Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Capítulo 9 – Como a IA Lê: Embeddings

Aléssio Miranda Jr.

Setembro - 2026/2

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

O que você verá neste capítulo

Ao longo de todo o Arco 1, nossas redes neurais operaram sobre dados essencialmente numéricos: planilhas CSV, pixels de imagens, tensores de floats. Mas a linguagem humana é feita de letras, não de números. Como, então, ensinamos uma máquina a "ler"? Neste capítulo, faremos a travessia do paradigma numérico para o paradigma linguístico, explorando a revolução dos **Embeddings** — a representação vetorial de palavras em espaços de alta dimensionalidade. Partiremos da falha catastrófica do *One-Hot Encoding*, passaremos pela *Hipótese Distribucional* de Firth (1957), chegaremos à arquitetura *Word2Vec* de Mikolov et al. (2013), e aterrisaremos no ecossistema industrial `spaCy` para extração de vetores semânticos em Python. Ao final, você será capaz de converter qualquer texto em um vetor denso e medir a similaridade entre conceitos usando álgebra linear pura.

1 O Problema Fundacional: Letras vs. Números

Nas Aulas 1 a 8 do Arco 1, todas as nossas redes neurais recebiam entradas que já eram números por natureza: coordenadas de pixels em imagens, colunas numéricas de planilhas, ou valores binários em portas lógicas. O `np.dot()` que vocês usaram extensivamente na implementação do Perceptron e da MLP era, na essência, uma multiplicação de matrizes — a mesma operação que vocês estudaram em Álgebra Linear quando multiplicavam $A_{m \times n} \cdot B_{n \times p}$.

Mas a linguagem humana é categoricamente diferente. Quando um ser humano escreve "O gato dormiu no sofá", cada palavra é uma sequência de caracteres sem valor numérico intrínseco. A letra "g" não é "maior" que a letra

CAPÍTULO EM REVISÃO (18 de agosto de 2025 a 15 de setembro de 2026). O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

"s"; a palavra "gato" não pode ser somada à palavra "sofá" de forma significativa. Redes neurais, contudo, operam exclusivamente sobre tensores de ponto flutuante. Portanto, o primeiro desafio que enfrentamos ao entrar no domínio do **Processamento de Linguagem Natural (NLP)** é: como converter texto em números sem perder o significado?

Figura 1: O problema central do NLP: como transformar texto (dado categórico) em um tensor numérico que preserve o significado semântico? A resposta a essa pergunta é o tema central deste capítulo.

2 A Primeira Tentativa: One-Hot Encoding

A abordagem mais ingênua — e historicamente a primeira utilizada — é o **One-Hot Encoding**. A ideia é elementar: atribuímos a cada palavra do vocabulário um índice inteiro e representamos esse índice como um vetor canônico (unitário) no $\mathbb{R}^V$, onde V é o tamanho do vocabulário. Se o vocabulário contém 50.000 palavras, cada palavra se torna um vetor de 50.000 dimensões com um único "1" na posição correspondente e 49.999 zeros.

Vocês já viram vetores canônicos em Álgebra Linear: são os vetores $\hat{e}_1 = [1, 0, 0, \ldots]$, $\hat{e}_2 = [0, 1, 0, \ldots]$, e assim por diante. O One-Hot Encoding nada mais é do que mapear cada palavra para um desses vetores da base canônica de $\mathbb{R}^V$.

Tabela 1: Representação One-Hot para um vocabulário de 5 palavras. Cada palavra é um vetor canônico ortogonal a todos os outros.

Palavra	d_1	d_2	d_3	d_4	d_5
gato	1	0	0	0	0
cachorro	0	1	0	0	0
rei	0	0	1	0	0
rainha	0	0	0	1	0
cadeira	0	0	0	0	1

CAPÍTULO EM REVISÃO (18 de agosto de 2025) – O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

△ Importante: title=Os Três Problemas Fatais do One-Hot Encoding

1. **Dimensionalidade explosiva:** Um vocabulário de 50.000 palavras gera vetores de 50.000 dimensões. A matriz de Embeddings de um corpus de 1 milhão de documentos teria $10^6 \times 5 \times 10^4 = 5 \times 10^{10}$ elementos — a maioria zeros. Computacionalmente insustentável.
2. **Perda total de semântica:** Como todos os vetores canônicos são ortogonais entre si (o produto escalar é zero para quaisquer dois vetores distintos), as palavras "bom" e "excelente" estão tão distantes vetorialmente quanto "bom" e "péssimo". A representação é **agnóstica ao significado**.
3. **Ausência de contexto:** A palavra "banco" recebe exatamente o mesmo vetor independentemente de significar "instituição financeira" ou "assento de madeira". O One-Hot não captura polissemia.

