Word2vec

Felipe Salvatore

USP, 28/03/2017

Introdução

Word2vec é um nome que abarca dois modelos:

- **Skip-gram**
- **Continuous Bag-of-Words (CBOW)**

Tarefa: Apreender de modo eficiente uma representação vetorial de palavras a partir de um corpus grande e não-estruturado.

Esse aprendizado é feito atraves das estatística de co-ocorrência das palavras em algum corpus. A ideia em si não é nova:

"You shall know a word by the company it keeps" (J.R Firth, 1957)

Versão simplificada de CBOW: modelo (i)

Dado um corpus, escolhemos:

- um vocabulário V.
- um tamanho N para a representação vetorial das palavras.

Vamos usar as matrizes $W \in \mathbb{R}^{|V|,N}$ e $W' \in \mathbb{R}^{N,|V|}$ para criar **duas** representações vetoriais de cada palavra $\mathbb{w}$:

- **input vector**: $v_{\mathbb{w}}$ (linha de W).
- **output vector**: $v'_{\mathbb{w}}$ (coluna de W').

Versão simplificada de CBOW: modelo (ii)

A tarefa do modelo vai ser prever uma palavra de centro dada uma palavra de contexto:

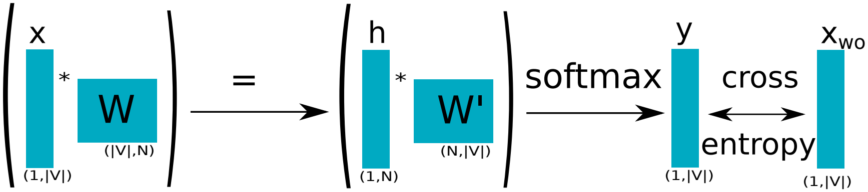
O primeiro rei de Portugal nasceu em ...

Observação $\Rightarrow$ (rei, primeiro)

$\Rightarrow$ (input word, output word)

$\Rightarrow$ $(\mathbb{w}_I, \mathbb{w}_O)$

Versão simplificada de CBOW: modelo (iii)



Versão simplificada de CBOW: modelo (iv)

Dado $(x_{\mathbb{w}_I}, x_{\mathbb{w}_O})$ one-hot de $(\mathbb{w}_I, \mathbb{w}_o)$ e $x = x_{\mathbb{w}_I}$ o modelo é:

$$h_i = \sum_{s=1}^{|V|} w_{si} x_s \;\; \text{com } i = 1, \ldots, N \tag{1}$$

$$u_j = \sum_{s=1}^{N} w'_{sj} h_s \;\; \text{com } j = 1, \ldots, |V| \tag{2}$$

$$y_j = p(\mathbb{w}_j | \mathbb{w}_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{|V|} \exp(u_{j'})} \;\; \text{com } j = 1, \ldots, |V| \tag{3}$$

$$E = CE(x_{\mathbb{w}_O}, y) = -\sum_{s=1}^{|V|} x_{\mathbb{w}_O s} \log(y_s) \tag{4}$$

Versão simplificada de CBOW: modelo (v)

Pela configuração de $x_{\mathbb{w}_I}$ e $x_{\mathbb{w}_O}$ podemos simplificar (1), (2), (3) e (4):

$$h = v_{\mathbb{w}_I} \tag{5}$$

$$u_j = {v'_{\mathbb{w}_j}}.^T v_{\mathbb{w}_I} \tag{6}$$

$$y_j = \frac{\exp({v'_{\mathbb{w}_j}}.^T v_{\mathbb{w}_I})}{\sum_{j'=1}^{|V|} \exp({v'_{\mathbb{w}_{j'}}}.^T v_{\mathbb{w}_I})} \tag{7}$$

$$E = -u_{j^*} + \log(\sum_{j'=1}^{|V|} \exp(u_{j'})) \tag{8}$$

onde j^* é o índice de $\mathbb{w}_o$.

