Inteligência Artificial 2

Aula 09: Como a IA Lê — Embeddings
Da Maldição One-Hot à Revolução Vetorial

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Setembro - 2026/2

Onde Paramos?

Até aqui, vocês dominaram o **Arco 1** completo da disciplina:

- O **Perceptron** como tomador de decisão: $z = X \cdot W + b$.
- O **MLP** com camadas ocultas e ReLU para problemas não-lineares.
- O **Backpropagation** calculando gradientes via Regra da Cadeia.
- O **Keras/TensorFlow** automatizando tudo com `model.fit()`.
- Classificação multiclasse com **Softmax** e combate ao **Overfitting**.

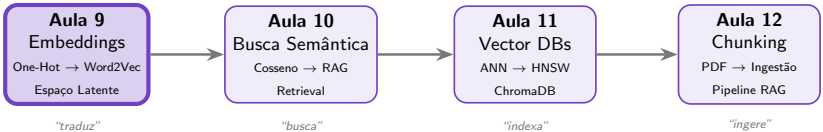
A Pergunta que Abre o Arco 2

Redes Neurais processam **números**.
Mas e se a entrada for **texto**? Como a IA "lê"?

Objetivos da Sessão

- **A Barreira:** Entender por que computadores não podem processar strings diretamente e a necessidade de representação numérica.
- **BoW e One-Hot:** Conhecer as abordagens históricas e suas falhas fatais (esparsidade, ausência de semântica).
- **Hipótese Distribucional:** A revolução de Firth — "Conhecerás uma palavra pelas companhias que ela tem".
- **Embeddings e Word2Vec:** A transição para vetores densos e as arquiteturas CBOW/Skip-Gram.
- **Aritmética Vetorial:** $\vec{v}_{\text{Rei}} - \vec{v}_{\text{Homem}} + \vec{v}_{\text{Mulher}} \approx \vec{v}_{\text{Rainha}}$.
- **Similaridade do Cosseno:** A métrica fundamental para medir proximidade semântica.
- **Ferramentas:** spaCy como ponte entre a teoria e a prática em Python.

O Mapa do Arco 2 — Revolução da Linguagem e Vetores

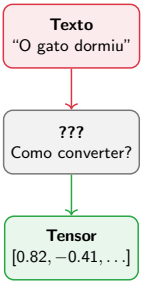


Conexão com o Arco 1

No Arco 1, a rede aprendeu a **decidir** com números. No Arco 2, vamos ensinar a IA a **ler** texto convertendo palavras em vetores — e essa conversão usa exatamente a mesma álgebra linear que vocês dominaram.

Computação e Linguagem Humana

- O cérebro humano processa linguagem associando sons/símbolos a conceitos complexos — de forma **analógica e contextual**.
- Redes Neurais operam exclusivamente com **tensores** de números em ponto flutuante.
- Tudo em Deep Learning exige que a informação seja reduzida a **matrizes numéricas** antes de qualquer operação.

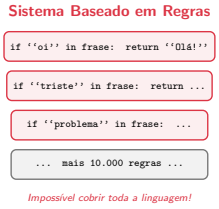


O Grande Desafio do NLP

Como mapear a vastidão do pensamento humano — com nuances, sarcasmo e contexto cultural — em um formato vetorial que um computador possa otimizar?

As Primeiras Abordagens: Regras e Dicionários

- Décadas de 1960–70: sistemas especialistas (ELIZA, SHRDLU).
- **Abordagem:** Milhares de regras `if-else` mapeando estruturas gramaticais.
- O programador escrevia explicitamente cada padrão linguístico.
- **Problema Fatal:** Linguagem humana é não-determinística — ambiguidade, ironia, erros e contexto cultural são imensuráveis.



O Paradigma Bag of Words (BoW)

- Anos 1990–2000: métodos **estatísticos** substituem regras manuais.
- **Bag of Words:** Trata o documento como um conjunto *não-ordenado* de palavras, contando a frequência de cada termo.

Frase A: "O gato mordeu o rato"

Frase B: "O rato mordeu o gato"

Vocabulário: [o, gato, mordeu, rato]

$v_A = [2, 1, 1, 1]$ $\longleftrightarrow$ IDÊNTICOS! $v_B = [2, 1, 1, 1]$

Limitação Fatal

"O gato mordeu o rato" e "O rato mordeu o gato" geram o **mesmo vetor BoW**. A ordem — e portanto o significado — se perde completamente.