A percepção de que o produto escalar $\hat{e}_i \cdot \hat{e}_j = 0$ para $i \neq j$ é a raiz do problema deveria soar familiar: em Álgebra Linear, vocês aprenderam que vetores ortogonais não compartilham nenhuma "informação" entre si. Precisamos de uma representação onde palavras semanticamente próximas tenham vetores **não-ortogonais** — isto é, cujo produto escalar (e portanto cujo cosseno do ângulo) seja positivo e significativo.

3 A Hipótese Distribucional de Firth

A solução teórica para o problema da semântica vetorial veio de uma ideia surpreendentemente antiga. Em 1957, o linguista britânico John Rupert Firth postulou aquela que se tornaria a pedra angular de toda a representação vetorial moderna:

• Teoria: title=A Hipótese Distribucional (Firth, 1957)

"You shall know a word by the company it keeps." (Conhecerás uma palavra pelas companhias que ela mantém.)
O fundamento teórico dos Embeddings é que o **significado** de uma palavra não reside nos seus caracteres, mas nos **contextos linguísticos** em que ela aparece. Se as palavras "gato" e "cachorro" aparecem frequentemente nos mesmos contextos ("O ___ dormiu no sofá", "Levei o ___ ao veterinário", "O ___ comeu a ração"), o modelo aprende que elas são semanticamente próximas — sem jamais ter "visto" um gato ou cachorro. A similaridade é deduzida *puramente* a partir de padrões estatísticos de coocorrência no texto.

Essa hipótese é elegante porque transforma um problema filosófico ("o que significa 'gato'?") em um problema estatístico ("em quais contextos a palavra

CAPÍTULO EM REVISÃO (18 de agosto de 2025) – O conteúdo pode sofrer alterações.

‘gato’ aparece?”). E problemas estatísticos são exatamente o que redes neurais sabem resolver.

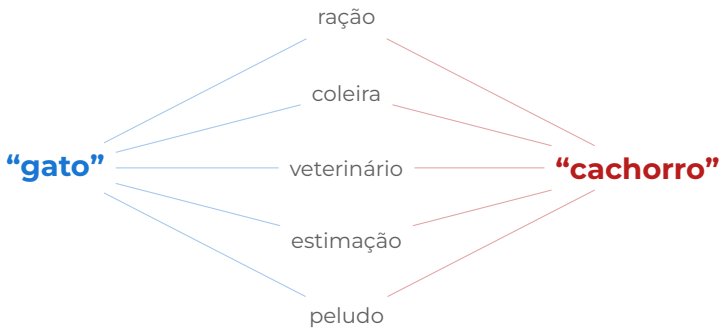


Figura 2: A Hipótese Distribucional em ação: “gato” e “cachorro” compartilham os mesmos contextos linguísticos (ração, coleira, veterinário), portanto devem ocupar regiões próximas no espaço vetorial.

4 A Revolução Semântica: Word2Vec (Mikolov et al., 2013)

A materialização computacional da Hipótese Distribucional veio em 2013, quando Tomáš Mikolov e sua equipe no Google publicaram a arquitetura **Word2Vec**. A ideia é magistral em sua simplicidade: em vez de dar a cada palavra um vetor canônico de $\mathbb{R}^V$ (One-Hot), atribuímos a cada palavra um **vetor denso** de coordenadas em um espaço de dimensionalidade muito menor (tipicamente $d = 300$). Chamamos esse espaço de **Espaço Latente**.

Se temos um espaço com 300 dimensões (cada uma representando uma característica abstrata como “realeza”, “feminilidade”, “concretude”, “movimento”), podemos mapear cada palavra do vocabulário como um ponto nesse hiper-espaço. A mágica desse mapeamento é que palavras com sentidos similares — ou que frequentemente coocorrem nos mesmos contextos — ficam **geometricamente próximas**.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2025 – 21:26) – O conteúdo pode sofrer alterações.

