Processamento de Linguagem Natural

Episode IV : Word2Vec: Skip-Gram Model

Alex Marino

14 de novembro de 2024

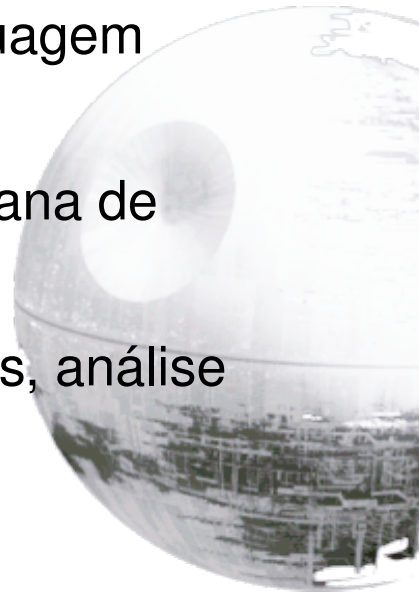
Conteúdo

Agenda da Aula

- Introdução ao Processamento de Linguagem Natural (NLP)
- Vetores de Palavras (Word Embeddings)
- Algoritmo Word2Vec
- Modelos Skip-Gram e CBOW
- Função Objetivo do Word2Vec
- Amostragem Negativa (Negative Sampling)
- Aplicações e Casos de Uso

Introdução ao Processamento de Linguagem Natural (NLP)

- O NLP envolve a interação entre computadores e linguagem humana.
- Objetivo: Entender, interpretar e gerar linguagem humana de maneira útil.
- Exemplo de Aplicações: Tradução automática, chatbots, análise de sentimentos.

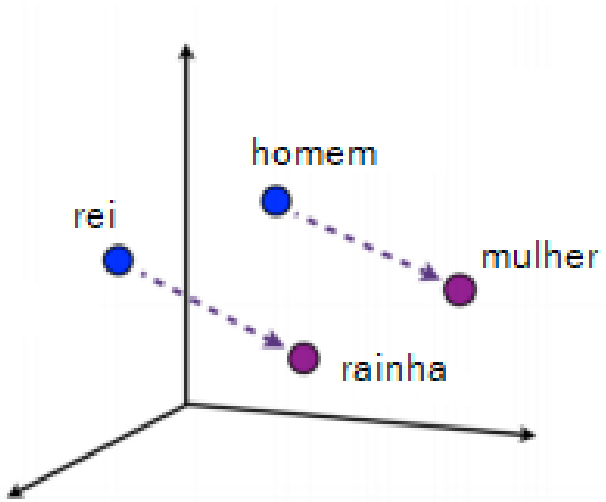


O Problema da Representação de Palavras

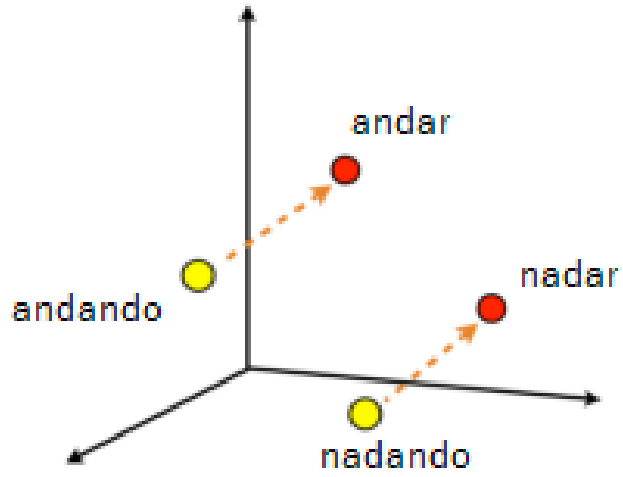
- Tradicionalmente, palavras são representadas como **vetores de uma única dimensão** (*one-hot encoding*).
- Problema: Não captura similaridade entre palavras.
- Exemplo: "hotel"e "motel"seriam representados como vetores completamente diferentes.

Vetores de Palavras (Word Embeddings)

- Ideia central: Representar palavras como vetores densos em um espaço contínuo.
- Palavras similares têm vetores próximos no espaço vetorial.
- Exemplo: Vetores para "rei"e "rainha"são próximos, e suas diferenças capturam relações como gênero.



Gênero



Conjugação verbal

Word2Vec: Conceito Principal

- Word2Vec é um algoritmo que aprende vetores de palavras a partir de grandes corpora de texto.
- Captura semelhanças semânticas e contextuais entre palavras.
- Exemplo prático: "rei- "homem"+ "mulher" "rainha".

Word2Vec: Duas Abordagens

O Word2Vec tem dois modelos principais:

- **Skip-Gram**: Prediz as palavras de contexto ao redor de uma palavra central.
- **CBOW (Continuous Bag of Words)**: Prediz a palavra central a partir de suas palavras de contexto.

