>> tokens[0].similarity(tokens[1])
0.6953801
```

- Exemplo 3: eu e livro

```
>>> tokens[0].similarity(tokens[1])
0.19232121
```

spaCy – Similaridade entre Palavras

- O cálculo da similaridade é feito por meio da medida do cosseno

$$scos(\vec{f},\vec{v}) = \frac{\vec{f}\cdot\vec{v}}{\left|\vec{f}\right||\vec{v}|} = \frac{\sum_{i=1}^{n} f_i v_i}{\sqrt{\sum_{i=1}^{n} f_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}$$

  - Intervalo [0-1], onde 0 representa vetores completamente diferentes e 1 representa vetores completamente similares.

spaCy – Exercícios de Fixação

- 1. Dada uma palavra, encontrar no córpus as 3 outras palavras que mais são próximas semanticamente e as 3 palavras que são mais distantes.

  - Faça o teste com o modelo do próprio spaCy

spaCy – Exercícios de Fixação

- 2. Encontrar os vetores de todas as palavras do córpus e descobrir quais são as palavras relacionadas.

  - Dica: testar a similaridade entre todas as palavras do córpus e ordenar o par (palavra1, palavra2) pelo seu valor de similaridade
    - Utilização de estrutura de repetição (**`for`** ou **`while`**)
    - Utilização de um algoritmo de ordenação ou estrutura de controle (**`if`**)

  - Analisar quais valores foram retornados e confirmar a relação entre as palavras.