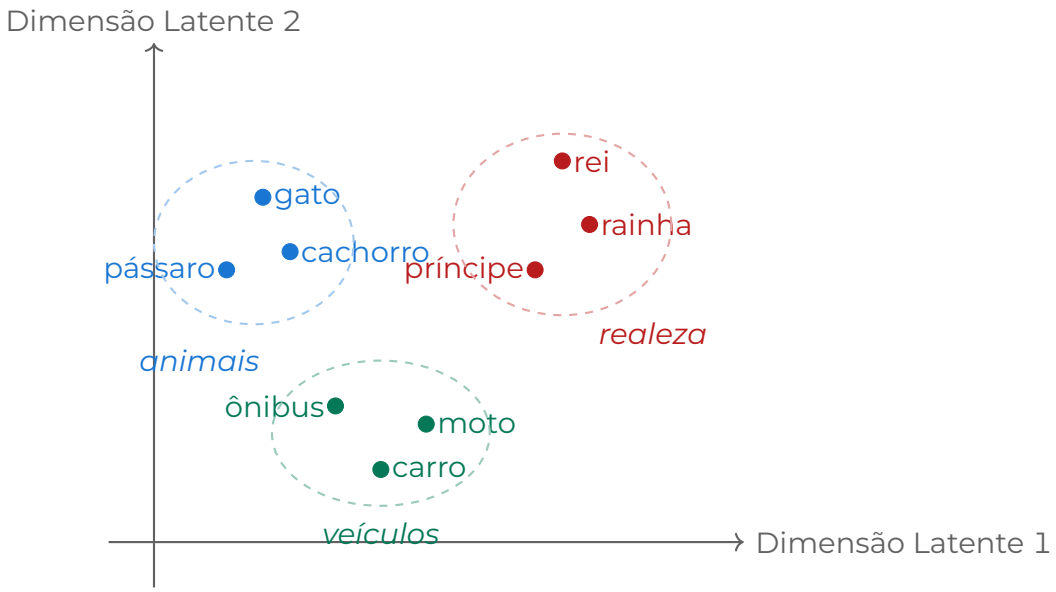


Figura 3: Visualização simplificada (2D) de um espaço vetorial de Embeddings. Palavras semanticamente similares formam *clusters* (agrupamentos) naturais. No espaço real, são 300+ dimensões.

4.1 As Duas Arquiteturas do Word2Vec

O Word2Vec utiliza uma rede neural superficial — com apenas **uma camada oculta** (exatamente como o Perceptron que vocês construíram na Aula 1, mas com uma única camada) — treinada em bilhões de tokens de texto. Existem duas variantes complementares:

- **CBOW (Continuous Bag of Words):** Recebe o contexto (as k palavras vizinhas) como entrada e tenta predizer a palavra central. É mais rápido para vocabulários grandes e funciona bem para palavras frequentes.
- **Skip-Gram:** Recebe a palavra central e tenta predizer o contexto (as palavras vizinhas). Funciona melhor para palavras raras e datasets menores, pois cada par (centro, contexto) gera um exemplo de treinamento.

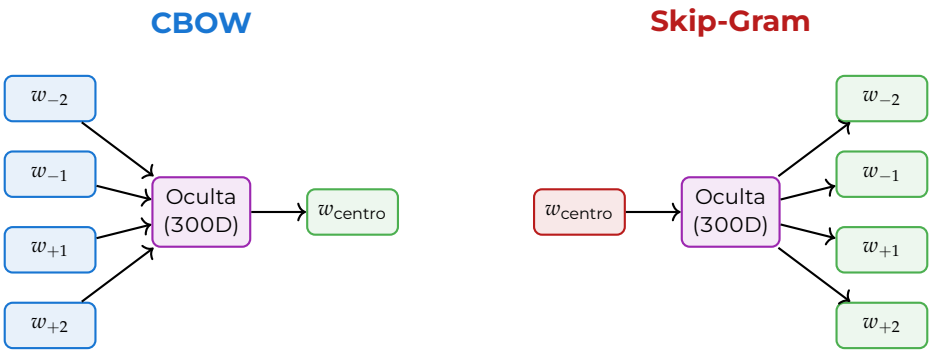


Figura 4: As duas arquiteturas do Word2Vec. **CBOW**: contexto → palavra central. **Skip-Gram**: palavra central → contexto. Em ambos os casos, os pesos da camada oculta *são* os vetores de Embedding.

