

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
Instituto de Informática  
Departamento de Informática Aplicada

## Aula 8: Representação de textos com Word Embeddings Fixas

Prof. Dennis Giovani Balreira



INF01221 - Tópicos Especiais em Computação XXXVI:  
Processamento de Linguagem Natural



# Conteúdo

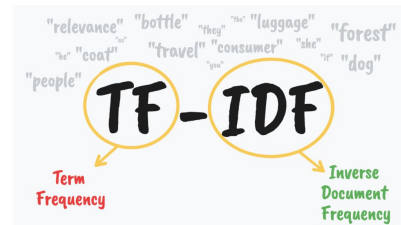
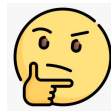
- Representação de textos com word embeddings fixas:
  - Hipótese distribucional
  - Semântica vetorial
  - One-hot vector
  - Word embeddings fixas
  - Word2Vec
  - Propriedades
  - Limitações

# Onde estamos em PLN?

- Algoritmos tradicionais
  - Predominantes entre o final dos anos 1990 até ~2016
  - BoW features + Aprendizado de Máquina
- Embeddings fixas + Deep Learning
  - Predominates de ~2014 até ~2019
  - Word2vec, Glove, FastText + LSTM
- Embeddings contextuais + Large Language Models
  - Estado da arte em diversas tarefas
  - BERT, GPT, etc.

# Introdução

- Vimos duas formas de representação de texto tradicionais:
  - Bag of Words
  - TF-IDF
- Ambas representam documentos como vetores
  - Vetores podem ser usados como entrada para algoritmos de aprendizado de máquina
- Quais problemas tem das abordagens acima?



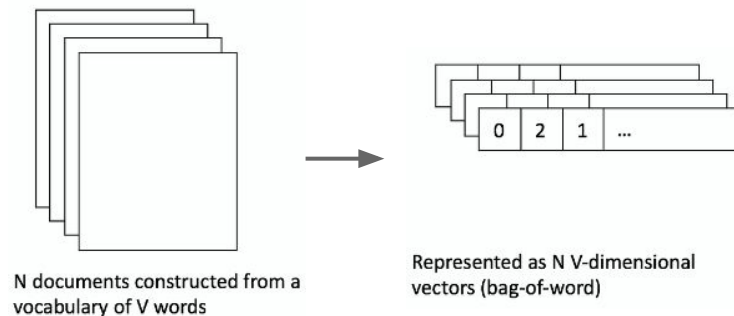
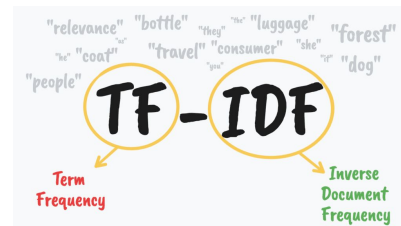
Document D1	<i>The child makes the dog happy</i> the: 2, dog: 1, makes: 1, child: 1, happy: 1
Document D2	<i>The dog makes the child happy</i> the: 2, child: 1, makes: 1, dog: 1, happy: 1



	child	dog	happy	makes	the	BoW Vector representations
D1	1	1	1	1	2	[1,1,1,1,2]
D2	1	1	1	1	2	[1,1,1,1,2]

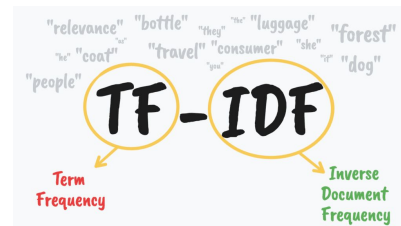
# Introdução

- Vimos duas formas de representação de texto tradicionais:
  - Bag of Words
  - TF-IDF
- Ambas representam **documentos como vetores**
  - Vetores podem ser usados como entrada para algoritmos de aprendizado de máquina
- Quais problemas tem das abordagens acima?
  - Vetores muito **longos e esparsos**



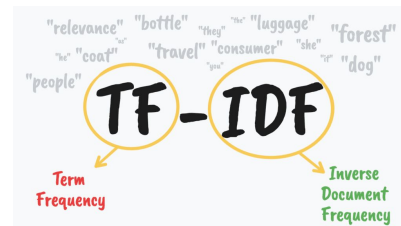
# Introdução

- Vimos duas formas de representação de texto tradicionais:
  - Bag of Words
  - TF-IDF
- Ambas representam **documentos como vetores**
  - Vetores podem ser usados como entrada para algoritmos de aprendizado de máquina
- Quais problemas tem das abordagens acima?
  - Vetores muito **longos e esparsos**
  - Não captam correlação (**similaridade**) entre palavras
    - Ex: Carro vs. Automóvel



# Introdução

- Vimos duas formas de representação de texto tradicionais:
  - Bag of Words
  - TF-IDF
- Ambas representam **documentos como vetores**
  - Vetores podem ser usados como entrada para algoritmos de aprendizado de máquina
- Quais problemas tem das abordagens acima?
  - Vetores muito **longos e esparsos**
  - Não captam correlação (**similaridade**) entre palavras
  - Não lida com **polissemia** (palavra com múltiplos sentidos)
    - Ex: Manga (da camisa X fruta)



# Hipótese distribucional

- Teoria linguística que afirma que o significado de uma palavra pode ser inferido a partir dos contextos em que ela aparece
- Ou seja:  
“Palavras que ocorrem em contextos similares tendem a ter significados similares” [Harris, 1954]



Harris, Z. S. (1954). Distributional structure. *Word*, 10, 146–162. Reprinted in J. Fodor and J. Katz, *The Structure of Language*, Prentice Hall, 1964 and in Z. S. Harris, *Papers in Structural and Transformational Linguistics*, Reidel, 1970, 775–794.



# Hipótese distribucional

- Teoria linguística que afirma que o significado de uma palavra pode ser inferido a partir dos contextos em que ela aparece
- Ou seja:

“Palavras que ocorrem em contextos similares tendem a ter significados similares” [Harris, 1954]
- Ideia: basear-se nesta ideia para aprender vetores de palavras levando em conta seu contexto
  - Ex:
    - Estou sem dinheiro. Fui ao banco.



Harris, Z. S. (1954). Distributional structure. *Word*, 10, 146–162. Reprinted in J. Fodor and J. Katz, *The Structure of Language*, Prentice Hall, 1964 and in Z. S. Harris, *Papers in Structural and Transformational Linguistics*, Reidel, 1970, 775–794.

# Hipótese distribucional

- Teoria linguística que afirma que o significado de uma palavra pode ser inferido a partir dos contextos em que ela aparece
- Ou seja:

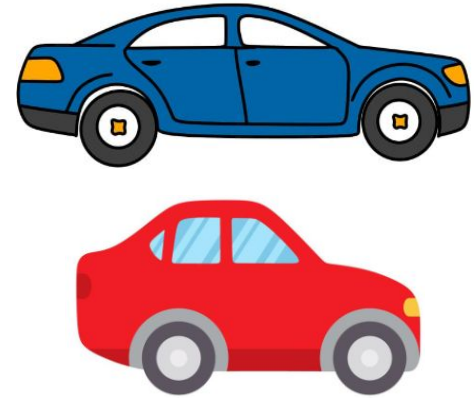
“Palavras que ocorrem em contextos similares tendem a ter significados similares” [Harris, 1954]
- Ideia: basear-se nesta ideia para aprender vetores de palavras levando em conta seu contexto
  - Ex:
    - Estou sem dinheiro. Fui ao banco.
    - Estou cansado. Vou ao banco descansar.



Harris, Z. S. (1954). Distributional structure. *Word*, 10, 146–162. Reprinted in J. Fodor and J. Katz, *The Structure of Language*, Prentice Hall, 1964 and in Z. S. Harris, *Papers in Structural and Transformational Linguistics*, Reidel, 1970, 775–794.

# Relações entre palavras

- **Sinonímia:** palavras com o mesmo ou quase o mesmo significado
  - Ex: carro e automóvel



# Relações entre palavras

- Sinonímia: palavras com o mesmo ou quase o mesmo significado
  - Ex: carro e automóvel
- **Similaridade:** palavras que têm significados relacionados ou próximos
  - Ex: cachorro e lobo



# Relações entre palavras

- Sinonímia: palavras com o mesmo ou quase o mesmo significado
  - Ex: carro e automóvel
- Similaridade: palavras que têm significados relacionados ou próximos
  - Ex: cachorro e lobo
- **Associação**: palavras que tendem a aparecer juntas devido a uma conexão de contexto ou conceito
  - Ex: café e xícara

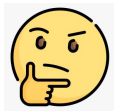


# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?



Wittgenstein, L. (1953). *Philosophical Investigations*. Blackwell.

# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?
  - Uma palavra **só possui significado “completo”** quando vista pelo seu **contexto** (palavras/frases próximas)

Wittgenstein, L. (1953). *Philosophical Investigations*. Blackwell.

# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?
  - Uma palavra **só possui significado “completo”** quando vista pelo seu **contexto** (palavras/frases próximas)

O que é “tesgüino”?



Wittgenstein, L. (1953). Philosophical Investigations. Blackwell.



# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

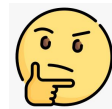
“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?
  - Uma palavra **só possui significado “completo”** quando vista pelo seu **contexto** (palavras/frases próximas)

Wittgenstein, L. (1953). Philosophical Investigations. Blackwell.

## O que é “tesgüino”?

- Tem uma garrafa de tezgüino na mesa
- Todos gostam de tezgüino
- Tezgüino me deixou bêbada
- Tezgüino é feito de milho



# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

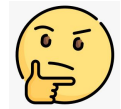
“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?
  - Uma palavra **só possui significado “completo”** quando vista pelo seu **contexto** (palavras/frases próximas)

Wittgenstein, L. (1953). Philosophical Investigations. Blackwell.

