# Modelos de Linguagens Neurais: Word2Vec

Marta Talitha Carvalho Freire Mendes
Orientador: Patrick Marques Ciarrelli

Artigo:A Primer on Neural Network Models for Natural Language Processing (2015) - Yoav Goldberg


Universidade Federal do Espírito Santo

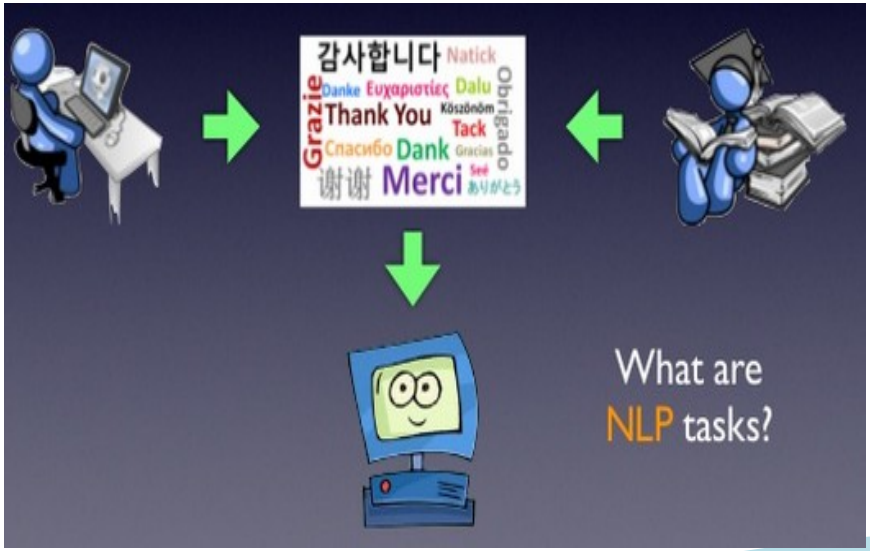
# Índice

- Introdução
- Tarefas PLN
- Como representar as palavras
- Arquitetura da rede neural
- Alguns Casos da Literatura
- Ferramentas
- Exemplo prático

# O que é PLN?

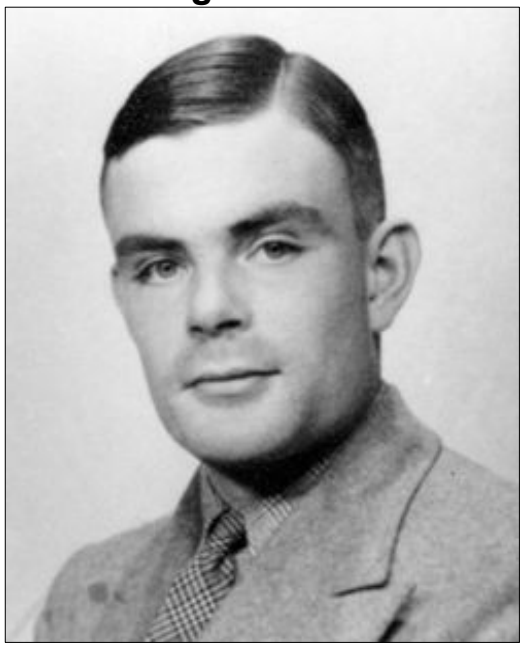
"É um campo da inteligência artificial preocupada com a **interação** entre **computadores** e a **linguagem natural humana**."

Ela Kumar (2011)



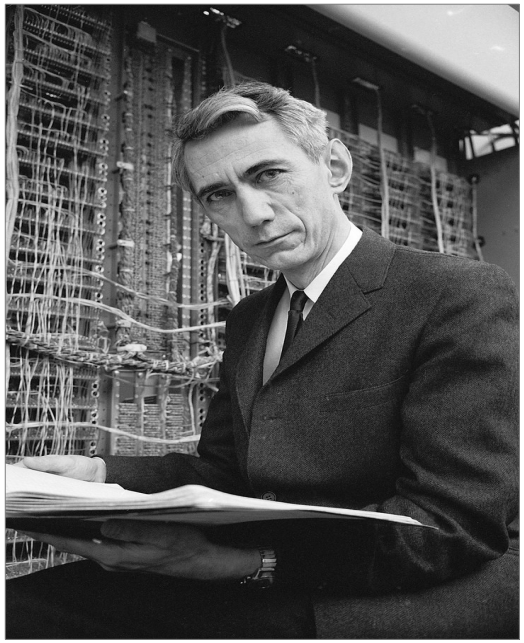
# O que é PLN?

**Alan Turing**



1913-1954
Matemático
Cientista da computação

**Claude Shannon**



1916-2001
Matemático
Engenheiro eletrônico

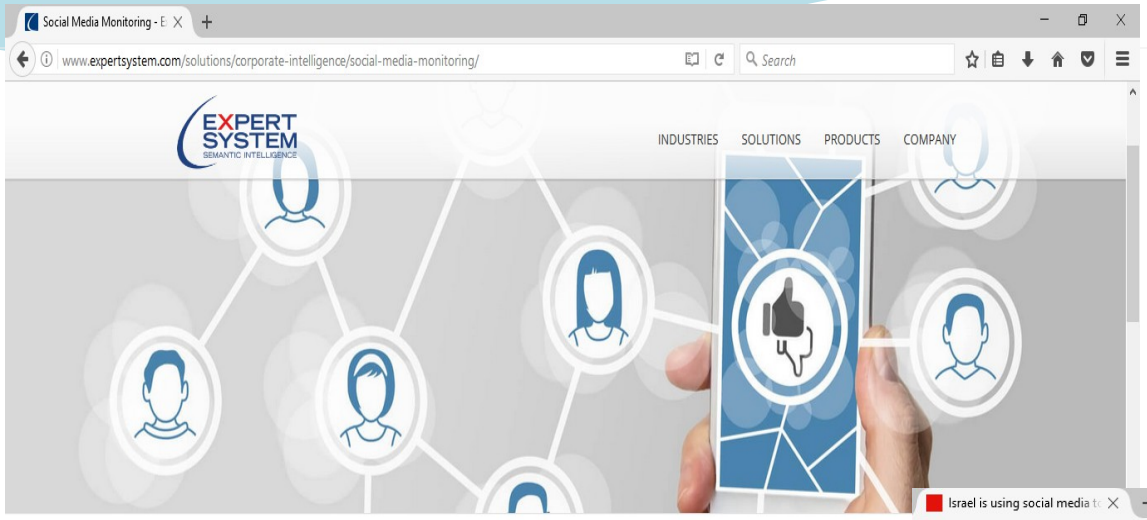
**Noam Chomsky**

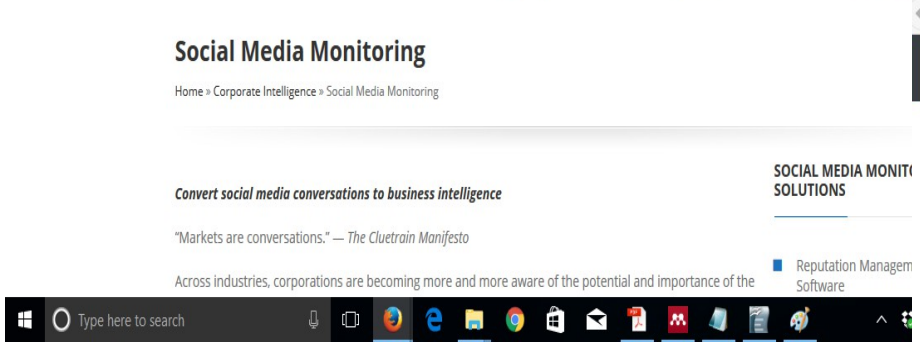
1928
Linguista

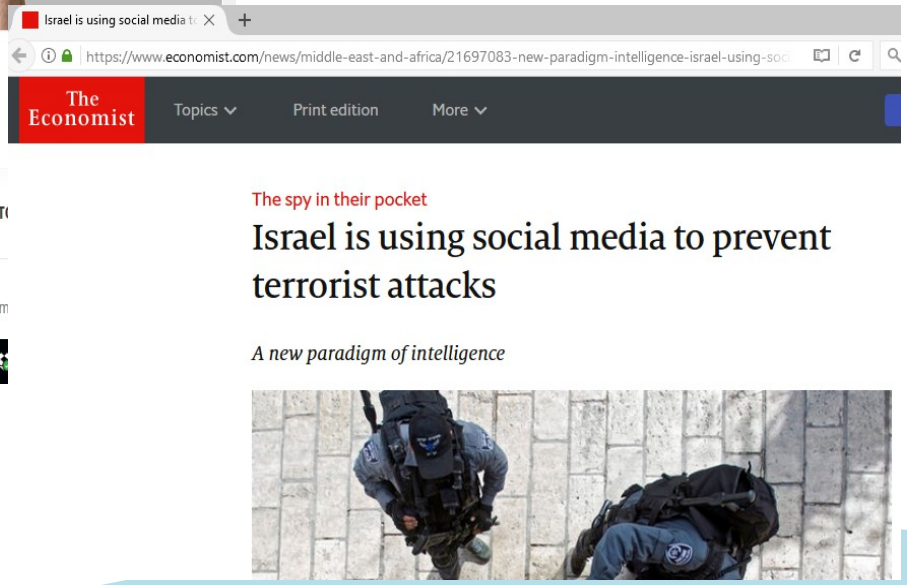
# Tarefas PLN

- Tradução automática
- Análise de sentimento
- Identificação léxico-sintática
- Corretor automático
- Sistemas de pergunta e respostas
- Reconhecimento da fala
- Mineração de texto
- Monitoramento das redes sociais
- etc

