

1

Deep Learning e Processamento de Linguagem Natural

Anderson da Silva Soares

www.inf.ufg.br/~anderson

www.deeplearningbrasil.com.br



NLP

- Sub-campo de data Science
 - Analisar
 - Compreender
 - Derivar informação
-
- Aplicações:
 - Sumarização
 - Tradução
 - Análise semântica
 - Reconhecimento de voz

Casos de sucesso

Automatic summarization

Coreference resolution

Discourse analysis

Machine translation

Morphological segmentation

Named entity recognition (NER)

Natural language generation

Word sense disambiguation

Relationship extraction

Speech processing

Part-of-speech tagging

sentence boundary disambiguation

Sentiment analysis

Optical character recognition (OCR)

Question answering

Parsing

Word segmentation

Natural language understanding

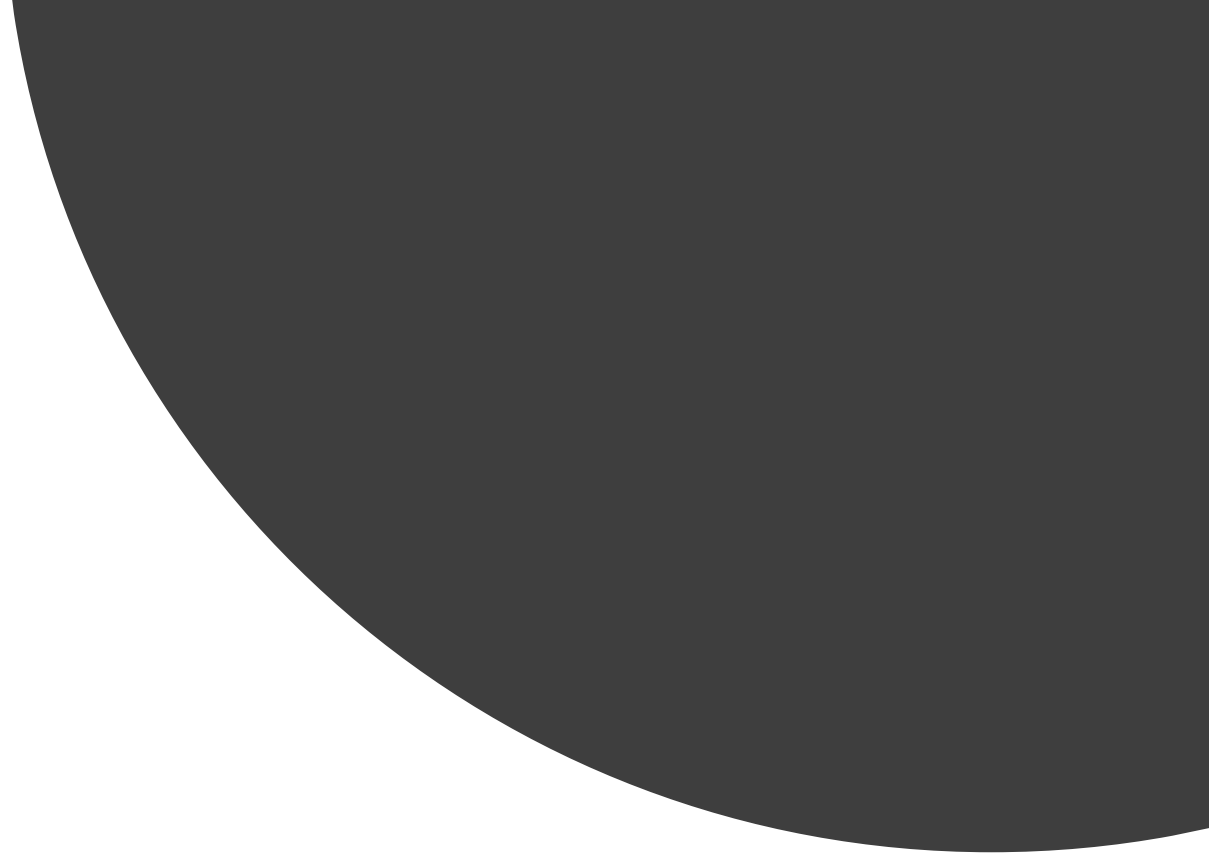
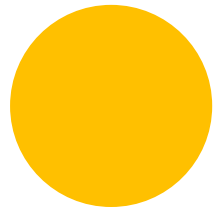
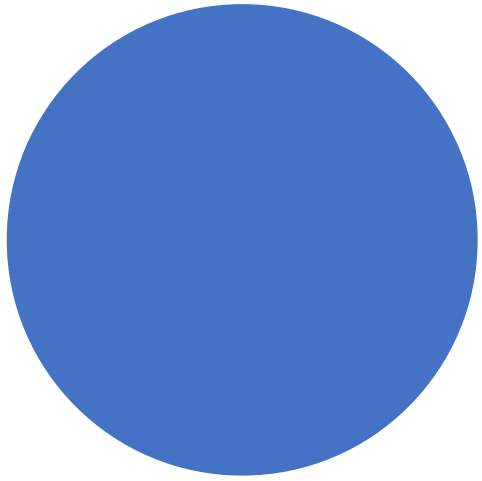
Information retrieval (IR)

Speech recognition

Topic segmentation and recognition

Speech segmentation

Information extraction (IE)



Pandas

Python for data
analysis

Prakhar Amlathe

Utah State University

Introdução ao Pandas :

Panel Data System

Criado em 2008,
atualmente mantido
por Jeff Reback e
colaboradores.

Principal pacote para
análise de dados em
python

Conceito open-
source

Overview

- Python Library para data analysis similar a: R, MATLAB, SAS
- Rich data structures com velocidade, facilidade e expressividade.
- Construída como front-end da NumPy
- Componentes chaves no Pandas : Duas novas estruturas de dados para Python

Series


DataFrame

Problemas em que panda é nosso amigo(a)...




- Munging data
- Cleaning data
- Analyzing data
- Modelling data
- Plotting graphs or Tabular displays of organized results.

Series : Pandas Data Structure

- Contém um vetor de dados (qualquer tipo de dados NumPy) com índices associados.
- 0 a N onde N = size -1

 jupyter Untitled9 Last Checkpoint: 3 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

         Code  CellToolbar   

```
In [1]: import pandas as pd

# create a Series with an arbitrary list
s = pd.Series([7, 'Joey', 3.14, 568970, 'Chandler!' , 'Data Science'])
s
```

```
Out[1]: 0          7
        1      Joey
        2      3.14
        3    568970
        4    Chandler!
        5    Data Science
        dtype: object
```

```
In [ ]: |
```


DataFrame : Pandas Data Structure

- Um dataframe é uma estrutura tabular composta de linhas e colunas, similar a uma planilha ou tabela de banco de dados.
- Pode ser tratada como uma série de objetos compartilhando um mesmo índice

```
In [3]: import pandas as pd

# create a DataFrame

sample = {'year': [2010, 2011, 2012, 2013, 2014, 2015],
          'winning team': ['CSK', 'Delhi daredevils', 'KKR', 'Pune warriors', 'Hyderabas sunrisers', 'KKR'],
          'wins': [15, 12, 10, 14, 11, 12],
          'losses': [5, 3, 6, 1, 4, 3]}
iplcricket = pd.DataFrame(sample, columns=['year', 'winning team', 'wins', 'losses'])
iplcricket
```

```
Out[3]:
```

	year	winning team	wins	losses
0	2010	CSK	15	5
1	2011	Delhi daredevils	12	3
2	2012	KKR	10	6
3	2013	Pune warriors	14	1
4	2014	Hyderabas sunrisers	11	4
5	2015	KKR	12	3

Operações realizáveis nas estruturas do pandas

Filtering

Summarizing

Group by –
split apply
combine

Merge, join,
aggregate

Time series/
Data
functionality

Plotting with
Matplotlib and
many more...

Estudo de caso

505 opiniões sobre um determinado produto

disciplinas 2017-2/Aula 8 x Pandas_limpeza_dados x TV Samsung Smart TV | P x

www.bondfaro.com.br/proc_unico?id=2852&xrc6937=598635&xro=107,6937&xrc107=1593

★ Bookmarks Inglês Call for papers AG Disciplinas Leitura Filhas Curso online Consultoria ZuzooVn/machine-lea Rec

CELULARES TV ELETRÔNICOS INFORMÁTICA ELETRODOMÉSTICOS CASA E DECORAÇÃO ESPORTE E LAZER GAM

Início / Eletrônicos / TV / Samsung / Smart TV

TV

PALAVRA-CHAVE

FILTRAR

Samsung x Smart TV x

VARIAÇÃO DE PREÇO

- ☐ R\$ 0 a R\$ 1.799 (4)
- ☐ R\$ 1.800 a R\$ 2.499 (1)
- ☐ R\$ 2.500 a R\$ 3.599 (2)
- ☐ R\$ 3.600 a R\$ 5.499 (3)
- ☐ R\$ 5.500 a R\$ 6.899 (2)

Máximo Máximo

Aguardando www.google.com...

Digite aqui para pesquisar

Smart TV Samsung Série 4 UN32J4300AG 32 polegadas LED Plana

505 opiniões

R\$ 1.093,88

Smart TV Samsung Série 5 UN40J5500AG 40 polegadas LED Plana

230 opiniões

R\$ 1.766,91

Links úteis

- <http://pandas.pydata.org/>
- <https://www.analyticsvidhya.com/blog/2016/01/12-pandas-techniques-python-data-manipulation/>
- <http://www.slideshare.net/maikroeder/pandas-16424935>
- <http://pandas.pydata.org/pandas-docs/stable/tutorials.html>

Table 1: The three components of learning algorithms.

Representation	Evaluation	Optimization
Instances	Accuracy/Error rate	Combinatorial optimization
<i>K</i> -nearest neighbor	Precision and recall	Greedy search
Support vector machines	Squared error	Beam search
Hyperplanes	Likelihood	Branch-and-bound
Naive Bayes	Posterior probability	Continuous optimization
Logistic regression	Information gain	Unconstrained
Decision trees	K-L divergence	Gradient descent
Sets of rules	Cost/Utility	Conjugate gradient
Propositional rules	Margin	Quasi-Newton methods
Logic programs		Constrained
Neural networks		Linear programming
Graphical models		Quadratic programming
Bayesian networks		
Conditional random fields		

- Alguns conceitos importantes:
- Tokenização – converter textos em tokens
- Tokens – palavras presentes em um texto
- Objeto de texto – Uma sentença ou frase

```
sudo easy_install pip  
Sudo pip install -U nltk
```

```
Import nltk nltk.download()
```

NLTK 3.2.4 documentation

[NEXT](#) | [MODULES](#) | [INDEX](#)

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

- Inclui pelo menos três fases:
- Remoção de ruídos
- Normalização Lexica
- Padronização de objetos

Utilizando dicionário de “ruídos”

```
# Sample code to remove noisy words from a text

noise_list = ["is", "a", "this", "..."]

def _remove_noise(input_text):
    words = input_text.split()
    noise_free_words = [word for word in words if word not in noise_list]
    noise_free_text = " ".join(noise_free_words)
    return noise_free_text

_remove_noise("this is a sample text")
>>> "sample text"
...
```


Normalização Léxica

Todas as formas abaixo tem um sentido: estudar

Gerúndio: estudando

Particípio passado: estudado

INDICATIVO

Presente	Pretérito perfeito	Pretérito imperfeito
eu estudo tu estudas ele/ela estuda nós estudamos vós estudaís eles/elas estudam	eu estudei tu estudaste ele/ela estudou nós estudámos / estudamos vós estudastes eles/elas estudaram	eu estudava tu estudavas ele/ela estudava nós estudávamos vós estudáveis eles/elas estudavam

Pret. mais-que-perfeito	Futuro /	CONDICIONAL /
	Futuro do presente	Futuro do pretérito
eu estudara tu estudaras ele/ela estudara nós estudáramos vós estudáreis eles/elas estudaram	eu estudarei tu estudarás ele/ela estudará nós estudaremos vós estudareis eles/elas estudarão	eu estudaria tu estudarias ele/ela estudaria nós estudariamos vós estudáreis eles/elas estudariam

CONJUNTIVO

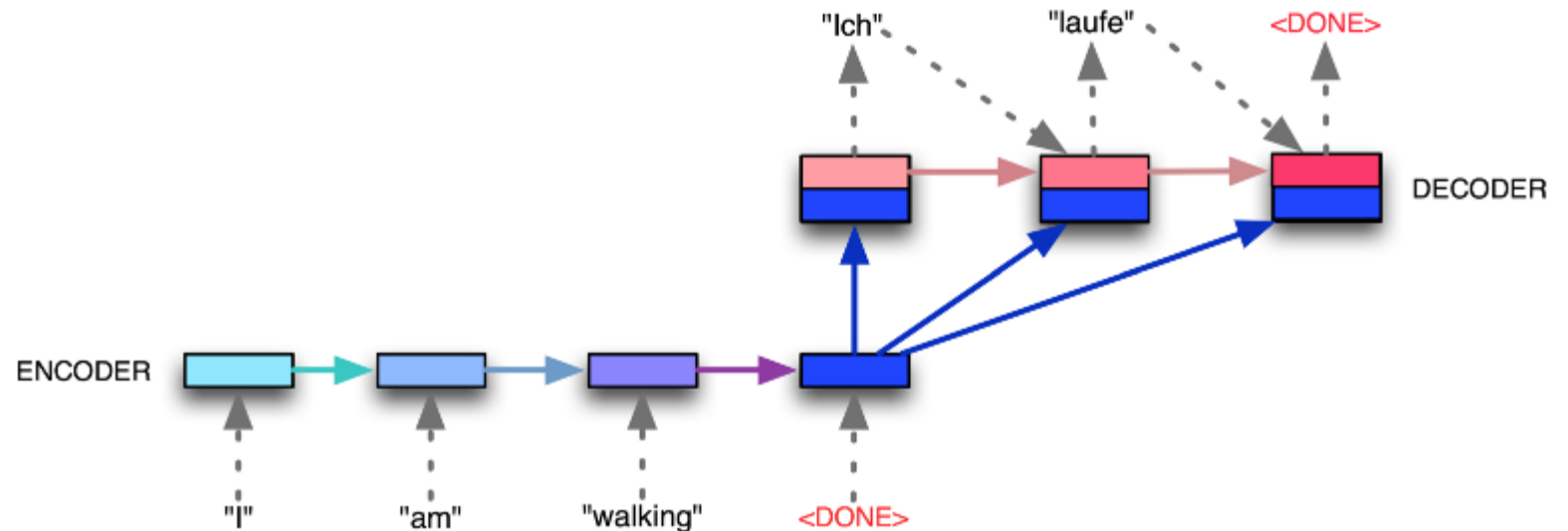
SUBJUNTIVO (BR)

Presente	Pretérito imperfeito	Futuro
que eu estude que tu estudas que ele/ela estude que nós estudemos que vós estudeis que eles/elas estudem	se eu estudasse se tu estudasses se ele/ela estudasse se nós estudássemos se vós estudásseis se eles/elas estudassem	quando eu estudar quando tu estudares quando ele/ela estudar quando nós estudarmos quando vós estudardes quando eles/elas estudarem

IMPERATIVO

afirmativo	negativo	INFINITIVO PESSOAL
— estuda tu estude ele/ela estudemos nós estudai vós estudem eles/elas	— não estudas tu não estude ele/ela não estudemos nós não estudeis vós não estudem eles/elas	para estudar eu para estudares tu para estudar ele/ela para estudarmos nós para estudardes vós para estudarem eles/elas

- NLP geralmente inclui as seguintes problemas:
- Classificação
- Rotulação
- Geração de sequência



- A visão moderna de NLP com Deep Learning pode ser resumida em 4 elementos básicos
 - Incorporar (Embedded)
 - Codificar (encoding)
 - Atenção
 - Predição

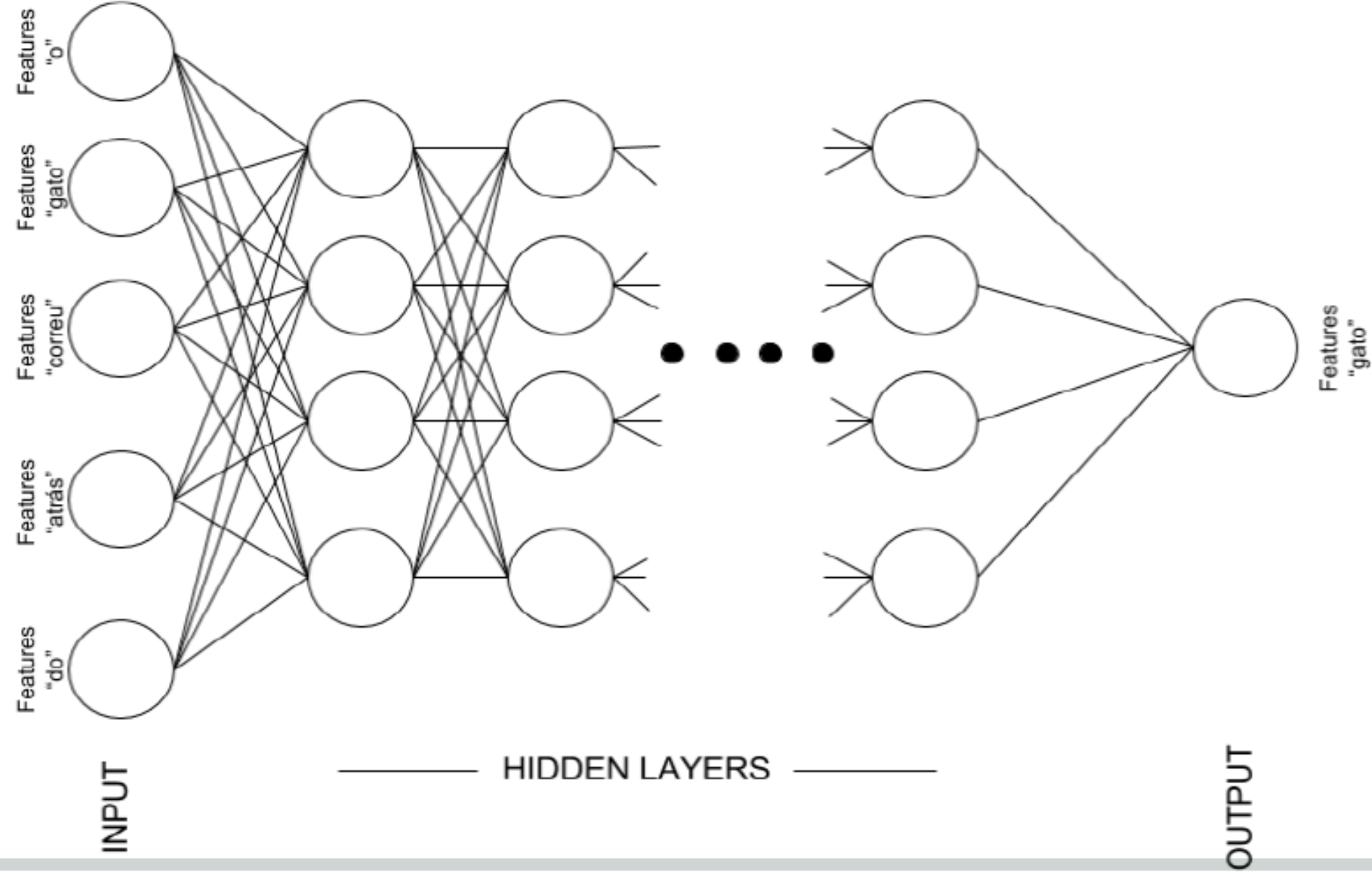
- Exemplo:

- “O gato corre atrás do _____”

- Será que a próxima palavra é:

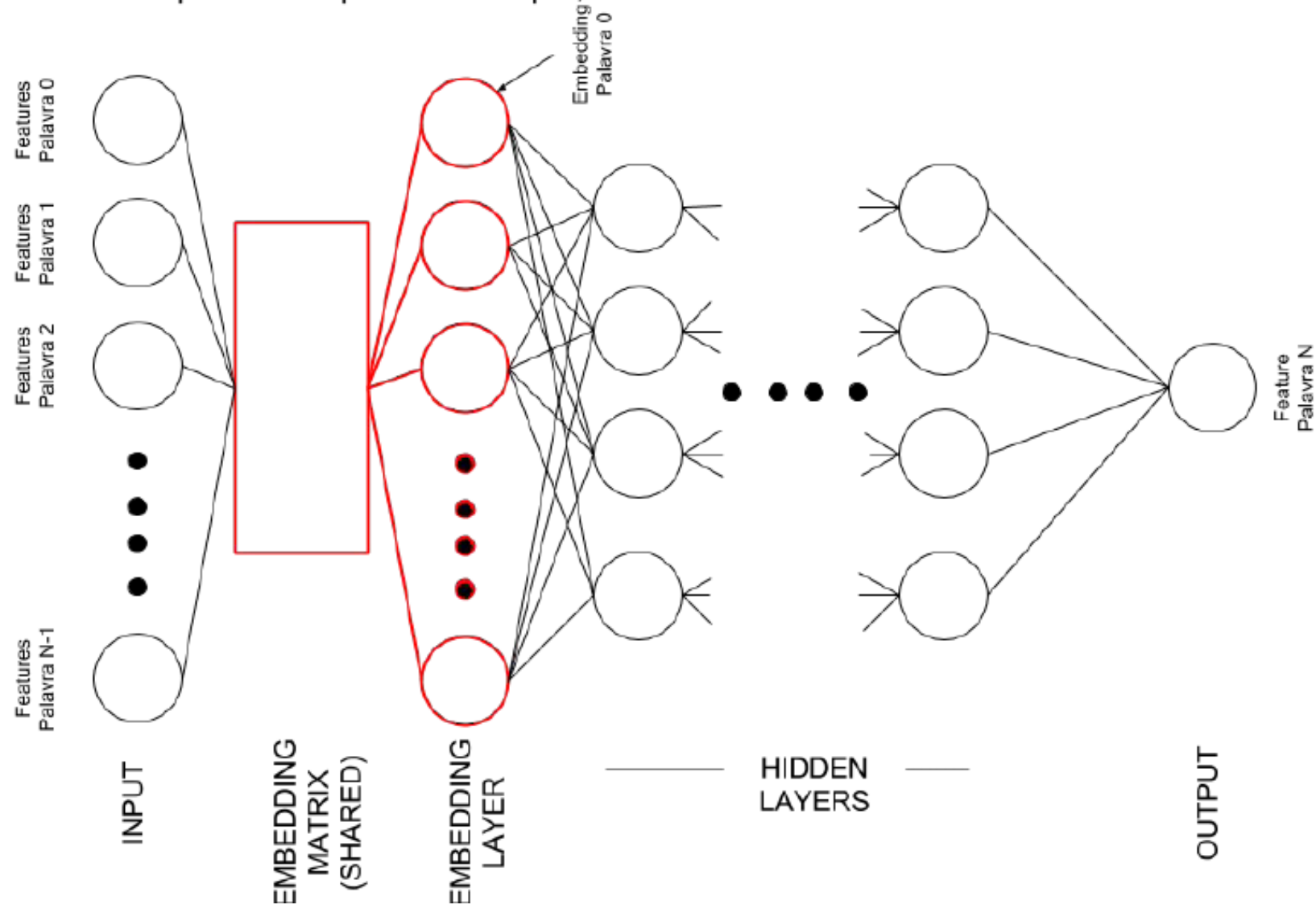
- “rato” ?
 - “cachorro” ?
 - “carro” ?