Usando o algorítimo de back propagration e SGD temos que a atualização dos pesos da camada mais externa é:

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \eta \, e_j \, h_i \tag{9}$$

em notação vetorial:

$$v'^{(new)}_{\mathbb{w}_j} = v'^{(old)}_{\mathbb{w}_j} - \eta \, e_j \, v_{\mathbb{w}_I} \tag{10}$$

onde $e = y - x_{\mathbb{w}_o}$

Versão simplificada de CBOW: atualização (ii)

- $\mathrm{w}_j \neq \mathrm{w}_o \Rightarrow -\eta\, e_j < 0 \Rightarrow$ subtraímos de v'_{w_j} uma proporção de $v_{\mathrm{w}_I} \Rightarrow$ aumentamos a distância cosseno entre v_{w_I} e v'_{w_j}.
- $\mathrm{w}_j = \mathrm{w}_o \Rightarrow -\eta\, e_j > 0 \Rightarrow$ adicionamos uma proporção de v_{w_I} em $v'_{\mathrm{w}_j} \Rightarrow$ diminuímos a distância cosseno entre v_{w_I} e v'_{w_j}.

Versão simplificada de CBOW: atualização (iii)

Continuando com o back propagation:

$$W^{(new)} = W^{(old)} - \eta\, xEH^T \tag{11}$$

$$v_{\mathrm{w}_I}{}^{(new)} = v_{\mathrm{w}_I}{}^{(old)} - \eta\, xEH^T_{(k_I,.)} \tag{12}$$

Onde $EH = e(W')^T$ e k_I é o índice de w_I.

Versão simplificada de CBOW

Repetindo esse processo com diferentes exemplos extraídos do corpus o efeito vai acumular e como resultado palavras com contexto similar vão ficar próximas entre si.

O que o modelo faz é capturar as estatísticas de co-ocorrência usando a distância cosseno.

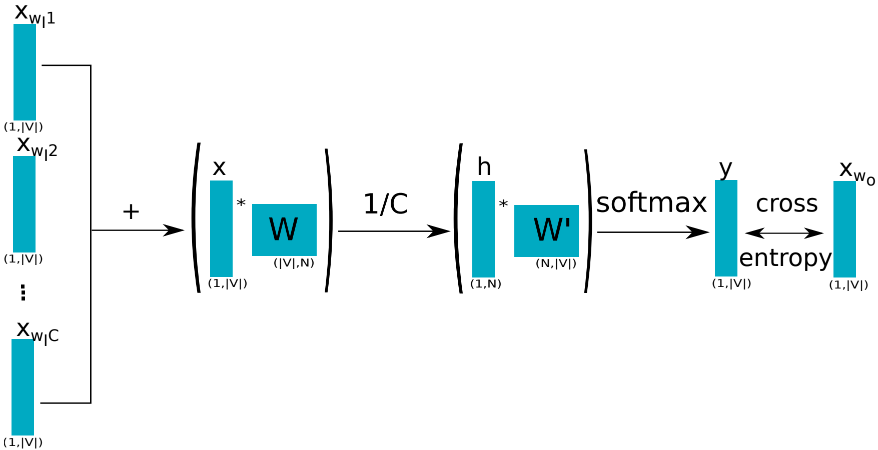
CBOW

Agora, partindo de uma janela arbitrária de tamanho C, vamos construir observações do tipo $([\mathbb{w}_{I_1}, \ldots, \mathbb{w}_{I_C}], \mathbb{w}_O)$.
Por exemplo com $C = 2$:

Nunca me acostumei com o cantor dessa banda, e nem ...

$$([com, o, dessa, banda], cantor)$$

CBOW: modelo (i)



CBOW: modelo (ii)

$$x = x_{\mathrm{w}_{I_1}} + \cdots + x_{\mathrm{w}_{I_C}} \tag{13}$$

$$h = \frac{1}{C}(v_{\mathrm{w}_{I_1}} + \cdots + v_{\mathrm{w}_{I_C}}) \tag{14}$$

$$u_j = \sum_{s=1}^{N} w'_{sj} h_s \tag{15}$$

$$y_j = p(\mathrm{w}_j | \mathrm{w}_{I_1}, \ldots, \mathrm{w}_{I_C}) = \frac{\exp(v'_{\mathrm{w}_j}.^T h)}{\sum_{j'=1}^{|V|} \exp(v'_{\mathrm{w}_{j'}}.^T h)} \tag{16}$$

$$E = -u_{j^*} + \log(\sum_{j'=1}^{|V|} \exp(u_{j'})) \tag{17}$$

CBOW: atualização

$$v'_{\mathrm{w}_j}{}^{(new)} = v'_{\mathrm{w}_j}{}^{(old)} - \eta\, e_j\, h \tag{18}$$

$$v_{\mathrm{w}_{I_c}}{}^{(new)} = v_{\mathrm{w}_{I_c}}{}^{(old)} - \frac{1}{C}\,\eta\, xEH^T_{(k_{I_c},.)} \tag{19}$$

para $c = 1, \ldots, C$. Onde $k_{I_1}, \ldots, k_{I_C}$ são os índices de $\mathrm{w}_{I_1}, \ldots, \mathrm{w}_{I_C}$ respectivamente.