Modelo Skip-Gram

- Objetivo: Dada uma palavra central, prever as palavras de contexto.
- O modelo aprende vetores que maximizam a probabilidade de prever as palavras ao redor.
- Exemplo: Dado "gato", prever palavras como "animal", "doméstico", etc.

Modelo CBOW (Continuous Bag of Words)

- Objetivo: Dado um conjunto de palavras de contexto, prever a palavra central.
- Exemplo: Dado "foi", "ao", "mercado", prever a palavra "João".

Função Objetivo do Word2Vec

Objetivo: Maximizar a probabilidade de palavras de contexto, dado uma palavra central.

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log P(w_{t+j}|w_t;\theta)$$

- O modelo ajusta vetores para maximizar essa probabilidade.
- Minimiza a função de custo para otimizar a similaridade entre palavras.

O Softmax e a Função de Probabilidade

- A probabilidade de uma palavra w_o ser um contexto da palavra central w_c é dada por:

$$P(w_o|w_c) = \frac{\exp(u_o^\top v_c)}{\sum_{w' \in V} \exp(u_{w'}^\top v_c)}$$

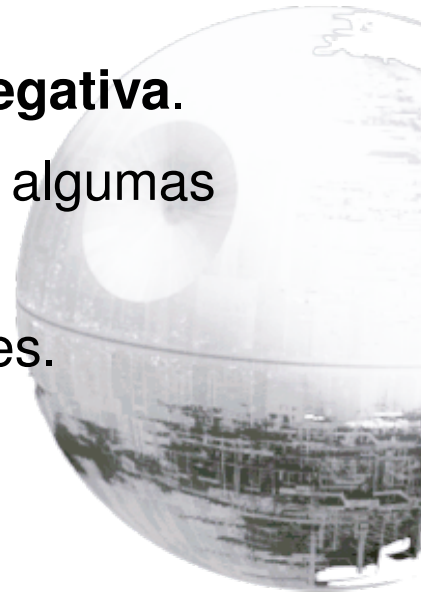
- O modelo utiliza a função **softmax** para garantir que a soma das probabilidades seja 1.

O Desafio do Cálculo do Softmax

- O cálculo do denominador no softmax é muito caro para grandes vocabulários.
- Cada atualização requer calcular a soma sobre todas as palavras do vocabulário.

Amostragem Negativa (Negative Sampling)

- Para reduzir a complexidade, usamos **amostragem negativa**.
- Em vez de calcular o softmax completo, selecionamos algumas "palavras negativas"aleatoriamente.
- Isso permite uma atualização mais eficiente dos vetores.



Exemplo Prático de Amostragem Negativa

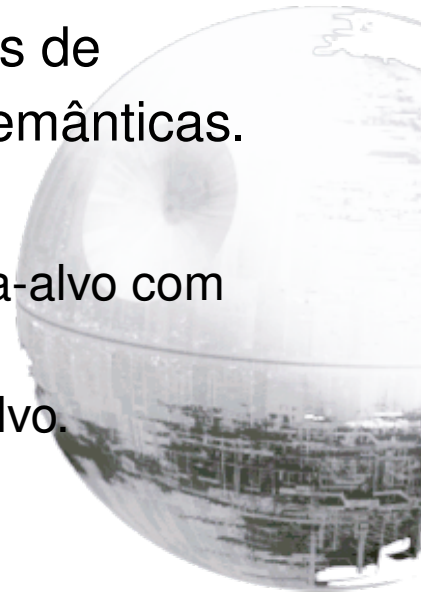
- Dado o par de palavras "gato"e "animal", o modelo tenta maximizar essa relação.
- Ao mesmo tempo, minimiza a relação entre "gato"e palavras aleatórias como "cadeira"ou "mesa".

Aplicações do Word2Vec

- Recuperação de Informação: Encontrar documentos relevantes com base em termos similares.
- Tradução Automática: Ajudar a encontrar traduções de palavras com base em vetores similares.
- Análise de Sentimentos: Capturar o contexto emocional de palavras em textos.

Introdução ao Word2Vec

- Word2Vec é um algoritmo usado para aprender vetores de palavras (word embeddings) que capturam relações semânticas.
- Dois modelos principais são utilizados:
  - **CBOW (Continuous Bag of Words)**: Prever a palavra-alvo com base no contexto.
  - **Skip-Gram**: Prever o contexto com base na palavra-alvo.