# Tarefas: monitoramento das redes sociais



# Tarefas: Responder perguntas


veja.com

Naufrágios Privatizações Revista VEJA VEJA Comer & Beber

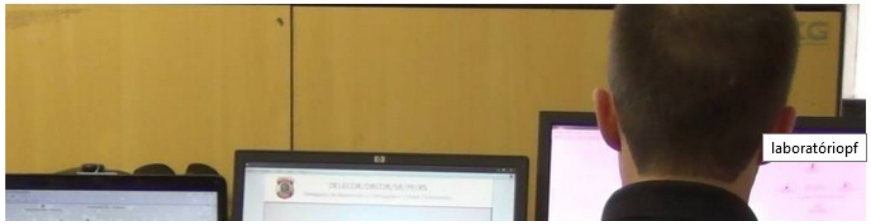
Brasil

## Watson, a inteligência artificial a serviço da PF gaúcha

Nome do sistema da IBM, usado nas investigações da 'Lava Jato gaúcha', é referência ao Dr. Watson, personagem que auxilia Sherlock Holmes nas investigações

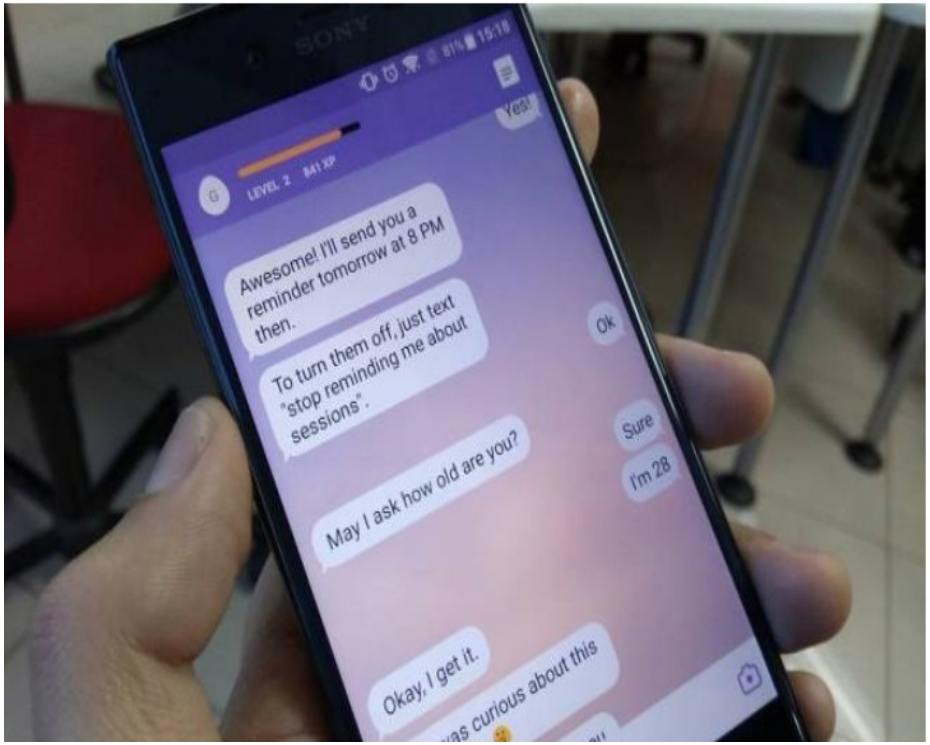
Por **Paula Sperb**

25 ago 2017, 19h32 - Publicado em 25 ago 2017, 18h55



# Tarefas: ChatBot

Replika é um chatbot que se propõe a conversar com o usuário e se alimenta de diversas outras informações a fim de evoluir aos poucos conforme o sistema conhece o usuário e se acostuma com sua personalidade.



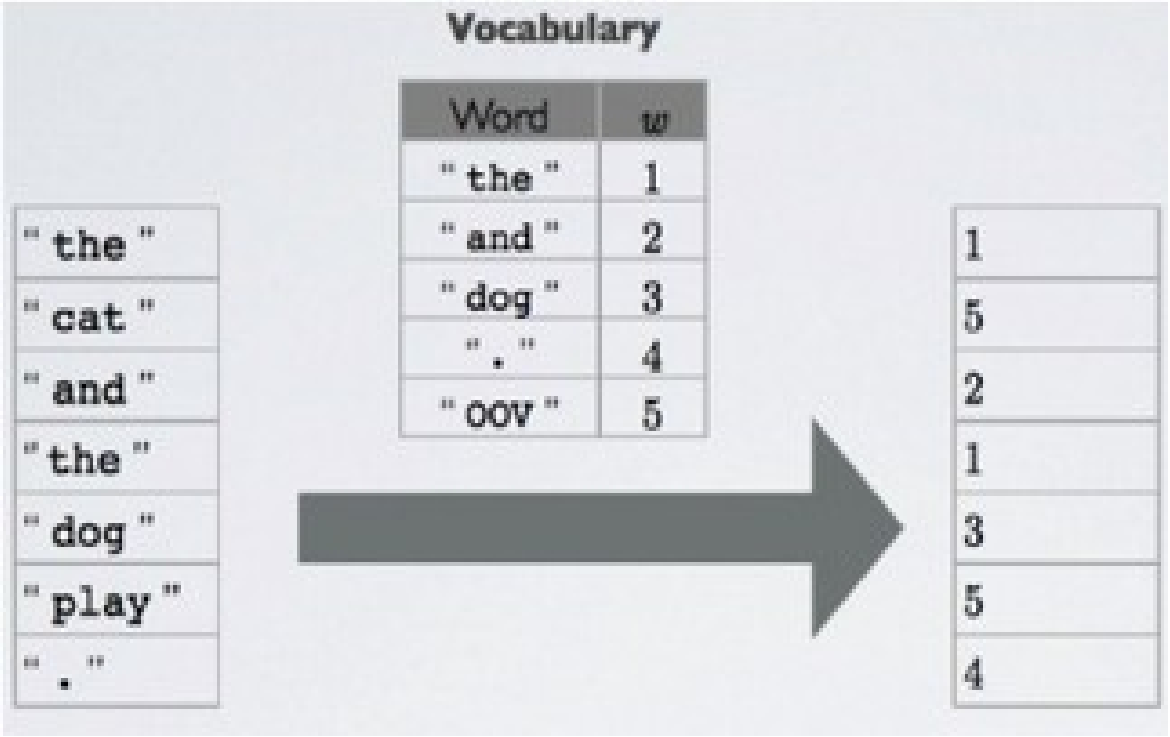
# Como representar palavras

Algumas formas de representar as palavras:

- Bag-of-words
- One-hot
- Vetor de probabilidades
- Word embeddings
- etc

# Dicionário

Mapeiam **todas** as palavras do texto(normalmente lematizadas) para seu único id.



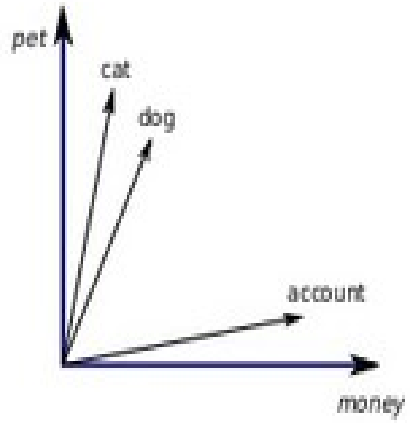
# One-hot

As palavras ou características são representadas como vetor de **alta dimensão**:

- dog = (0,0,0,0,1,0,0,0,0,....)
- cat = (0,0,0,0,0,0,0,1,0,....)
- eat = (0,1,0,0,0,0,0,0,0,....)

- dogs = (1,0,0,0,0,0,0,....)
- dog = (0,0,0,0,1,0,0,....)

- laptop = (0,0,1,0,0,0,0,0,0,0....)
- computador = (0,0,0,0,0,0,0,0,0,1....)

# Bag of Words

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

```
[
    "John",
    "likes",
    "to",
    "watch",
    "movies",
    "Mary",
    "too",
    "also",
    "football",
    "games"
]
```

# Term frequency

(1) John likes to watch movies. Mary likes movies too.

(2) John also likes to watch football games.