Em ambos os casos, o ponto crucial é que os **pesos da camada oculta** da rede treinada *são* os vetores de Embedding. O treinamento é, na verdade, um

CAPÍTULO EM REVISÃO (18 de agosto de 2025) – O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

efeito colateral: o objetivo declarado era prever palavras, mas o verdadeiro tesouro são as coordenadas aprendidas na camada oculta. Essa ideia de extrair representações úteis dos pesos internos de uma rede é a essência do **Transfer Learning** — conceito que reaparecerá com força no Arco 3 (Transformers).

5 Matemática Linguística: O Rei e a Rainha

A consequência mais espetacular de tratar semântica como coordenadas algébricas é que operações vetoriais sobre conceitos linguísticos passam a produzir resultados semanticamente coerentes. O exemplo mais célebre da literatura de Embeddings é a analogia vetorial:

$$\vec{\text{rei}} - \vec{\text{homem}} + \vec{\text{mulher}} \approx \vec{\text{rainha}} \tag{1}$$

Vamos destrinchar essa equação com números concretos. Suponha um espaço latente simplificado com apenas 4 dimensões:

Tabela 2: Aritmética vetorial com dimensões interpretáveis. A subtração de “homem” e adição de “mulher” navega no eixo de gênero sem alterar o eixo de realeza.

Palavra	Realeza	Gênero	Idade	Poder
Rei	0.9	0.1	0.7	0.9
Homem	0.0	0.1	0.5	0.3
Mulher	0.0	0.9	0.5	0.3
Rei − Homem + Mulher	0.9	0.9	0.7	0.9
Rainha (real)	0.9	0.9	0.7	0.85

A diferença entre o resultado calculado e o vetor real de “Rainha” é mínima ($\Delta_{\text{Poder}} = 0.05$). O vizinho mais próximo no espaço vetorial é, de fato, “Rainha”. Vocês estão literalmente fazendo **álgebra sobre conceitos humanos** — e a resposta emerge da geometria do espaço, não de regras programadas.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2025 a 2026). O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

▷ **Exemplo: title=Outras Analogias Vetoriais Funcionais**

A aritmética vetorial captura relações semânticas diversas:

- **Países e capitais:** $\vec{\text{Paris}} - \vec{\text{França}} + \vec{\text{Itália}} \approx \vec{\text{Roma}}$
- **Tempos verbais:** $\vec{\text{caminhando}} - \vec{\text{caminhar}} + \vec{\text{nadar}} \approx \vec{\text{nadando}}$
- **Superlativos:** $\vec{\text{melhor}} - \vec{\text{bom}} + \vec{\text{grande}} \approx \vec{\text{maior}}$

Cada uma dessas analogias demonstra que os vetores codificam **relações abstratas** (capital-de, gerúndio-de, superlativo-de) como direções específicas no espaço latente.

6 A Similaridade de Cosseno: Medindo Proximidade Semântica

Se cada palavra é um ponto no $\mathbb{R}^{300}$, como medimos o quão "parecidas" duas palavras são? A resposta é a **Similaridade de Cosseno**, que vocês já conhecem implicitamente da Álgebra Linear: é o cosseno do ângulo θ entre dois vetores, derivado diretamente da definição geométrica do produto escalar.

$$\cos(\theta) = \frac{\mathbf{a} \cdot \mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||} = \frac{\sum_{i=1}^{d} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{d} a_i^2} \cdot \sqrt{\sum_{i=1}^{d} b_i^2}} \tag{2}$$

O valor varia de -1 (vetores diametralmente opostos) a $+1$ (vetores perfeitamente alinhados). Na prática de NLP, valores acima de 0.85 indicam alta similaridade semântica, enquanto valores abaixo de 0.4 indicam assuntos distintos. A grande vantagem do cosseno sobre a distância euclidiana é que ele ignora a **magnitude** dos vetores: um parágrafo inteiro e uma frase curta sobre o mesmo tema terão cosseno alto, mesmo que a euclidiana os considere "distantes" pelo tamanho.