Representação de Palavras: One-Hot Encoding

- Cada palavra recebe um ID exclusivo e é transformada em vetor binário de comprimento $|V|$.
- Apenas a posição correspondente ao ID recebe valor 1; todo o resto é 0.

Exemplo: $V = [\text{gato, cachorro, carro, avião}]$

Palavra	Vetor One-Hot
gato	$[1, 0, 0, 0]$
cachorro	$[0, 1, 0, 0]$
carro	$[0, 0, 1, 0]$
avião	$[0, 0, 0, 1]$

Conexão com Álgebra Linear

Vocês reconhecem esses vetores? São **vetores canônicos** $e_1, e_2, \ldots, e_n$ — a base ortonormal do $\mathbb{R}^n$ que vocês viram em Álgebra Linear. Cada palavra vive em um eixo independente do espaço.

O Fracasso do One-Hot: Dimensionalidade e Esparsidade

- **Maldição da Dimensionalidade**: Dicionário de Português $\approx$ 500.000 palavras $\Rightarrow$ cada vetor tem 500.000 posições.
- 99,9998% dos valores são zero: vetores **esparsos**.
- Multiplicações matriciais com matrizes enormes e quase vazias **paralisam** GPUs.

Ortogonalidade Semântica

$v_{\text{gato}} \cdot v_{\text{cachorro}} = 0$

O produto escalar é **zero**!

Para a máquina, "gato" é tão diferente de "cachorro" quanto de "avião".

Não há conexão com o significado.

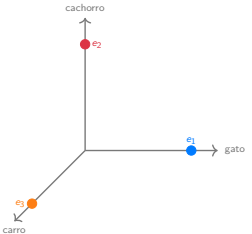


Tabela Comparativa: One-Hot vs. Embeddings

	One-Hot	**Embedding (Denso)**
Dimensão	$\|V\|$ (ex: 500.000)	Fixa (ex: 300)
Valores	Binários (0 ou 1)	Contínuos ($\mathbb{R}$)
Esparsidade	99,9% zeros	Denso (sem zeros)
Semântica	Nenhuma	Captura contexto
"gato" $\sim$ "cachorro"	$\cos\theta = 0$	$\cos\theta \approx 0.85$
Memória	Explosiva	Compacta

A Conclusão

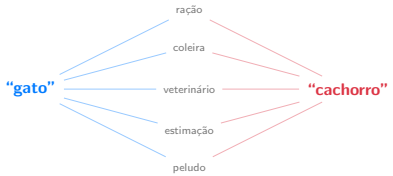
One-Hot trata palavras como entidades isoladas e ortogonais. Embeddings capturam **significado** — e essa captura usa a mesma álgebra matricial que vocês já dominam.

A Hipótese Distribucional

Firth, 1957

"You shall know a word by the company it keeps"
(Conhecerás uma palavra pelas companhias que ela tem)

- O **significado** de uma palavra não está nos seus caracteres, mas no seu **contexto de uso**.
- Se "gato" e "cachorro" aparecem frequentemente perto de "ração", "coleira", "veterinário" e "estimação", devem compartilhar significados semelhantes.



Contexto compartilhado ⇒ significado similar

A Transição: De Vetores Esparsos a Densos

- Como capturar essa semântica distribucional? Através dos **Embeddings** (Representações Densas).
- Ao invés de um vetor binário esparso de dimensão 500.000, usamos um vetor **contínuo e compacto** de dimensão fixa (ex: 300):

$$e_{\text{gato}} = [\underbrace{0.85}_{1}, \underbrace{-0.42}_{2}, \underbrace{0.11}_{3}, \ldots, \underbrace{-0.93}_{300}] \in \mathbb{R}^{300}$$

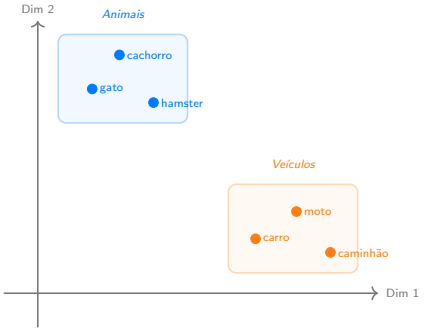
- Cada dimensão codifica uma **característica latente** abstrata (gênero, animacidade, realeza...), descoberta **automaticamente** durante o treinamento.

Conexão com o Arco 1

Lembram da multiplicação $X \cdot W$? A matriz de Embeddings *é* uma camada `Dense` especial — uma **lookup table** treinável que converte IDs em vetores densos.

