Python para PLN

spaCy + Embeddings

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

spaCy – Relembrando...

- Biblioteca Python para **uso em produção**
- **Modelos de linguagem** robustos para o português
- A maioria do processamento gira em torno dos objetos **`Doc`** e **`Token`**
- Tarefas de PLN facilmente realizadas por meio de **atributos**
  - **`lemma_`**, **`pos_`**, **`morph`**, **`ents`**, **`label_`**, **`dep_`**, ...
- **Visualização gráfica** de algumas tarefas de PLN pelo **displaCy**

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
```

Similaridade do Cosseno

- 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.

Introdução à *Word Embeddings*

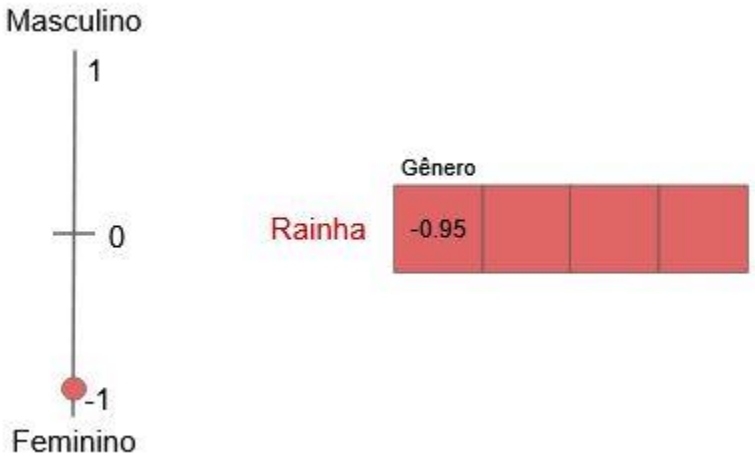
- Hipótese distribucional
  - Palavras tem significados parecidos quando são usadas em contextos parecidos.

- Modelos de linguagem
  - Predizem a próxima palavra, dado um conjunto de palavras
  - Exemplo: "O gato corre atrás do _____"
    - Qual a próxima palavra? "rato"? "cachorro"? "carro"?
  - Os modelos de linguagem são usados para tarefas como processamento de voz, autocorreção de ortografia, etc.

INTRODUÇÃO À *WORD EMBEDDINGS*

- *WORD EMBEDDING*: representação vetorial de uma palavra.
  - texto -> números

- Exemplo: Definição da palavra "rainha" com uma escala "Gênero", que vai de -1 a 1: quanto mais perto de -1, mais feminina:



Fonte: https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057

Introdução à *Word Embeddings*

- Porém, só com a informação sobre gênero não é possível representar bem a palavra...

- Podem ser adicionadas várias outras dimensões, ou quadradinhos, com a escala que a palavra tem mais a ver.

  - No exemplo anterior, imagine "rainha" e "rei" em escalas de "Realeza", "Fruta" e "Violência", por exemplo:

	Rainha	Rei
Gênero	-0.95	0.789
Realeza	0.89	0.96
	...	...
Fruta	0.015	-0.05
Violência	0.56	0.8

Fonte: https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057

Introdução à *Word Embeddings*

- E como esses valores são atribuídos?
- A partir de aprendizado de máquina!
  - Usa-se algum algoritmo para gerar, a partir do seu contexto.
- E o tamanho ideal do vetor, ou seja, a quantidade de dimensões?
  - Depende do seu corpus/*dataset* de treinamento: quanto menor, menos dimensões
  - Geralmente é um valor entre 100 e 1000.
- Um algoritmo muito utilizado para obter as *word embeddings* é chamado **Word2Vec**.

Fonte: https://medium.com/turing-talks/word-embedding-fazendo-o-computador-entender-o-significado-das-palavras-92fe22745057

spaCy – Similaridade: Word2vec

- O Word2Vec é uma técnica cuja a ideia é transformar cada token do texto em um vetor numérico para representação semântica.

- É uma das técnicas mais utilizadas no pré-processamento de textos e aprendizado de ***word embeddings***.

- É possível a utilização dessa técnica dentro do spaCy
  - É parecido com o atributo **`similarity()`**, porém, como geralmente usam-se modelos maiores e treinados com mais dados, **pode ser** mais eficiente o uso do word2vec.

Tomas Mikolov et al. **Efficient Estimation of Word Representations in Vector Space**. 2013.

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

  - 1. Encontrar modelos de *embeddings* treinados
  - 2. Converter o modelo para o spaCy
  - 3. Adequar o código da aplicação no spaCy para utilização do word2vec

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 1. Encontrar modelos de *embeddings* treinados

  - Existem vários modelos de *word embeddings* treinados, um para cada fim. Utilizaremos as *word embeddings* do NILC, que estão aqui.

    - Dois modelos são disponibilizados: CBOW e SKIP-GRAM
      - CBOW: modelo utilizado para **descobrir a palavra central** de uma sentença, baseado nas palavras que o cercam.
      - SKIP-GRAM: modelo utilizado para **descobrir as palavras de contexto** a partir de uma palavra central.

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 2. Converter o modelo para o spaCy