7 O Ecossistema spaCy

Para obter Embeddings de forma ágil e industrial, utilizaremos o **spaCy** (Honnibal & Montani, 2017), um framework de NLP de nível industrial para Python. O spaCy fornece modelos pré-treinados em múltiplas línguas (incluindo Português) que já carregam milhões de tokens processados com seus vetores devidamente mapeados no espaço latente. Os modelos "médios" (`_md`) e "grandes" (`_lg`) incluem vetores de 300 dimensões baseados em arquiteturas inspiradas no Word2Vec.

CAPÍTULO EM REVISÃO (18 de agosto de 2025 – 2026). O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

▶ Prática: title=Extraindo Embeddings com o spaCy

```python
import spacy

# Carregar modelo medio com vetores (300 dimensoes)
nlp = spacy.load("pt_core_news_md")

# Processar textos
doc1 = nlp("O gato dormiu no sofa")
doc2 = nlp("O felino descansou no movel")
doc3 = nlp("O carro bateu no poste")

# Similaridade semantica (0 a 1)
print(doc1.similarity(doc2))  # ~0.85 (alta!)
print(doc1.similarity(doc3))  # ~0.35 (baixa)

# Vetor de uma palavra (300 dimensoes)
palavra = nlp("inteligencia")
print(palavra.vector.shape)   # (300,)
print(palavra.vector[:5])     # [0.12, -0.34, ...]
```

Observe que o `doc1.similarity(doc2)` está calculando internamente a Similaridade de Cosseno entre os vetores médios dos dois documentos. Sem que você perceba, o spaCy executa exatamente a fórmula $\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$ que derivamos na seção anterior. A abstração é elegante, mas o motor é pura álgebra linear — a mesma que vocês dominaram desde o primeiro semestre da graduação.

7.1 Modelos de Embeddings Modernos

Desde o Word2Vec em 2013, a tecnologia de representação vetorial evoluiu significativamente, acompanhando o avanço das arquiteturas de redes neurais:

CAPÍTULO EM REVISÃO (18 de agosto de 2025) - O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

Tabela 3: Evolução cronológica dos modelos de Embedding. Note a progressão de palavras isoladas para frases inteiras e o aumento da dimensionalidade.

Modelo	Ano	Dim.	Granularidade	Diferencial
Word2Vec	2013	300	Palavra	CBOW + Skip-gram
GloVe	2014	300	Palavra	Coocorrência global (Stanford)
FastText	2017	300	Subpalavra	Lida com palavras novas (Facebook)
ELMo	2018	1024	Contexto	Embedding contextualizado (LSTM)
SBERT	2019	768	Frase	Frase inteira → vetor
Ada-002	2022	1536	Frase	API OpenAI (SaaS)
`text-emb-3`	2024	3072	Frase	API OpenAI (último)

8 Conclusão

A capacidade de converter linguagem humana em vetores numéricos densos é o alicerce sobre o qual todo o restante deste Arco 2 será construído. Aprendemos que o One-Hot Encoding desperdiça dimensões e destrói semântica; que a Hipótese Distribucional permite inferir significado a partir de coocorrência; que o Word2Vec materializa essa hipótese em vetores densos de 300 dimensões; e que o spaCy nos dá acesso industrial a esses vetores em Python.

Na próxima aula, utilizaremos esses vetores para construir um **buscador semântico**: um sistema que encontra documentos não pela presença de palavras-chave, mas pelo *significado* da consulta — eliminando os demônios da sinonímia e da polissemia que há décadas atormentam os motores de busca tradicionais.

CAPÍTULO EM REVISÃO (18 de agosto de 2025) - O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

✔ Takeaways

- O **One-Hot Encoding** gera vetores canônicos ortogonais no $\mathbb{R}^V$ — explosão dimensional e zero semântica.
- A **Hipótese Distribucional** (Firth, 1957): o significado de uma palavra emerge dos contextos em que ela aparece.
- O **Word2Vec** (Mikolov, 2013) treina uma rede superficial (CBOW/Skip-Gram) cujos pesos da camada oculta *são* os Embeddings.
- A **Aritmética Vetorial** funciona: $\vec{\text{Rei}} - \vec{\text{Homem}} + \vec{\text{Mulher}} \approx \vec{\text{Rainha}}$.
- A **Similaridade de Cosseno** ($\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$) mede a proximidade semântica independente da magnitude.
- O **spaCy** fornece vetores pré-treinados de 300D em Português, acessíveis com `token.vector`.