Skip-Gram

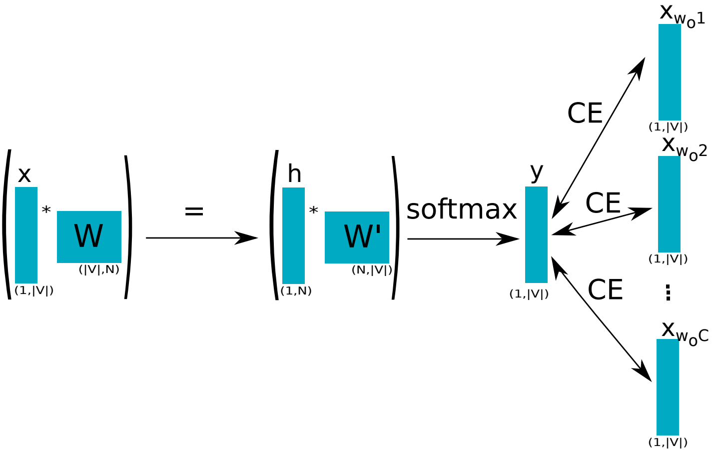
Skip-gram é o "contrário" do CBOW.

Com esse modelo vamos tentar prever o contexto dado a palavra de centro.

Observação $\Rightarrow (\mathbb{w}_I, [\mathbb{w}_{O_1}, \ldots, \mathbb{w}_{O_C}])$

$\Rightarrow$ (cantor, [com, o, dessa, banda])

Skip-Gram: modelo (i)



Skip-Gram: modelo (ii)

As definições de x, h, u, e y são as mesmas que em (1), (2) e (3). Nesse modelo queremos minimizar a soma da entropia cruzada que é o mesmo que maximizar $p(\mathbb{w}_{o_1}, \ldots, \mathbb{w}_{o_C} \mid \mathbb{w}_I)$:

$$
\begin{aligned}
E &= \sum_{c=1}^{C} (-\sum_{s=1}^{V} \mathbb{w}_{o_c\,s} \log(y_s)) \\
&= -\sum_{c=1}^{C} \log(y_{j_c^*}) \\
&= -\log(\prod_{c=1}^{C} y_{j_c^*}) \\
&= -\log(\prod_{c=1}^{C} p(\mathbb{w}_{o_c} \mid \mathbb{w}_I)) \\
&= -\log p(\mathbb{w}_{o_1}, \ldots, \mathbb{w}_{o_C} \mid \mathbb{w}_I)
\end{aligned}
$$

Simplificando a equação de erro:

$$E = -\sum_{c=1}^{C} u_{j_c^*} + C \log\left(\sum_{j'=1}^{V} \exp(u_{j'})\right) \tag{20}$$

Skip-Gram: atualização

$$v'_{\mathrm{w}_j}{}^{(new)} = v'_{\mathrm{w}_j}{}^{(old)} - \eta \, e_j \, v_{\mathrm{w}_I} \tag{21}$$

$$v_{\mathrm{w}_I}{}^{(new)} = v_{\mathrm{w}_I}{}^{(old)} - \eta \, xEH^T_{(k_I,.)} \tag{22}$$

Onde $e = (Cy - \sum_{c=1}^{C} x_{\mathrm{w}_c})$ e $EH = e(W')^T$.

Otimização

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{|V|} \exp(u_{j'})}$$

Muito custoso se for fazer isso para cada instância de treinamento

- Amostragem negativa
- Softmax hierárquico

Vamos nos concentrar no modelo Skip-gram.

Note: podemos implementar esse modelo de modo a prever apenas uma palavra de contexto:

$$(cantor, [com, o, dessa, banda])$$

$$(cantor, com), (cantor, o), (cantor, dessa), (cantor, banda)$$

Note

- Skip-gram $\Rightarrow h = v_{\mathbb{W}_I}$
- CBOW $\Rightarrow h = \frac{1}{C} \sum_{c=1}^{C} v_{\mathbb{W}_{I_c}}$

Amostragem negativa

Vamos manter x, W e W e h como antes. Para calcular a função erro vamos usar uma distribuição $P_n(\mathbb{w})$ sobre as palavras do corpus. Exemplo:

$$P_n(\mathbb{w}) = \frac{U(\mathbb{w})^{\frac{3}{4}}}{Z}$$

Usando $P_n(\mathbb{w})$ vamos amostrar $\mathbb{w}_{i_1}, \ldots, \mathbb{w}_{i_K}$; garantindo que $\mathbb{w}_o$ não está entre elas.