Modelos de linguagem são usados para coisas como processamento de voz, autocorreção de ortografia, etc.



- Um pouco depois de se começar a usar redes neurais para modelos de linguagem, percebeu-se se que era mais interessante colocar uma camada adicional *-linear, de tamanho fixo-* entre o vetor de features de cada palavra e as camadas da rede
- Para reduzir a dimensionalidade dos vetores de palavras e ter representações de tamanho fixo
- Essa camada foi chamada de *Embedding Layer*

Para cada sequência de N palavras no corpus, fazer:



01

Depois de projetada na camada de Embedding, cada palavra vira um vetor em um espaço vetorial.

02

Neste espaço, a proximidade entre vetores representa proximidade de padrão de uso, ou seja, palavras que são usadas no mesmo contexto ficam próximas umas das outras.

01

Em meados de 2010, foi identificado que embeddings poderiam ser usados para melhorar resultados de praticamente todas as tarefas normais de NLP, como tagueamento POS, SRL, NER, classificação de textos, etc.

02

Em 2013, foi feito um estudo sobre a semântica desses vetores (chamados de *Word Embeddings*) e foi descoberto que eles eram tão bons que se podia até fazer operações aritméticas com eles.

03

Exemplo: a operação aritmética

- $\text{Embedding}(\text{"king"}) - \text{Embedding}(\text{"man"}) + \text{Embedding}(\text{"woman"})$
- Dá como resultado
 - $\text{Embedding}(\text{"queen"})$

Embedded

- Interpretação depende do conjunto e não dos elementos pontuais (palavras)
- Exemplo:
 - Eu vou passar neste curso, só que não !!!
 - Eu vou passar roupa.
- Dois problemas:
 - “só que não” contraria o sentido do início da frase
 - “passar” tem sentidos diferentes em razão das outras palavras

Bag of Words - Desvantagens

- BoW funciona bem para textos simples ou formais
- Falha evidente: negação de sentido
- A negação é muito estudada como um contra-exemplo para os modelos BoW
- Exemplo:
 - “O filme carece de inteligência e humor”
 - contém “inteligência” and “humor”
 - Um típico BoW classificaria como sentimento positivo

Bag of Words - Desvantagens

- Existem negações simples e complexas
- Many possible forms of negation that cannot be listed *a priori*
 - “Não tem humor”
 - “Está faltando humor”
 - “Desprovido de humor”
- A ordem importa
 - “sem risos e com muitos momentos chatos”
 - “muitos risos e sem momentos chatos”
 - Mesmo BoW – Mas sentimentos distintos

Embedded

- Feedforward neural networks
- Recurrent neural networks
- Backpropagation
- Assum que as entradas e saídas são vetores
- Como representar palavras em um vetor?

“One-Hot Encoding”

- Técnica “Naïve”
- Palavra = $w \in \mathbb{R}^n, n = \text{dom}(V)$
- Exemplos
 - $V = [\text{“dog”}, \text{“bites”}, \text{“man”}]$
 - “dog bites man” = $[[1,0,0],[0,1,0],[0,0,1]]$
 - “man bites dog” = $[[0,0,1],[0,1,0],[1,0,0]]$

```
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
import numpy as np

feature_labels = np.array([1, 1, 2, 3, 4, 3, 4])
encoder = LabelEncoder()
encoder.fit(feature_labels)
feature_labels = encoder.transform(feature_labels)
feature_labels = np_utils.to_categorical(feature_labels)

'''
array([[ 1.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```


“One-Hot Encoding” Drawbacks

- Tamanho do vetor x tamanho do vocabulário
 - Pre-determinado
- “Out-of-Vocabulary”
 - Como tratar palavras não vistas no conjunto de teste?
 - Possível solução: “UNKNOWN” para representar palavras pouco usadas
- Problemas que persistem:
 - Eu gosto de redes neurais
 - Podemos inferir que gosto de REDES (de computador?)



John Rupert Firth

“You shall know a word by the company it keeps”

-1957

- English linguist
- Muito Famoso em NLP
- Interpretação moderna: Co-ocorrência é um bom indicador de significado

“One-Hot Encoding” Drawbacks

- Tudo isso significa BANCO

government debt problems turning into banking crises as has happened in
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Matriz de co-
ocorrências

- Como representar palavras vizinhas?
 - Opção 1: documento inteiro
 - Opção 2: janela de vizinhança

Matriz de co-ocorrências

- Eu gosto de redes neurais
- Eu gosto de linguagem natural
- Eu aprecio leitura

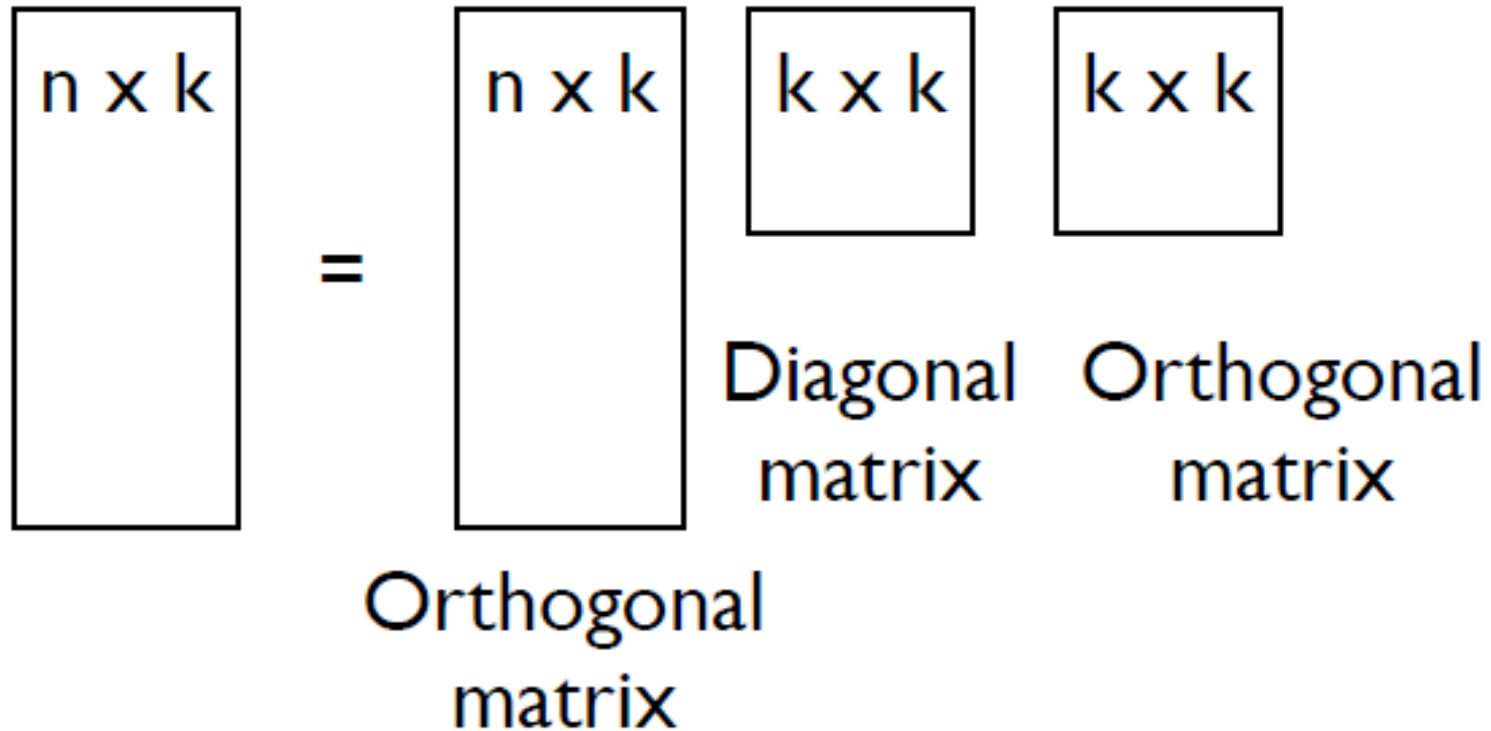
	Eu	gosto	de	redes	neurais	linguagem	natural	aprecio	leitura	.
Eu	0	2	0	0	0	0	0	1	0	0
gosto	2	0	2	0	0	0	0	0	0	0
de	0	0	0	1	1	0	0	0	0	0
redes	0	0	0	0	0	1	0	0	0	0
neurais	0	0	0	0	0	0	0	0	0	1
linguagem	0	0	0	0	0	0	1	0	0	0
natural	0	0	0	0	0	0	0	0	0	0
aprecio	1	0	0	0	0	0	0	0	1	1
leitura	0	0	0	0	0	0	0	0	0	0
.	0	0	0	1	0	0	1	0	1	0

Matriz de co-ocorrências