## O que é “tezgüino”?

- Tem uma garrafa de tezgüino na mesa
- Todos gostam de tezgüino
- Tezgüino me deixou bêbada
- Tezgüino é feito de milho



# Semântica vetorial

- Um **modelo perfeito** que consiga lidar com todos os aspectos do **significado** das palavras é **inatingível**

“O significado de uma palavra é o seu uso na linguagem” [Wittgenstein, 1953]

- O que isso significa?
  - Uma palavra **só possui significado “completo”** quando vista pelo seu **contexto** (palavras/frases próximas)

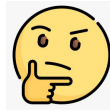
Wittgenstein, L. (1953). *Philosophical Investigations*. Blackwell.

## O que é “tesgüino”?



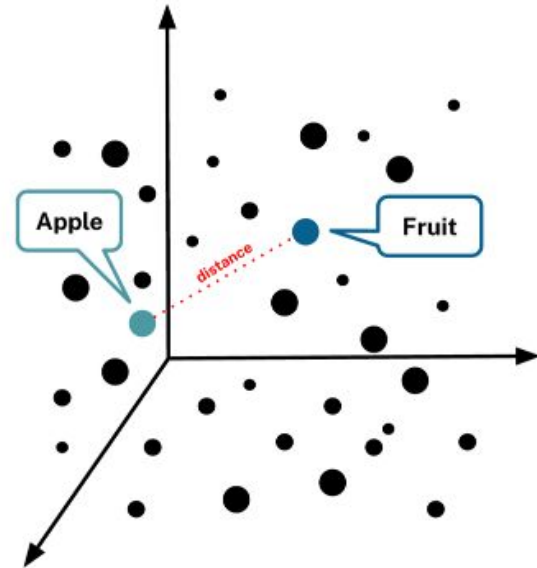
# Semântica vetorial

- Sabemos que palavras podem ser vetores!
- Mas como criar vetores “semânticos”?



# Semântica vetorial

- Sabemos que palavras podem ser vetores!
- Mas como criar vetores “semânticos”?
  - Vetores BoW e TF-IDF são:
    - Longos (comprimento de  $|V|$  muito alto)
    - Esparsos (maioria dos elementos é zero)
  - Alternativa usando vetores:
    - Curtos (poucas dimensões)
    - Densos (maioria não é zero)
- A primeira utilização foi na técnica **Indexação Semântica Latente** [Deerwester et al. 1990]



Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. JASIS, 41(6), 391–407.

# Indexação Semântica Latente (LSI)

- Técnica usada em **recuperação de informações** e **análise de texto** que busca **identificar relações ocultas entre palavras** em grandes conjuntos de documentos
- Suposição: palavras usadas em **contextos** semelhantes possuem **significados** parecidos
- Baseada no conceito matemático de **decomposição de valores singulares** (SVD):
  - 1. Criação de matriz termo-documento (BoW ou TF-IDF)
  - 2. Aplicação de SVD, gerando matrizes menores
  - 3. Projeção em um espaço de menor dimensão

The diagram illustrates the SVD decomposition of a matrix  $M$  into three matrices:  $U$ ,  $\Sigma$ , and  $V^*$ .

**Matrix  $M$  ( $m \times n$ ):** A 4x4 grid of gray squares.

**Matrix  $U$  ( $m \times m$ ):** A 4x4 grid with columns colored green, blue, green, and green.

**Matrix  $\Sigma$  ( $m \times n$ ):** A 4x4 grid with diagonal elements colored orange, yellow, and yellow, and all other elements white.

**Matrix  $V^*$  ( $n \times n$ ):** A 4x4 grid with rows colored purple, purple, purple, and pink.

The equation is shown as:

$$M = U \Sigma V^*$$

**Matrix  $U$  ( $m \times m$ ):** A 4x4 grid with columns colored green, blue, green, and green.

**Matrix  $U^*$  ( $m \times m$ ):** A 4x4 grid with rows colored green, green, blue, and green.

**Matrix  $\Sigma$  ( $m \times n$ ):** A 4x4 grid with diagonal elements colored orange, yellow, and yellow, and all other elements white.

The equation is shown as:

$$U U^* = I_m$$

**Matrix  $V$  ( $n \times n$ ):** A 4x4 grid with columns colored purple, purple, purple, and pink.

**Matrix  $V^*$  ( $n \times n$ ):** A 4x4 grid with rows colored purple, purple, purple, and pink.

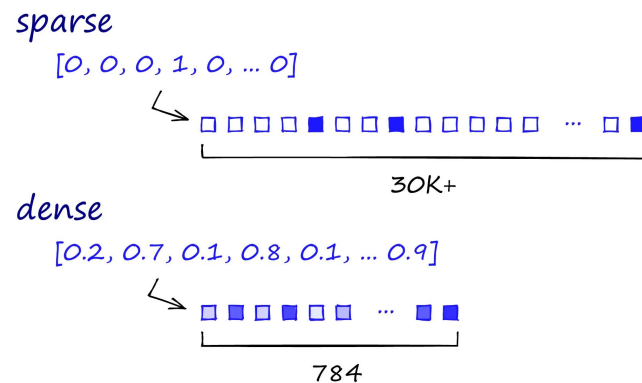
**Matrix  $\Sigma$  ( $m \times n$ ):** A 4x4 grid with diagonal elements colored orange, yellow, and yellow, and all other elements white.

The equation is shown as:

$$V V^* = I_n$$

# Por que usar vetores densos (vs. esparsos)?

- Vetores **densos** (e curtos) são mais fáceis de usar como **features** em algoritmos de aprendizado de máquina!
  - **Menos pesos** para ajustar
  - Em geral, possuem **melhor poder de generalização**
  - Conseguem capturar **sinonímia** (carro e automóvel)
  - Na prática, funcionam melhor!
- Estes vetores focam em **codificar palavras** buscando capturar as suas **semânticas**!
  - Diferem de **BoW** ou **TF-IDF**:
    - Focam em codificar documentos
    - Ignoram a ordem das palavras



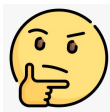
# Outra representação (palavras): one-hot vector

- Técnica usada para representar palavras individuais em um vetor binário
- Cada palavra é representada por um vetor cujo tamanho é igual ao número de palavras no vocabulário
  - Cada dimensão do vetor representa uma palavra
  - Valor “1” indica presença da palavra e “0” ausência
  - Apenas um valor “1” por vetor

Rome = [1, 0, 0, 0, 0, 0, ..., 0]  
Paris = [0, 1, 0, 0, 0, 0, ..., 0]  
Italy = [0, 0, 1, 0, 0, 0, ..., 0]  
France = [0, 0, 0, 1, 0, 0, ..., 0]

Diagram illustrating one-hot vectors for words. Arrows point from the word labels above to the corresponding elements in the vectors: Rome points to the first element (1), Paris points to the second element (1), and word V points to the last element (0).

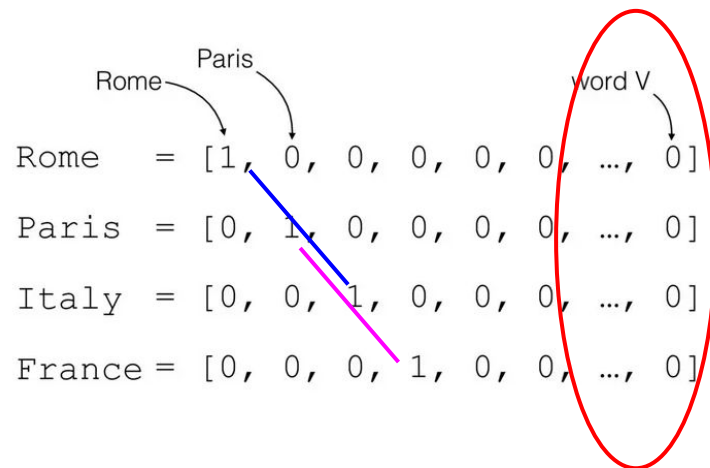
- Problemas?





# Outra representação (palavras): one-hot vector

- Técnica usada para representar **palavras individuais** em um **vetor binário**
- Cada **palavra** é representada por um vetor cujo tamanho é igual ao **número de palavras no vocabulário**
  - Cada dimensão do vetor representa uma palavra
  - Valor “1” indica presença da palavra e “0” ausência
  - Apenas um valor “1” por vetor
- Problemas?
  - Alta dimensionalidade (vocabulário)
  - Falta de semântica entre palavras



## Outra representação (palavras): one-hot vector

- Técnica usada para representar **palavras individuais** em um **vetor binário**
- Cada **palavra** é representada por um vetor cujo tamanho é igual ao **número de palavras no vocabulário**
  - Cada dimensão do vetor representa uma palavra
  - Valor “1” indica presença da palavra e “0” ausência
  - Apenas um valor “1” por vetor
- Problemas?
  - **Alta dimensionalidade (vocabulário)**
  - **Falta de semântica entre palavras**
- Solução ideal: **palavras relacionadas** deveriam ter **score** de **similaridade maior**!

laranja	[1, 0, 0, 0]
morango	[0, 1, 0, 0]
cidade	[0, 0, 1, 0]
Londres	[0, 0, 0, 1]