Amostragem negativa

$$(\mathbb{w}_I, \mathbb{w}_O)$$

Exemplo positivo

$$(\mathbb{w}_I, \mathbb{w}_{i_1}), \ldots, (\mathbb{w}_I, \mathbb{w}_{i_K})$$

Exemplos negativos

Amostragem negativa: o modelo (i)

$$p(D = 1 \mid \mathbb{w}_I, \mathbb{w}) = \sigma(v'_{\mathbb{w}}\,.^T\ h)$$

probabilidade do par $(\mathbb{w}_I, \mathbb{w})$ ocorrer no corpus

$$p(D = 0 \mid \mathbb{w}_I, \mathbb{w})$$

probabilidade do par $(\mathbb{w}_I, \mathbb{w})$ não ocorrer no corpus

O objetivo de treinamento agora é maximizar as probabilidades

$$p(D = 1 \mid \mathbb{w}_I, \mathbb{w}_O),\ p(D = 0 \mid \mathbb{w}_I, \mathbb{w}_{i_1}), \ldots,\ p(D = 0 \mid \mathbb{w}_I, \mathbb{w}_{i_K})$$

Amostragem negativa: o modelo (ii)

Assim, vamos minimizar a seguinte função erro:

$$
\begin{aligned}
E &= -\log(p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) \,.\, \prod_{s=1}^{K} p(D=0 \mid \mathrm{w}_I, \mathrm{w}_{i_s})) \\
&= -(\log p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) + \log(\prod_{s=1}^{K} p(D=0 \mid \mathrm{w}_I, \mathrm{w}_{i_s}))) \\
&= -(\log p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) + \sum_{s=1}^{K} \log(p(D=0 \mid \mathrm{w}_I, \mathrm{w}_{i_s}))) \\
&= -\log \sigma(v'_{\mathrm{w}_O} \,.^T\, h) - \sum_{s=1}^{K} \log(\sigma(-v'_{\mathrm{w}_{i_s}} \,.^T\, h))
\end{aligned}
$$

Amostragem negativa: atualização

$$v'_{\mathbb{w}_O}{}^{(new)} = v'_{\mathbb{w}_O}{}^{(old)} - \eta\,(\sigma(v'_{\mathbb{w}_O}\,.^T\ h) - 1)\,h \tag{23}$$

$$v'_{\mathbb{w}_{i_s}}{}^{(new)} = v'_{\mathbb{w}_{i_s}}{}^{(old)} - \eta\,\#(i_s)\,\sigma(v'_{\mathbb{w}_{i_s}}\,.^T\ h)\,h \tag{24}$$

$$W^{(new)} = W^{(old)} - \eta\,xEH^T \tag{25}$$

Onde

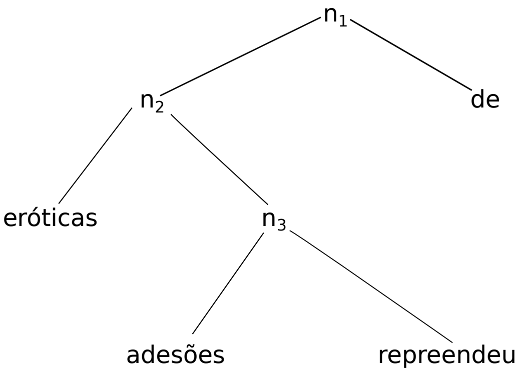
$$EH = (\sigma(v'_{\mathbb{w}_O}\,.^T\ h) - 1)\,v'_{\mathbb{w}_O} + \sum_{s=1}^{K} \sigma(v'_{\mathbb{w}_{i_s}}\,.^T\ h)v'_{\mathbb{w}_{i_s}}$$

e $\#(i_s)$ é a contagem de i_s.