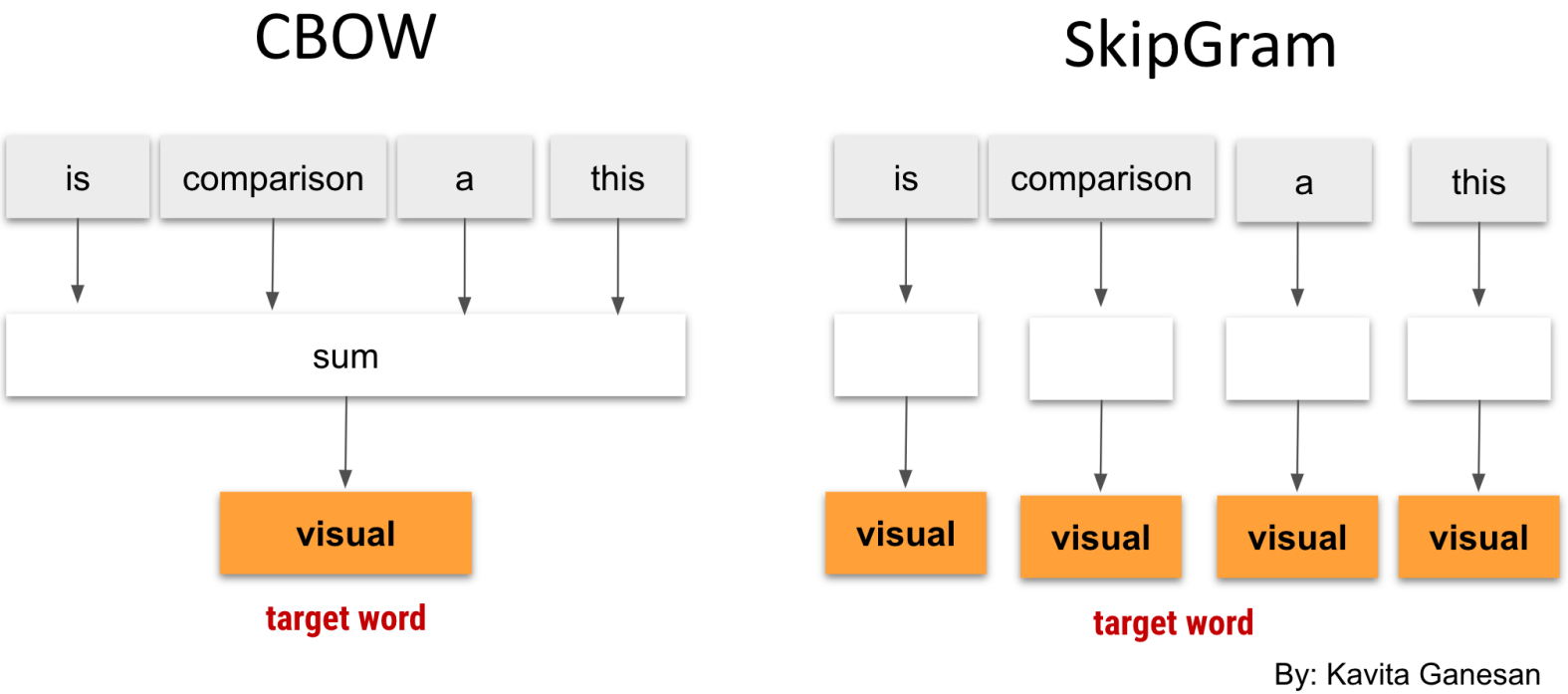
O que é Skip-Gram?

- Skip-Gram tenta prever palavras de contexto (ao redor) a partir de uma palavra-alvo.
- Objetivo: Maximizar a probabilidade de prever corretamente as palavras de contexto.
- Exemplo: Dada a palavra "gato", prever palavras ao redor como "animal", "felino", "domestico".

O que é CBOW?

- CBOW tenta prever a palavra-alvo a partir das palavras de contexto ao redor.
- Objetivo: Maximizar a probabilidade de prever corretamente a palavra-alvo com base no contexto.
- Exemplo: Dado o contexto "animal", "felino", prever a palavra "gato".

Skix-gram x cbow



This is a visual comparison

Direção dos Modelos

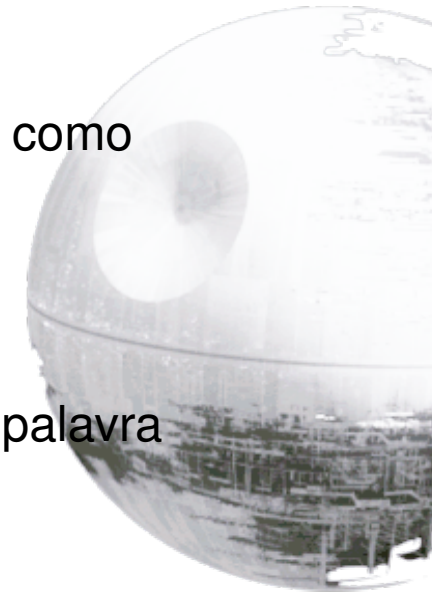
- **Skip-Gram**:
  - Dada a palavra-alvo, prever as palavras de contexto.
  - Direção: Palavra-Alvo $\rightarrow$ Palavras de Contexto.
- **CBOW**:
  - Dado o contexto, prever a palavra-alvo.
  - Direção: Palavras de Contexto $\rightarrow$ Palavra-Alvo.