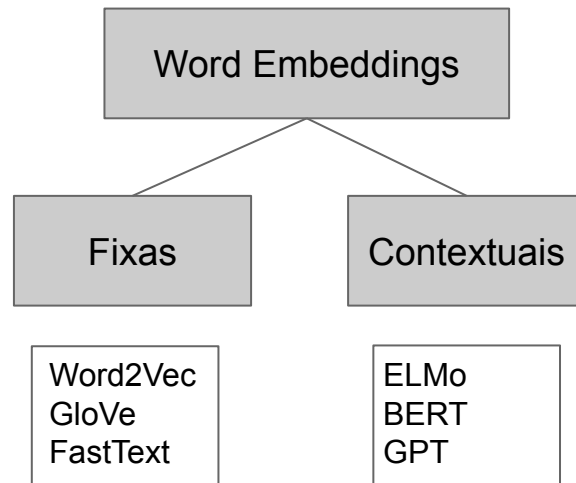


>>

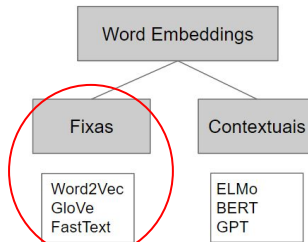


# Word Embeddings

- Representações numéricas densas de palavras em um espaço vetorial de dimensões reduzidas
- São geradas por modelos de **aprendizado de máquina**
  - Palavras **semanticamente semelhantes** têm **vetores próximos** no espaço vetorial
  - Permite capturar o **significado** e as **relações semânticas** entre as palavras
- **Word Embeddings** podem ser classificadas em dois tipos:
  - Fixas (Word2Vec, GloVe) [~2013]
  - Contextuais (BERT, GPT) [~2018]

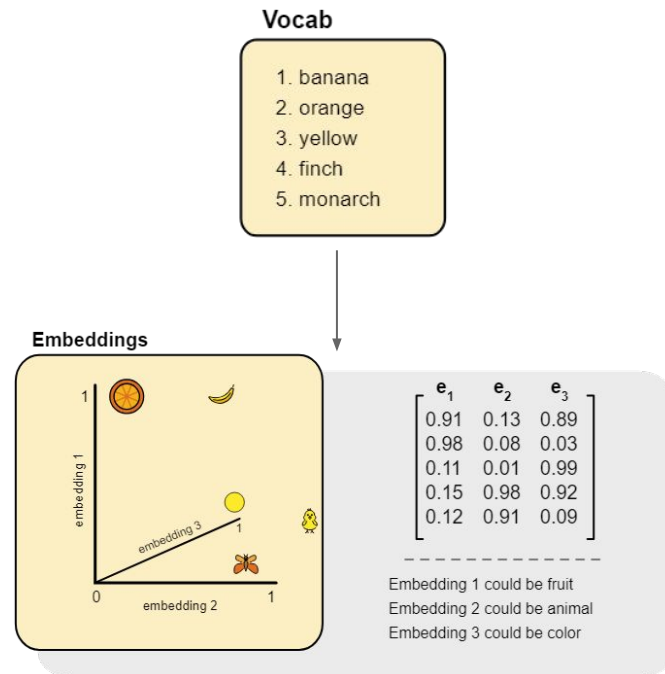


# Word Embeddings

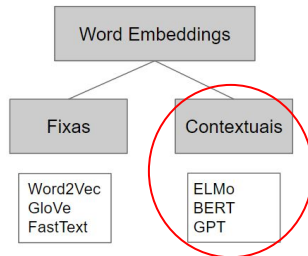


- Word Embeddings Fixas

- Cada palavra tem uma **única representação vetorial, independentemente do contexto** em que é usada
- Uma vez treinadas, fornecem **vetores estáticos**, um para cada palavra no vocabulário
- Capturam **similaridades semânticas “gerais” (estáticas)** de cada palavra **com base nos textos treinados**
- Modelos principais:
  - **Word2Vec**
  - **GloVe (Global Vectors for Word Representation)**
  - **FastText**

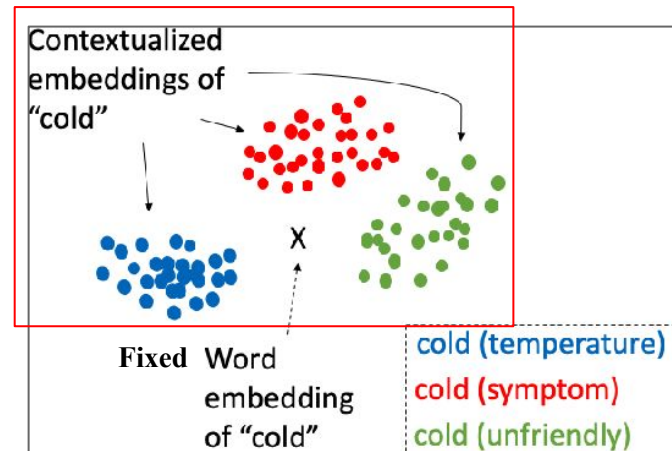


# Word Embeddings



- Word Embeddings Contextuais

- Cada “ocorrência” de palavra possui uma **representação vetorial única**
- O **vetor da palavra** é ajustado baseado nas **palavras próximas**, levando em conta o **contexto** (sentença ou parágrafo) onde a palavra ocorre
- Capturam similaridades “**customizadas**” (**dinâmicas**) de cada palavra com base nos textos treinados
- Modelos principais
  - ELMo (*Embeddings from Language Models*)
  - BERT (*Bidirectional Encoder Representations from Transformers*)
  - GPT (*Generative Pre-trained Transformer*)



# Word Embeddings: fixas vs. contextuais

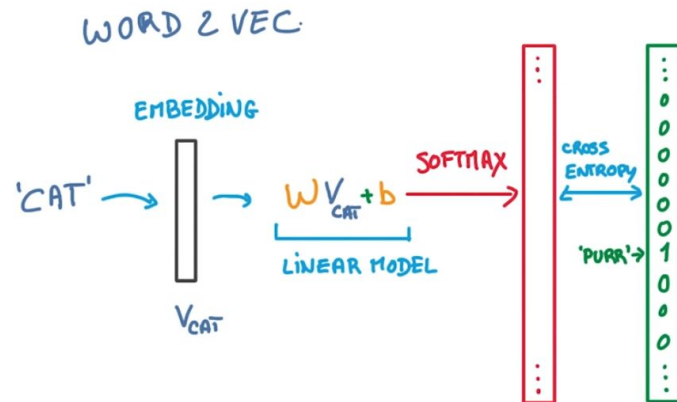
Aspecto	Word Embeddings Fixas	Word Embeddings Contextuais
Representação por Palavra	Uma única representação por palavra.	Diferentes representações dependendo do contexto.
Capacidade de Capturar Contexto	Não captura o contexto de uso da palavra.	Captura o contexto específico da palavra.
Exemplos de Modelos	Word2Vec, GloVe, FastText	BERT, ELMo, GPT
Captura de Polissemia	Não, usa o mesmo vetor para todos os significados de uma palavra.	Sim, gera diferentes vetores para diferentes significados da palavra.
Complexidade Computacional	Baixa, rápida para uso após o treinamento.	Alta, exige mais recursos computacionais devido à complexidade.
Aplicações	Tarefas simples de PLN que não exigem grande contexto.	Tarefas que dependem do contexto, como análise de sentimento e tradução automática.

# Word Embeddings Fixas: Word2Vec

- Algoritmo criado para gerar **representações vetoriais densas** de palavras (*word embeddings*)
- Desenvolvido por Tomas Mikolov et al. (2013) (Google)
- Método **muito popular!**
  - Rápido de treinar
  - Código aberto
- Ideia geral: **prever** em vez de **contar**
  - Tenta prever palavras com base em suas **palavras vizinhas**, capturando as **relações semânticas**



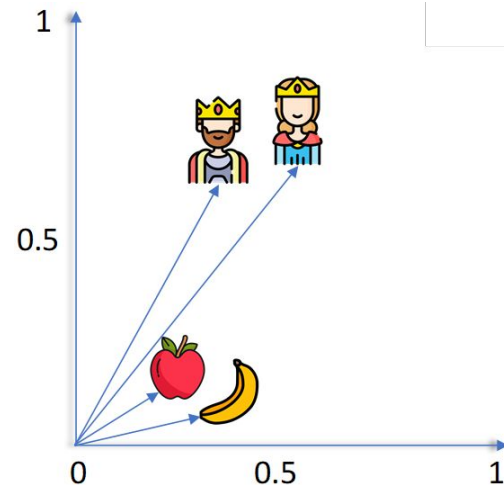
Tomas Mikolov







- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a). Distributed representations of words and phrases and their compositionality. In NIPS 13, 3111–3119. (mais de 29 mil citações!)
- Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In NAACL HLT 2013, 746–751.