```
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 #Eu gosto de redes neurais.
12 #Eu gosto de linguagem natural.
13 #Eu aprecio leitura.
14 la = np.linalg
15 palavras = ["Eu", "gosto", "de", "redes", "neurais", "linguagem", "natural", "aprecio", "leitura", "."]
16 X = np.array([[0,2,0,0,0,0,0,1,0,0],
17               [2,0,2,0,0,0,0,0,0,0],
18               [0,0,0,1,1,0,0,0,1,0],
19               [0,0,1,0,0,1,0,0,0,0],
20               [0,0,0,1,0,0,0,0,0,1],
21               [0,0,1,0,0,0,1,0,0,0],
22               [0,0,0,0,0,1,0,0,0,1],
23               [1,0,0,0,0,0,0,0,1,0],
24               [1,0,0,0,0,0,0,0,0,1],
25               [0,0,0,1,0,0,1,0,1,0]])
26
27 U,S,V = la.svd(X,full_matrices=False)
28
29
30 axes = plt.gca()
31 axes.set_xlim([-1,1])
32 axes.set_ylim([-1,1])
33 for i in xrange(len(palavras)):
34     print U[i,0],U[i,1]
35     plt.text(U[i,0],U[i,1],palavras[i])
36
```

Matriz de co-ocorrências

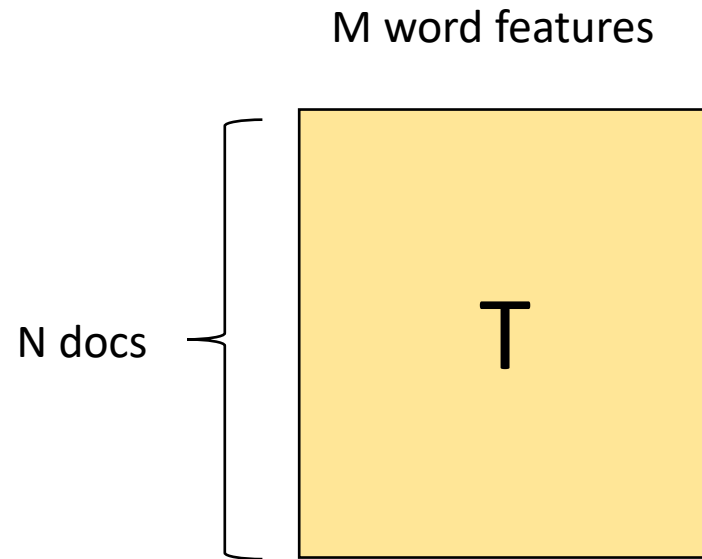
$$\mathbf{X} = \mathbf{U} \mathbf{D} \mathbf{V}^T$$



Latent semantic analysis studies documents in **Bag-Of-Words format** (1988).

i.e. Dada uma matriz T que codifica alguns documentos:

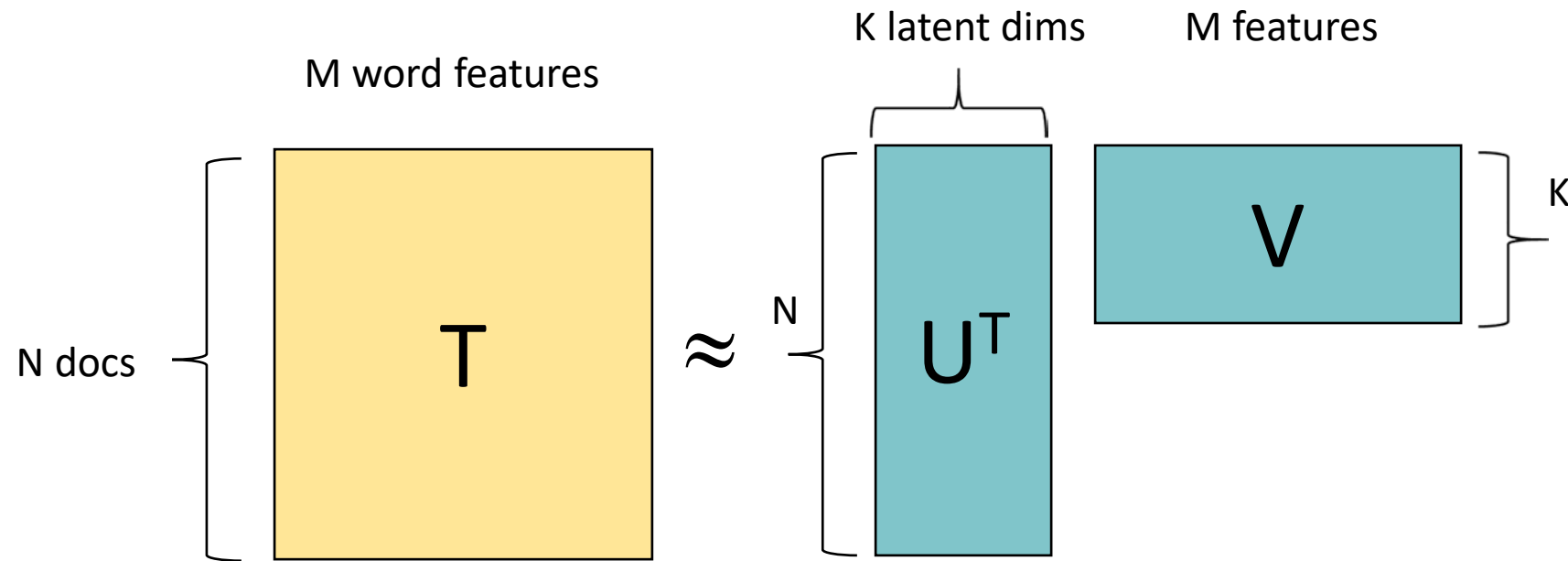
T_{ij} é a contagem da palavra j no document i .



Dado a bag-of-words T , calcular a fatorização $T \approx U^T * V$ (e.g. a best L_2 approximation to T)

Fatores codificam **contextos similares de documentos**.

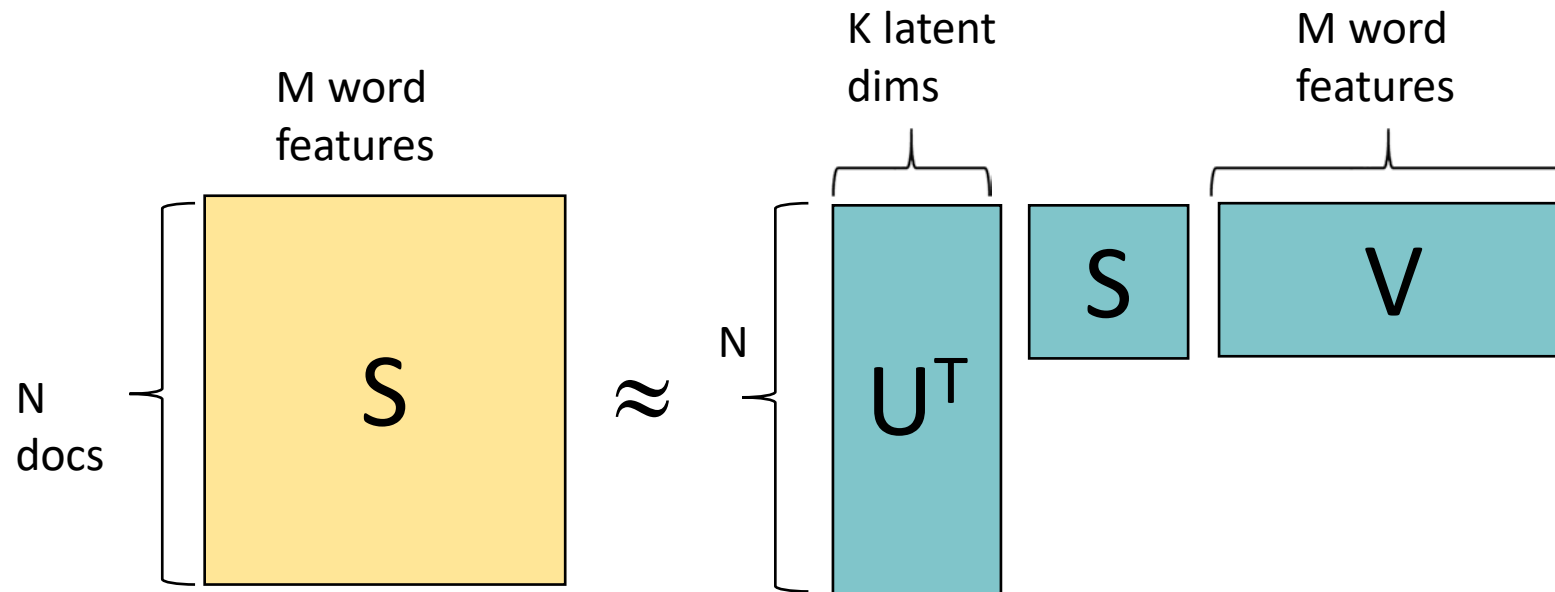
Fatores são linhas de V .



Se U e V são ortogonais, S é uma matriz de valores singulares.

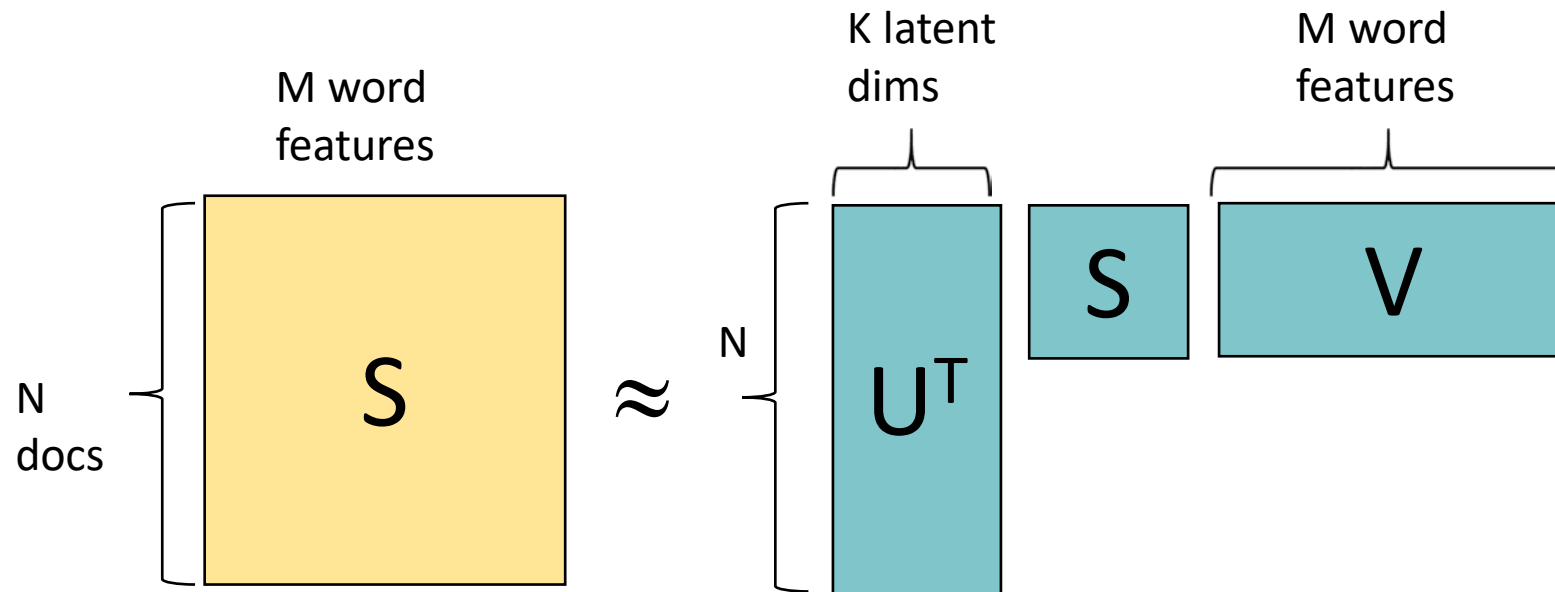
$v = Vt$ é um “embedding” de um document no espaço latente.

$t' = U^T v = U^T V t$ é a decodificação de uma sentença.

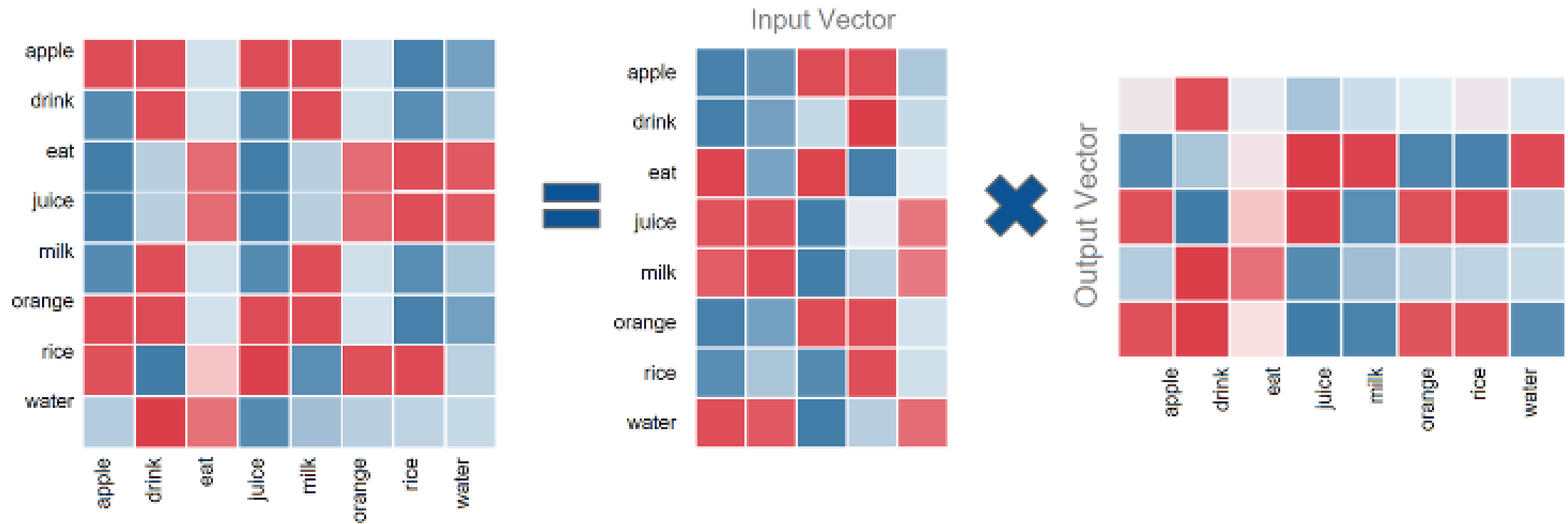


$t' = U^T v = U^T V t$ é a codificação da sentença.

Uma fatoração SVD fornece a melhor reconstrução possível do document a partir do espaço considerado.



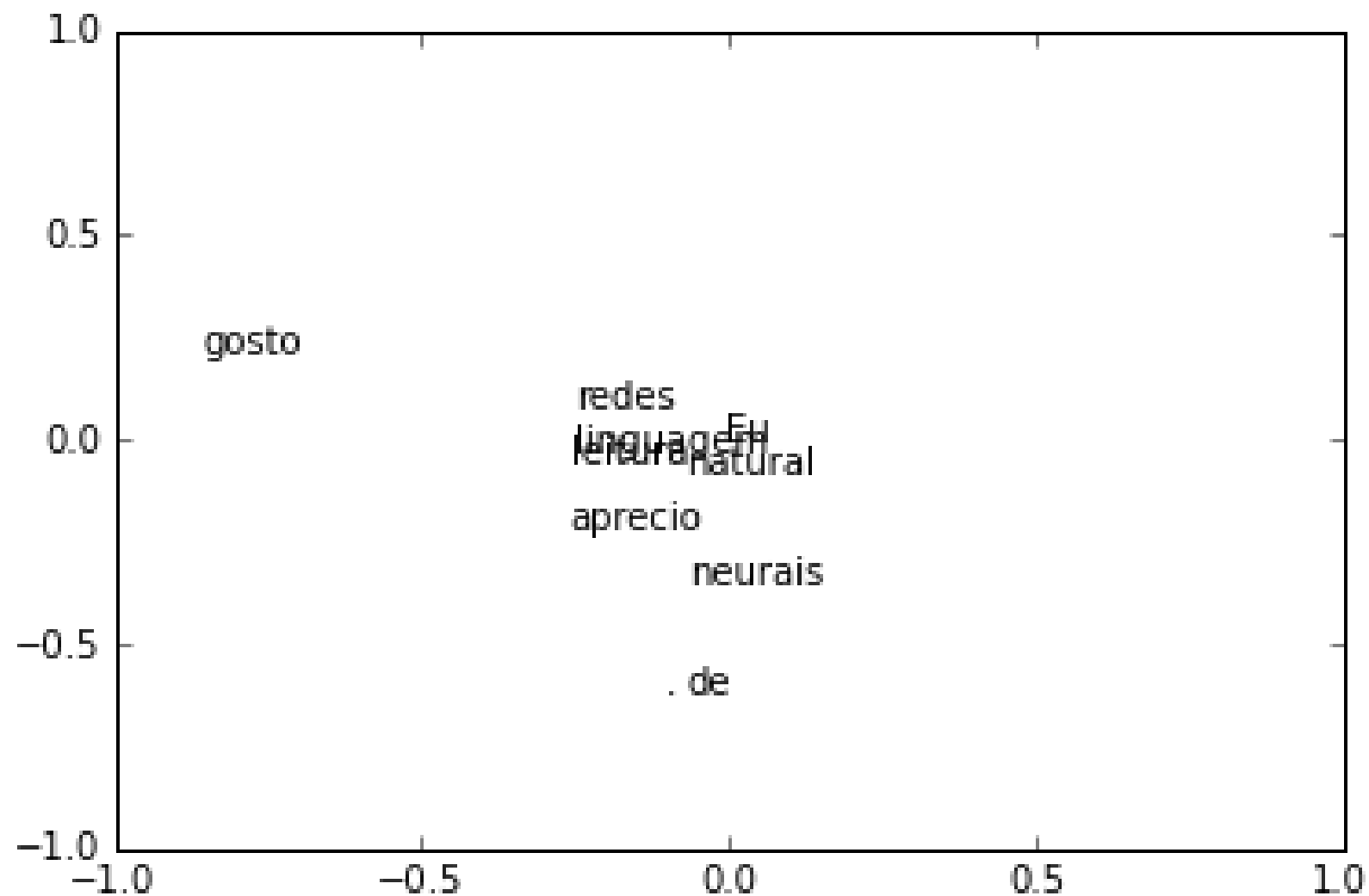
Matriz de co-ocorrências

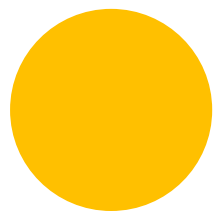
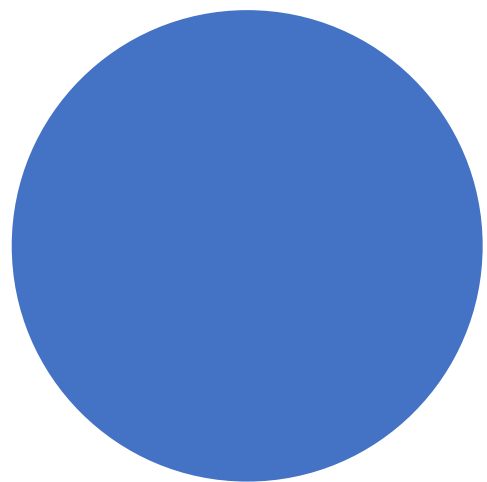


Matriz de co-ocorrências