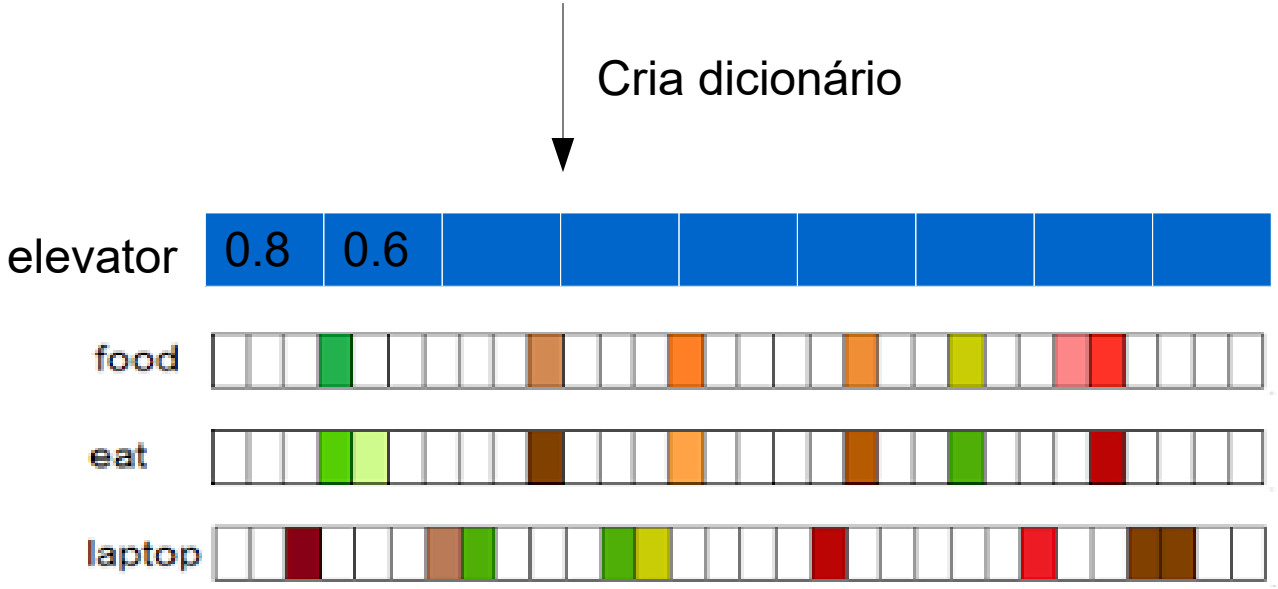
```
(1) [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
(2) [1, 1, 1, 1, 0, 0, 0, 1, 1, 1]
```

```
[
    "John",
    "likes",
    "to",
    "watch",
    "movies",
    "Mary",
    "too",
    "also",
    "football",
    "games"
]
```

# Distribuição de probabilidades

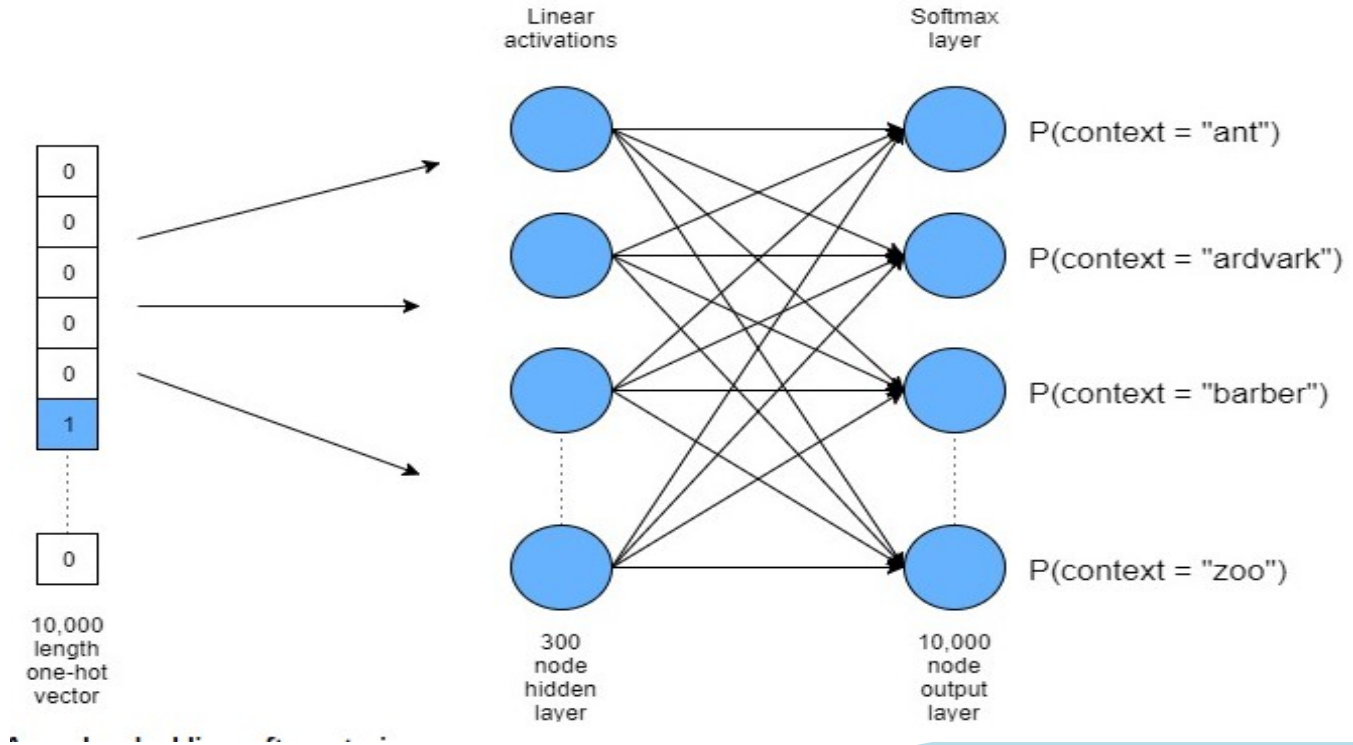
As palavras ou características são representadas como vetor de **alta dimensão**:

**W= 5** The left elevator goes down to the second floor



# Word embbeding

Os vetores são obtidos da representação interna de modelos de redes neurais de texto. A ideia é treinar um classificador com um corpus de texto, a fim de cada palavra tenha uma dimensão que a caracterize suficientemente (Mikolov et al 2013):



# Word embbeding

Dada a entrada (“an”, “efficient”, “method”, “for”, “high”, “quality”, “distributed”, “vector”), nós queremos maximizar a probabilidade de obter a saída “learning”


**Efeito de janelamento:**
*Janelas muito grandes* tendem a produzir similaridades mais  a nivel de “tópico” (estão relacionadas ao mesmo evento)
Exemplo: (dog,  bark , leash)  (walked, run, walking)

*Janelas muito pequenas* tendem a produzir similaridades morfológica e funcionais.
Exemplo: (poodle, pitbull,rottweiler) (walking,running,approching)

Goldberg (2015)

# Dimensionalidade

How many dimensions should we allocate for each feature? Unfortunately, there are no theoretical bounds or even established best-practices in this space. Clearly, the dimensionality should grow with the number of the members in the class (you probably want to assign more dimensions to word embeddings than to part-of-speech embeddings) but how much is enough? In current research, the dimensionality of word-embedding vectors range between about 50 to a few hundreds, and, in some extreme cases, thousands. Since the dimensionality of the vectors has a direct effect on memory requirements and processing time, a good rule of thumb would be to experiment with a few different sizes, and choose a good trade-off between speed and task accuracy.

Entre 50 a centenas => testando

# Word embbeding

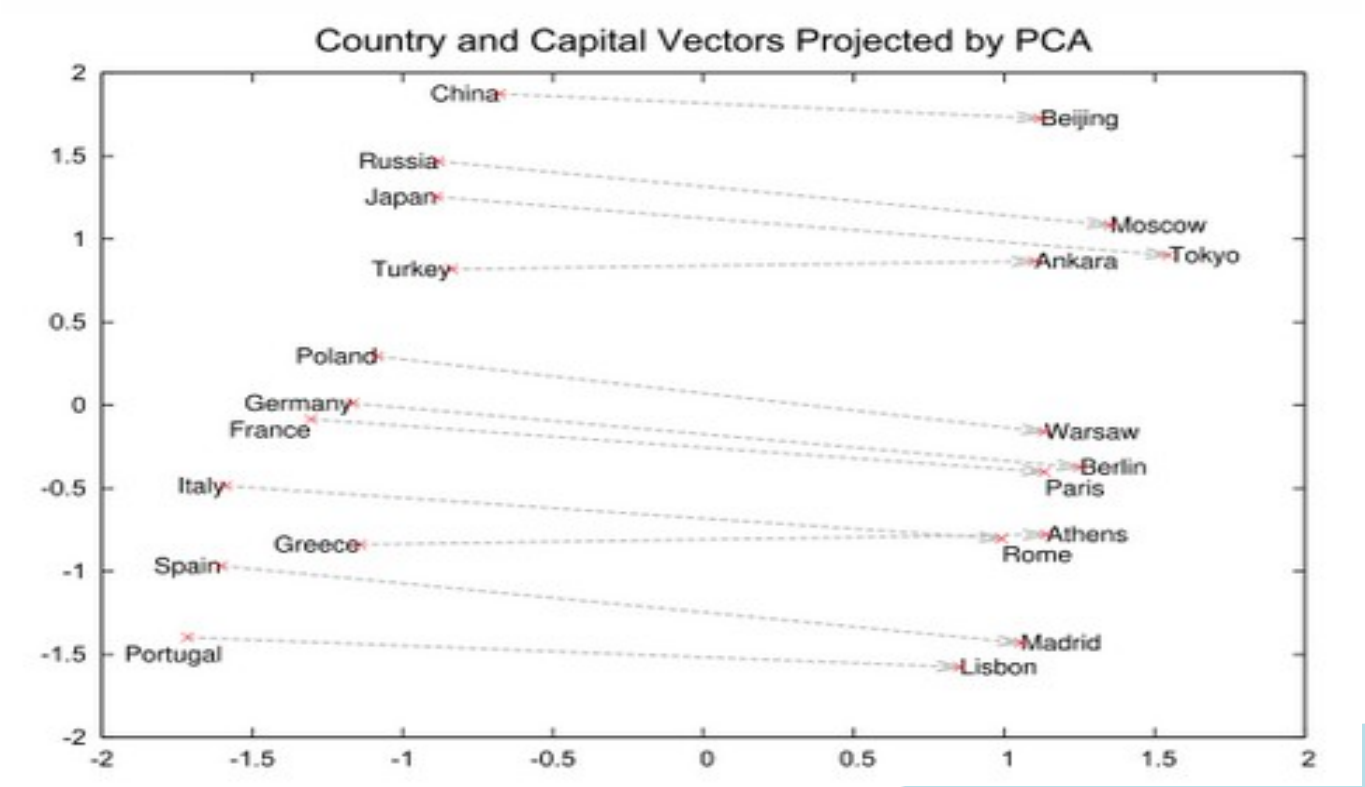
Palavras que ocorrem em contexto similares tem similares word embbeding. Por exemplo, Se os dados de treino para o modelo de linguagem contém os seguintes n-grams:

- *but the cute dog jumped*
- *but the cute cat jumped*
- *child hugged the cat closely*
- *child hugged the dog closely*
- *like to watch cat videos*
- *like to watch dog videos*

Então o modelo de linguagem será beneficiado pelo conhecimento de **dog** e **cat** que ocorrem em contextos similares e são “intercambiáveis”.

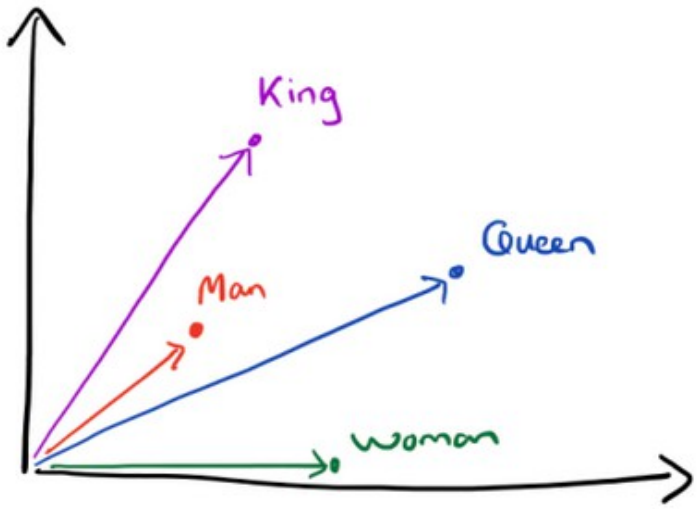
# Word embbeding

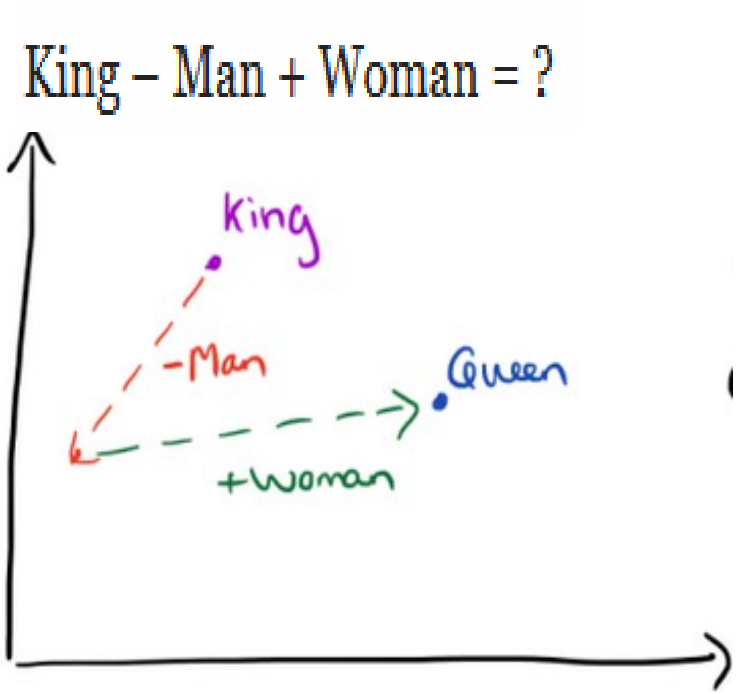
Um dos efeitos da aplicação dessa técnica é a captura de relação entre conceitos: a figura ilustra a habilidade de organizar os conceitos automaticamente e aprender implicitamente o relacionamento entre eles.



# Word embbeding

Operações com vetores



# Word embbeding

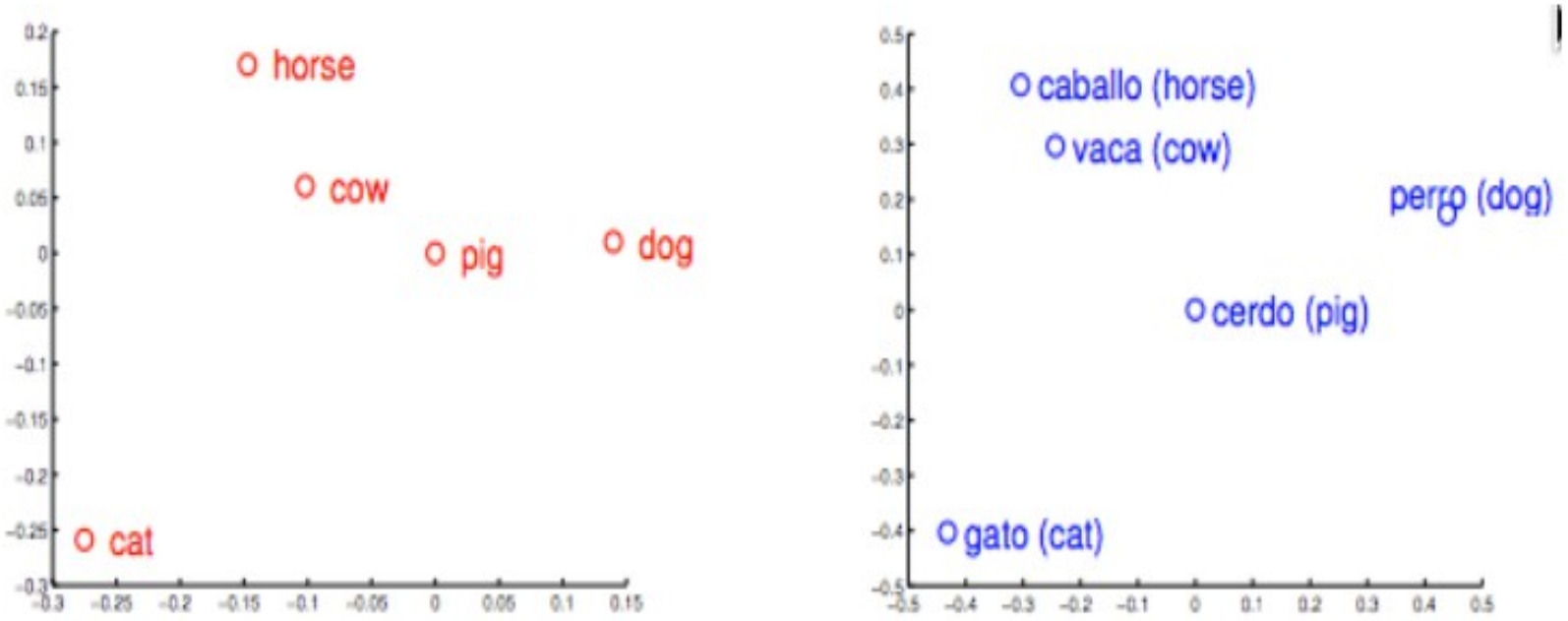



Figure 15.12: Word embeddings for words in English and Spanish (projected down to two dimensions). Similar words have similar representations and appear close to each other in the graph. Words and their translations have similar configurations (taken from Mokolov (2013)).