# Word Embeddings Fixas: Word2Vec

- **Características** dos embeddings fixos:
  - **Vetor denso**: cada palavra é representada por um vetor denso, com valores contínuos que capturam as relações semânticas entre palavras
  - **Relações semânticas**: Palavras que aparecem frequentemente no mesmo contexto terão vetores próximos
  - **Operações semânticas**: Word2Vec permite operações aritméticas com palavras (vetores)
    - Um exemplo famoso é:  
"rei" - "homem" + "mulher"  $\approx$  "rainha"
  - Captura relações como **gênero**, **singular/plural**, ou **hierarquias semânticas**



	0.25	0.16		0.33	0.10
	0.29	0.68		0.51	0.71



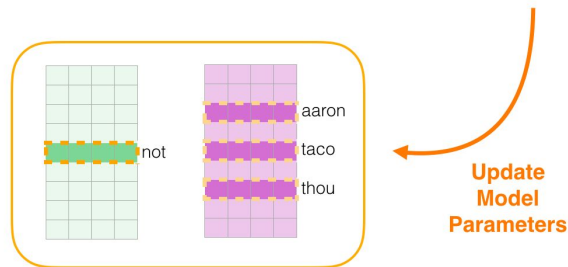
# Word Embeddings Fixas: Word2Vec

- Ideia: **em vez de contar** com que frequência cada palavra  $w$  ocorre perto de “morango”
  - Treinar um **classificador** em uma tarefa de **predição binária**:
    - É provável que  $w$  apareça perto de “morango”?
  - Não é a tarefa de interesse**, mas os **pesos do classificador** podem servir como **embeddings**

... limão, uma colher de geleia de morango uma pitada sal ...  
c1 c2 alvo c3 c4



input word	output word	target	input • output	sigmoid()	Error
not	thou	1	0.2	0.55	0.45
not	aaron	0	-1.11	0.25	-0.25
not	taco	0	0.74	0.68	-0.68



# Word Embeddings Fixas: Word2Vec

- Ideia: **em vez de contar** com que frequência cada palavra  $w$  ocorre perto de “morango”
  - Treinar um **classificador** em uma tarefa de **predição binária**:
    - É provável que  $w$  apareça perto de “morango”?
  - **Não é a tarefa de interesse**, mas os **pesos do classificador** podem servir como **embeddings**
- Texto é usado como **dados de treinamento supervisionado**
  - Exemplo:
    - Palavras **próximas de “morango”** são **positivas**
    - **Outras palavras** são **negativas**
- **Não requer dados manualmente anotados!**
  - *Ground truth* vem **naturalmente** de **palavras próximas**



# Word Embeddings Fixas: Word2Vec

- **Treinamento:**
  - As relações são aprendidas utilizando uma **rede neural superficial** (não profunda), treinada em **grandes quantidades de dados de texto**
  - A saída final é um **vetor denso** de uma dimensão escolhida
    - Por exemplo: 100, 300
  - Cada palavra possui uma **representação própria** e contínua (vetor)

# Word Embeddings Fixas: Word2Vec

- Existem dois principais métodos usados pelo **Word2Vec**:

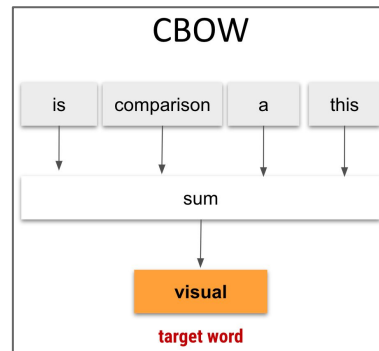
- CBOW (Continuous Bag of Words):**

- Prevê a palavra-alvo com base nas palavras de contexto
    - Exemplo:

Contexto: "cachorro está no" → Predição: "parque"

- Skip-Gram:

This is a visual comparison



# Word Embeddings Fixas: Word2Vec

- Existem dois principais métodos usados pelo **Word2Vec**:
  - CBOW (Continuous Bag of Words):**

- Prevê a palavra-alvo com base nas palavras de contexto
- Exemplo:

Contexto: "cachorro está no" → Predição: "parque"

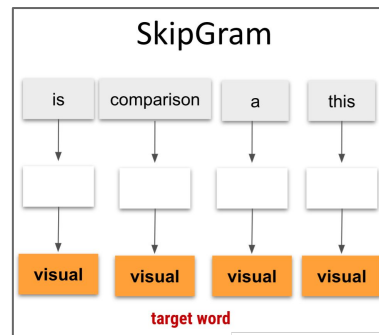
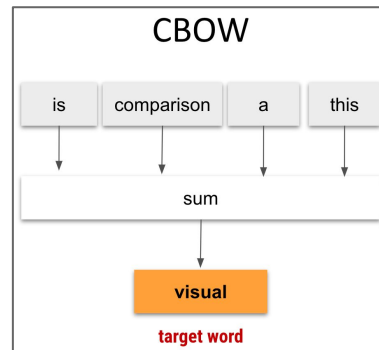
- Skip-Gram:**

- Prevê as palavras de contexto com base na palavra-alvo
- Exemplo:

Palavra-alvo: "cachorro" → Predição: "está", "no", "parque"

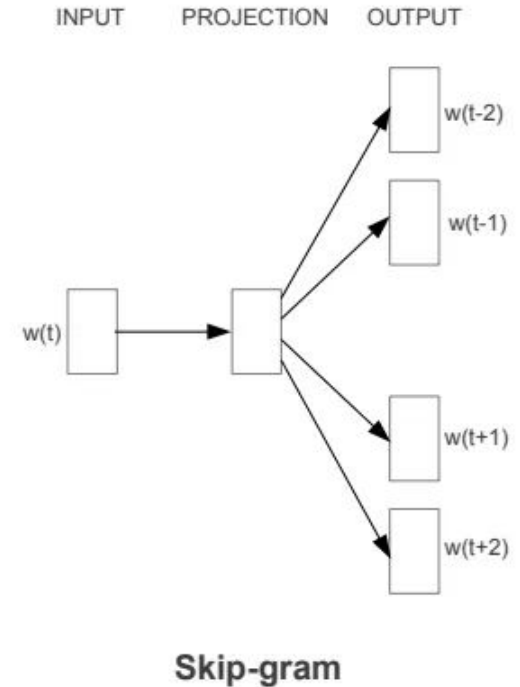
- **Vamos ver mais detalhes do Skip-Gram**

This is a visual comparison



# Word Embeddings Fixas: Word2Vec

- Algoritmo Skip-gram:
  - 1. Trate a **palavra alvo** e seu **contexto** (i.e., palavras vizinhas) como **exemplos positivos**
  - 2. Pegue uma **amostra aleatória** de **outras palavras do vocabulário** para servir como **exemplos negativos**
  - 3. Use uma **regressão logística** para **treinar um classificador** a distinguir entre os dois casos
  - 4. Use os **pesos da regressão** como **embeddings**



# Word Embeddings Fixas: Word2Vec

- Exemplo (Skip-gram):

- Sentença de **treinamento** (supondo janela de contexto de 2 palavras):

... limão, uma colher de geleia de morango uma pitada sal ...  
c1 c2 alvo(t) c3 c4

- Dada uma tupla  $(t, c) = \text{target}, \text{context}$



- Retorne a **probabilidade** que  $c$  seja uma palavra do contexto de  $t$

$$P(+ | t, c)$$

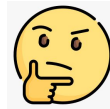
$$P(- | t, c) = 1 - P(+ | t, c)$$

# Word Embeddings Fixas: Word2Vec

- Sabendo que palavras parecidas tendem a aparecer próximas de suas similares
  - Podemos modelar suas similaridades usando o produto escalar

$$\text{Similaridade}(t,c) \propto t \cdot c$$

- Problema:
  - Produto escalar não é uma probabilidade (nem o cosseno)
- Qual função permite transformar “valores” em probabilidades?





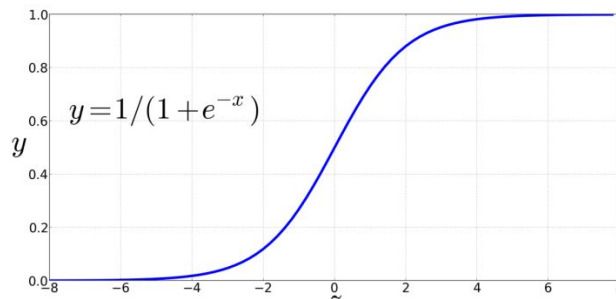
# Word Embeddings Fixas: Word2Vec

- Sabendo que palavras parecidas tendem a aparecer próximas de suas similares
  - Podemos modelar suas similaridades usando o produto escalar

$$\text{Similaridade}(t,c) \propto t \cdot c$$

- Problema:
  - Produto escalar não é uma probabilidade (nem o cosseno)
- Qual função permite transformar “valores” em probabilidades?
  - Sigmóide (usada em regressão logística) transforma valores em números entre 0 e 1

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



# Word Embeddings Fixas: Word2Vec

- Transformando o produto escalar em probabilidade:

$$\text{Similaridade}(t, c) \propto t \cdot c$$

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$

$$P(-|t, c) = 1 - P(+|t, c) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

# Word Embeddings Fixas: Word2Vec

- Estendendo a ideia para todas as **palavras de contexto**:
  - Para cada palavra **target**  $t$ , queremos maximizar a **probabilidade** (~likelihood) de observar palavras de contexto  $c_1, c_2, \dots, c_k$ , onde  $k$  é a **janela de contexto** (assumindo que as palavras são independentes)
  - O **produto final** representa a **probabilidade geral** de observar **todas as palavras de contexto** dentro da janela de uma **palavra alvo**

geleia de morango uma pitada  
c1 c2 alvo(t) c3 c4

$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}}$$



$$P(+|t, c_{1:k}) = \prod_{i=1}^k \frac{1}{1 + e^{-t \cdot c_i}}$$

Produto é muito caro computacionalmente!

$$P(-|t, c) = 1 - P(+|t, c) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

# Word Embeddings Fixas: Word2Vec

- Estendendo a ideia para todas as **palavras de contexto**:
  - Para cada palavra **target**  $t$ , queremos maximizar a **probabilidade** (~likelihood) de observar palavras de contexto  $c_1, c_2, \dots, c_k$ , onde  $k$  é a **janela de contexto** (assumindo que são as palavras são independentes)
  - O **produto final** representa a **probabilidade geral** de observar **todas as palavras de contexto** dentro da janela de uma **palavra alvo**

geleia de morango uma pitada  
c1 c2 alvo(t) c3 c4

$$P(+|t, c) = \frac{1}{1 + e^{-t \cdot c}} \quad \nearrow \quad P(+|t, c_{1:k}) = \prod_{i=1}^k \frac{1}{1 + e^{-t \cdot c_i}}$$