```
8 import numpy as np
9 import matplotlib.pyplot as plt
10
11 #Eu gosto de redes neurais.
12 #Eu gosto de linguagem natural.
13 #Eu aprecio leitura.
14 la = np.linalg
15 palavras = ["Eu", "gosto", "de", "redes", "neurais", "linguagem", "natural", "aprecio", "leitura", "."]
16 X = np.array([[0,2,0,0,0,0,0,1,0,0],
17               [2,0,2,0,0,0,0,0,0,0],
18               [0,0,0,1,1,0,0,0,1,0],
19               [0,0,1,0,0,1,0,0,0,0],
20               [0,0,0,1,0,0,0,0,0,1],
21               [0,0,1,0,0,0,1,0,0,0],
22               [0,0,0,0,0,1,0,0,0,1],
23               [1,0,0,0,0,0,0,0,1,0],
24               [1,0,0,0,0,0,0,0,0,1],
25               [0,0,0,1,0,0,1,0,1,0]])
26
27 U,S,V = la.svd(X,full_matrices=False)
28
29
30 axes = plt.gca()
31 axes.set_xlim([-1,1])
32 axes.set_ylim([-1,1])
33 for i in xrange(len(palavras)):
34     print U[i,0],U[i,1]
35     plt.text(U[i,0],U[i,1],palavras[i])
36
```

Matriz de co-ocorrências





Representações modernas de NLP



01

Vetor de palavras
devem ser
significativos

02

Treinar uma rede
neural para fazer
tarefas simples

03

A camada oculta
de uma rede
neural é um
detector de
características

04

Usar a camada da
rede neural como
word embedding

Word2Vec



Abordagens para representar palavras

Distribuição semântica (*Count*)

- Usado desde 90's
- Matriz esparsa de context PMI/PPMI
- Decomposição com SVD

Word Embeddings (*Predict*)

- Inspirado para deep learning
- word2vec (*Mikolov et al., 2013*)
- GloVe (*Pennington et al., 2014*)

Teoria subjacente: **The Distributional Hypothesis** (*Harris, '54; Firth, '57*)

“Similar words occur in similar contexts”



Abordagens para representar palavras

Ambas abordagens:

- Fundamentos na **mesma teoria de linguagens**
- São matematicamente relacionadas
 - “Neural Word Embedding as Implicit Matrix Factorization” (NIPS 2014)
- Pq word embeddings são melhores?
 - “Don’t Count, Predict!” (Baroni et al., ACL 2014)



Word Embeddings

Algoritmos novos

(objective + training method)

- Skip Grams + Negative Sampling
- CBOW + Hierarchical Softmax
- Noise Contrastive Estimation
- GloVe
- ...

Novos hyperparametros

(preprocessing, smoothing, etc.)

- Subsampling
- Dynamic Context Windows
- Context Distribution Smoothing
- Adding Context Vectors
- ...

O que realmente aprimora a performance?

- Definição de word embedding:
 - **Vetor distribuído, denso, contínuo, de tamanho fixo, que representa uma palavra**
 - Distribuído: cada conceito é representado por uma feature. Isso diminui a dimensionalidade necessária.
 - Denso: não esperso
 - Contínuo: não binário
 - Tamanho fixo: palavras projetadas no mesmo espaço para que possam ser separadas

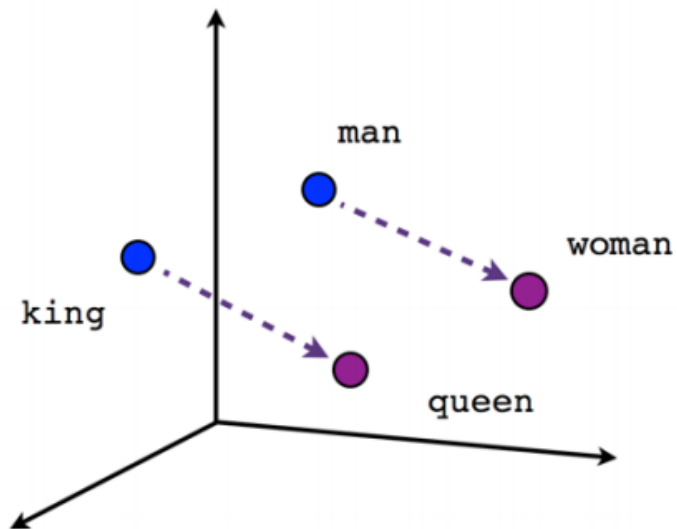
- Rede neural treinada para prever context de palavras
- Considera poucos aspectos linguísticos

Word2Vec

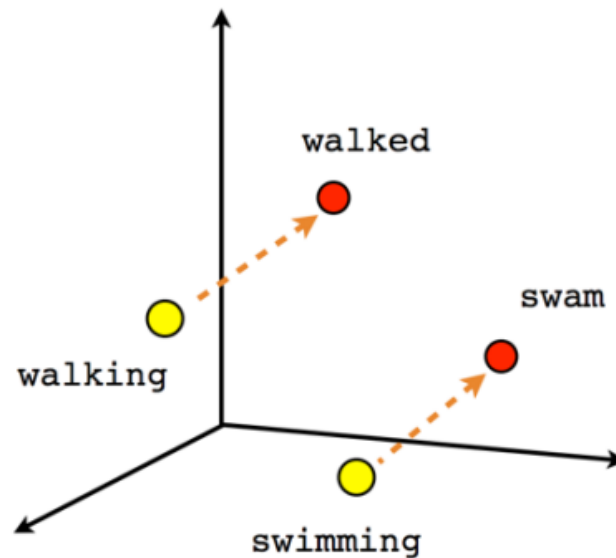
Word2Vec

Método tradicional - Bag of Words	Word Embeddings
<ul style="list-style-type: none">• Usa one hot encoding• Cada palavra no vocabulário representada por uma posição em um vetor• Informação de context é pouco utilizada	<ul style="list-style-type: none">• Cada palavra possui uma posição no espaço (euclidiano por exemplo), onde é representada por um vetor de tamanho fixo• Aprendizado não-supervisionado• Por exemplo, “Hello” pode ser representado por: [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02]

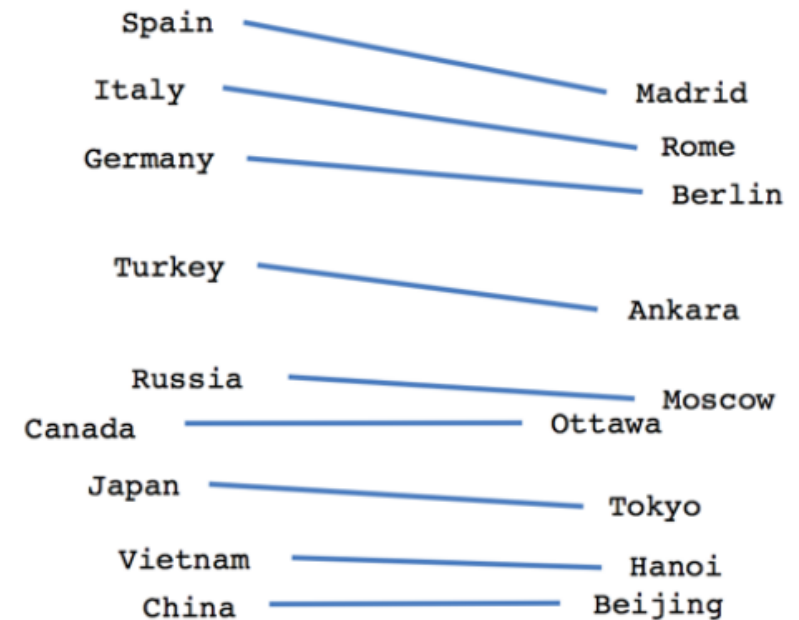
Exemplos



Male-Female



Verb tense



Country-Capital

$$\text{vector[Queen]} = \text{vector[King]} - \text{vector[Man]} + \text{vector[Woman]}$$



- Espresso? But I ordered a cappuccino!
- Don't worry, the cosine distance between them is so small that they are almost the same thing.

Em um
mundo
perfeito...

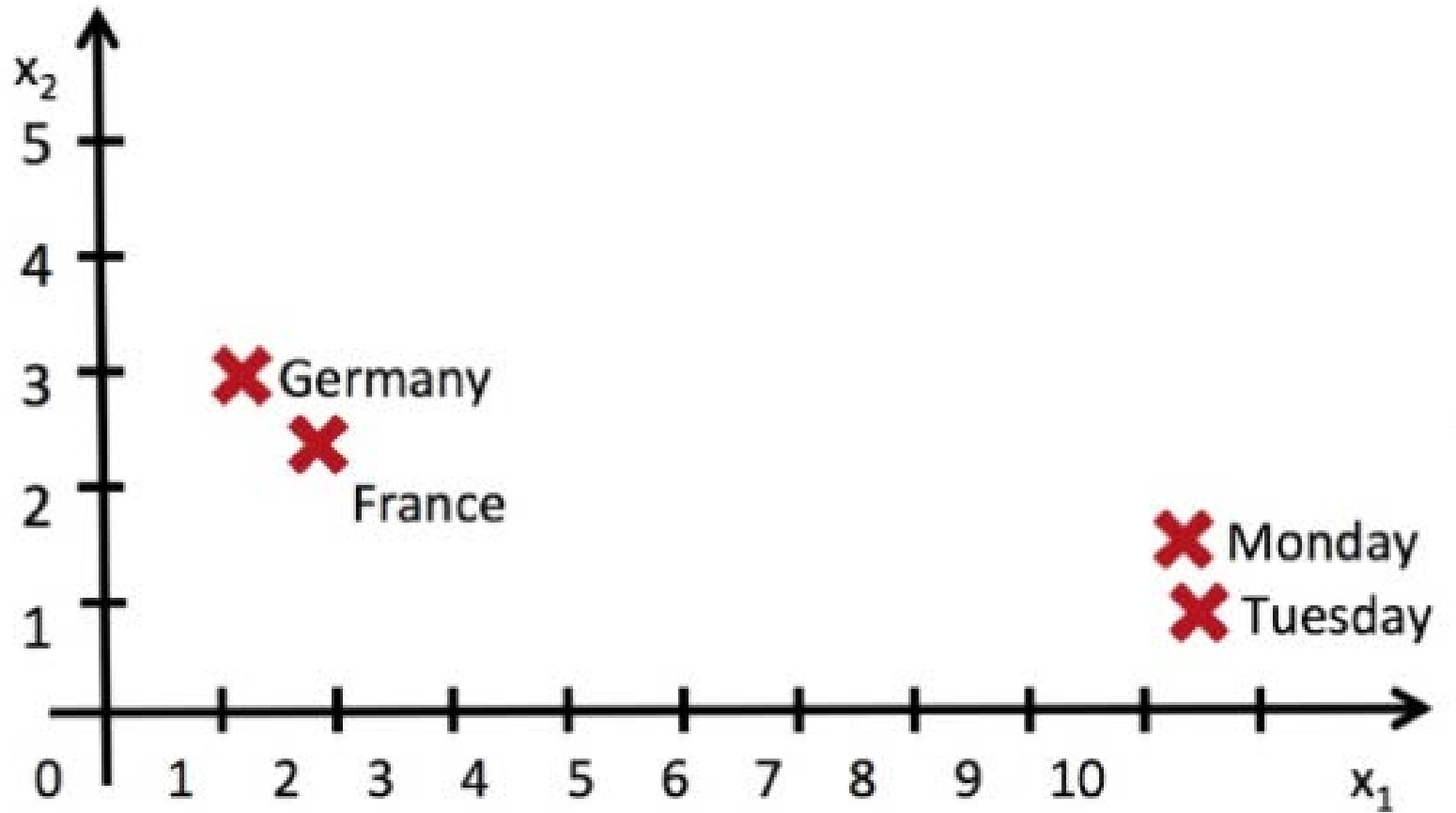
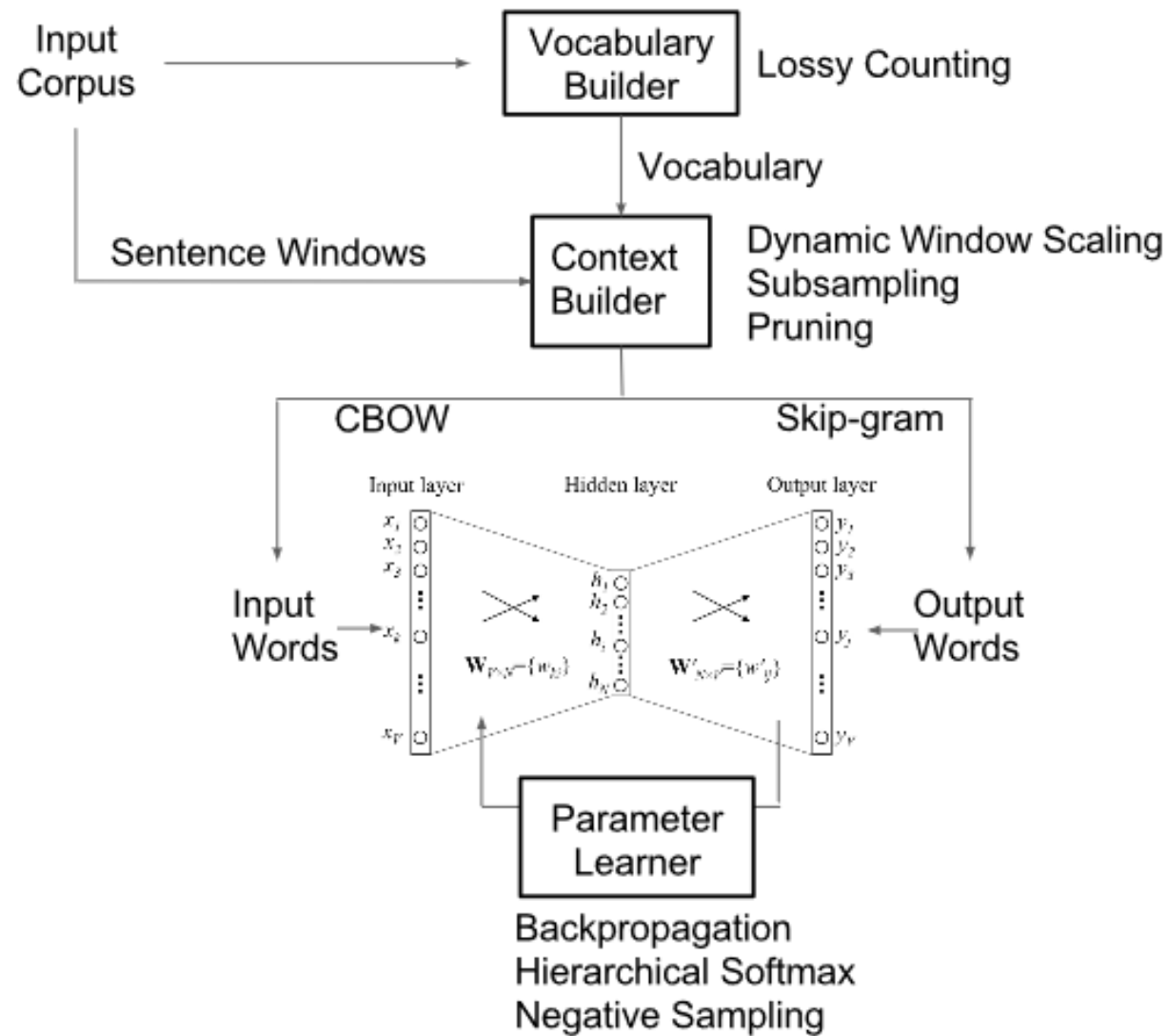


Figure (edited) from Bengio, "Representation Learning and Deep Learning", July, 2012, UCLA

Arquitetura

- O treinamento pode utilizar duas abordagens:
- CBOW (continuous bag of words)
 - Treinamento para prever uma palavra dado os vizinhos da janela
- Skip-Grams
 - Treinamento para prever o contexto dado uma palavra.

Arquitetura



Arquitetura

Hidden Layer
Linear Neurons

Output Layer
Softmax Classifier

Probability that the word at a
randomly chosen, nearby
position is **"abandon"**

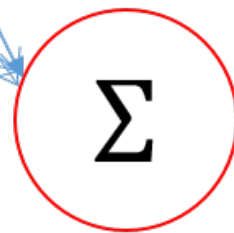
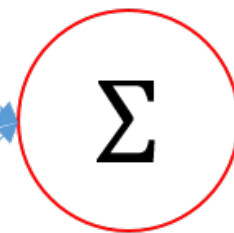
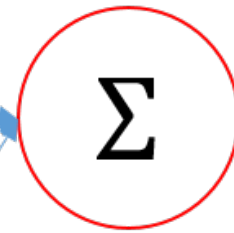
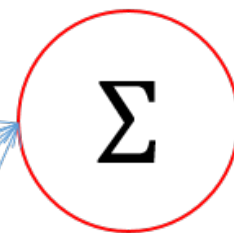
... **"ability"**

... **"able"**

... **"zone"**



300 neurons



10,000

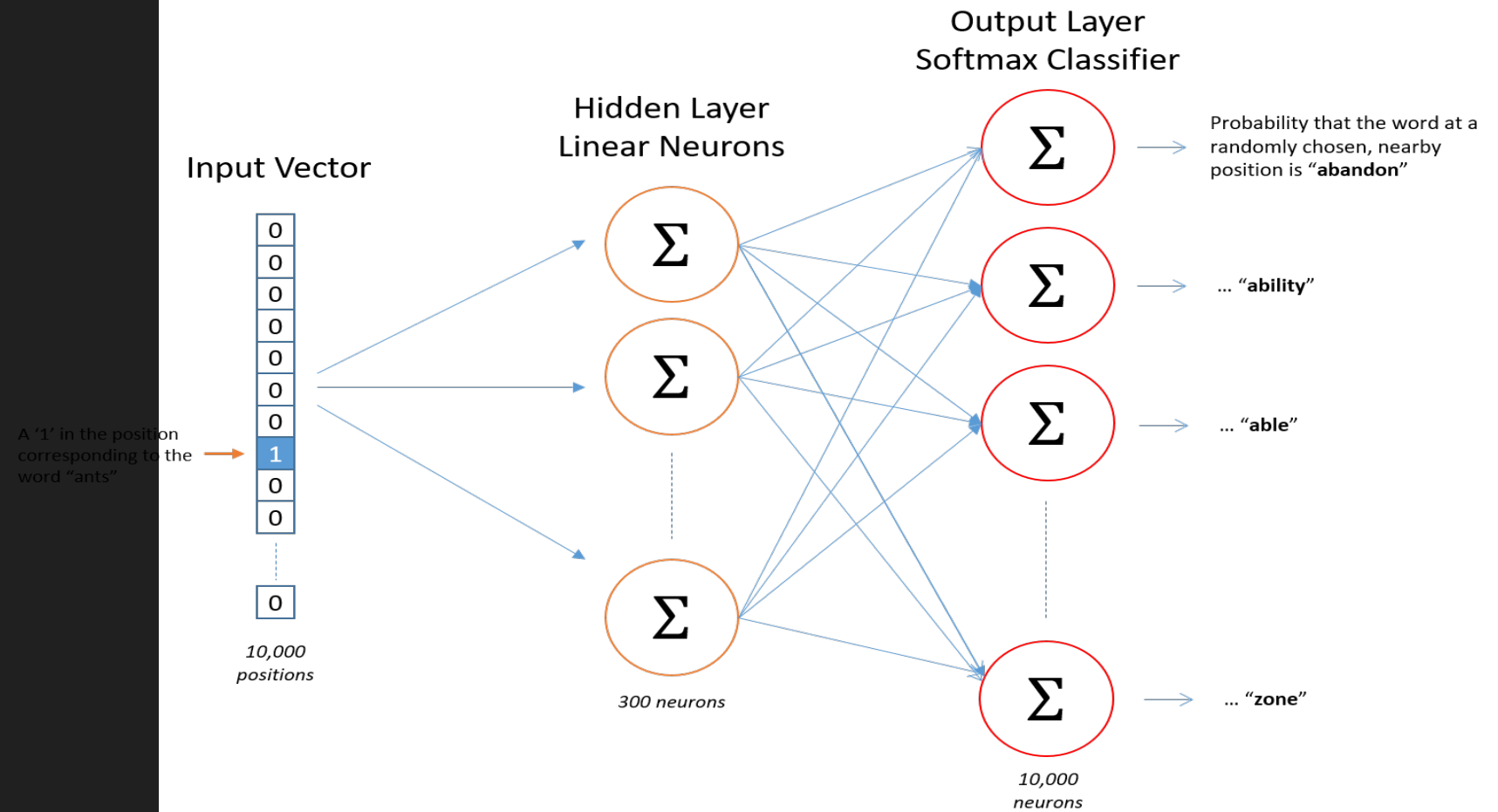


10,000
positions

A '1' in the position
corresponding to the
word "ants"

Arquitetura

- Não tem função de ativação
- Camada de saída usa softmax.



Context windows

- Context can be anything – a surrounding n-gram, a randomly sampled set of words from a fixed size window around the word

For example, assume context is defined as the word following a word.

i.e. $\text{context}(w_i) = w_{i+1}$

Corpus : I ate the cat

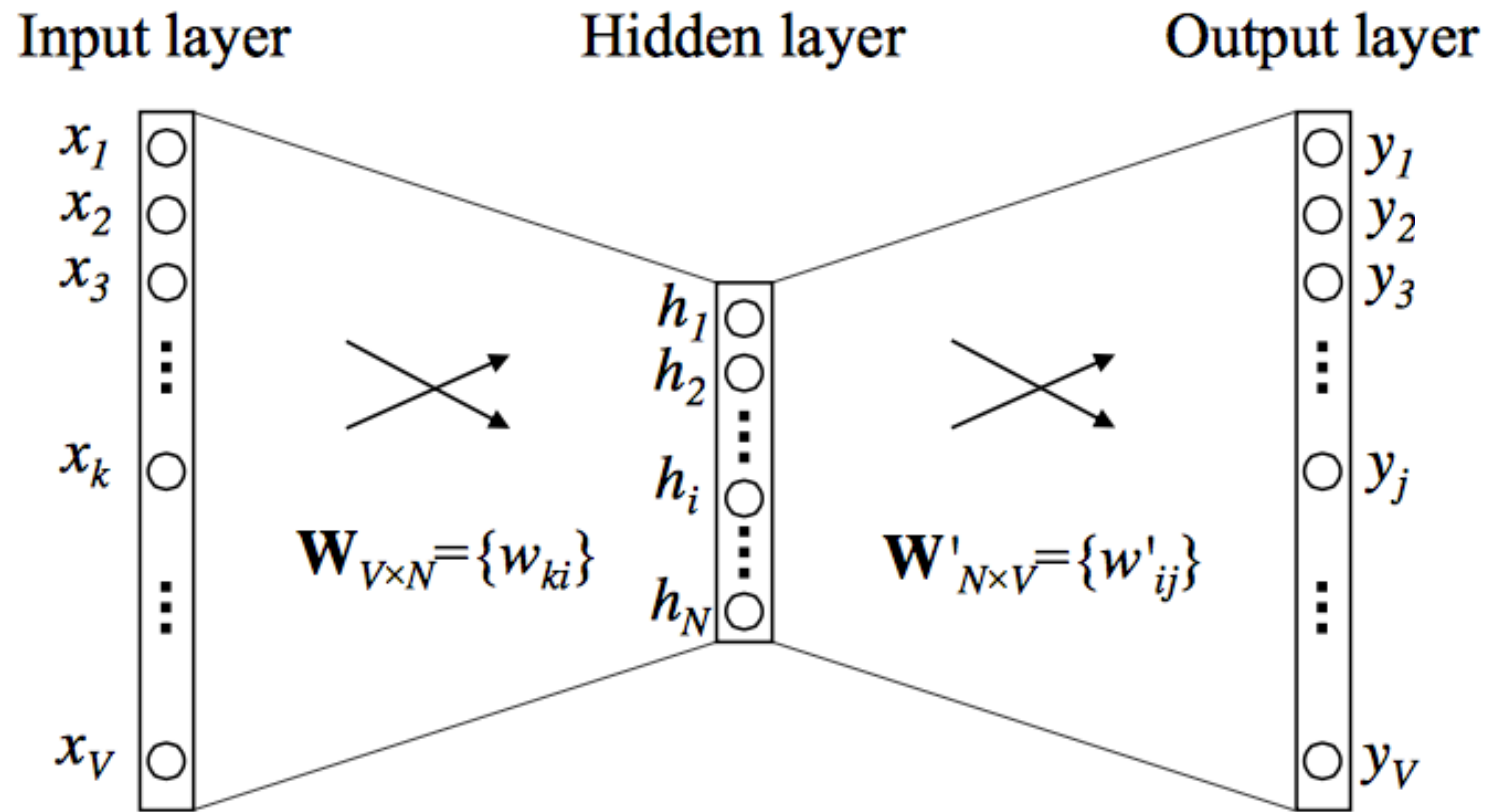
Training Set : I|ate, ate|the , the|cat, cat|.

1. eat | apple
2. eat | orange
3. eat | rice
4. drink | juice
5. drink | milk
6. drink | water
7. orange | juice
8. apple | juice
9. rice | milk
10. milk | drink
11. water | drink
12. juice | drink

Treinamento :

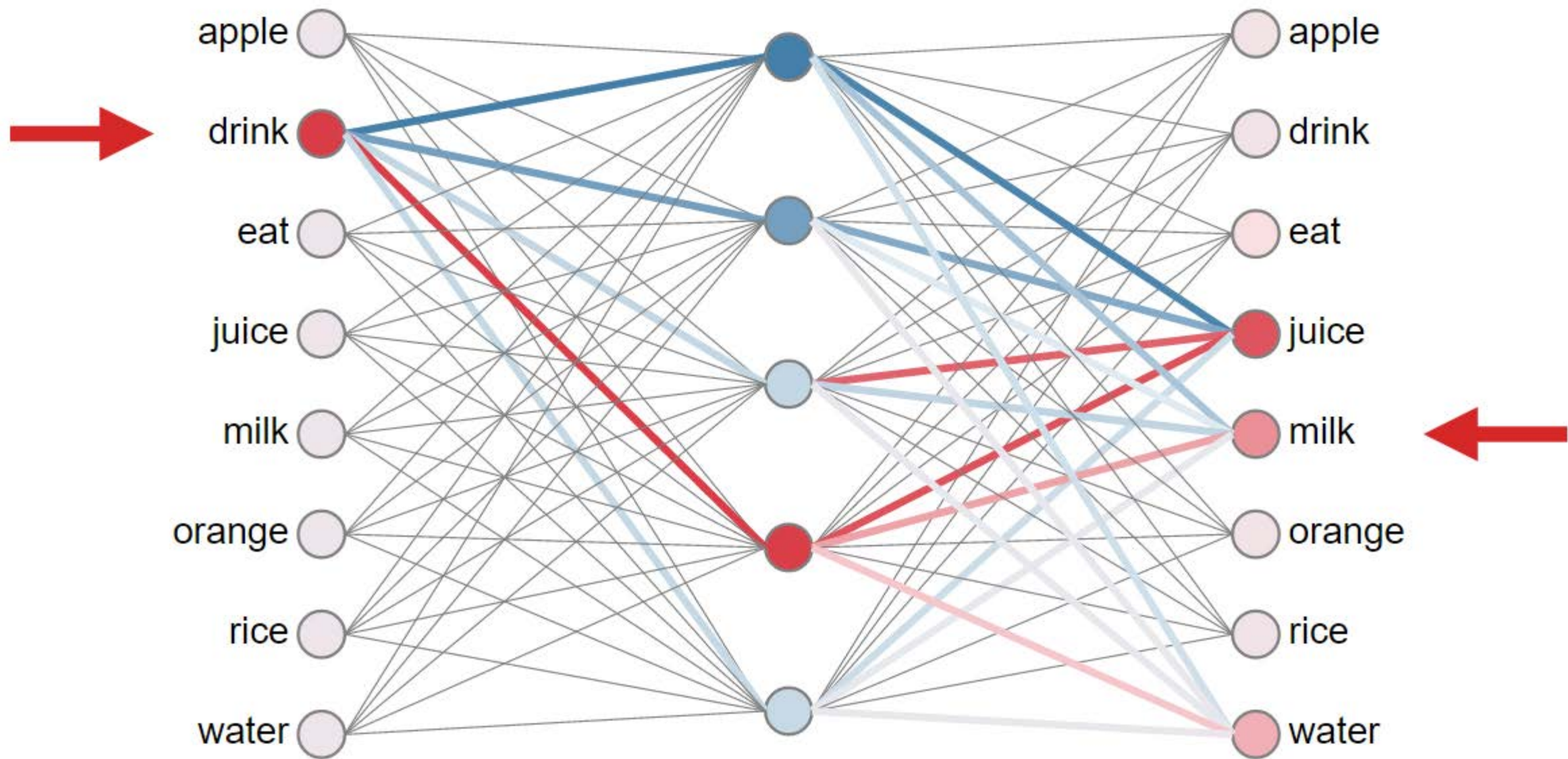
1. Milk and Juice are drinks
2. Apples, Oranges and Rice can be eaten
3. Apples and Orange are also juices
4. Rice milk is actually a type of milk!

Ideia intuitiva



$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t)$$

$$p(w_O | w_I) = \frac{\exp(v'_{w_O}{}^\top v_{w_I})}{\sum_{w=1}^W \exp(v'_w{}^\top v_{w_I})}$$



Word2Vec Resultados interesantes

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks



Word2Vec Resultados interessantes

- Interesting/Humorous Word2Vec Math

Input Equation	Closest Word
King-Man+Woman	Queen
Human-Animal	Ethics
Library-Books	Hall
Obama-USA+Russia	Putin
President-Power	Prime Minister

Exemplos:

```
from gensim.models import KeyedVectors
# cria o modelo
pt_model = KeyedVectors.load_word2vec_format('wiki.pt.vec')
# Escolhe uma palavra de busca
find_similar_to = 'sabão'