The basic idea is to form a hierarchical description of a word as a sequence of $O(\log |V|)$ *decisions, and to learn to take these probabilistic decisions instead of directly predicting each word's probality.(Morin and Bengio 2005, p.247)*

Softmax hierárquico: árvore de Huffman

[(de, 24480774), (repreendeu, 401), (eróticas, 424), (adesões, 400)]



{ de:1, eróticas: 00, adesões:010, repreendeu:011}.

Softmax hierárquico: notação

- Dado um vocabulário V, temos $|V| - 1$ vértices que não são folhas (*nós internos*).
- O caminho da raiz até a folha vai ser usado para estimar a probabilidade da palavra representada pela folha.
- Dado a palavra $\mathbb{w}$, $L(\mathbb{w})$ é o comprimento do caminho da raiz até $\mathbb{w}$ e $n(\mathbb{w}, 1), \ldots, n(\mathbb{w}, L(\mathbb{w}) - 1)$ são todos os nós internos no caminho da raiz até $\mathbb{w}$.
- $H(\mathbb{w})_j$ vai denotar o j-ésimo número de código de $\mathbb{w}$. 1 vai codificar esquerda e -1 vai codificar direita.

{ de:-1, eróticas: 11, adesões:$1 - 11$, repreendeu:$1 - 1 - 1$}.

Softmax hierárquico: o modelo

Dado uma observação $(\mathbb{w}_I, \mathbb{w}_O)$ os cálculos de x e h vão ser os mesmos; W ainda guarda os *input vectors*. No lugar dos *output vectors* temos:

$$v'_{n(\mathbb{w},j)}$$

para cada nó interno $n(\mathbb{w}, j)$.

Softmax hierárquico

Dado $\mathbb{w}_I$, cada nó interno tem uma probabilidade associada de ir para a esquerda ou para a direita:

$$p(n(\mathbb{w}, j), esquerda) = \sigma({v'_{n(\mathbb{w},j)}}^T \, . \, h) \tag{26}$$

$$p(n(\mathbb{w}, j), direita) = \sigma(-{v'_{n(\mathbb{w},j)}}^T \, . \, h) \tag{27}$$

Desse modo, temos que dado a ocorrência de $\mathbb{w}_I$ a probabilidade da palavra $\mathbb{w}$ ser $\mathbb{w}_O$ é:

$$p(\mathbb{w} = \mathbb{w}_O \mid \mathbb{w}_I) = \prod_{j=1}^{L(\mathbb{w}_O)-1} \sigma(H(\mathbb{w}_O)_j . {v'_{n(\mathbb{w}_O,j)}}^T \, . \, h) \tag{28}$$

Softmax hierárquico

Como queremos maximizar $p(\mathbb{w} = \mathbb{w}_O \mid \mathbb{w}_I)$ a função de erro que queremos minimizar é

$$E = -\log p(\mathbb{w} = \mathbb{w}_O \mid \mathbb{w}_I) \tag{29}$$

Softmax hierárquico: atualização

$$v'_{n(\mathrm{w}_O,j)}{}^{(new)} = v'_{n(\mathrm{w}_O,j)}{}^{(old)} - \eta\,(\sigma(v'_{n(\mathrm{w}_O,j)}{}^{T}\, h) - t_j)\, h \tag{30}$$

$$W^{(new)} = W^{(old)} - \eta\, x EH^T \tag{31}$$

Onde,

$$t_j = \begin{cases} 1, & \text{se } H(\mathrm{w})_j = 1 \\ 0, & \text{se } H(\mathrm{w})_j = -1 \end{cases}$$

$$EH = \sum_{j=1}^{L(\mathrm{w}_O)-1} (\sigma(v'_{n(\mathrm{w},j)}{}^{T}\, h) - t_j) v'_{n(\mathrm{w},j)}$$

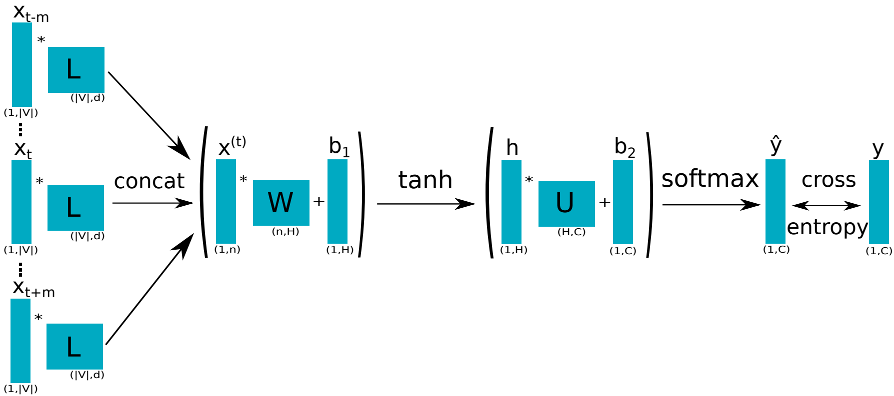
NER: named entity recognition (extração de entidades nomeadas). Dada uma sentença queremos saber quais entidades ocorrem nela, assim temos de um lado sentenças e de outro categorias (pessoa, localização, organização).