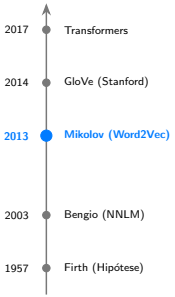
O Conceito de Espaço Latente

- O **Espaço Latente** é um hiperespaço de N dimensões onde cada palavra é um **ponto** (coordenada).
- Se treinado corretamente, palavras com contextos similares ficam **fisicamente próximas** nesse espaço.
- Conceitos semanticamente distantes ficam afastados.



A Invenção do Word2Vec

- Em **2013**, Tomas Mikolov (Google) publicou o **Word2Vec** — a primeira arquitetura de Redes Neurais Rasas que escalava para **bilhões de palavras** em poucas horas.
- Não foi a primeira tentativa de Embeddings, mas foi a que **revolucionou** a NLP pela eficiência.
- A NLP abandonou o formalismo discreto e migrou para o **contínuo e estocástico**.
- Word2Vec é a base que alimenta Transformers e LLMs modernos.

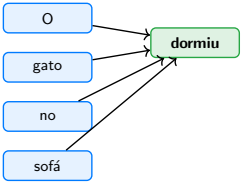


As Duas Arquiteturas do Word2Vec

O Word2Vec engloba duas metodologias de aprendizado *self-supervised* (o texto é seu próprio rótulo):

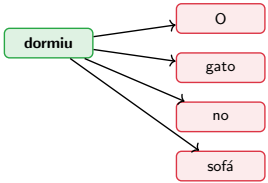
1. CBOW

Continuous Bag-of-Words



Contexto → Palavra Central

2. Skip-Gram



Palavra Central → Contexto

CBOW: A Matemática da Predição

- Corpus como sequência $w_1, w_2, \ldots, w_T$, com janela de contexto c.
- **Objetivo:** Maximizar a probabilidade da palavra central dado o contexto:

$$P(w_t \mid w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c})$$

- A rede calcula a **média** dos vetores de contexto e propaga para camada densa com `softmax` sobre todo o vocabulário.
- O **backpropagation** ajusta as coordenadas latentes para que palavras que co-ocorrem fiquem próximas.

Conexão com Aula 3

A função `softmax` é a mesma que vocês usaram na classificação multiclasse! Aqui, cada "classe" é uma palavra do dicionário.

- No **Skip-Gram**, a função objetivo inverte a relação — prediz contexto a partir da central:

$$J = \frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-c \leq j \leq c \\ j \neq 0}} \log P(w_{t+j} \mid w_t)$$

O Problema

O Softmax sobre $|V|$ palavras (ex: 500K) é computacionalmente proibitivo a cada iteração.

A Solução: Negative Sampling

Classifica o par real como positivo e amostra k pares aleatórios como negativos. Converte um problema de complexidade $O(|V|)$ em regressão logística local $O(k)$.

Analogias e Aritmética Vetorial

A descoberta mais fascinante do Word2Vec: relações semânticas viram **operações vetoriais lineares**.

- O vetor translação "Itália" $\rightarrow$ "Roma" é quase idêntico ao de "França" $\rightarrow$ "Paris":

$$\vec{v}_{\text{Roma}} - \vec{v}_{\text{Itália}} \approx \vec{v}_{\text{Paris}} - \vec{v}_{\text{França}}$$

- A rede internalizou relações **abstratas** (capital-de, gênero, tempo verbal) como direções no espaço.

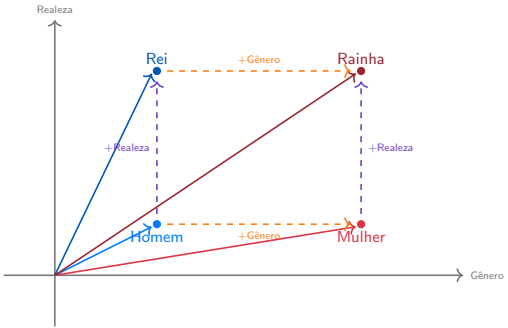
Operação		**Resultado**
Itália − Roma	$\approx$	França −
Espanha + (Paris − França)	$\approx$	Madrid
Rei − Homem + Mulher	$\approx$	Rainha
Walking − Walk	$\approx$	Swimming

O Exemplo Clássico

Isolar o "eixo de gênero" e aplicar o deslocamento:

$$\vec{v}_{\text{Rei}} - \vec{v}_{\text{Homem}} + \vec{v}_{\text{Mulher}} \approx \vec{v}_{\text{Rainha}}$$

- "Rei" contém a semântica *masculino* + *realeza*.
- Subtrair "Homem" remove masculinidade; somar "Mulher" injeta feminilidade.
- O resultado cai perto de "Rainha" no espaço vetorial.