# Busca por similaridade [default= top 10]
for similar_word in pt_model.similar_by_word(find_similar_to):
    print("Palavras: {0}, Similarity: {1:.2f}".format(
        similar_word[0], similar_word[1]
    ))
```

Exemplos:

Saída

Word: cars, Similarity: 0.83

Word: automobile, Similarity: 0.72

Word: truck, Similarity: 0.71

Word: motorcar, Similarity: 0.70

Word: vehicle, Similarity: 0.70

Word: driver, Similarity: 0.69

Word: drivecar, Similarity: 0.69

Word: minivan, Similarity: 0.67

Word: roadster, Similarity: 0.67

Word: racecars, Similarity: 0.67

Exemplos:

```
# Test words
```

```
word_add = ['dhaka', 'india']
```

```
word_sub = ['bangladesh']
```

```
# Word vector addition and subtraction
```

```
for resultant_word in en_model.most_similar(
```

```
    positive=word_add, negative=word_sub
```

```
):
```

```
    print("Word : {0} , Similarity: {1:.2f}".format(
```

```
        resultant_word[0], resultant_word[1]
```

```
    ))
```

Exemplos:

Resultado da adição semântica:

Políticos + dinheiro + Empreiteiras + Leis

```
Dimensão do vetor de palavras: 300
Palavra similar: politico, Grau de similaridade: 0.79
Palavra similar: politicas, Grau de similaridade: 0.77
Palavra similar: politica, Grau de similaridade: 0.77
Palavra similar: politicagens, Grau de similaridade: 0.74
Palavra similar: br/politica/politicos, Grau de similaridade: 0.72
Palavra similar: politicagem, Grau de similaridade: 0.72
Palavra similar: políticos, Grau de similaridade: 0.71
Palavra similar: desaparecidospoliticos, Grau de similaridade: 0.69
Palavra similar: politiche, Grau de similaridade: 0.67
Palavra similar: politici, Grau de similaridade: 0.67
Resultados adição: empreiteiros , Similaridade: 0.70
Resultados adição: empreiteira , Similaridade: 0.70
Resultados adição: licitações , Similaridade: 0.69
Resultados adição: propinas , Similaridade: 0.69
Resultados adição: desonerações , Similaridade: 0.68
Resultados adição: subvenções , Similaridade: 0.68
Resultados adição: empreiteiro , Similaridade: 0.66
Resultados adição: despesas , Similaridade: 0.66
pcfapeg@pcfapeg:~/Downloads/anderson/PLN$
```

www.deeeplearningbrasil.com.br

Word2Vec Vantagens

- Word2vec é apenas uma rede neural
- Equivalente a one-hot encoding
- ...então pq usar word2vec?
- Pre-training

Word2Vec Desvantagens

- Ainda sujeita ao problema Out-of-vocabulary (OOV)
- Como gerar vetores de palavras desconhecidas?
- Como aprimorar a representação de palavras pouco usadas?
 - Usar “unknown”?



word2vec

- word2vec não é um simples algoritmo
- É um pacote de **software package** para representar palavras como vetores, contendo:
 - Dois modelos distintos
 - CBoW
 - Skip-Gram
 - Vários métodos de treinamento
 - Negative Sampling
 - Hierarchical Softmax
 - Um pipeline de pré-processamento rico
 - Dynamic Context Windows
 - Subsampling
 - Deleting Rare Words



word2vec

- Negative sampling

$$\log \sigma(v'_{w_O}{}^\top v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-v'_{w_i}{}^\top v_{w_I}) \right]$$

- Note que a parte positiva modela a probabilidade de a palavra (w_O) co-ocorrer com a palavra de saída (w_I). A parte negativa está tentando minimizar a probabilidade esperada de uma palavra aleatória co-ocorrendo com a palavra de entrada.



Skip-Grams com Negative Sampling (SGNS)

Marco viu um pequeno **vampiro** peludo pendurado na árvore.

words

vampiro

vampiro

vampiro

vampiro

...

contexts

peludo

pequeno

pendurado

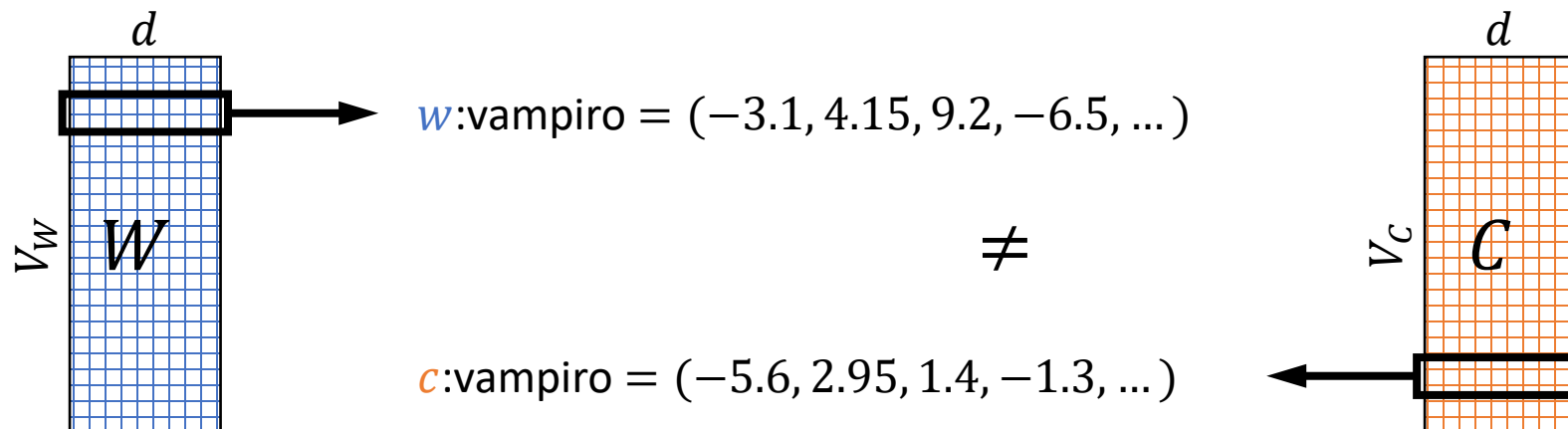
na

...

D (data)

Skip-Grams com Negative Sampling (SGNS)

- SGNS procura um vetor \vec{w} para cada palavra w no vocabulário V_W
- Cada vetor tem d dimensões (e.g. $d = 100$)
- Efetivamente, ele aprende uma matriz W que representa V_W
- **Ponto chave:** matriz C do contexto dos vetores
- Em fato, cada palavra tem dois embeddings





Skip-Grams com Negative Sampling (SGNS)

- **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$
 - c foi **observada** com w

palavras

vampiro

vampiro

vampiro

vampiro

contexto

peludo

pequeno

segurando

no



Skip-Grams with Negative Sampling (SGNS)

• **Maximize:** $\sigma(\vec{w} \cdot \vec{c})$

- c foi **observada** com w

words

vampiro

vampiro

vampiro

vampiro

contexts

peludo

pequeno

segurando

no

• **Minimize:** $\sigma(\vec{w} \cdot \vec{c}')$

- c' com w

words

vampiro

vampiro

vampiro

vampiro

contexts

Australia

virtual

o

1985

Word2Vec

- Sumário
 - Objetivo simples (prever palavras vizinhas) permite aprender relações úteis
 - Tarefas simples podem ser usadas para treinar representações para problemas complexos
 - “You shall know a word by the company it keeps”

Global Vectors for Word Representation (GloVe)

Conceito

- Usa razão de probabilidades de co-ocorrência, em vez de probabilidades de co-ocorrência em si

Probability and Ratio	$k = \text{solid}$	$k = \text{gas}$	$k = \text{water}$	$k = \text{fashion}$
$P(k \text{ice})$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k \text{steam})$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k \text{ice})/P(k \text{steam})$	8.9	8.5×10^{-2}	1.36	0.96

- Seja $P(k | w)$ a probabilidade da palavra k aparecer no contexto da palavra w . Considere uma palavra fortemente relacionada ao gelo, mas não ao vapor, como o sólido. $P(\text{sólido} | \text{gelo})$ será relativamente alto, e $P(\text{sólido} | \text{vapor})$ será relativamente baixo. Assim, a proporção de $P(\text{sólido} | \text{gelo}) / P(\text{sólido} | \text{vapor})$ será grande. Se tomarmos uma palavra, como o gás relacionado ao vapor, mas não ao gelo, a proporção de $P(\text{gás} | \text{gelo}) / P(\text{gás} | \text{vapor})$ será pequena. Para uma palavra relacionada ao gelo e ao vapor, como a água, esperamos que a proporção seja próxima de uma. Também esperamos uma proporção próxima a uma palavra relacionada ao gelo e ao vapor.

GloVe – Algoritmo de treinamento

- Calcule a estatística e co-ocorrência na matriz X. Cada element X_{ij} de uma matriz representa o quanto uma palavra i aparece no context da palavra j , porém ponderando a distância:

$\text{decay} = 1/\text{offset}$

- Defina restrições para cada par de palavras: $w_i^T w_j + b_i + b_j = \log(X_{ij})$

w_i – vetor para a palavra principal,

w_j – vetor para o context

- Defina uma função de custo

$$J = \sum_{i=1}^V \sum_{j=1}^V f(X_{ij}) (w_i^T w_j + b_i + b_j - \log X_{ij})^2$$

GloVe – Algoritmo de treinamento

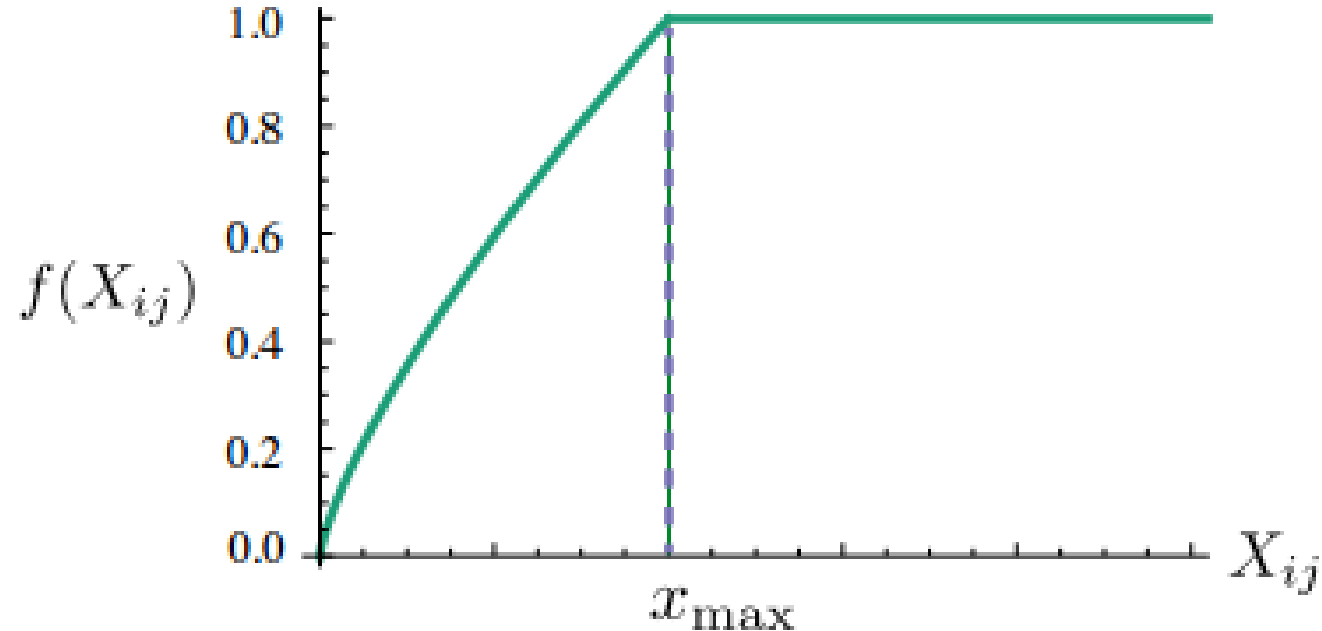
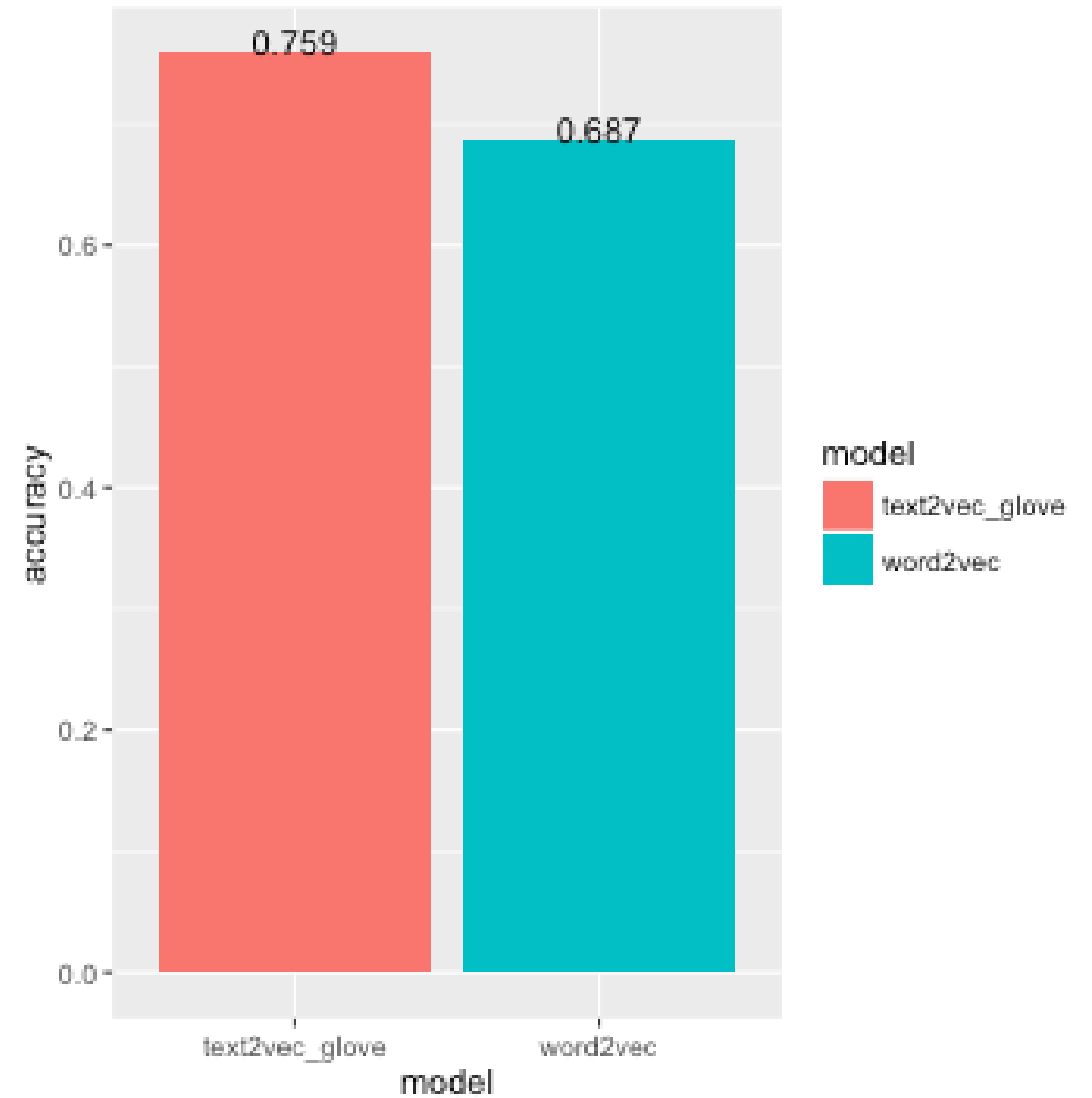
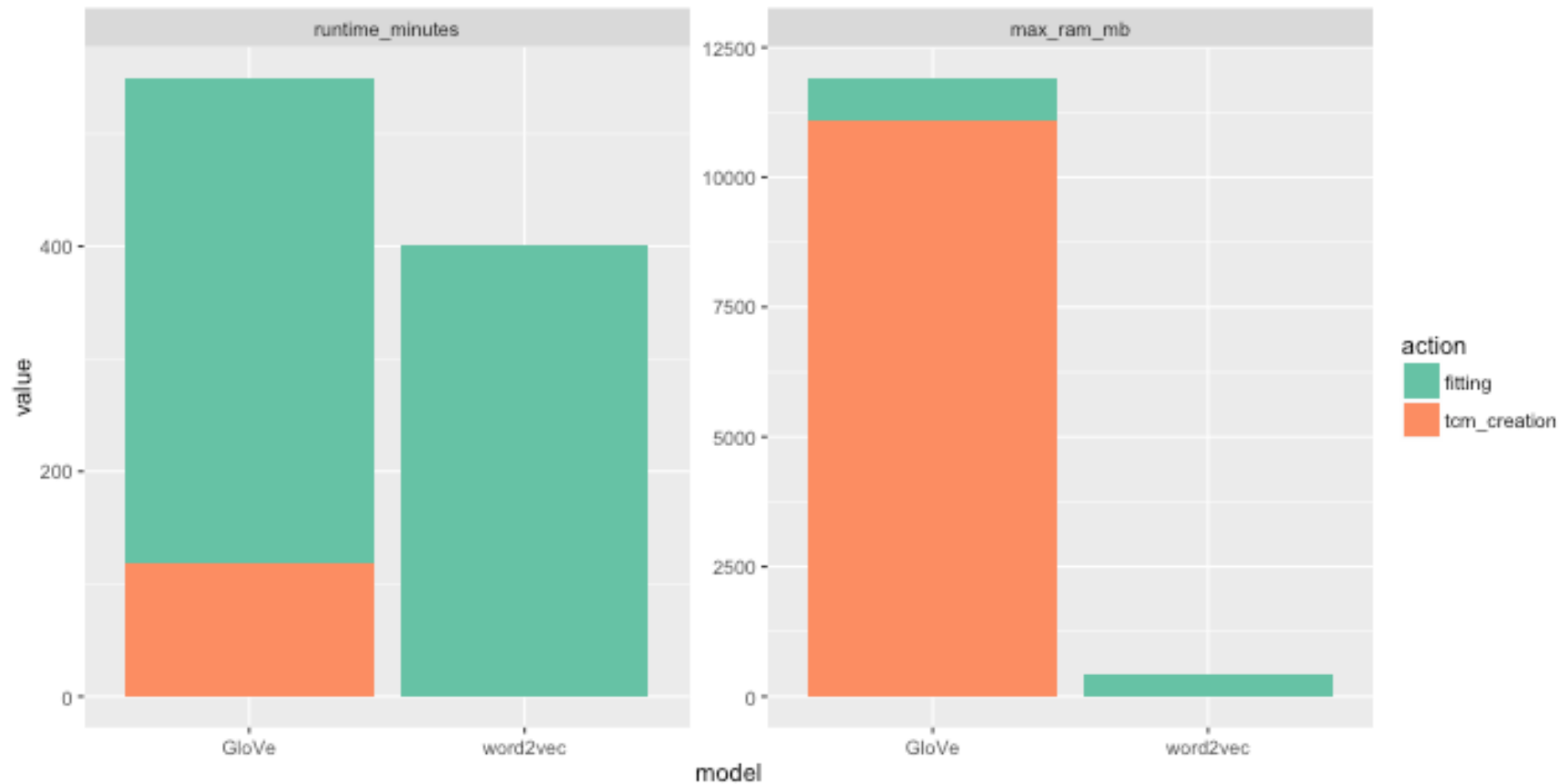


Figure 1: Weighting function f with $\alpha = 3/4$.

Word2Vec versus GloVe

Fonte: <http://dsnotes.com/post/glove-enwiki/>

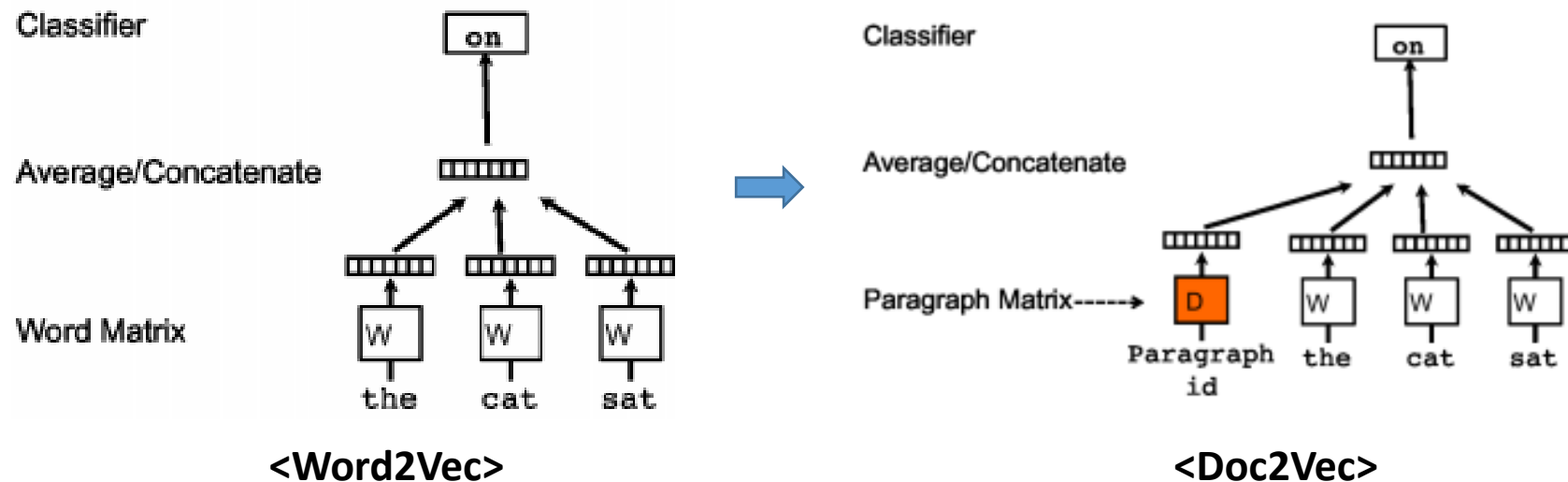




Word2Vec versus GloVe

Fonte: <http://dsnotes.com/post/glove-enwiki/>

Doc2Vec



Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents

- Extensão de Word2Vec: um document é considerado um palavra
- Como resultado, documentos são representados no espaço contínuo

Bag of Tricks for Efficient Text Classification

Fast text classification

- BoW model on text classification and tag prediction

Starsmith (born Finlay Dow-Smith 8 July 1988 Bromley England) is a British songwriter producer remixer and DJ. He studied a classical music degree at the University of Surrey majoring in performance on saxophone. He has already received acclaim for the remixes he has created for Lady Gaga Robyn Timbaland Katy Perry Little Boots Passion Pit Paloma Faith Marina and the Diamonds and Frankmusik amongst many others.

ARTIST

Rikkavesi is a medium-sized lake in eastern Finland. At approximately 63 square kilometres (24 sq mi) it is the 66th largest lake in Finland. Rikkavesi is situated in the municipalities of Kaavi Outokumpu and Tuusniemi. Rikkavesi is 101 metres (331 ft) above the sea level. Kaavinjärvi and Rikkavesi are connected by the Kaavinkoski Canal. Ohtaanselkä strait flows from Rikkavesi to Juojärvi.

Natural Place

- A very strong (and fast) **baseline**, often on-par with SOTA approaches
- Ease of use is at the core of the library