✎ Questões: Autoavaliação

1 **[Direta]** Explique por que o produto escalar entre dois vetores One-Hot distintos é sempre zero e qual a consequência disso para a semântica.

2 **[Direta]** Descreva a diferença entre as arquiteturas CBOW e Skip-Gram do Word2Vec, indicando em que cenário cada uma é mais apropriada.

3 **[Direta]** Dado $\vec{A} = [3, 4]$ e $\vec{B} = [6, 8]$, calcule a Similaridade de Cosseno e interprete o resultado geometricamente.

4 **[Reflexiva]** O Word2Vec deduz que "gato" e "cachorro" são similares sem jamais ter visto um gato ou cachorro. Isso significa que a IA "entende" linguagem? Discuta os limites dessa representação puramente estatística.

5 **[Reflexiva]** Os vetores de Embedding capturam vieses sociais presentes nos textos de treinamento (ex: associações de gênero com profissões). Como engenheiro, que responsabilidades éticas surgem ao usar modelos pré-treinados em produção?

CAPÍTULO EM REVISÃO (18 de agosto de 2025 a 20 de setembro de 2026). O conteúdo pode sofrer alterações.

Lab. 09: Extração Vetorial com spaCy

Capítulo em Revisão

Este conteúdo está sendo revisado. Algumas informações podem estar preliminares, conter erros de formatação ou sofrer alterações na versão final.

⋆ Objetivo do Laboratório

Bem-vindo ao **Módulo 2** — "A Revolução da Linguagem e dos Vetores". Nesta atividade, você tangibilizará o conceito de *Embeddings* na prática. Utilizaremos o **spaCy** para extrair representações vetoriais de palavras em português, inspecionar propriedades matemáticas dos vetores, calcular a matriz de similaridade semântica completa, e verificar a aritmética vetorial "Rei − Homem + Mulher ≈ Rainha". Ao final, todo o pipeline será validado pelo CI/CD do GitLab.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2025) - O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

9 Parte 1: O Problema I — Estilo Maratona

Problema I: O Tradutor Conceitual da NeuroLex

Contexto: A *NeuroLex*, uma startup de legaltech, está construindo um buscador semântico para contratos jurídicos. O time de produto precisa de uma **prova de conceito (PoC)** que demonstre: (1) que palavras podem ser convertidas em vetores numéricos, (2) que esses vetores capturam semântica real ("contrato" é mais parecido com "acordo" do que com "bicicleta"), e (3) que operações algébricas sobre conceitos produzem resultados coerentes. Sua missão é construir essa PoC e apresentá-la ao CTO com evidências numéricas irrefutáveis.

Modelo de Entrada: Palavras em português $\rightarrow$ modelo spaCy (`pt_core_news_md`).

Modelo de Saída: Três artefatos obrigatórios:

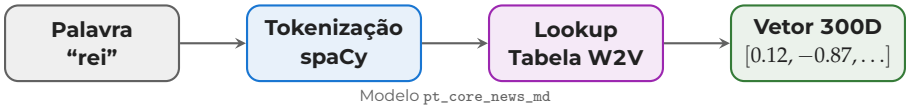
1. Vetor de 300 dimensões para cada palavra (tipo `float32`)
2. Matriz de similaridade semântica completa (5×5 palavras)
3. Resultado da aritmética vetorial: $\vec{\text{Rei}} - \vec{\text{Homem}} + \vec{\text{Mulher}} \approx \vec{\text{Rainha}}$

Critérios de Aprovação:

1. Vetores com exatamente 300 dimensões
2. Palavras do mesmo campo semântico $\rightarrow$ cosseno > 0.5
3. Aritmética vetorial coerente: resultado mais próximo de "rainha" do que de "cadeira"

10 Parte 2: Entendendo o Pipeline — De Palavra a Vetor

Antes de abrir o editor de código, entenda o que vai acontecer. O spaCy é uma biblioteca de NLP que traz **modelos pré-treinados** com vetores de palavras embutidos. Quando você carrega o modelo médio (`pt_core_news_md`), ele traz consigo uma tabela gigante com ~20.000 vetores pré-calculados — um para cada palavra do vocabulário português. O diagrama abaixo resume o fluxo:



Modelo `pt_core_news_md`
20.000 vetores pré-treinados

CAPÍTULO EM REVISÃO (18 de agosto de 2025) — O conteúdo pode sofrer alterações.