Melhor usar soma por meio dos logs

$$P(-|t, c) = 1 - P(+|t, c) = \frac{e^{-t \cdot c}}{1 + e^{-t \cdot c}}$$

$$\log P(+|t, c_{1:k}) = \sum_{i=1}^k \log \frac{1}{1 + e^{-t \cdot c_i}}$$

# Word Embeddings Fixas: Word2Vec

Algoritmo Skip-gram:

## 0. Passos iniciais:

- Definição da quantidade de elementos do vetor (~300)
- Cada palavra terá um vetor correspondente (denso)
- Definir janela de contexto conforme número de palavras ck (*context*)

Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings

# Word Embeddings Fixas: Word2Vec

Algoritmo Skip-gram:

1. Tratar a palavra *target* e seu contexto como exemplos positivos:
  - Determinar exemplos positivos conforme janela de contexto definida

Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings

... limão, uma colher de geleia de morango uma pitada sal ...

c1 c2 alvo c3 c4

**Exemplos positivos +**

$t$	$c$
morango	geleia
morango	de
morango	uma
morango	pitada

# Word Embeddings Fixas: Word2Vec

Algoritmo Skip-gram:

## 2. Pegar amostras aleatórias para exemplos negativos

- Para cada exemplo positivo criação de k exemplos negativos
- Utilização de palavras de ruído (qualquer palavra que não seja t)
- Existem estratégias para escolher “boas” palavras negativas

... limão, uma colher de geleia de morango uma pitada sal ...  
c1 c2 alvo c3 c4

Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings

## Exemplos negativos - k=2

<u>t</u>	<u>c</u>	<u>t</u>	<u>c</u>
morango	sapato	morango	barco
morango	casa	morango	exército
morango	aonde	morango	olá
morango	teclado	morango	suíno

# Word Embeddings Fixas: Word2Vec

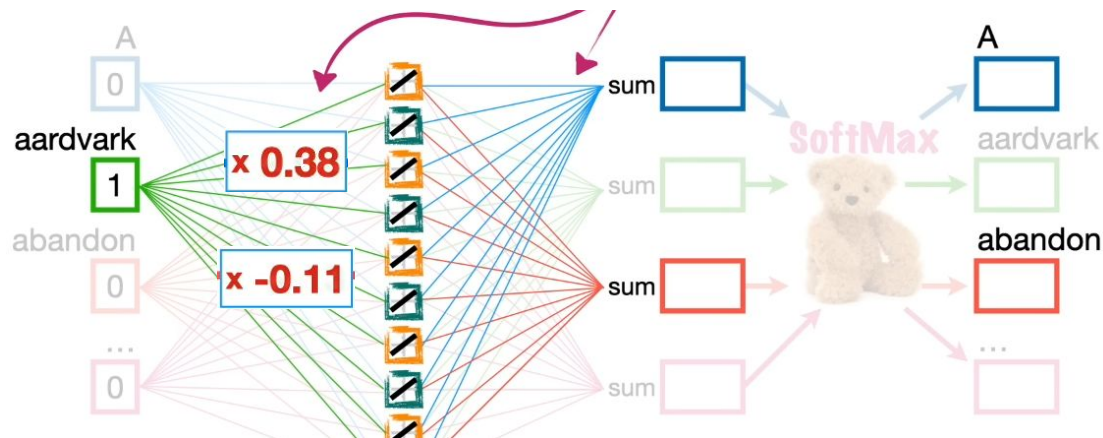
## Algoritmo Skip-gram:

### 3. Usar regressão logística para treinar o classificador

- Iniciar com vetores inicializados aleatoriamente
- Sobre todo o conjunto de treinamento, ajustar os vetores

## Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings



Word Embedding e Word2Vec, clearly explained!!! (Canal StatQuest with Josh Starmer). Link: <https://www.youtube.com/watch?v=viZrOnJciY0>



# Word Embeddings Fixas: Word2Vec

Algoritmo Skip-gram:

## 3. Usar regressão logística para treinar o classificador

- Iniciar com vetores inicializados **aleatoriamente**
- Sobre todo o **conjunto de treinamento**, ajustar os vetores
- Ajustar de forma a:
  - **Maximizar** a similaridade dos pares (t,c) (ou seja, target, context) [**positivos**]
  - **Minimizar** a similaridade dos pares (t,c) [**negativos**]
- Ou seja, queremos **maximizar a função objetivo** (*loss function*):

$$\sum_{(t,c) \in +} \log P(+|t, c) + \sum_{(t,c) \in -} \log P(-|t, c)$$

Soma dos logaritmos das probabilidades de que os pares (t,c) que aparecem juntos no corpus sejam previstos corretamente

Soma dos logaritmos das probabilidades de que os pares (t,c) que não aparecem juntos no corpus sejam previstos corretamente como ausentes

Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings

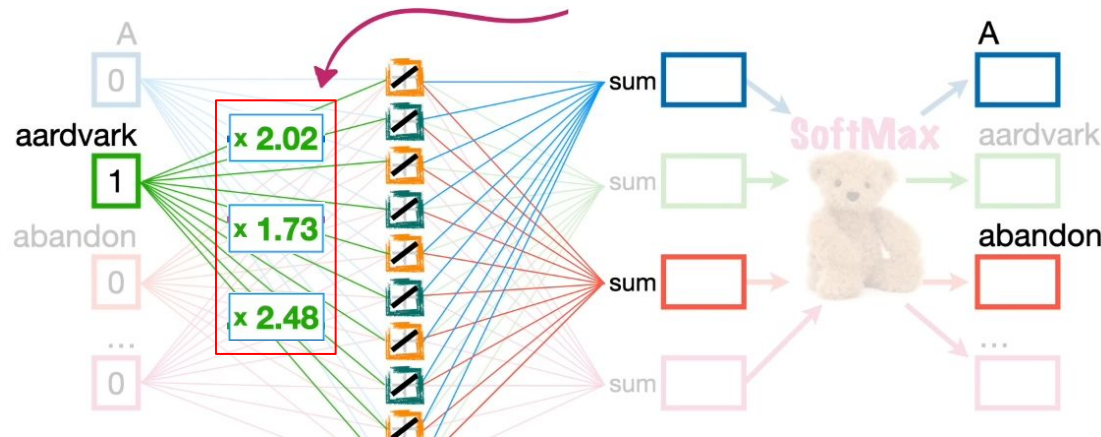
# Word Embeddings Fixas: Word2Vec

## Algoritmo Skip-gram:

4. Usar os pesos (otimizados) da regressão logística como embeddings

### Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings



(aardvark,a) [positivo]

(aardvark,abandon) [negativo]

Word Embedding e Word2Vec, clearly explained!!! (Canal StatQuest with Josh Starmer). Link: <https://www.youtube.com/watch?v=viZrOnJciY0>

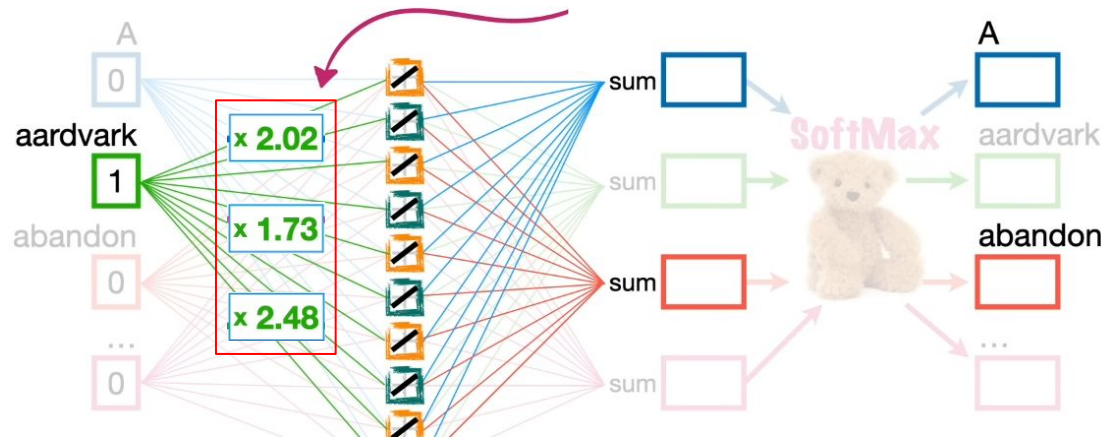
# Word Embeddings Fixas: Word2Vec

## Algoritmo Skip-gram:

4. Usar os pesos (otimizados) da regressão logística como embeddings

### Algoritmo Skip-gram:

1. Trate a palavra alvo e seu contexto (i.e., palavras vizinhas) como exemplos positivos
2. Pegue uma amostra aleatória de outras palavras do vocabulário para servir como exemplos negativos
3. Use uma regressão logística para treinar um classificador a distinguir entre os dois casos
4. Use os pesos da regressão como embeddings



(aardvark,a) [positivo]

(aardvark,abandon) [negativo]

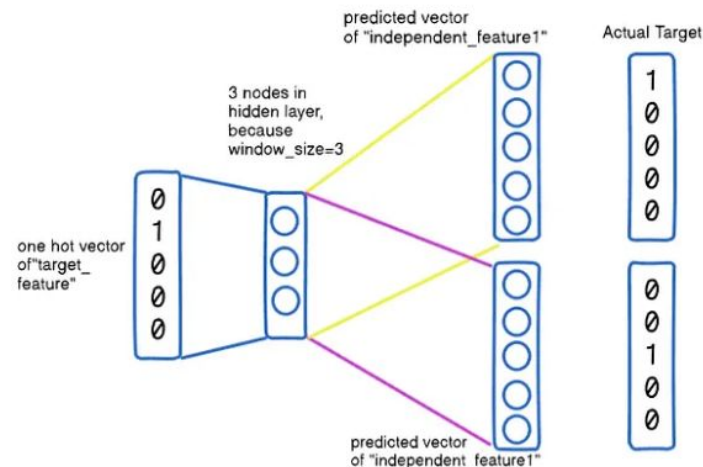
O classificador pode ser descartado  
(só queremos as embeddings)