```
./fasttext supervised -input data/dbpedia.train -output data/dbpedia
```

```
./fasttext test data/dbpedia.bin data/dbpedia.test
```

Model

- Model probability of a **label** given a **paragraph**

feature for paragraph \mathcal{P} : $h_{\mathcal{P}}$

classifier for label l : v_l

$$p(l|\mathcal{P}) = \frac{e^{h_{\mathcal{P}}^{\top} v_l}}{\sum_{k=1}^K e^{h_{\mathcal{P}}^{\top} v_k}}$$

- Paragraph feature

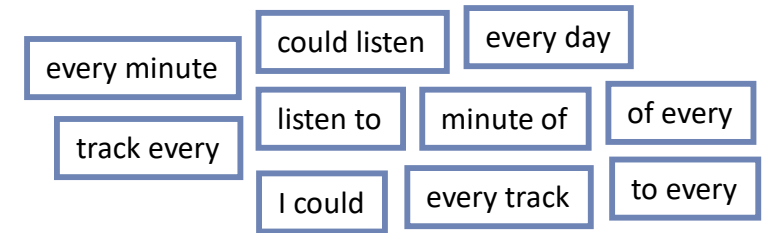
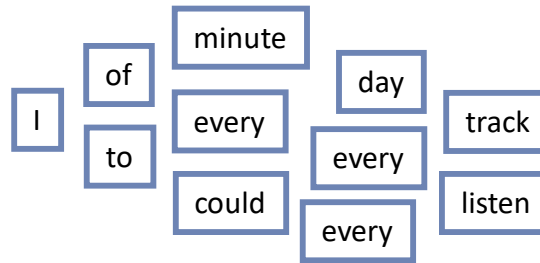
$$h_{\mathcal{P}} = \sum_{w \in \mathcal{P}} x_w$$

- Word vectors are **latent** and not useful *per se*
- If **scarce** supervised data, use **pre-trained** word vectors
- Fasttext é um classificador em cima de um modelo de sentença2vec e é isso.

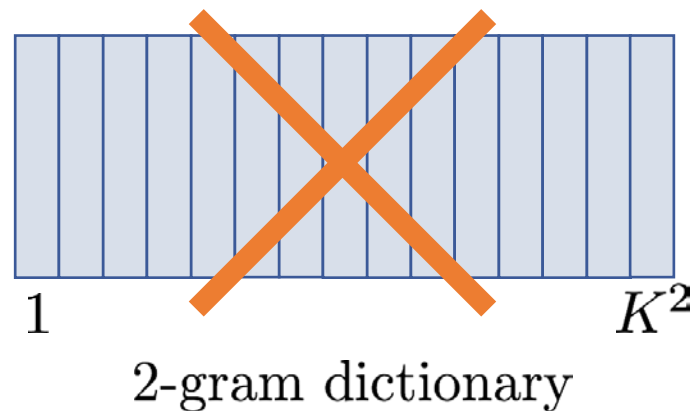
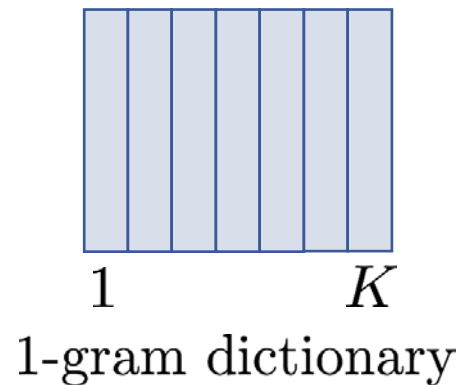
n-grams

- Possible to add higher-order features

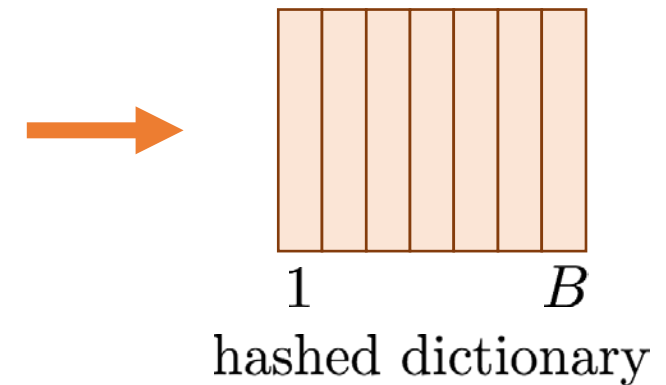
*I could listen to every track
every minute of every day.*



- Avoid building n-gram dictionary



Use a hashed dictionary!



- Um vetor para uma palavra é feito a partir de uma soma de n grams caracteres.
- Por exemplo o vetor “apple” é uma soma de vetores de n -grams “<ap”, “app”, “appl”, “apple”, “apple>”, “ppl”, “pple”, “pple>”, “ple”, “ple>”, “le>” (assumindo *hyperparâmetros* $ngram[minn]$ sendo 3 e o maior $ngram[maxn]$ sendo 6).

Glove x word2vec x Fasttext

- Glove trata cada palavra no corpus como uma entidade atômica e gera um vetor para cada palavra. Neste sentido, Glove é muito parecida com word2vec - ambas tratam as palavras como a unidade mais pequena para treinar.
- Fasttext, que é essencialmente uma extensão do modelo word2vec, trata cada palavra como composta de caracteres ngrams. Então, o vetor para uma palavra é feito da soma desses caracteres n gramas.
- Gerar melhores relações para palavras raras (mesmo que as palavras sejam raras, seu n gram ainda são compartilhados com outras palavras)
- Palavras fora de vocabulário - eles podem construir o vetor para uma n gram, mesmo que a palavra não apareça no corpo de treinamento. Tanto a Glove como o word2vec não podem.

Sentiment analysis - performance

Model	AG	Sogou	DBP	Yelp P.	Yelp F.	Yah. A.	Amz. F.	Amz. P.
BoW (Zhang et al., 2015)	88.8	92.9	96.6	92.2	58.0	68.9	54.6	90.4
ngrams (Zhang et al., 2015)	92.0	97.1	98.6	95.6	56.3	68.5	54.3	92.0
ngrams TFIDF (Zhang et al., 2015)	92.4	97.2	98.7	95.4	54.8	68.5	52.4	91.5
char-CNN (Zhang and LeCun, 2015)	87.2	95.1	98.3	94.7	62.0	71.2	59.5	94.5
char-CRNN (Xiao and Cho, 2016)	91.4	95.2	98.6	94.5	61.8	71.7	59.2	94.1
VDCNN (Conneau et al., 2016)	91.3	96.8	98.7	95.7	64.7	73.4	63.0	95.7
<code>fastText</code> , $h = 10$	91.5	93.9	98.1	93.8	60.4	72.0	55.8	91.2
<code>fastText</code> , $h = 10$, bigram	92.5	96.8	98.6	95.7	63.9	72.3	60.2	94.6

Table 1: Test accuracy [%] on sentiment datasets. `FastText` has been run with the same parameters for all the datasets. It has 10 hidden units and we evaluate it with and without bigrams. For char-CNN, we show the best reported numbers without data augmentation.

Sentiment analysis - runtime

	Zhang and LeCun (2015)		Conneau et al. (2016)			fastText
	small char-CNN	big char-CNN	depth=9	depth=17	depth=29	h = 10, bigram
AG	1h	3h	24m	37m	51m	1s
Sogou	-	-	25m	41m	56m	7s
DBpedia	2h	5h	27m	44m	1h	2s
Yelp P.	-	-	28m	43m	1h09	3s
Yelp F.	-	-	29m	45m	1h12	4s
Yah. A.	8h	1d	1h	1h33	2h	5s
Amz. F.	2d	5d	2h45	4h20	7h	9s
Amz. P.	2d	5d	2h45	4h25	7h	10s

Table 2: Training time for a single epoch on sentiment analysis datasets compared to char-CNN and VDCNN.



Tag prediction

- Using Flickr Data
- Given an image caption
- Predict the most likely tag
- Sample outputs:

Input	Prediction
taiyoucon 2011 digitals: individuals digital photos from the anime convention taiyoucon 2011 in mesa, arizona. if you know the model and/or the character, please comment.	#cosplay
2012 twin cities pride 2012 twin cities pride parade	#minneapolis
beagle enjoys the snowfall	#snow

Model	prec@1	Running time	
		Train	Test
Freq. baseline	2.2	-	-
Tagspace, h = 50	30.1	3h8	6h
Tagspace, h = 200	35.6	5h32	15h
fastText, h = 50	31.2	6m40	48s
fastText, h = 50, bigram	36.7	7m47	50s
fastText, h = 200	41.1	10m34	1m29
fastText, h = 200, bigram	46.1	13m38	1m37

Table 5: Prec@1 on the test set for tag prediction on YFCC100M. We also report the training time and test time. Test time is reported for a single thread, while training uses 20 threads for both models.

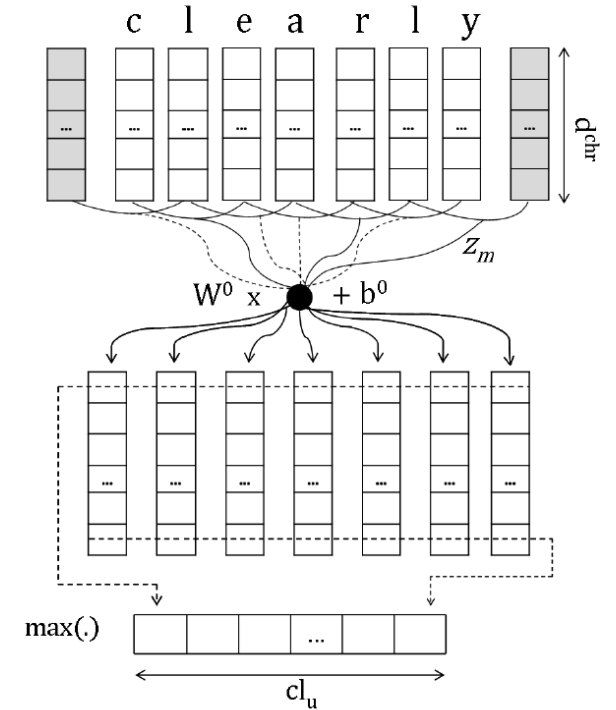
Modelos baseados em caracteres

Modelos baseados em caracteres

- Palavras são construídas por caracteres
 - Gera modelos a partir de palavras conhecidas
 - Expressões semelhantes compartilham modelos similares
 - Resolve o problema OOV

Modelos baseados em caracteres

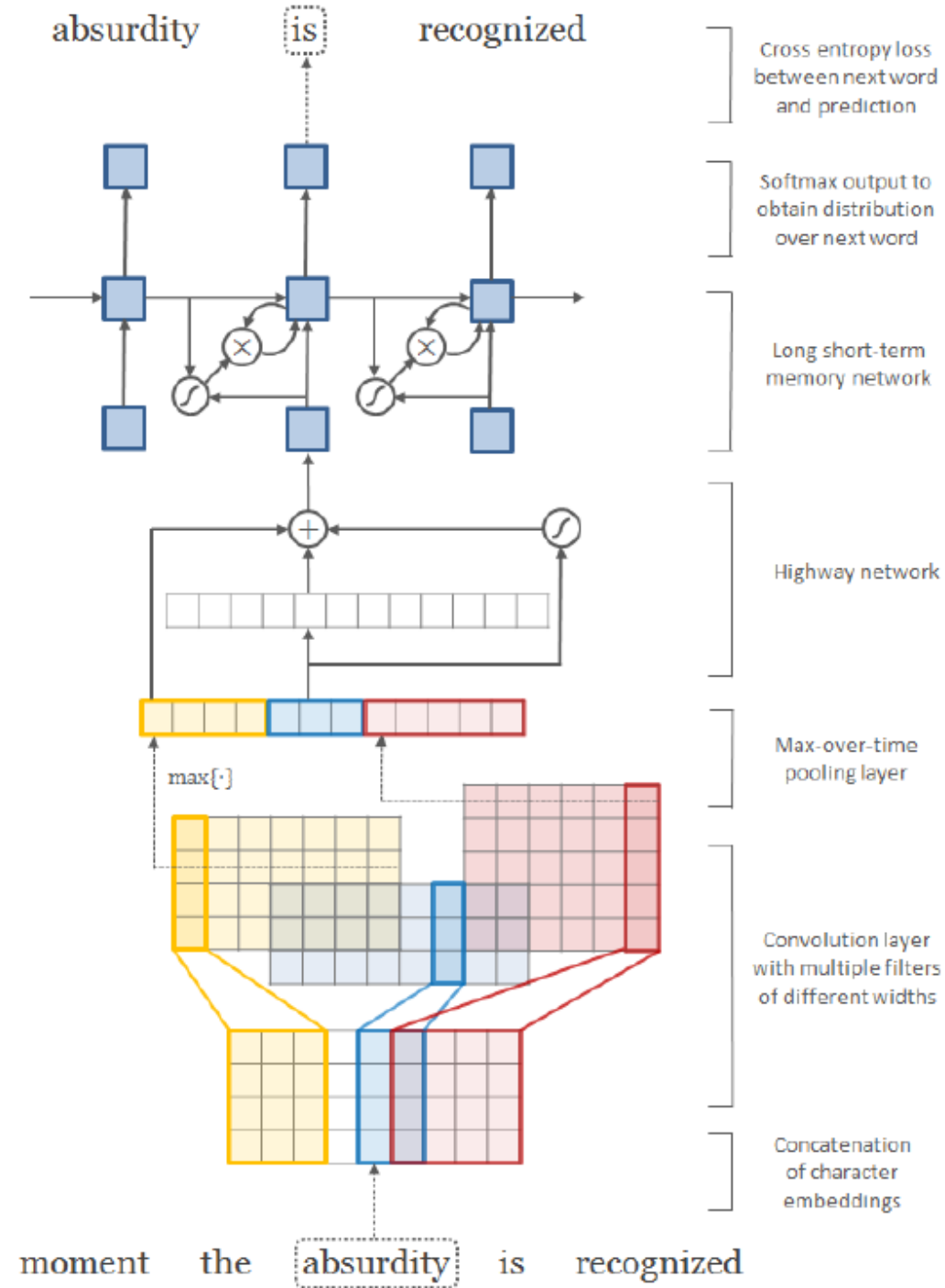
- Dos Santos and Zadrozny (2014) Learning Character level Representations for Part-of-Speech Tagging
- Convoluções sobre os caracteres
- Janela fixa





Modelos baseados em caracteres

- Kim, Jernite, Sontag, and Rush (2015) Character-Aware Neural Language Models
- Utiliza convolução, LSTM



Modelos baseados em caracteres

- Cao and Rei (2016) A Joint Model for Word Embedding and Word Morphology