Exemplo Numérico Passo-a-Passo

Vamos simular com vetores simplificados de 4 dimensões:

Palavra	**Vetor (4D simplificado)**
Rei	[0.8, 0.9, 0.2, 0.1]
Homem	[0.7, 0.1, 0.3, 0.0]
Mulher	[0.1, 0.1, 0.8, 0.0]
Rei − Homem	[0.1, 0.8, −0.1, 0.1]
+ Mulher	[0.2, 0.9, 0.7, 0.1]
Rainha (real)	[0.2, 0.9, 0.8, 0.1]

Observação

O resultado $[0.2, 0.9, 0.7, 0.1]$ é **muito próximo** do vetor real de "Rainha" $[0.2, 0.9, 0.8, 0.1]$. A aritmética vetorial captura relações abstratas que emergem do treinamento!

Métrica de Proximidade: Similaridade do Cosseno

- A Distância Euclidiana pode falhar com vetores de magnitudes diferentes (palavras frequentes vs. raras).
- A solução padrão: **Similaridade do Cosseno**, que mede apenas o **ângulo** entre vetores:

Equação da Similaridade de Cosseno

$$\cos\_\text{sim}(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Resultado de -1 a 1:** ≈ 1 = mesma direção (semântica idêntica); ≈ 0 = ortogonais (sem relação); ≈ -1 = opostos (raro em NLP).

Cosseno: Calculando na Prática

Usando os vetores simplificados do exemplo anterior:

$A = \text{gato} = [0.8, 0.3, 0.1]$
$B = \text{cachorro} = [0.7, 0.4, 0.2]$

Produto Escalar:
$A \cdot B = 0.8 \times 0.7 + 0.3 \times 0.4 + 0.1 \times 0.2 = 0.70$

Normas:
$\|A\| = \sqrt{0.64 + 0.09 + 0.01} = 0.860$
$\|B\| = \sqrt{0.49 + 0.16 + 0.04} = 0.831$

Cosseno:
$\cos\theta = \frac{0.70}{0.860 \times 0.831} = \mathbf{0.979}$

Interpretação

Similaridade de $0.979 \approx 1.0$ indica que "gato" e "cachorro" são **semanticamente muito próximos** no espaço vetorial.

Conexão com ÁL

O produto escalar e a norma que vocês viram em Álgebra Linear — $\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos\theta$ — é **exatamente** esta fórmula.

O Desafio do Treinamento: Transfer Learning

- Treinar Embeddings do zero (Random Init) com Skip-Gram + corpus robusto (Wikipedia, Common Crawl) exige **supercomputadores e dias de GPU**.
- A indústria **não treina do zero**. Usa **Transfer Learning**: download de matrizes pré-treinadas.

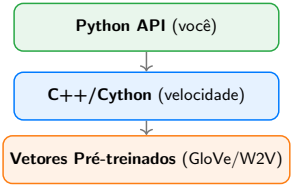
Projeto	**Organização**	**Diferencial**
Word2Vec	Google	Pioneiro (Skip-Gram/CBOW)
GloVe	Stanford	Fatoração de co-ocorrência
fastText	Meta (Facebook)	Subwords (lida com neologismos)

Na Prática

Você faz `download` de uma matriz $|V| \times D$ pré-treinada e a pluga na sua rede. É como "importar conhecimento linguístico" de bilhões de textos!

Processamento Industrial de NLP: O spaCy

- **spaCy** é a biblioteca de referência industrial para NLP veloz e coesa em Python.
- Motor em C++ sob o capô: tokenização, extração de entidades, parsing sintático — tudo hiper-otimizado.
- Hospeda download direto de vetores em diversas línguas, incluindo `pt_core_news_md` para Português.



Usando spaCy em Python

Após instalar (`pip install spacy`) e baixar o modelo pelo CLI:

```python
import spacy

# Carregando o pipeline em portugues com Embeddings ("md" = medium)
nlp = spacy.load("pt_core_news_md")

# Processando duas palavras
token_rei = nlp("rei")[0]
token_rainha = nlp("rainha")[0]

# Visualizando as coordenadas do espaco latente (Vetor de 300
    dimensoes)
vetor = token_rei.vector
print(f"Dimensoes do vetor de Rei: {vetor.shape}") # Saida: (300,)
print(f"Algumas coordenadas: {vetor[:5]}")
```