Diferenças no Uso

- **Skip-Gram**:
  - Funciona melhor com palavras raras e grandes corpora.
  - Ideal para capturar relações semânticas de palavras incomuns.
- **CBOW**:
  - Mais eficiente e rápido para corpora menores.
  - Funciona bem com palavras comuns.

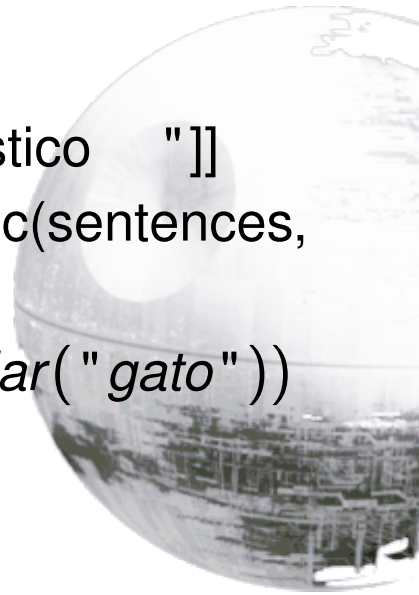
Comparação Matemática

- **Skip-Gram**:
  - Maximiza $P(\textit{contexto}|\textit{palavra\_alvo})$
  - Exemplo: Dada a palavra-alvo "gato", prever palavras como "animal", "felino", "doméstico".
- **CBOW**:
  - Maximiza $P(\textit{palavra\_alvo}|\textit{contexto})$
  - Exemplo: Dado o contexto "animal", "felino", prever a palavra "gato".



Exemplo prático: Skip-Gram em Python

Corpus de exemplo sentences = [[
"gato", "animal", "felino"], ["cachorro", "animal", "domstico "]]
Treinamento do modelo Skip-Gram (sg=1) model = Word2Vec(sentences,
$\text{vector}_s ize = 100, window = 5, sg = 1, min_c ount = 1)$
Ver palavras semelhantes a "gato"print(model.wv.$\text{most}_s imilar("gato"))$



Exemplo prático: CBOW em Python

Ver palavras semelhantes a "gato"print(model$_c$*bow*.*wv*.*most*$_s$*imilar*("*gato*"))

Eficiência Computacional

- **Skip-Gram**:
  - Mais lento para treinar, especialmente com corpora grandes.
  - Capta bem palavras raras, mas requer mais tempo.
- **CBOW**:
  - Mais rápido para treinar.
  - Funciona bem em corpora menores e para palavras comuns.

Vantagens e Desvantagens

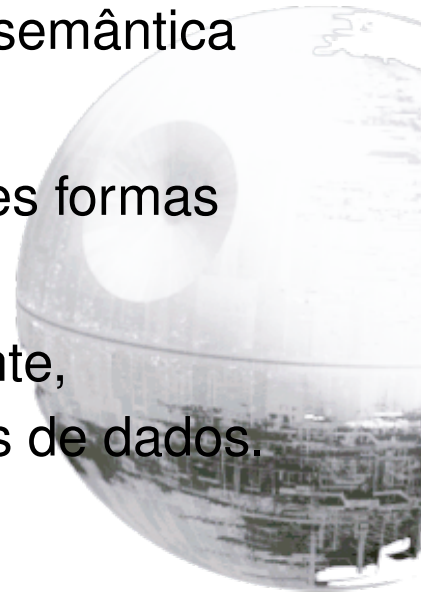
- **Skip-Gram**:
  - Vantagens: Captura bem palavras raras e relações semânticas.
  - Desvantagens: Mais lento e computacionalmente caro.
- **CBOW**:
  - Vantagens: Mais rápido e eficiente em corpora pequenos.
  - Desvantagens: Não captura bem palavras raras.

Conclusão

- Skip-Gram é preferido para grandes corpora e para capturar nuances semânticas de palavras raras.
- CBOW é mais eficiente para corpora pequenos e quando o desempenho é uma prioridade.
- Ambos os modelos têm suas aplicações específicas dependendo do tamanho e complexidade do corpus.

Conclusão

- O Word2Vec é uma técnica poderosa para capturar a semântica das palavras.
- Modelos como Skip-Gram e CBOW oferecem diferentes formas de capturar relações contextuais.
- Amostragem negativa torna o treinamento mais eficiente, permitindo que o modelo escale para grandes volumes de dados.



MAY THE

SOURCE

BE WITH

YOU