Modelo de janela : observações do tipo

$$([\mathbb{w}_{t-m}, \ldots, \mathbb{w}_t, \ldots, \mathbb{w}_{t+m}], c)$$

("A comissão europeia discorda do tratado proposto", ORG)

Exemplo de aplicação: NER



Como avaliar o modelo?

- **avaliação intrínseca :** avaliação rápida feita numa tarefa intermediaria: e.g., predição de analogias semânticas.
- **avaliação extrínseca :** avaliação feita numa tarefa real de NLP: e.g., NER.

Como avaliar o modelo? Avaliação intrínseca

$\mathbb{w}_a$ está para $\mathbb{w}_b$ assim como $\mathbb{w}_c$ está para _

rapaz moça irmãos irmãs

trabalhou trabalham gerar geram

homem mulher rei rainha

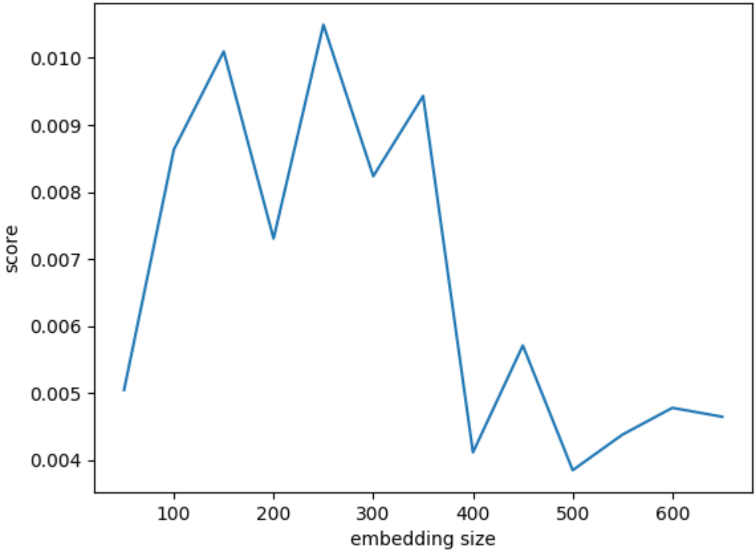
$$\mathbb{w}_a : \mathbb{w}_b \rightarrow \mathbb{w}_c :?$$

$$\mathbb{w}_d = argmax_x \frac{(b - a + c).^T x}{||b - a + c||} \tag{32}$$

$$\mathbb{w}_d = argmax_x \; b.^T x - a.^T x + c.^T x \tag{33}$$

Qual a palavra cuja representação eh similar a $\mathbb{w}_b$ e $\mathbb{w}_c$ e dissimilar de $\mathbb{w}_a$?

Avaliação intrínseca



Implementação: https://github.com/felipessalvatore/Word2vec-pt

Categoria	Word2vec-pt	Gensim
capital-common-countries	(15/306)	(8/90)
capital-world	(7/1155)	(4/173)
curency	(0/106)	(0/54)
city-in-state	(1/1171)	(1/208)
family	(41/342)	(132/306)
gram1-adjective-to-adverb	(1/552)	(5/380)
gram2-opposite	(0/182)	(3/90)
gram3-comparative	(5/30)	(12/30)
gram4-superlative	(3/20)	(4/6)
gram5-present-participle	(0/702)	(61/462)
gram6-nationality-adjective	(0/412)	(37/739)
gram7-past-tense	(8/1056)	(124/506)
gram8-plural	(1/992)	(25/380)
gram9-plural-verbs	(9/552)	(29/306)

Bibliografia

- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representation in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111-3119.
- Morin, F., Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246-252.
- Rong, X. (2016). Word2vec Parameter Learning Explained. *arXiv preprint arXiv:1411.2738*.