Calculando Similaridade com spaCy

O método `.similarity()` calcula automaticamente o cosseno entre dois vetores:

```
import spacy
nlp = spacy.load("pt_core_news_md")

doc = nlp("cachorro gato carro moto bicicleta maca banana")

# Comparando similaridades semanticas via Algebra Vetorial
print("cachorro vs gato:", doc[0].similarity(doc[1])) # Alta (Ex:
    0.85)
print("cachorro vs carro:", doc[0].similarity(doc[2])) # Baixa (Ex:
    0.12)
print("carro vs moto:", doc[2].similarity(doc[3]))    # Alta (Ex:
    0.78)
print("maca vs banana:", doc[5].similarity(doc[6]))   # Alta (Ex:
    0.75)
print("moto vs banana:", doc[3].similarity(doc[6]))   # Muito baixa
```

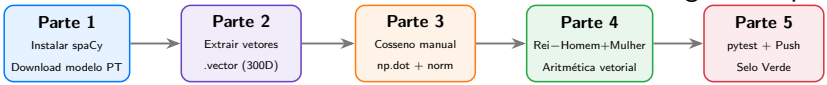
Recapitulação — O que Levar Desta Aula

1. **Texto $\neq$ Número**: Redes Neurais exigem representação numérica.
2. **BoW e One-Hot falham**: Esparsidade, explosão dimensional e zero semântica.
3. **Hipótese Distribucional**: O significado emerge do contexto de uso.
4. **Embeddings**: Vetores densos de dimensão fixa ($\mathbb{R}^{300}$) que capturam semântica.
5. **Word2Vec**: CBOW (contexto $\rightarrow$ palavra) e Skip-Gram (palavra $\rightarrow$ contexto).
6. **Aritmética Vetorial**: Rei $-$ Homem $+$ Mulher $\approx$ Rainha.
7. **Cosseno**: A métrica que ignora magnitude e foca na *direção* semântica.

Próxima Aula

Agora que sabemos converter palavras em vetores, como **buscamos** o documento mais relevante? A Busca Semântica e a arquitetura RAG que mudou a indústria.

Missão: Extrair vetores semânticos e realizar aritmética de Embeddings com spaCy + NumPy.



Objetivo

Não apenas observar similaridades — mas reconstruir algebraicamente as analogias usando NumPy puro e validar com a distância euclidiana para o resultado esperado.

Lab Passo 1: Preparando o Terreno

- O spaCy já está instalado no ambiente, mas precisamos garantir o download do modelo em português.
- Vocês carregarão o modelo `pt_core_news_md`. O "md" (medium) garante que o modelo venha com os vetores de 300 dimensões (o "sm", small, não tem vetores!).

```
import spacy
nlp = spacy.load('pt_core_news_md')
```

Lab Passo 2: Extração de Vetores

- Cada palavra processada vira um `Token`.
- A propriedade `.vector` revela a "alma" matemática da palavra.

```
palavra_rei = nlp("rei")[0]
v_rei = palavra_rei.vector  # O Array Numpy de 300 dimensoes
```

Vocês farão isso para extrair os vetores base da analogia: rei, homem, e mulher.

Lab Passo 3: O Cosseno Manual (A Base)

- Vocês poderiam usar `token1.similarity(token2)`, mas a missão aqui é provar que não há mágica.
- Vocês vão implementar a fórmula do Cosseno usando `np.dot` (produto escalar) e `np.linalg.norm` (norma L2).

```
def cosseno(a, b):
    # SUA MISSAO AQUI: a.b / (|a| * |b|)
    pass
```

Lab Passo 4: Aritmética Vetorial na Prática

- Vocês calcularão: `v_resultado = v_rei - v_homem + v_mulher`.
- E então, usarão a função de cosseno que construíram para medir a similaridade entre `v_resultado` e o vetor real de "rainha".
- Verão a pontuação atingir valores surpreendentes, provando o funcionamento empírico dos Embeddings!

Lab Passo 5: Testes e CI/CD

- O pytest vai injetar vetores mockados na sua função `cosseno` para garantir que a matemática está impecável.
- Também vai verificar se o pipeline extraiu vetores de 300 dimensões, e não 96 ou outra arquitetura.
- Como sempre: `git add .`, `git commit`, e `git push`. O **Selo Verde** aguarda!

Dúvidas?

Email: alessio@cefetmg.br
Aula: Segunda e Quarta-Feira, 7h as 8:40h (Sala 39)
WhatsApp: Dúvidas rápidas

Lembrete

Confira sua Issue no GitLab!

Próxima aula: Busca Semântica e Similaridade

100% da Aula 9 Concluída!