- Mesmo objetivo da word2vec, porém com caracteres
- LSTM Bi-direcional
- Modelo capaz de inferir origem das palavras

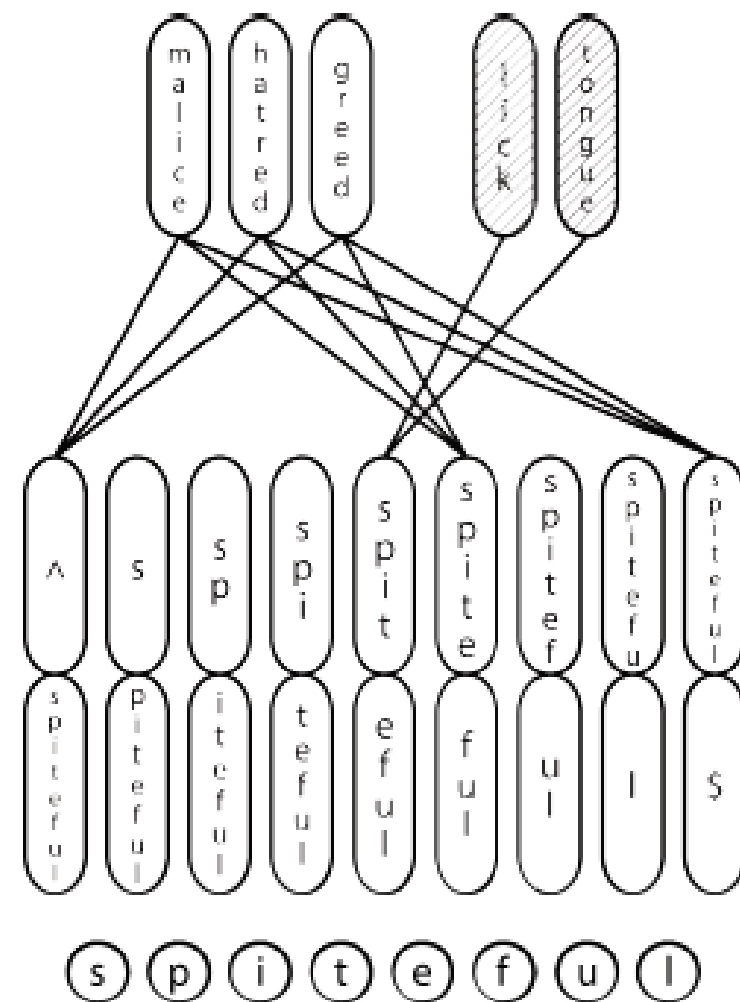


Figure 3: An illustration of the attention model (start and end of word symbols omitted). The root morpheme contributes the most to predicting the context, and is upweighted. In contrast, another potential split is inaccurate, and predicts the wrong context words. This is downweighted.

- Bag of Words

- Recurrent Networks

- Convolutional Networks

- Tree-Structured Networks



Fácil

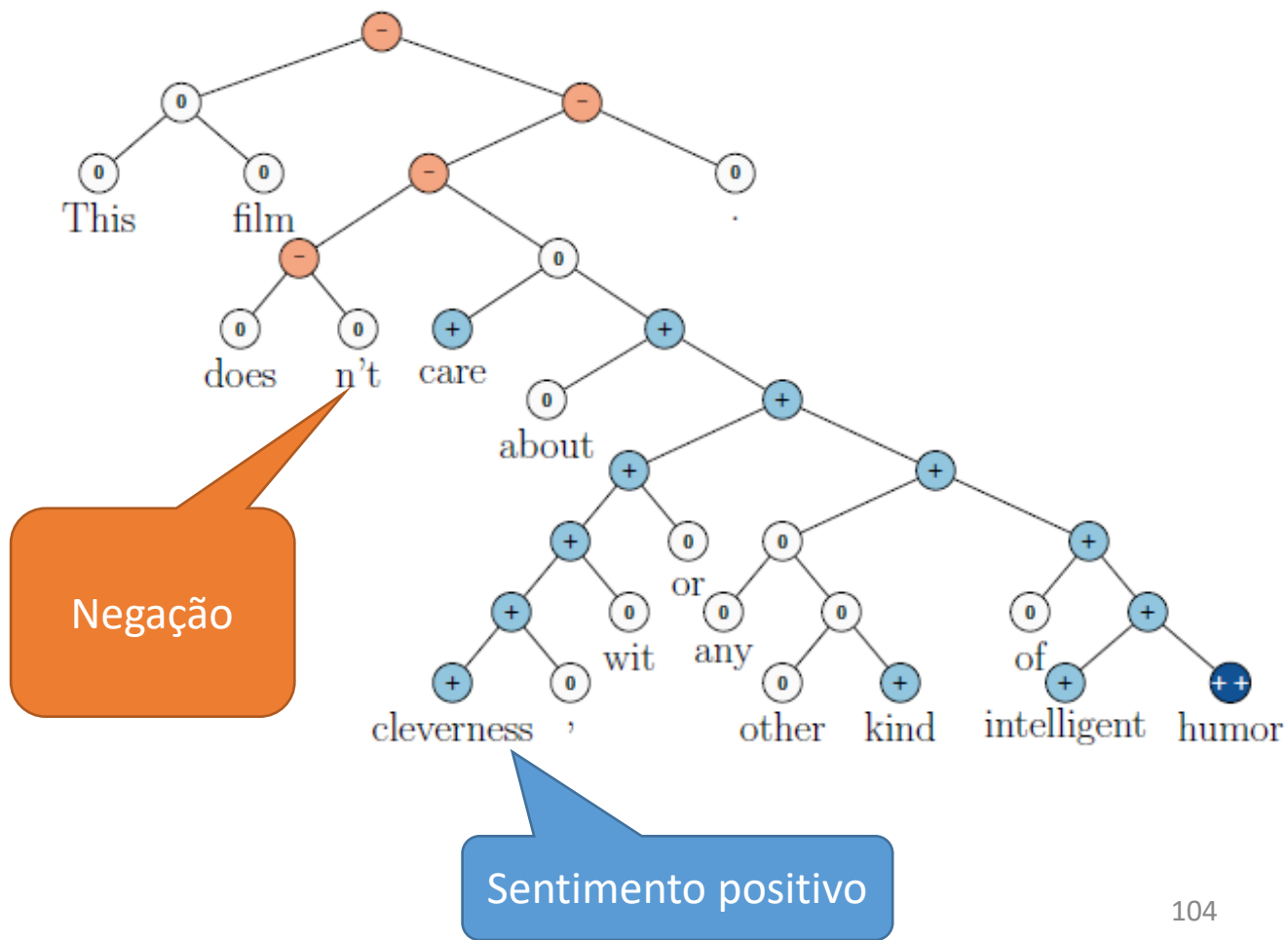
Difícil

Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank

Socher, Perelygin, Wy, Chuang, Manning, Ng, and Potts (2013)

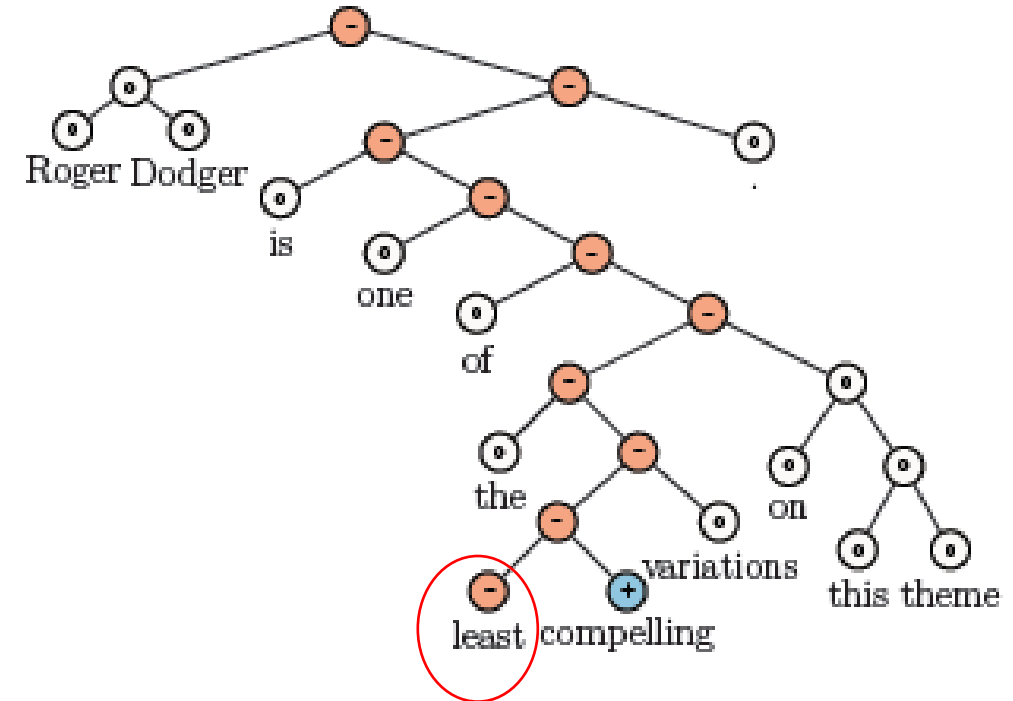
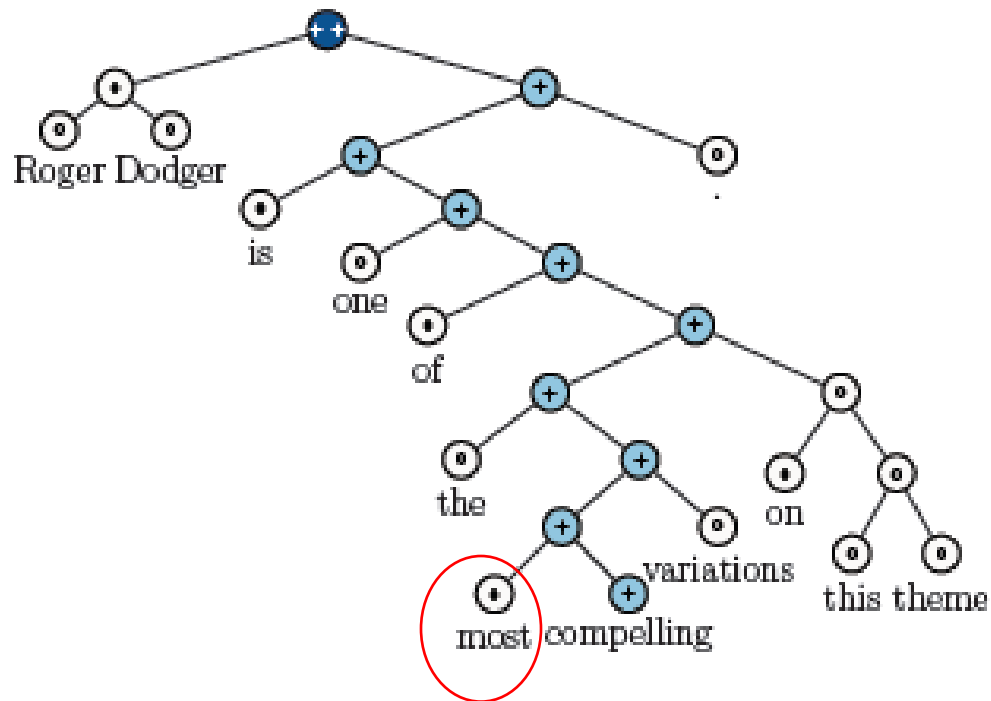
Recursive Compositionality

- Frases são combinadas recursivamente
- Positivo e negativo podem ser negados



Recursive Sentiment Calculations

- Duas sentenças similares



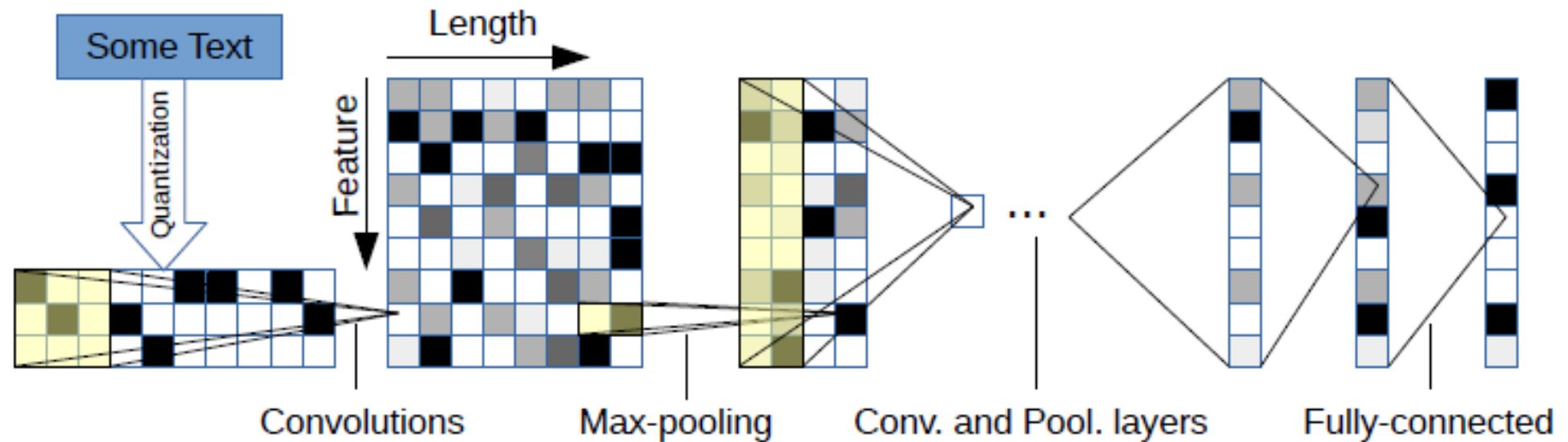


Character-level Convolutional Networks for Text Classification

Zhang, Zhao and LeCun (2016)

- Algumas soluções utilizam estruturas
 - Caracteres
 - Palavras
 - Frases
 - Sentenças
 - Parágrafos
 - Documentos
- LeCun usa convolução sobre caracteres
 - Estrutura pode ser “aprendida”
 - Sem necessidade de tokenizing/parsing/tagging/etc.

Convolução



$$h(y) = \sum_{x=1}^k f(x) \cdot g(y \cdot d - x + c)$$

1D Convolution

$$h(y) = \max_{x=1}^k g(y \cdot d - x + c)$$

1D Pooling

Data Augmentation

- Data augmentation em NLP
 - Troque palavras por sinônimos
 - LibreOffice thesaurus
 - LibreOffice thesaurus construída a partir da WordNet

Convolução 1D

Estado da arte

- **Dynamic Convolutional Neural Network – DCNN - 2015**

A Convolutional Neural Network for Modelling Sentences

Nal Kalchbrenner

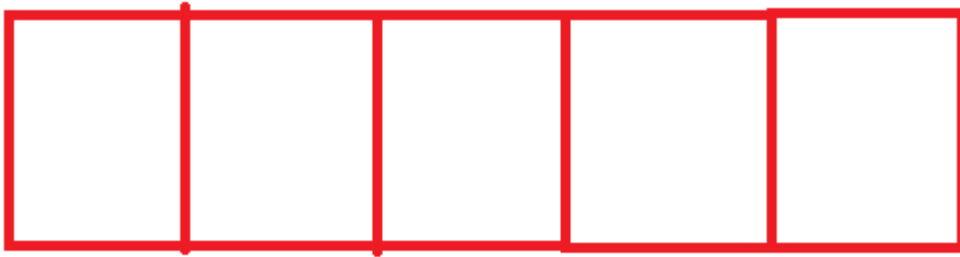
Edward Grefenstette

Phil Blunsom

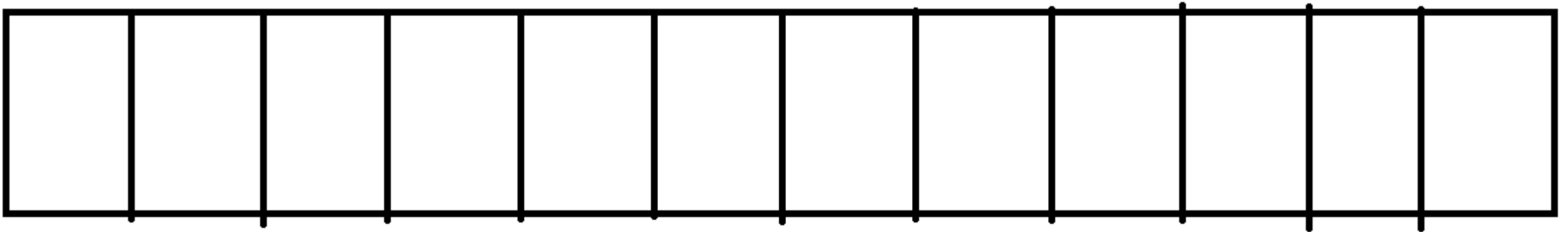
`{nal.kalchbrenner, edward.grefenstette, phil.blunsom}@cs.ox.ac.uk`

Department of Computer Science
University of Oxford

Convolução 1D

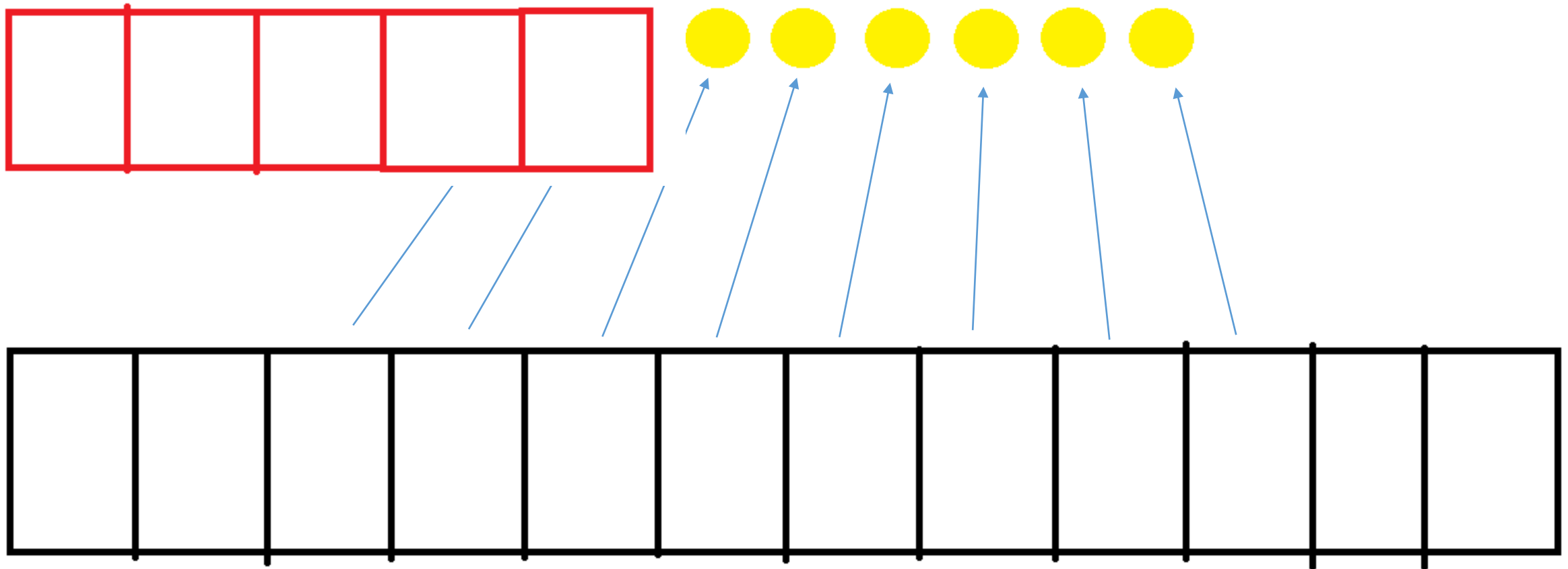


Vetor de pesos
Size: m

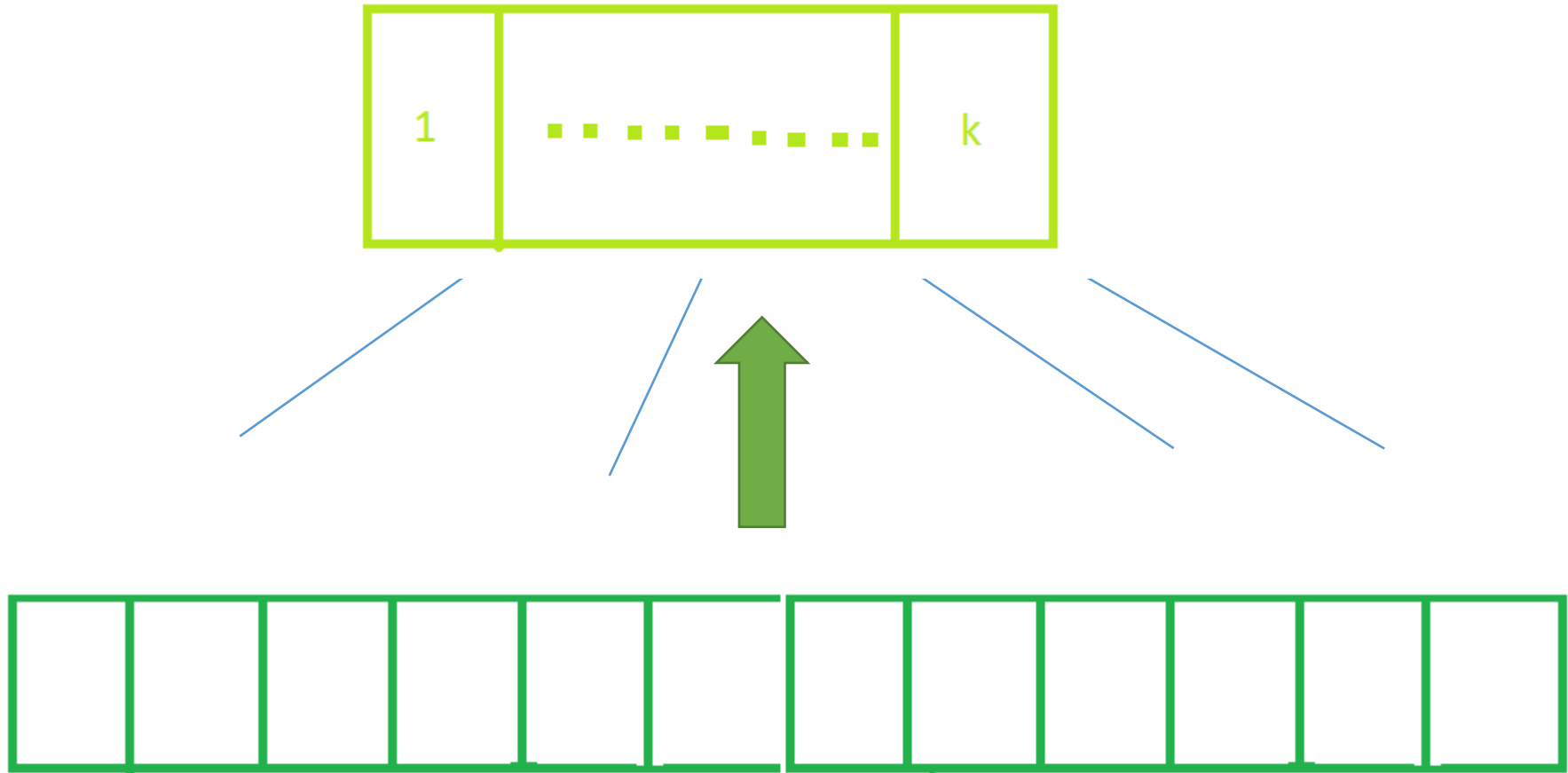


Sentença

Convolução 1D



k-Max Pooling



k-Max Pooling

- k características mais ativas
- Preserva a ordem
- Insensitiva para posições
- Pode detectar múltiplas ocorrências

Dynamic k-Max Pooling

$$k_l = \max(k_{top}, \lceil \frac{L-l}{L} s \rceil)$$

S = tamanho da sentença = 18

l = camada atual

L = Total de camadas = 3

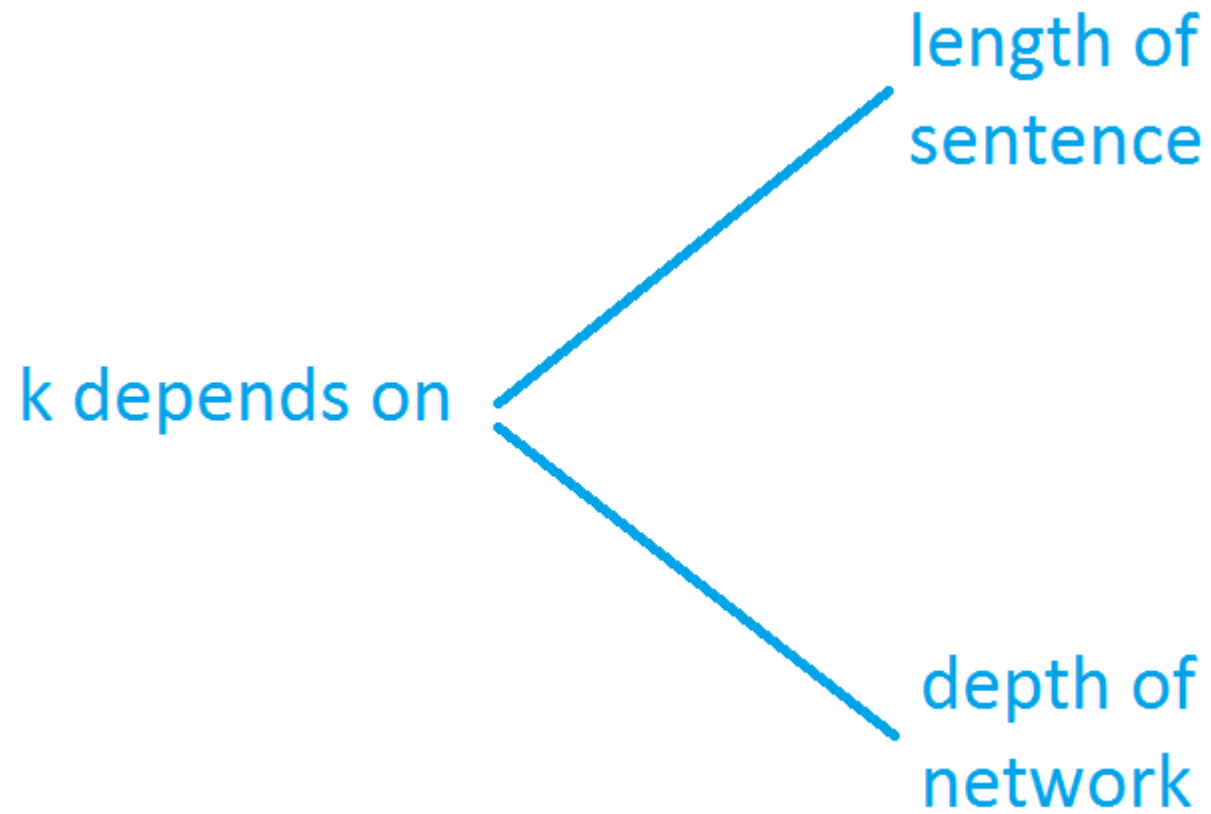
Ktop = fixo = 3

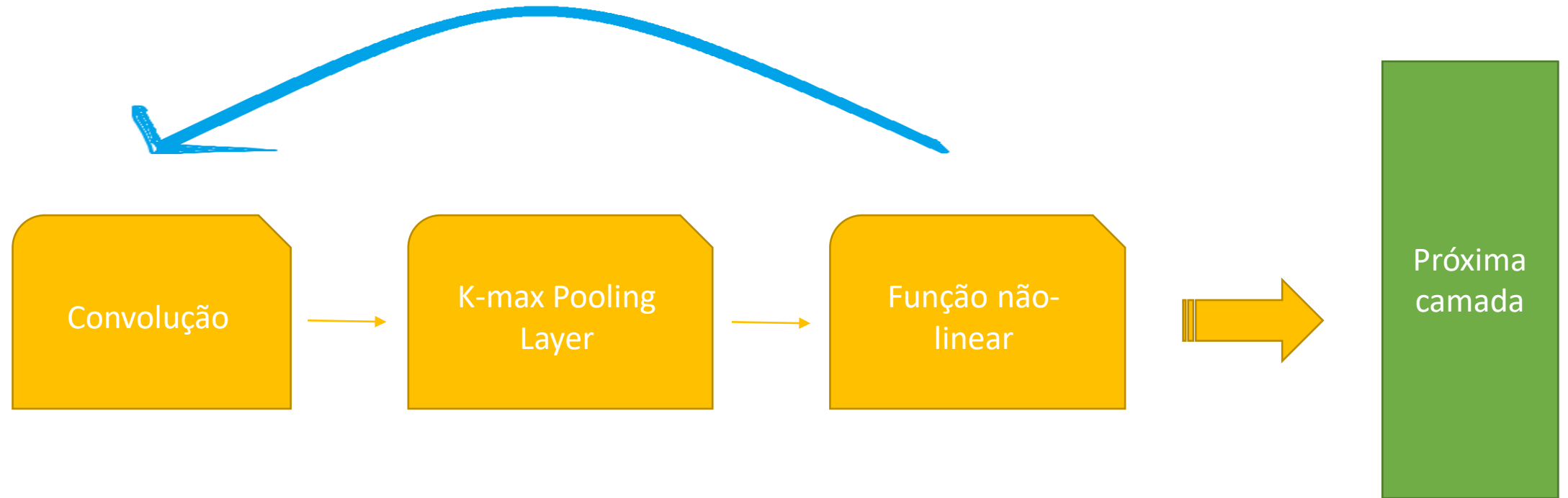
K1 = 12

K2 = 6

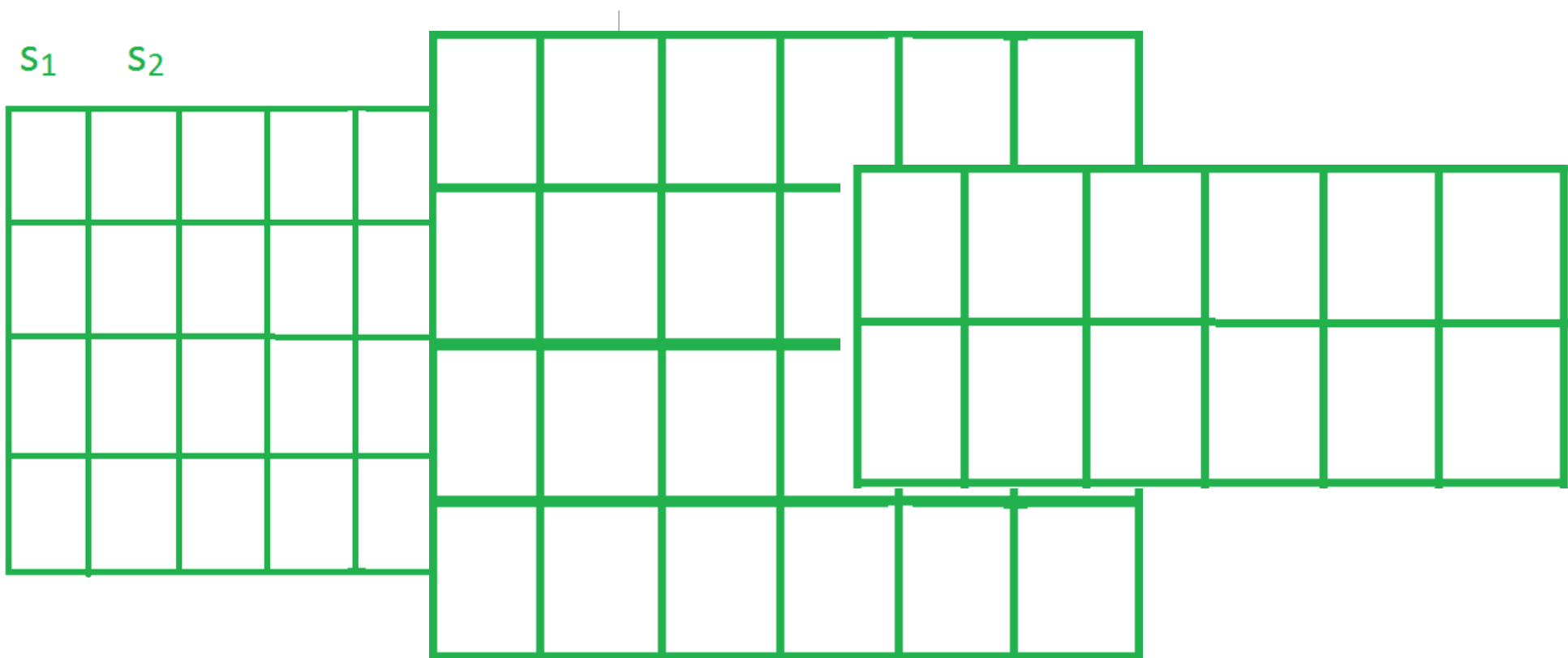
K3 = 3

K-Max Pooling





- Importante: linhas diferentes são independentes



Twitter Sentiment Prediction with Distant Supervision

- Data set gigante de tweets
- Tweet é rotulado como positive ou negative

Classifier	Accuracy (%)
SVM	81.6
BiNB	82.7
MaxEnt	83.0
Max-TDNN	78.8
NBoW	80.9
DCNN	87.4

Sentiment Prediction in Movie Reviews

- Predizer sentiment de comentário de filme - Stanford Sentiment Treebank
- Saída é binária para o experiment 1 (“positive”, “negative”) e de multipla classes no 2 “negative, somewhat negative, neutral, somewhat positive, positive”

TRAINING	8544
DEVELOPMENT	1101
TEST	2210

TRAINING	6920
DEVELOPMENT	872
TEST	1821

Classifier	Fine-grained (%)	Binary (%)
NB	41.0	81.8
BiNB	41.9	83.1
SVM	40.7	79.4
RECNTN	45.7	85.4
MAX-TDNN	37.4	77.1
NBoW	42.4	80.5
DCNN	48.5	86.8

Question Type Classification

- TREC question dataset
- Seis diferentes tipos de questões

TRAINING	5452
TEST	500

Classifier	Features	Acc. (%)
HIER	unigram, POS, head chunks NE, semantic relations	91.0
MAXENT	unigram, bigram, trigram POS, chunks, NE, supertags CCG parser, WordNet	92.6
MAXENT	unigram, bigram, trigram POS, wh-word, head word word shape, parser hypernyms, WordNet	93.6
SVM	unigram, POS, wh-word head word, parser hypernyms, WordNet 60 hand-coded rules	95.0
MAX-TDNN	unsupervised vectors	84.4
NBOW	unsupervised vectors	88.2
DCNN	unsupervised vectors	93.0

Problema exemplo

- **Classificar posts** provenientes de 20 diferentes grupos de notícias,
- **As categorias são bastante semanticamente distintas e, portanto, terão palavras bastante diferentes associadas a elas.**

comp.sys.ibm.pc.hardware
comp.graphics
comp.os.ms-windows.misc
comp.sys.mac.hardware
comp.windows.x
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey

Exemplo completo: <https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html>

Problema exemplo

- Converter as amostras de texto no conjunto de dados em seqüências de índices de palavras.
- Preparar uma "matriz de incorporação" que conterà no índice i o vetor de incorporação para a palavra índice i em nosso índice de palavras.
- Carregar a matriz “embeded” em uma camada da rede neural, configurada para congelar (seus pesos, os vetores de incorporação, não serão atualizados durante o treinamento).
- Construa uma rede neural convolutiva 1D, terminando em uma saída de softmax para 20 categorias.

Problema exemplo

- Calcular um índice de palavras-chave para embeddings conhecidos (GloVe, word2Vec, etc)