Cada palavra do vocabulário tem um endereço fixo na tabela. Quando você pede o vetor de "rei", o spaCy simplesmente faz um *lookup* nessa tabela e retorna os 300 números. Palavras fora do vocabulário recebem vetores nulos.

• Teoria: Qual Modelo Usar? _sm vs _md vs _lg

O spaCy oferece modelos de três tamanhos. Apenas os modelos médio e grande incluem vetores de palavras densos:

Modelo	Vetores?	Dimensões	Quando usar
`pt_core_news_sm`	× Não	—	Tokenização apenas
`pt_core_news_md`	✓ Sim	300D	**Recomendado**
`pt_core_news_lg`	✓ Sim	300D	Vetores mais precisos

Se você usar o modelo `_sm` por engano, o `token.vector` retornará apenas hashes (96D) sem significado semântico. Este é o erro mais comum deste lab.

11 Parte 3: Configuração do Ambiente (5 min)

Abra o repositório da disciplina, navegue para a pasta da Aula 09, e instale o spaCy com o modelo médio. O download do modelo leva alguns segundos:

```
pip install spacy
python -m spacy download pt_core_news_md
```

O comando `python -m spacy download` baixa os pesos do modelo do servidor do spaCy e os instala como um pacote Python. Após a instalação, você pode carregar o modelo em qualquer script com `spacy.load("pt_core_news_md")`.

12 Parte 4: Dissecando a Anatomia de um Embedding (10 min)

A. Extraindo o Vetor de uma Palavra

Crie o arquivo `src/extrator_vetores.py`. Nesta primeira etapa, vamos carregar o modelo e examinar o vetor de uma única palavra para entender sua estrutura interna:

```
import spacy
import numpy as np

# 1. Carrega o modelo em portugues com vetores (Word Embeddings)
print("Carregando o modelo NLP...")
nlp = spacy.load("pt_core_news_md")

```

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2026) - O conteúdo pode sofrer alterações.

```python
# 2. Inspecionando o vetor de uma unica palavra
token = nlp("inteligencia")[0]
vetor = token.vector

print(f"\nPalavra: '{token.text}'")
print(f"Dimensoes do vetor: {vetor.shape}")
print(f"Primeiros 5 valores: {vetor[:5]}")
print(f"Norma L2: {np.linalg.norm(vetor):.4f}")
print(f"Tipo: {vetor.dtype}")
```

O que cada linha revela:

- **shape = (300,)**: Cada palavra é um ponto no espaço de 300 dimensões. Você pode imaginar como uma "coordenada" extremamente detalhada no universo semântico.
- **dtype = float32**: Cada dimensão é um número decimal de precisão simples — a mesma precisão usada em GPUs para treinar redes neurais.
- **Norma L2**: O "comprimento" do vetor. Vetores de palavras comuns costumam ter norma entre 5 e 10.

B. Verificando a Densidade

Um vetor esparso teria muitos zeros (como one-hot encoding). Um embedding denso tem quase 100% de valores não-zero — isso é o que torna os embeddings tão poderosos:

```python
# 3. O vetor e denso (nao-esparso): quase nenhum valor e zero
valores_nao_zero = np.count_nonzero(vetor)
print(f"Valores nao-zero: {valores_nao_zero}/{vetor.shape[0]} "
      f"({valores_nao_zero/vetor.shape[0]:.0%})")
```

⚠ Importante: Ponto de Verificação

A saída deve mostrar **300 dimensões**, tipo `float32`, e quase 100% de valores não-zero. Se aparecer `(96,)` ou valores todos zero, você provavelmente está usando o modelo `_sm`. Reinstale o `_md`.