```
python -m spacy init vectors pt <local_emb> <nome_pasta>
```

  - <nome_da_pasta> é a identificação de onde será armazenado o modelo convertido
  - <local_emb> é o caminho que se encontra o modelo baixado anteriormente

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 2. Converter o modelo para o spaCy

```
python -m spacy init vectors pt <local_emb> <nome_pasta>
```

```
C:\Users\roney\Desktop>python -m spacy init vectors pt cbow_s50.txt vectors_spacy
ℹ Creating blank nlp object for language 'pt'
[2021-03-23 15:25:10,224] [INFO] Reading vectors from cbow_s50.txt
929606it [00:24, 38397.92it/s]
[2021-03-23 15:25:34,474] [INFO] Loaded vectors from cbow_s50.txt
✔ Successfully converted 929606 vectors
✔ Saved nlp object with vectors to output directory. You can now use the path to
it in your config as the 'vectors' setting in [initialize].
C:\Users\roney\Desktop\vectors_spacy
```

  - Ao final, é criada uma pasta com vários itens que são usados pelo spaCy na manipulação dos vetores.

SPACY – SIMILARIDADE: WORD2VEC

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 3. Adequar o código no spaCy para utilização do word2vec

```
35  import spacy
36  from spacy import util as spc_util
37
38  palavras = "conversar falar"
39  nlp = spacy.load("pt_core_news_lg")
40  doc = nlp(palavras)
41  tokens = [token for token in doc]
42
43  print("Similaridade - spaCy:", tokens[0].similarity(tokens[1]))
44
45  pathw2v = 'vectors_spacy'
46  spc_util.load_model(pathw2v, vocab=nlp.vocab)
47
48  print("Similaridade - word2vec:", tokens[0].similarity(tokens[1]))
```

```
Similaridade - spaCy: 0.73501545
Similaridade - word2vec: 0.7504553
```

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 3. Adequar o código no spaCy para utilização do word2vec
  - Perceba que a similaridade aumentou com o modelo word2vec treinado em comparação com o modelo de linguagem do spaCy
    - E se testarmos a similaridade entre justiça e trabalho?

spaCy – Similaridade: Word2vec

- Precisamos seguir 3 passos para usar os princípios do word2vec no spaCy:

- 3. Adequar o código no spaCy para utilização do word2vec
  - Perceba que a similaridade aumentou com o modelo word2vec treinado em comparação com o modelo de linguagem do spaCy
    - E se testarmos a similaridade entre justiça e trabalho?

```
Similaridade - spaCy: 0.31346163
Similaridade - word2vec: 0.2301711
```

    - O modelo do spaCy foi melhor. Veja que vai depender muito do modelo word2vec treinado e de quantas dimensões os vetores estão dispostos.
      - Como resolve? **Testes, testes e mais testes...**!

Mais informações sobre o Word2Vec no Google Colab AQUI!

spaCy – Similaridade: Word2vec

- Vamos fazer aquele teste clássico:

  MADRI – ESPANHA + FRANÇA ≈ PARIS

  - Precisamos fazer **operações entre vetores**.
  - O spaCy tem um atributo que retorna o vetor do token em questão: **`vector`**

    - Para as operações com vetores utilizaremos o módulo **Numpy**
      - Instalação: **`pip install numpy`**
    - Para o cálculo da similaridade, utilizaremos o método pronto proveniente do módulo **Scikit-learn**
      - Instalação: **`pip install -U scikit-learn`**

SPACY – SIMILARIDADE: WORD2VEC

- Vamos fazer aquele teste clássico:

MADRI – ESPANHA + FRANÇA ≈ PARIS

```
32  import spacy
33  from spacy import util as spc_util
34  import numpy as np
35  from sklearn.metrics.pairwise import cosine_similarity
36
37  palavras = "madri espanha frança paris"
38  nlp = spacy.load("pt_core_news_lg")
39  doc = nlp(palavras)
40  tokens = [token for token in doc]
41
42  pathw2v = 'vectors_spacy'
43  spc_util.load_model(pathw2v, vocab=nlp.vocab)
44
45  # Madri - Espanha + França
46  vetor_res = np.array(tokens[0].vector) - np.array(tokens[1].vector) + np.array(tokens[2].vector)
47
48  # É necessário remodelar o vetor retornado pelo spaCy,
49  # pois ele está em 1 dimensão e para o uso do cosseno, é necessário um vetor de 2 dimensões
50  vetor_res = vetor_res.reshape(1,-1)
51  vetor_paris = tokens[3].vector.reshape(1,-1)
52
53  similaridade = cosine_similarity(vetor_res,vetor_paris)
54  print(similaridade)
```

```
[[0.8048478]]
```