Word Embedding e Word2Vec, clearly explained!!! (Canal StatQuest with Josh Starmer). Link: <https://www.youtube.com/watch?v=viZrOnJciY0>

# Word Embeddings Fixas: Word2Vec

- Resumo do Word2Vec (skip-gram):
  - Comece com vetores  $V$  de  $N$  dimensões ( $N \sim 300$ ) inicializados aleatoriamente
  - Use regressão logística, o classificador mais “básico” depois do Naive Bayes
  - Pegue um corpus e use as palavras que coocorrem como exemplos positivos
  - Use pares de palavras que não coocorrem como exemplos negativos
  - Treine um classificador para distinguir entre os dois grupos ajustando todas as embeddings para melhorar o desempenho do classificador
  - Descarte o classificador e fique com as embeddings

## Skip-gram



# Word Embeddings Fixas: Outros modelos

- FastText:
  - Baseado no modelo **skip gram** do Word2Vec
  - **Mais eficiente** (mais rápido e mais simples de treinar)
  - Utiliza **informação de subpalavras** (para lidar com palavras fora do vocabulário)
  - Modelos treinados disponíveis em 157 idiomas
  - Mais detalhes em: <http://www.fasttext.cc/>
- GloVe:
  - Leva em consideração as **estatísticas de coocorrências** do corpus
    - Matriz de coocorrência -> fatoração
  - Tenta combinar as vantagens da **LSI (Indexação Semântica Latente)** e do w2v skip gram
  - Artigo mostra ganhos
    - Mas na prática, os resultados são parecidos com os do w2v em muitas tarefas
  - Mais detalhes em: <http://nlp.stanford.edu/projects/glove/>

# Word Embeddings Fixas: Propriedades

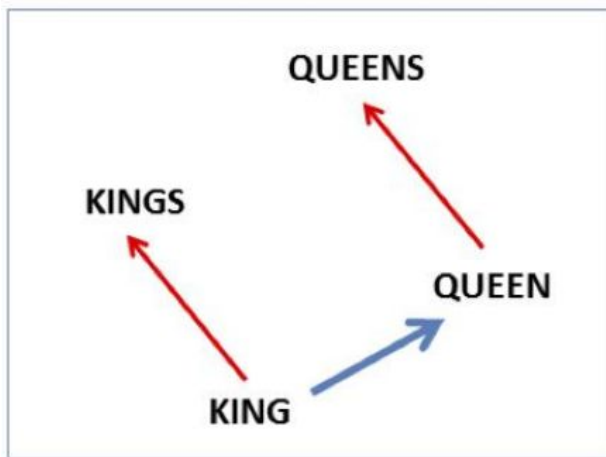
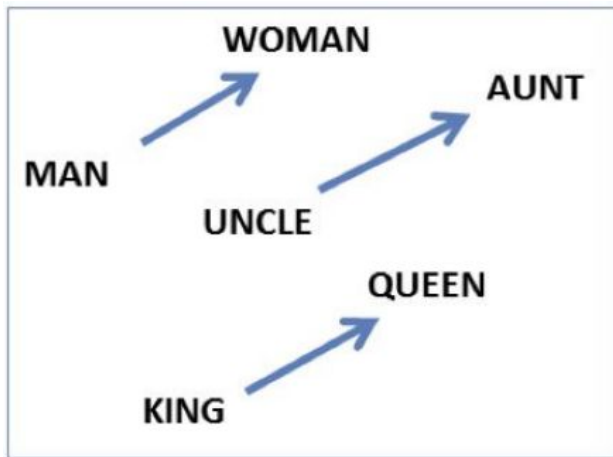
- A similaridade depende do tamanho da janela de contexto

Target Word	C = ~2	C = ~5
batman	nightwing aquaman catwoman superman manhunter	superman superboy aquaman catwoman batgirl
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood
turing	nondeterministic non-deterministic computability deterministic finite-state	non-deterministic finite-state nondeterministic buchi primality

Levy, O. and Goldberg, Y. (2014a). Dependency-based word embeddings. In ACL 2014.

# Word Embeddings Fixas: Propriedades

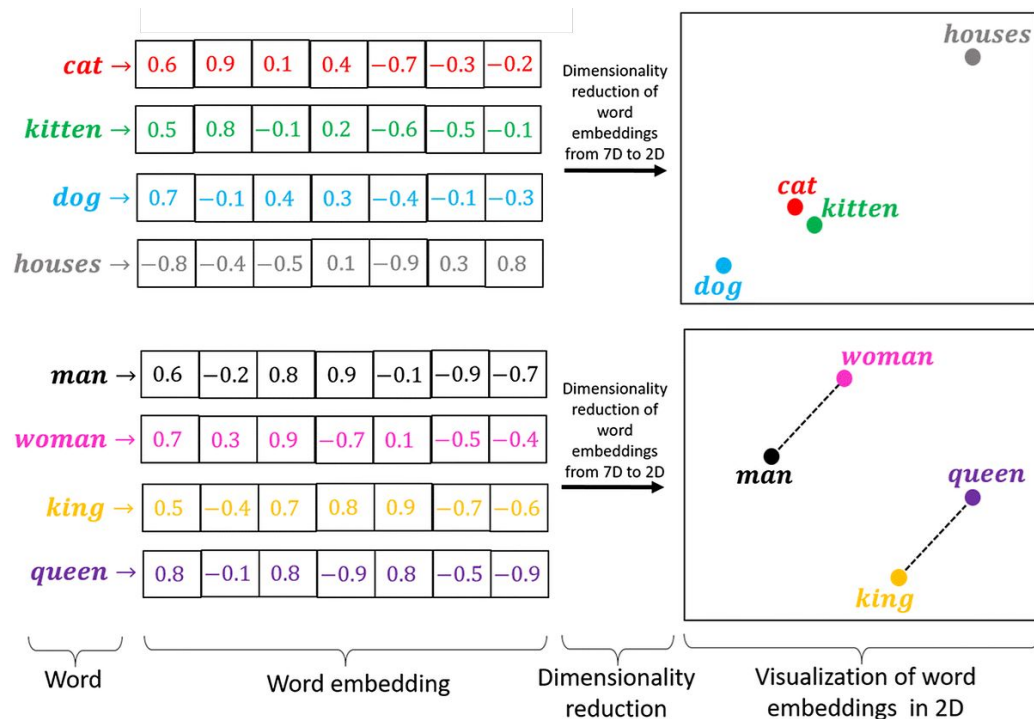
- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras  
 $\text{vector}(\text{'king'}) - \text{vector}(\text{'man'}) + \text{vector}(\text{'woman'}) \approx \text{vector}(\text{'queen'})$



Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In NAACL HLT 2013, 746–751.

# Word Embeddings Fixas: Propriedades

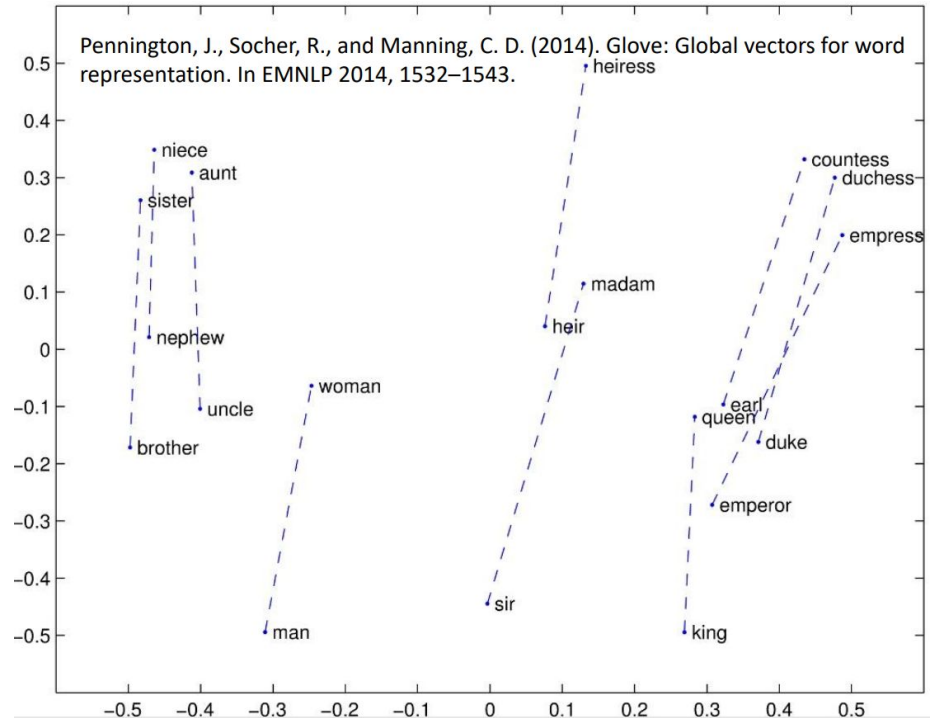
- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras





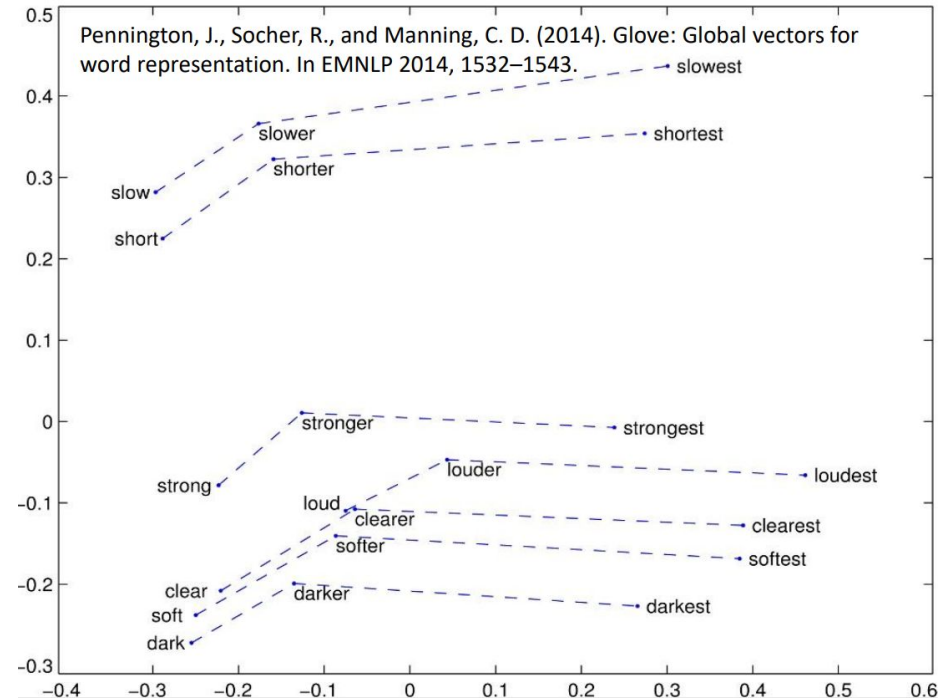
# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho da janela de contexto**
- Conseguem capturar **relações** entre as palavras



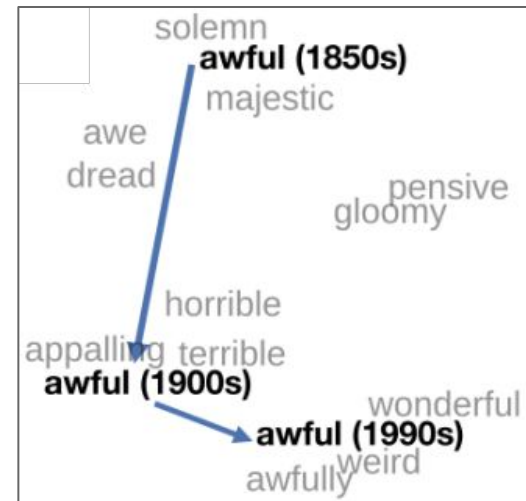
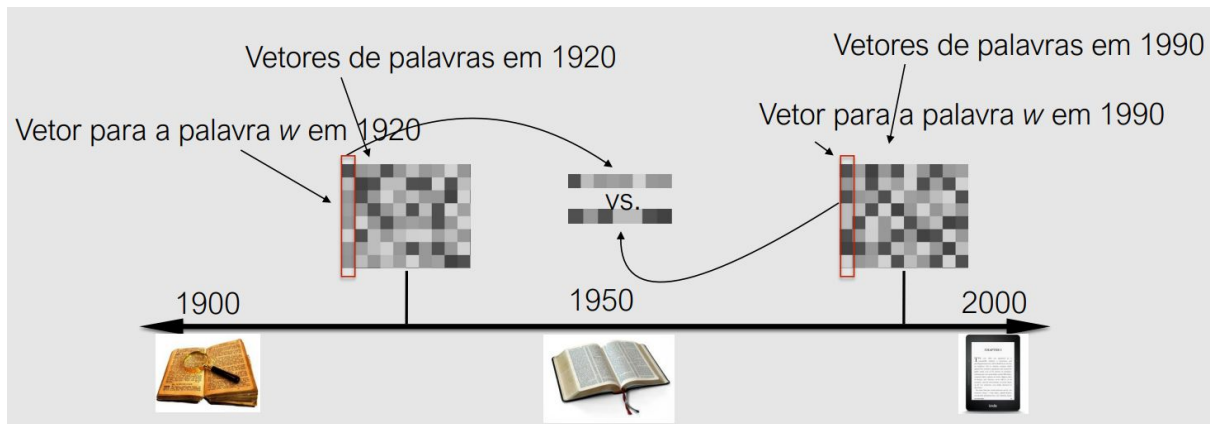
# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras



# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras
- Ajudam a estudar a **história dos significados** das palavras



~30 milhões de livros, 1850-1990, Google Books

Hamilton, W. L., Leskovec, J., and Jurafsky, D. (2016). Diachronic word embeddings reveal statistical laws of semantic change. In ACL 2016.

# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras
- Ajudam a estudar a **história dos significados** das palavras
- **Vieses e estereótipos**
  - Caliskan et al. (2017) descobriram que
    - Nomes **Afro-americanos** (*Leroy, Shaniqua*) tiveram um **cosseno** maior com palavras **desagradáveis** (*abuse, stink, ugly*)
    - Nomes **Europeus** (*Brad, Greg, Courtney*) tiveram um **cosseno** maior com palavras **agradáveis** (*love, peace, miracle*)



Caliskan, Aylin, Joanna J. Brusson and Arvind Narayanan. 2017. Semantics derived automatically from language corpora contain human-like biases. *Science* 356:6334, 183-186.

# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras
- Ajudam a estudar a **história dos significados** das palavras

- **Vieses e estereótipos**
  - Bolukbasi et al. (2016) investigaram relações



Ask “Paris : France :: Tokyo : x”

– x = Japan

Ask “father : doctor :: mother : x”

– x = nurse

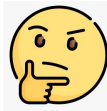
Ask “man : computer programmer :: woman : x”

– x = homemaker

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai.  
“Man is to computer programmer as woman is to homemaker? debiasing word  
embeddings.” In Advances in Neural Information Processing Systems, pp. 4349-4357. 2016.

# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras
- Ajudam a estudar a **história dos significados** das palavras
- **Vieses e estereótipos**
  - Então as **embeddings** são malvadas?



# Word Embeddings Fixas: Propriedades

- A **similaridade** depende do **tamanho** da **janela de contexto**
- Conseguem capturar **relações** entre as palavras
- Ajudam a estudar a **história dos significados** das palavras
- **Vieses e estereótipos**
  - Então as *embeddings* são malvadas?
  - **Não!** Elas **apenas refletem e replicam** todos os **vieses maldosos** da **sociedade!**



# Word Embeddings Fixas em Português

- NILC - Núcleo Interinstitucional de Linguística Computacional - USP de São Carlos

Link: <http://nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc>

## Download Word Embeddings Pré-treinadas

Para cada modelo, foram disponibilizados vetores de palavras gerados em várias dimensões. Alguns modelos como Word2vec, FastText e Wang2vec possuem as variações CBOW e Skip-Gram, que diferenciam-se pela forma como preveem as palavras. Em "Ver Detalhes" pode-se ter acesso às rotinas de processamento, limpeza e avaliação. No corpus, foram feitas tratativas de tokenização, remoção de stopwords, stemming e outras.

### Word2Vec

Modelo	Corpora STIL 2017
CBOW 50 dimensões	<a href="#">download</a>
CBOW 100 dimensões	<a href="#">download</a>
CBOW 300 dimensões	<a href="#">download</a>
CBOW 600 dimensões	<a href="#">download</a>
CBOW 1000 dimensões	<a href="#">download</a>
SKIP-GRAM 50 dimensões	<a href="#">download</a>
SKIP-GRAM 100 dimensões	<a href="#">download</a>
SKIP-GRAM 300 dimensões	<a href="#">download</a>
SKIP-GRAM 600 dimensões	<a href="#">download</a>
SKIP-GRAM 1000 dimensões	<a href="#">download</a>

[Ver Detalhes »](#)

### FastText

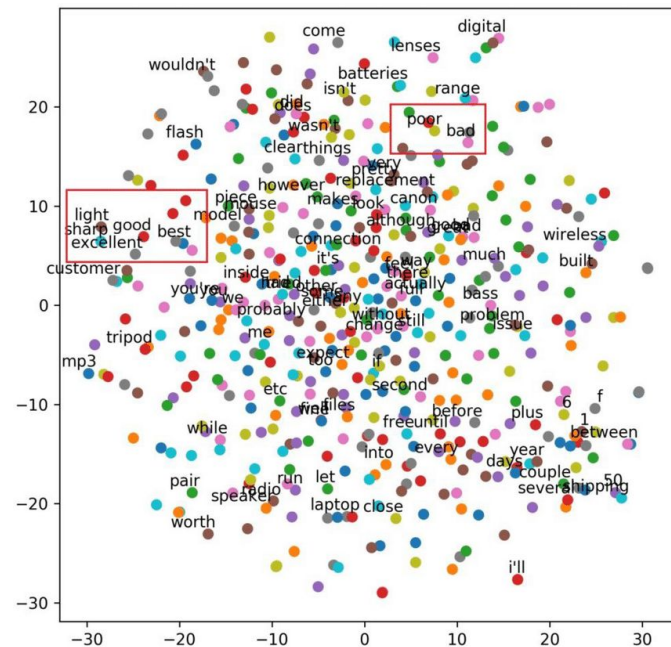
Modelo	Corpora STIL 2017
CBOW 50 dimensões	<a href="#">download</a>
CBOW 100 dimensões	<a href="#">download</a>
CBOW 300 dimensões	<a href="#">download</a>
CBOW 600 dimensões	<a href="#">download</a>
CBOW 1000 dimensões	<a href="#">download</a>
SKIP-GRAM 50 dimensões	<a href="#">download</a>
SKIP-GRAM 100 dimensões	<a href="#">download</a>
SKIP-GRAM 300 dimensões	<a href="#">download</a>
SKIP-GRAM 600 dimensões	<a href="#">download</a>
SKIP-GRAM 1000 dimensões	<a href="#">download</a>