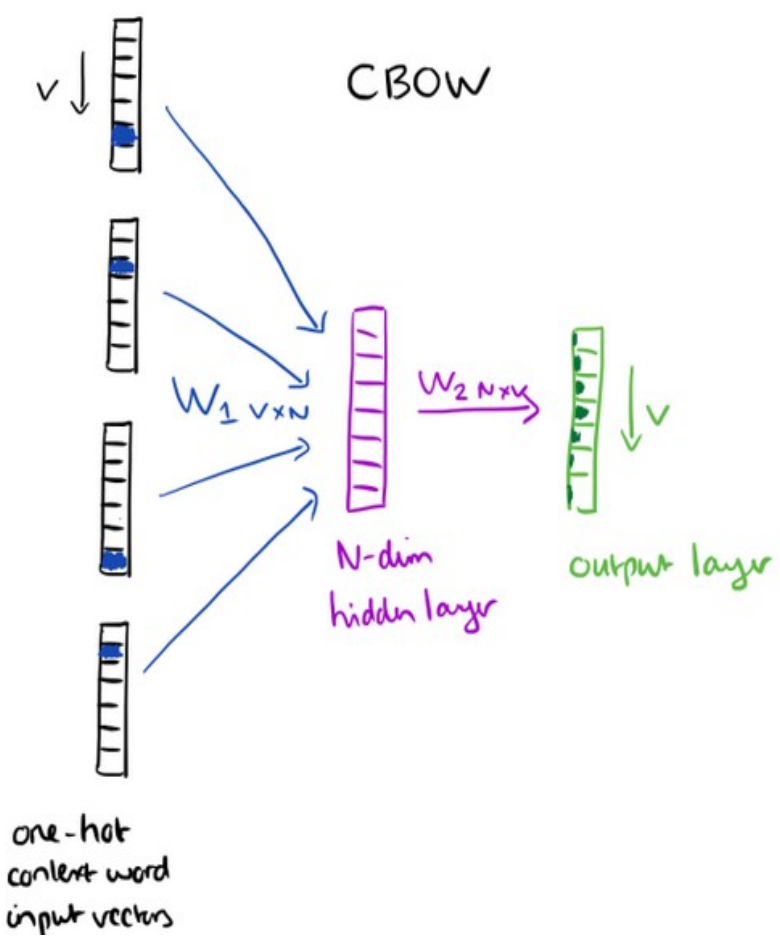
# Word embbeding

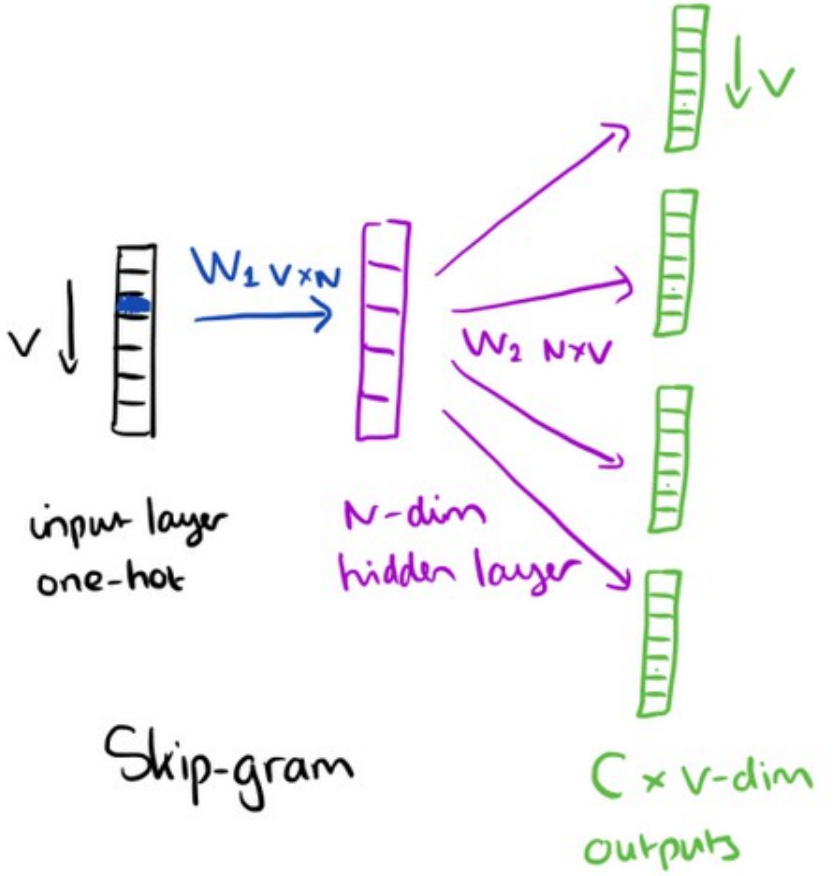
| | | |
|---|---|---|
| Glove | Stanford | 2014 |
| Word2Vec | Google | 2013 |
| FastText | Facebook | 2016 |

- Word2vec foi criado por um time de pesquisadores da Google liderado por Tomas Mikolov.

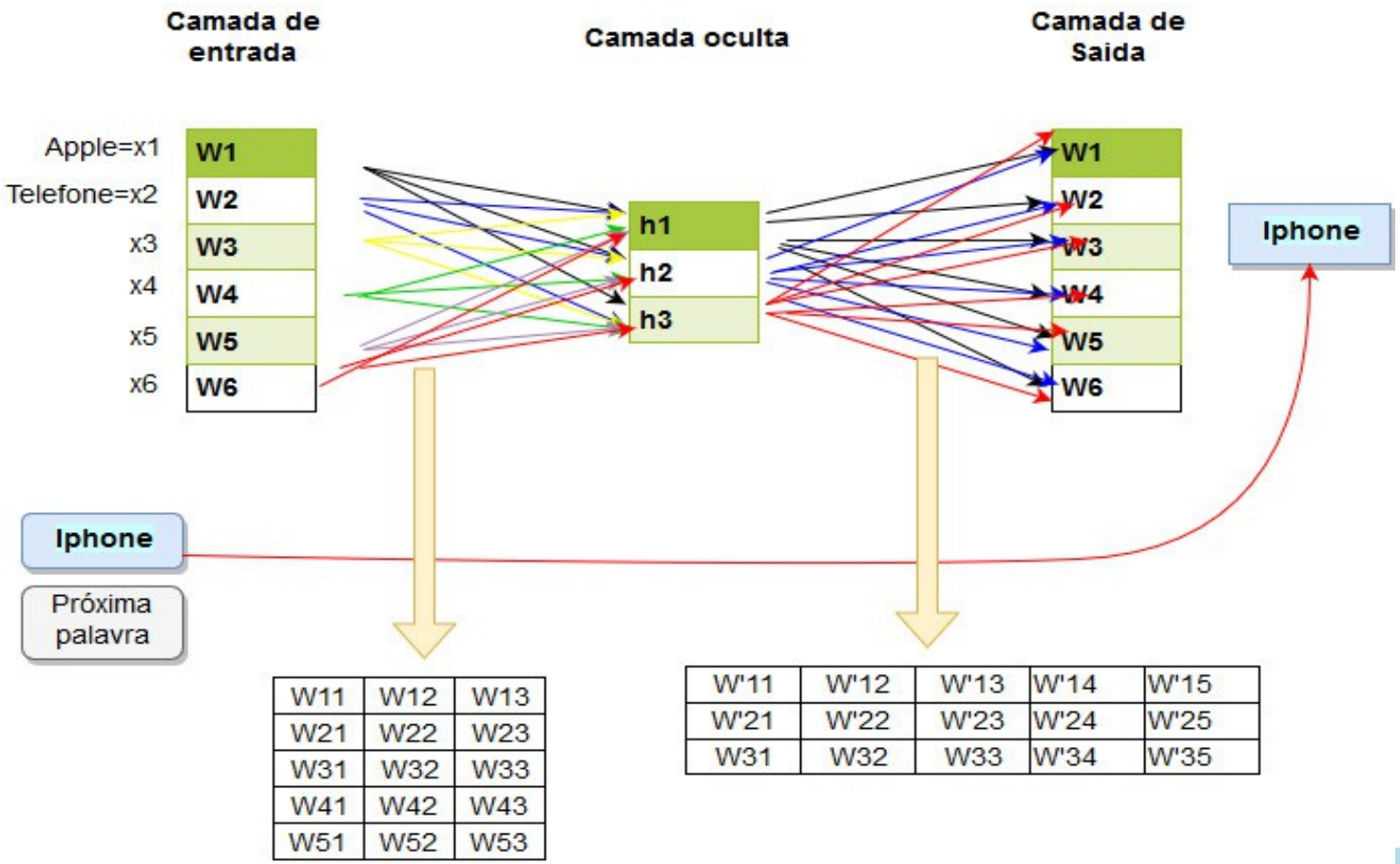
# Arquitetura redes neurais

**P(w/c)**



# Arquitetura redes neurais



# Passo 1

**Forward propagation:** combinação linear dos pesos com a entrada

$$\begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix} = W^T x = \begin{bmatrix} W11 & W12 & W13 & W14 & W15 \\ W21 & W22 & W23 & W24 & W25 \\ W31 & W32 & W33 & W34 & W35 \\ W41 & W42 & W43 & W44 & W45 \\ W51 & W52 & W53 & W54 & W55 \end{bmatrix} \begin{bmatrix} x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{bmatrix}$$

- Exemplo, seja uma rede com 5 entradas, então temos: $\mathbf{x} \in \mathbb{R}^{d_{in}}, \quad \mathbf{W} \in \mathbb{R}^{d_{in} \times d_{out}}$

$$h1 = (W_{11^*} x_1 + W_{21^*} x_2 + W_{31^*} x_3 + W_{41^*} x_4 + W_{51^*} x_5)$$
$$h2 = (W_{12^*} x_1 + W_{22^*} x_2 + W_{32^*} x_3 + W_{42^*} x_4 + W_{52^*} x_5)$$
$$h3 = (W_{13^*} x_1 + W_{23^*} x_2 + W_{33^*} x_3 + W_{43^*} x_4 + W_{52^*} x_5)$$

Some care has to be taken in the way the random initialization is performed. The method used by the effective word2vec implementation (Mikolov et al., 2013; Mikolov, Sutskever, Chen, Corrado, & Dean, 2013) is to initialize the word vectors to uniformly sampled random numbers in the range $[-\frac{1}{2d}, \frac{1}{2d}]$ where $d$ is the number of dimensions. Another option is to use *xavier initialization* (see Section 6.3) and initialize with uniformly sampled values from $\left[-\frac{\sqrt{6}}{\sqrt{d}}, \frac{\sqrt{6}}{\sqrt{d}}\right]$.