```
embeddings_index = {}  
f = open(os.path.join(GLOVE_DIR, 'glove.6B.100d.txt'))  
for line in f:  
    values = line.split()  
    word = values[0]  
    coefs = np.asarray(values[1:], dtype='float32')  
    embeddings_index[word] = coefs  
f.close()  
  
print('Found %s word vectors.' % len(embeddings_index))
```

```
embedding_matrix = np.zeros((len(word_index) + 1, EMBEDDING_DIM))  
for word, i in word_index.items():  
    embedding_vector = embeddings_index.get(word)  
    if embedding_vector is not None:  
        # words not found in embedding index will be all-zeros.  
        embedding_matrix[i] = embedding_vector
```


Problema exemplo

- Incorpore a camada na rede neural

```
from keras.layers import Embedding

embedding_layer = Embedding(len(word_index) + 1,
                             EMBEDDING_DIM,
                             weights=[embedding_matrix],
                             input_length=MAX_SEQUENCE_LENGTH,
                             trainable=False)
```

Problema exemplo

- Rede completa

```
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,), dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
x = Conv1D(128, 5, activation='relu')(embedded_sequences)
x = MaxPooling1D(5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = MaxPooling1D(5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = MaxPooling1D(35)(x) # global max pooling
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
preds = Dense(len(labels_index), activation='softmax')(x)

model = Model(sequence_input, preds)
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['acc'])

# happy learning!
model.fit(x_train, y_train, validation_data=(x_val, y_val),
        epochs=2, batch_size=128)
```

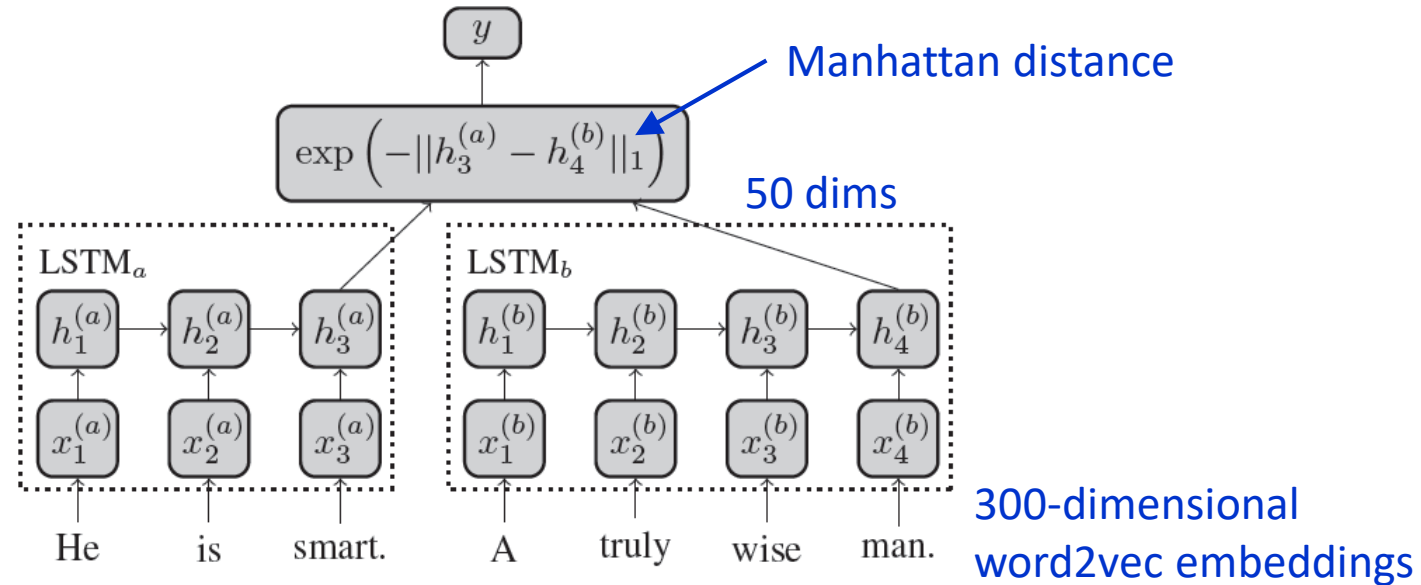
Problema
exemplo

Sem corpus

```
embedding_layer = Embedding(len(word_index) + 1,  
                             EMBEDDING_DIM,  
                             input_length=MAX_SEQUENCE_LENGTH)
```

Curiosidade...

This network is trained on pairs of sentences a, b with a similarity label y .



Parameters are shared between the two networks.

From "Siamese Recurrent Architectures for Learning Sentence Similarity"
Jonas Mueller, Aditya Thyagarajan, AAAI-2016

The network is trained on Semeval similar sentence pairs, expanded by substituting for random words using WordNet (a dataset of synonyms). Results:

Method	r	ρ	MSE
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	0.8822	0.8345	0.2286

From “Siamese Recurrent Architectures for Learning Sentence Similarity”
Jonas Mueller, Aditya Thyagarajan, AACL-2016

The network is trained on Semeval similar sentence pairs, expanded by substituting for random words using WordNet (a dataset of synonyms). Results:

Method	r	ρ	MSE
Illinois-LH (Lai and Hockenmaier 2014)	0.7993	0.7538	0.3692
UNAL-NLP (Jimenez et al. 2014)	0.8070	0.7489	0.3550
Meaning Factory (Bjerva et al. 2014)	0.8268	0.7721	0.3224
ECNU (Zhao, Zhu, and Lan 2014)	0.8414	–	–
Skip-thought+COCO (Kiros et al. 2015)	0.8655	0.7995	0.2561
Dependency Tree-LSTM (Tai, Socher, and Manning 2015)	0.8676	0.8083	0.2532
ConvNet (He, Gimpel, and Lin 2015)	0.8686	0.8047	0.2606
MaLSTM	0.8822	0.8345	0.2286

r = Pearson correlation, ρ = Spearman's rank correlation.

Tutoriais

- Socher Tutorial (2013)
 - Video: <http://techtalks.tv/events/312/573/>
 - Slides: <http://lxmls.it.pt/2014/socher-lxmls.pdf>
- Liang Tutorial (ICML 2015)
 - Video: http://videolectures.net/icml2015_liang_language_understanding/
 - Slides: <http://icml.cc/2015/tutorials/icml2015-nlu-tutorial.pdf>

Bibliotecas para NLP

- Natural Language Toolkit (NLTK): The complete toolkit for all NLP techniques.
- Pattern – A web mining module for the with tools for NLP and machine learning.
- TextBlob – Easy to use nlp tools API, built on top of NLTK and Pattern.
- spaCy – Industrial strength NLP with Python and Cython.
- Gensim – Topic Modelling for Humans
- Stanford Core NLP – NLP services and packages by Stanford NLP Group.

Bibliotecas para NLP

- Dica de leitura:
- <https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html>

Referências

- Tai, Socher, and Manning (2015) Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks
<https://arxiv.org/pdf/1503.00075v3.pdf>
- Mikolov et al. (2013) Efficient Estimation of Word Representations in Vector Space <https://arxiv.org/pdf/1301.3781v3.pdf>
- Socher et al. (2013) Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank
http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf
- Bowman et al. (2016) A Fast Unified Model for Parsing and Sentence Understanding <https://arxiv.org/pdf/1603.06021.pdf>

Referências (continued)

- Dos Santos and Zadrozny (2014) Learning Character-level Representations for Part-of-Speech Tagging
<http://jmlr.org/proceedings/papers/v32/santos14.pdf>
- Kim, Jernite, Sontag, and Rush (2015) Character-Aware Neural Language Models <https://arxiv.org/pdf/1508.06615.pdf>
- Cao and Rei (2016) A Joint Model for Word Embedding and Word Morphology <https://aclweb.org/anthology/W/W16/W16-1603.pdf>
- Irsoy and Cardie (2014) Deep recursive neural networks for compositionality in language
<https://www.cs.cornell.edu/~oirsoy/files/nips14drsv.pdf>

O curso está
terminando...



Mas ainda teremos:

- Reinforcement
- GAN's