[Ver Detalhes »](#)



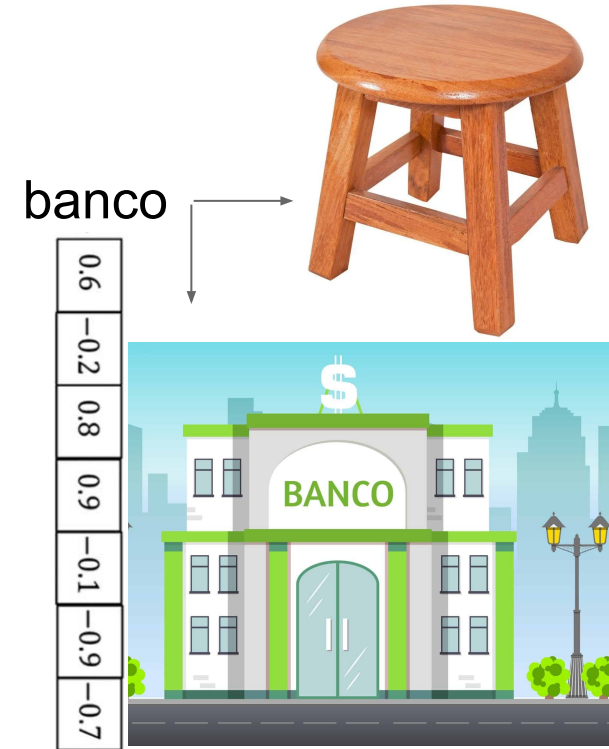
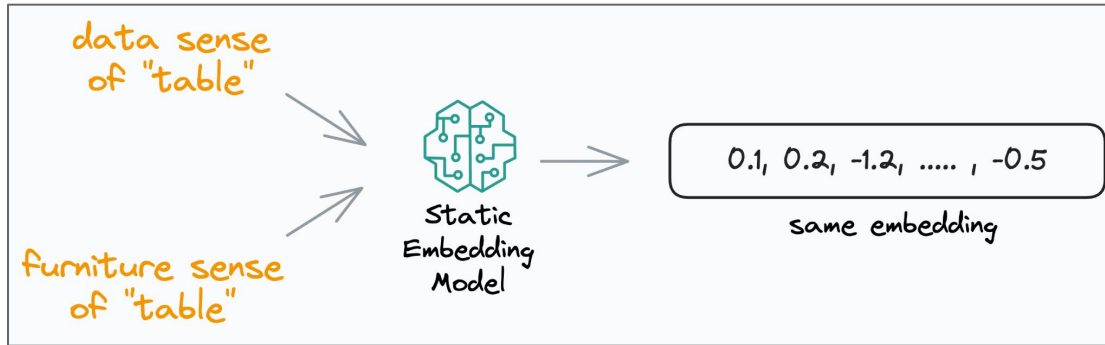
# Word Embeddings Fixas: Problemas

- *Embeddings Fixas* revolucionaram PLN de 2013 até 2018
  - Produzem **bons resultados** até **hoje!**
  - Entretanto possuem **limitações...**



# Word Embeddings Fixas: Problemas

- Falta de sensibilidade ao contexto
  - Cada palavra tem apenas um vetor de representação
  - Palavras com múltiplos significados tem o mesmo vetor



# Word Embeddings Fixas: Problemas

- **Falta de sensibilidade ao contexto**
  - Cada palavra tem apenas um vetor de representação
  - Palavras com múltiplos significados tem o mesmo vetor
- **Incapacidade de capturar relações sintáticas complexas**
  - Capturam a essência de palavras próximas, mas perdem a ordem das palavras e suas dependências

"O cachorro mordeu o homem."

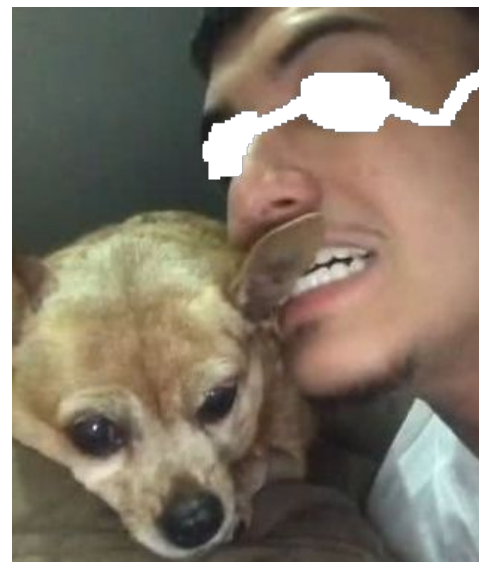


# Word Embeddings Fixas: Problemas

- **Falta de sensibilidade ao contexto**
  - Cada palavra tem apenas um vetor de representação
  - Palavras com múltiplos significados tem o mesmo vetor
- **Incapacidade de capturar relações sintáticas complexas**
  - Capturam a essência de palavras próximas, mas perdem a ordem das palavras e suas dependências

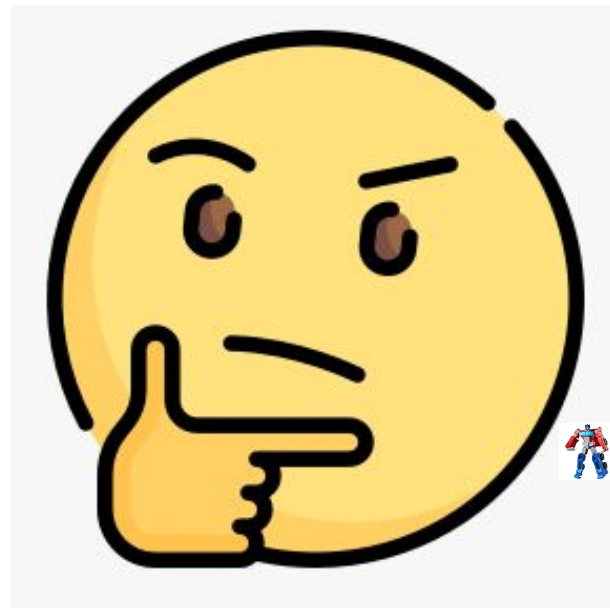
"O cachorro mordeu o homem."

"O homem mordeu o cachorro."



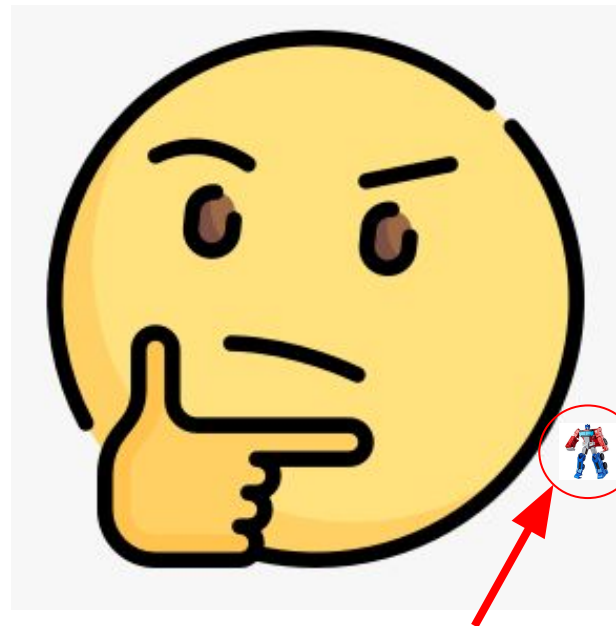
# Word Embeddings Fixas: Problemas

- **Falta de sensibilidade ao contexto**
  - Cada palavra tem apenas um vetor de representação
  - Palavras com múltiplos significados tem o mesmo vetor
- **Incapacidade de capturar relações sintáticas complexas**
  - Capturam a essência de palavras próximas, mas perdem a ordem das palavras e suas dependências
- **Representação estática e limitada**
  - Uma vez treinadas para um conjunto específico, não são mais modificadas para outros textos
- **Como resolver?**



# Word Embeddings Fixas: Problemas

- **Falta de sensibilidade ao contexto**
  - Cada palavra tem apenas um vetor de representação
  - Palavras com múltiplos significados tem o mesmo vetor
- **Incapacidade de capturar relações sintáticas complexas**
  - Capturam a essência de palavras próximas, mas perdem a ordem das palavras e suas dependências
- **Representação estática e limitada**
  - Uma vez treinadas para um conjunto específico, não são mais modificadas para outros textos
- **Como resolver?**



## Word Embeddings Fixas: Problemas

# LLMs (TRANSFORMERS)



- sensibilidade a contexto
- Representação
- Um modelo específico para cada tarefa
- Como resolver?

Mas antes disso precisamos estudar  
Redes Neurais para textos...

# Próximas aulas

- Aula prática (Laboratório 4)
- Aula teórica:
  - Redes neurais para textos [1]



UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
Instituto de Informática  
Departamento de Informática Aplicada

**Obrigado pela atenção!**  
**Dúvidas?**

Prof. Dennis Giovani Balreira  
(Material adaptado da Profa. Viviane Moreira e do Prof. Dan Jurafsky)



INF01221 - Tópicos Especiais em Computação XXXVI:  
Processamento de Linguagem Natural