Pesos com distribuição uniforme => produz “bons” vetores

# Passo 2

**Forward propagation:** combinação linear dos pesos com a camada oculta

$$Net(O) = W^T h = \begin{bmatrix} W'11 & W'21 & W'31 \\ W'12 & W'22 & W'32 \\ W'13 & W'23 & W'33 \\ W'14 & W'24 & W'34 \\ W'15 & W'25 & W'35 \end{bmatrix} \begin{bmatrix} h1 \\ h2 \\ h3 \end{bmatrix}$$

- Exemplo:

$$Net(O1) = u1 = (W'_{11^*} h_1 + W'_{21^*} h_2 + W'_{31^*} h_3)$$
$$Net(O2) = u2 = (W'_{12^*} h_1 + W'_{22^*} h_2 + W'_{32^*} h_3)$$
$$Net(O3) = u3 = (W'_{13^*} h_1 + W'_{23^*} h_2 + W'_{33^*} h_3)$$
$$Net(O4) = u4 = (W'_{14^*} h_1 + W'_{24^*} h_2 + W'_{34^*} h_3)$$
$$Net(O5) = u5 = (W'_{15^*} h_1 + W'_{25^*} h_2 + W'_{35^*} h_3)$$

# Passo 3

**Forward propagation:** Probabilidade da saída (Usando o softmax)

Exemplo da saída O1, O2 e O4:

$$Out(O1) = y1 = \frac{e^{Net(O1)}}{\left(e^{Net(O1)} + e^{Net(O2)} + e^{Net(O3)} + e^{Net(O4)} + e^{Net(O5)}\right)} = \frac{e^{u1}}{\left(e^{u1} + e^{u2} + e^{u3} + e^{u4} + e^{u5}\right)}$$

$$Out(O2) = y2 = \frac{e^{Net(O2)}}{\left(e^{Net(O1)} + e^{Net(O2)} + e^{Net(O3)} + e^{Net(O4)} + e^{Net(O5)}\right)} = \frac{e^{u2}}{\left(e^{u1} + e^{u2} + e^{u3} + e^{u4} + e^{u5}\right)}$$

$$Out(O4) = y4 = \frac{e^{(Net(O4))}}{\left(e^{(Net(O1))} + e^{(Net(O2))} + e^{(Net(O3))} + e^{(Net(O4))} + e^{(Net(O5))}\right)} = \frac{e^{u4}}{\left(e^{u1} + e^{u2} + e^{u3} + e^{u4} + e^{u5}\right)}$$

# Passo 3

**Forward propagation:** Probabilidade da saída (Usando o softmax)

Podemos generalizar a probabilidade de saída como:

$$P(W_j/W_i) = y_j = \frac{e^{Net(O_j)}}{\left(\sum_{j'=1}^{V} e^{Net(O_{j'})}\right)} = \frac{e^{u_j}}{\left(\sum_{j'=1}^{V} e^{u_{j'}}\right)}$$

**Wj = palavra alvo ou "atual"**
**Wi = palavras do contexto**

# Passo 4

**Backpropagation:** Cálculo do erro

Função de perda ou custo pode ser definida como a maximização:

$$\max P(Wo/Wi) = \max(y_j) = \max(\log(y_j))$$

# Passo 4

**Backpropagation:** Cálculo do erro

**Exemplificando o erro da saida 4:**

$$E(O_4) = P(W_{O4}/W_i) = \log\left(\frac{e^{u4}}{(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})}\right)$$

$$\log e^{(e^{u4})} - \log e^{(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})}$$

$$u_4 - \log e^{(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})}$$

# Passo 4

## **Backpropagation:** Cálculo do erro residual

Como queremos minimizar o erro então:

$$E = -\log P(Wo/Wi)$$

$$E(O_4) = \log e^{(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})} - u_4$$

- A função de perda também pode ser entendida como um caso especial da **medida cross entropy**, que é uma forma de medir a "distância" entre duas distribuições probabilísticas. Cross entropy não é simétrica.

Podemos generalizar:

$$E = \log \sum_{j'=1}^{V} e^{u_{j'}} - u_{j^*}$$

Onde o j* é o index da palavra atual

# Passo 5

## Backpropagation: Calculando a derivada do erro com relação a saída u4

A fim de aprender da função softmax via descida de gradiente, calcula-se a derivada:

$$\frac{(d(E(O_4)))}{(d(u_4))} = \frac{(d(\ln(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5}) - u_4))}{(d(u_4))}$$

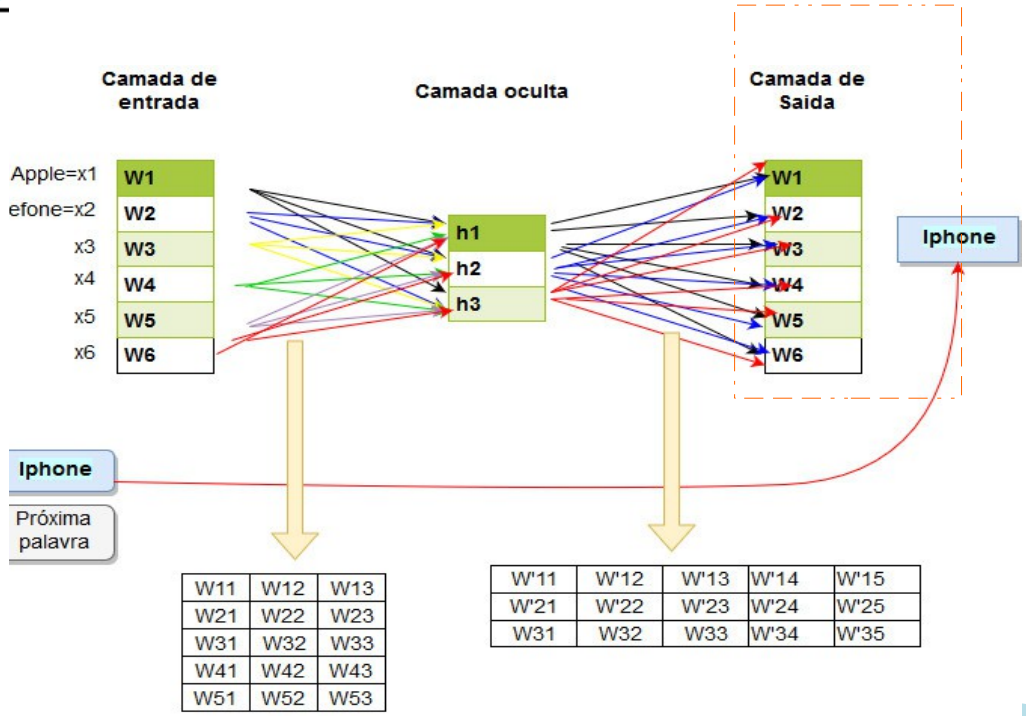
*aplicando regra da soma e diferença*

$$\frac{(d(\ln(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})))}{(d(u_4))} - \frac{(d(u_4))}{(d(u_4))}$$

$$\frac{e^{u4}}{(e^{u1}+e^{u2}+e^{u3}+e^{u4}+e^{u5})} - 1$$

$$y_4 - 1$$

$$\frac{(d(E(O_4)))}{(d(u_4))} = y_4 - 1$$



# Passo 5

**Backpropagation:** Calculando o gradiente do erro com relação a saída

Generalizando:

$$\frac{(dE)}{(du_j)} = \frac{\left(d\left(\log \sum_{j'=1}^{V} e^{u_{j'}} - u_j^*\right)\right)}{(d(u_j))} = \frac{(e^{u_j})}{\left(\sum_{j'=1}^{V} e^{u_{j'}}\right)} - \frac{(d(u_j^*))}{(d(u_j))} = y_j - t_j := e_j$$

$$t_j = 1 \; If \; (t_j = t_j^*) \; else \; t_j = 0$$

Dois casos: Se t_j= 1 quando nosso nó j representa nossa palavra de saída, caso contrário t_j=0.

e_j => erro total
t_j => valor alvo
y_j => probabilidade calculada

# Passo 6

**Backpropagation:** Pegando o gradiente do erro com relação ao peso W'11

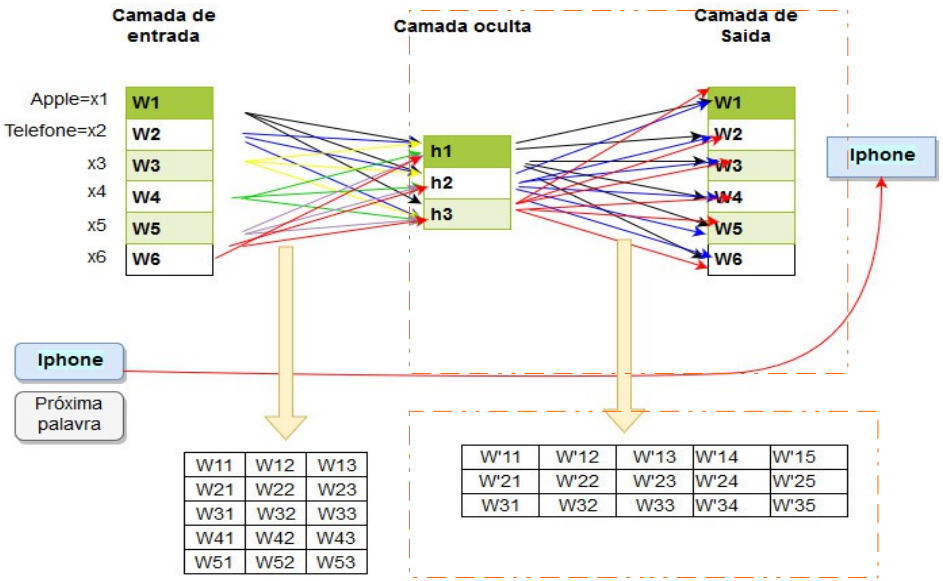
$$\frac{(dE(O_1))}{(d(W'_{11}))} = \frac{(dE(O_1))}{(d(u_1))} X \frac{du_1}{(dW'_{11})}$$

*aplicando a regra da cadeia*

$$\frac{(dE(O_1))}{(d(u_1))} = (y_1 - 0) = e_1$$

$$\frac{du_1}{(dW'_{11})} = \frac{(d((W'_{11})h1 + (W'_{21})h2 + (W'_{31})h3))}{(dW'_{11})} = h_1$$

$$\frac{(dE(O_1))}{(d(W'_{11}))} = \frac{(dE(O_1))}{(d(u_1))} X \frac{(d(u_1))}{(d(W'_{11}))} = e_1 h_1$$




Atualizando o peso de W'11 **(da camada oculta para camada de saída)**:

$$(W'_{11})^{New} = W'_{11} - \eta \frac{(dE(O_1))}{(dW'_{11})} = W'_{11} - \eta(e_1 * h_1)$$

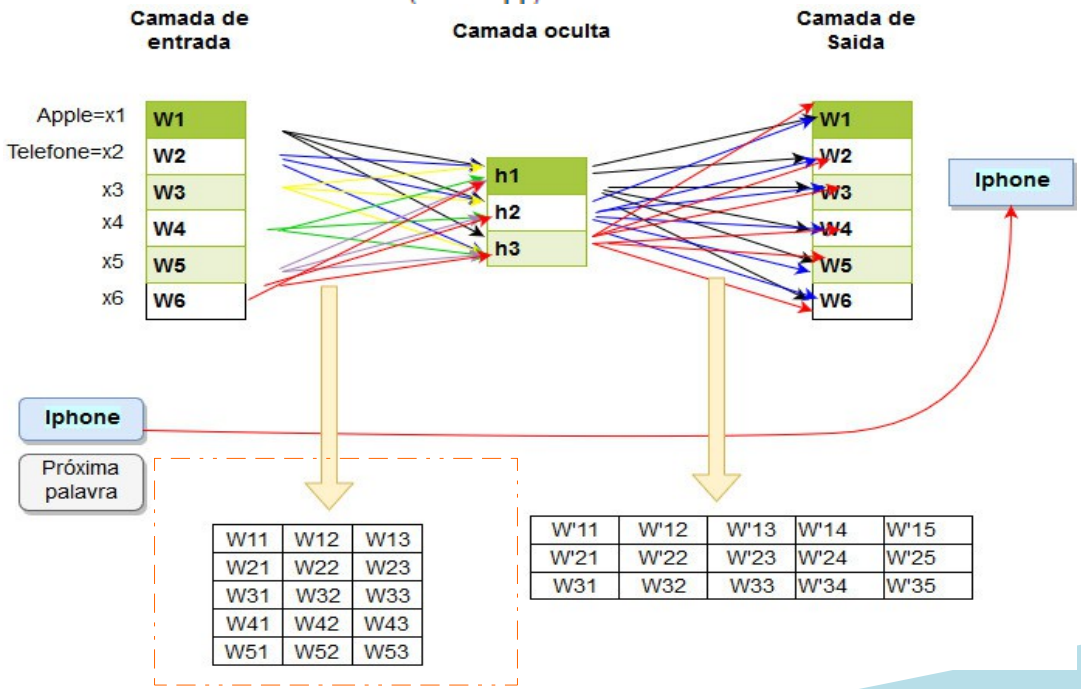
A taxa de aprendizagem compreende range [0 a 1]. Similarmente podemos atualizar o peso de W12,W13,W21,...W53. Repita todos os passos até obter pesos de interesse.

# Passo 7

**Backpropagation:** Atualizando o peso W11

- Atualizando os pesos da **camada de entrada para oculta:**

$$(W_{11})^{New} = W_{11} - \eta \frac{(dE(O_1))}{(dW_{11})} = W_{11} - \eta(e_1 * h_1)$$

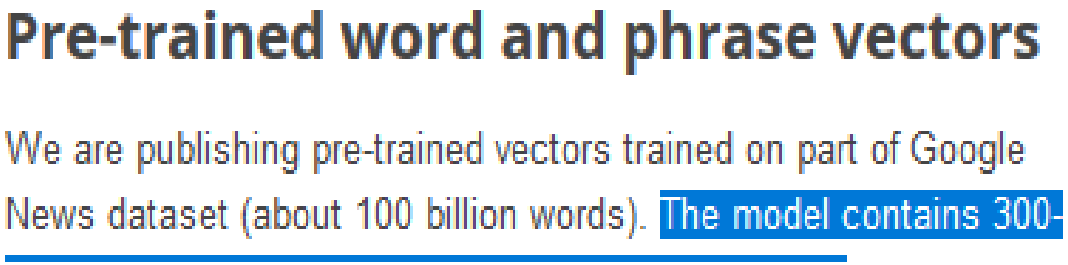



**Atualizar até encontrar pesos adequados.**

# Benchmark

**Base treinada**

- 300 features é que o Google usa em seu modelo treinado:




- TEXT8 - 17 milhões de palavras

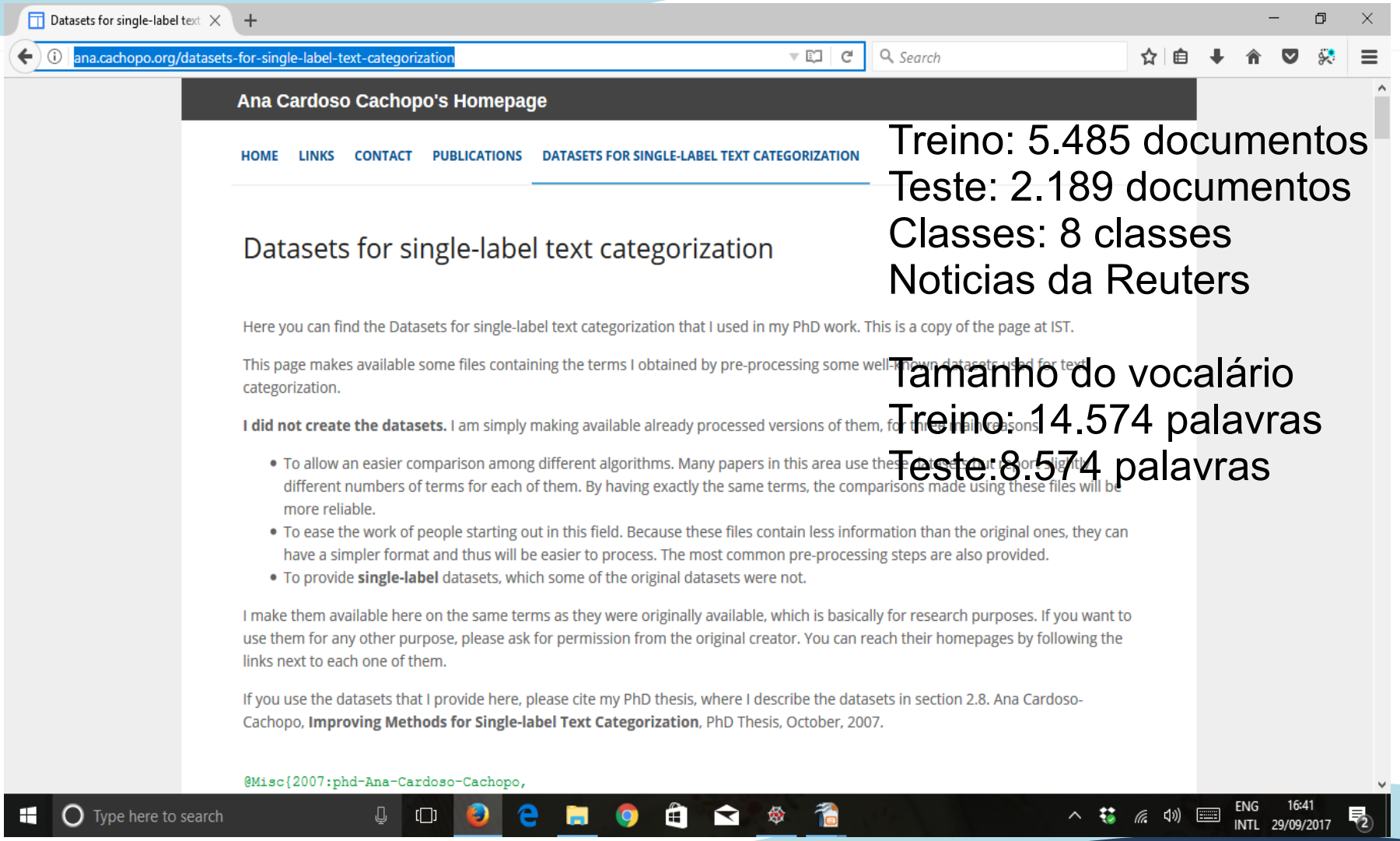
# Ferramentas

**Java:**
DeepLearning4J

**Python:**
Theano, Pytorch , TensorFlow, Keras etc

**WordEmbbeding:**
Gensim , Spark

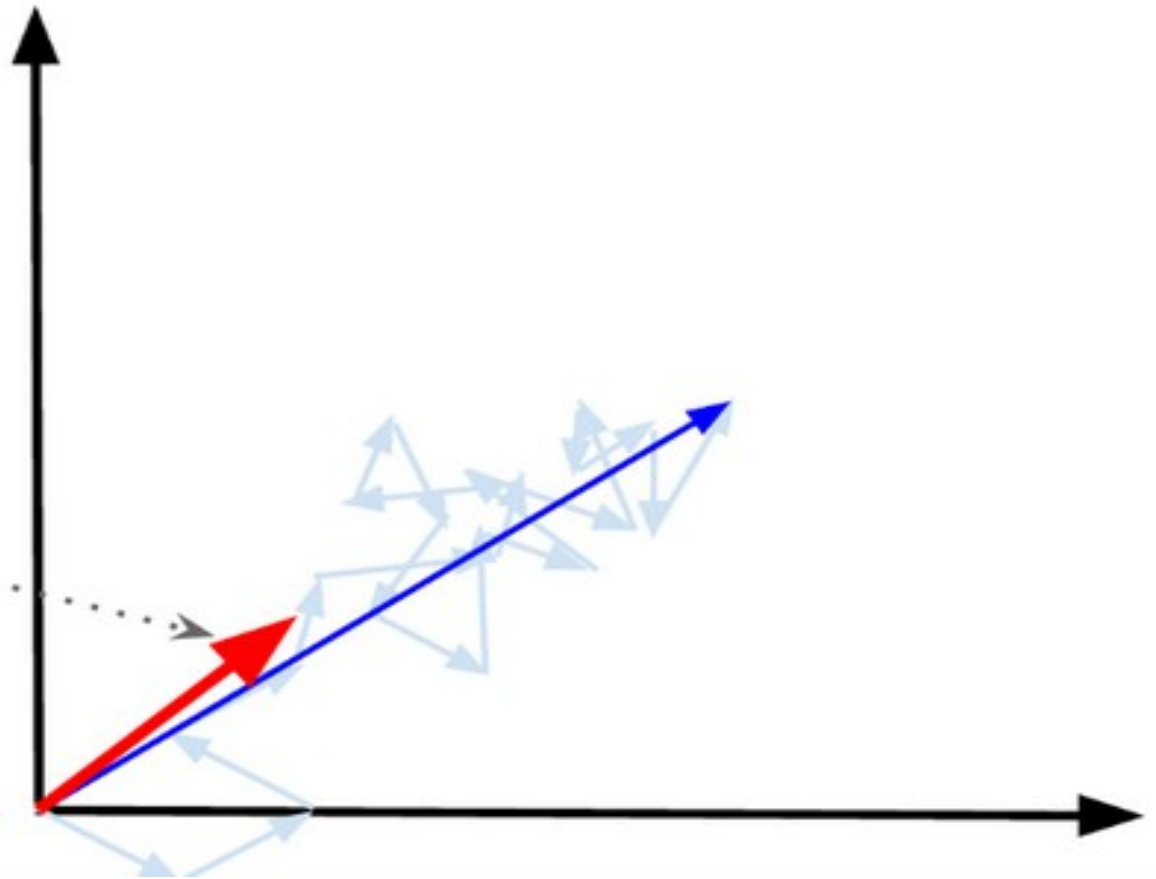

# Resultados




Treino: 5.485 documentos
Teste: 2.189 documentos
Classes: 8 classes
Noticias da Reuters

Tamanho do vocalário
Treino: 14.574 palavras
Teste:8.574 palavras

# Resultados



# Resultados

Datasets for single-label text

ana.cachopo.org/datasets-for-single-label-text-categorization

Search

### Some results

Just to give an idea of the relative hardness of each dataset, I have determined the accuracy that some of the most common classification methods achieve with them. As usual, tfidf term weighting is used to represent document vectors, and they were normalized to unitary length. The stemmed train and test sets were used for each dataset.

The "dumb classifier" is included as a baseline. It ignores the query and always gives as the predicted class the most frequent class in the training set.

| Accuracy Values | | | | | |
|---|---|---|---|---|---|
| **Classification Method** | **R8** | **R52** | **20Ng** | **Cade12** | **WebKb** |
| Dumb classifier | 0.4947 | 0.4217 | 0.0530 | 0.2083 | 0.3897 |
| Vector Method | 0.7889 | 0.7687 | 0.7240 | 0.4142 | 0.6447 |
| kNN (k = 10) | 0.8524 | 0.8322 | 0.7593 | 0.5120 | 0.7256 |
| Centroid (Normalized Sum) | 0.9356 | 0.8717 | 0.7885 | 0.5148 | 0.8266 |
| Naive Bayes | 0.9607 | 0.8692 | 0.8103 | 0.5727 | 0.8352 |
| SVM (Linear Kernel) | 0.9698 | 0.9377 | 0.8284 | 0.5284 | 0.8582 |

**K=1 => Recall 0.85**

Note that, because **R8**, **R52**, and **WebKB** are very skewed, the dumb classifier has a ``reasonable'' performance for these datasets. Also, it is worth noting that, while for **R8**, **R52**, **20Ng**, and **webKB** it is possible to find good classifiers, that is, classifiers that achieve a high accuracy, for **Cade12** the best we can get does not reach 58% accuracy, even with some of the best classifiers available.

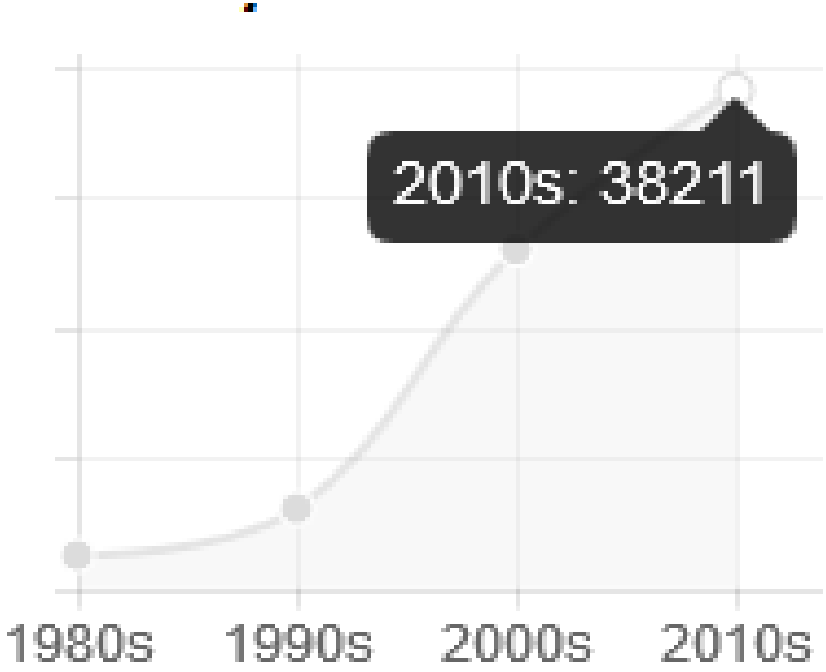
20NG-TEST-ALL-TERMS.TXT (10555K) ANA CARDOSO CACHOPO, 30 APR 2015, 10:31 V.1

20NG-TEST-NO-SHORT.TXT (9335K) ANA CARDOSO CACHOPO, 30 APR 2015, 10:32 V.1

Type here to search

ENG INTL 16:41 29/09/2017

# Casos na Literatura : ACM Libray




Automatic Image Annotation using Deep Learning Representations (2015)

Vocabulary Expansion Using Word Vectors for Video Semantic Indexing(2015)

Skipping Word: A Character-Sequential Representation based Framework for Question Answering (2016)

# Bibliografia

1 – Vídeo aula no youtube - Dr. Niraj
2 – Neural Networks Language Models - Philipp Koehn (2016)
3 – Distributed Representations of Words and Phrases and their Compositionality – Tomas Mikolov et al (2013)
4 – A Primer on Neural Network Models for Natural Language Processing - Yoav Goldberg (2015)
5 – Blog The amazing power of word vectors
6 – Text Understanding from Scratch  - Yann LeCun (2015)

# THANKS!

**Any questions?**