13 Parte 5: Construindo a Matriz de Similaridade (12 min)

Agora que você sabe que cada palavra é um vetor de 300 números, vamos comparar palavras entre si. O spaCy tem um método `.similarity()` embutido que calcula o cosseno entre dois vetores:

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2026) - O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

```python
# 4. Palavras de interesse para comparar
palavras = ["rei", "rainha", "homem", "mulher", "cadeira"]

# 5. Processa os tokens para obter representacao computacional
tokens = nlp(" ".join(palavras))

# 6. Matriz de Similaridade (cosine similarity embutido)
print("\n=== MATRIZ DE SIMILARIDADE SEMANTICA ===")
print(f"{'':>10}", end="")
for t in tokens:
    print(f"{t.text:>10}", end="")
print()

for t1 in tokens:
    print(f"{t1.text:>10}", end="")
    for t2 in tokens:
        sim = t1.similarity(t2)
        print(f"{sim:>10.3f}", end="")
    print()
```

A saída será uma tabela 5×5 simétrica. Dedique alguns minutos para analisar os valores:

• Teoria: Interpretando a Matriz

Observe os padrões que emergem dos números:

- **Diagonal:** Todos os valores são 1.0 (cada palavra é idêntica a si mesma — cosseno de um vetor consigo mesmo é sempre 1).
- **Rei $\leftrightarrow$ Rainha:** Similaridade alta (~ 0.7–0.8) por compartilharem o campo semântico de "realeza".
- **Rei $\leftrightarrow$ Cadeira:** Similaridade baixa (~ 0.2–0.3) por pertencerem a campos completamente distintos.
- **Homem $\leftrightarrow$ Mulher:** Similaridade moderada-alta por compartilharem o campo "ser humano".

Esses números **não foram programados manualmente** — eles emergem do treinamento sobre milhões de textos jornalísticos em português. O modelo "aprendeu" que reis e rainhas aparecem em contextos parecidos.

14 Parte 6: Aritmética Vetorial — A Prova Definitiva (15 min)

CAPÍTULO EM REVISÃO (18 de agosto de 2025 – 2026). O conteúdo pode sofrer alterações.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

A. A Ideia por Trás da Aritmética

Esta é a parte mais fascinante dos Embeddings. A hipótese é: se os vetores capturam significado, então **operações algébricas sobre vetores devem produzir resultados semanticamente coerentes**. A analogia mais famosa é:

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

A dimensão "gênero" é trocada, mantendo a dimensão "realeza"

A intuição é: subtraindo "Homem" de "Rei", removemos a componente "masculino" e ficamos com a "essência de realeza". Somando "Mulher", adicionamos a componente "feminino". O resultado deve apontar na direção de "Rainha".

B. Implementando e Medindo

Para verificar, precisamos de uma função que meça o ângulo entre dois vetores. Implemente a Similaridade de Cosseno usando apenas NumPy:

```python
# 7. Aritmetica Vetorial: Rei - Homem + Mulher = ???
vetor_rei = nlp("rei").vector
vetor_homem = nlp("homem").vector
vetor_mulher = nlp("mulher").vector
vetor_rainha = nlp("rainha").vector

# Operacao algebrica sobre conceitos
resultado = vetor_rei - vetor_homem + vetor_mulher

# 8. Medindo distancia do resultado ao alvo esperado
from numpy.linalg import norm

def cosseno(a, b):
    """Calcula a Similaridade de Cosseno entre dois vetores."""
    return np.dot(a, b) / (norm(a) * norm(b))

sim_rainha = cosseno(resultado, vetor_rainha)
sim_rei = cosseno(resultado, vetor_rei)
sim_cadeira = cosseno(resultado, nlp("cadeira").vector)

print("\n=== ARITMETICA VETORIAL ===")
print(f"Rei - Homem + Mulher ~= ???")
print(f"  Similaridade com 'rainha':  {sim_rainha:.4f}")
print(f"  Similaridade com 'rei':     {sim_rei:.4f}")
print(f"  Similaridade com 'cadeira': {sim_cadeira:.4f}")

# 9. Verificando qual palavra e mais proxima
if sim_rainha > sim_cadeira:
    print("\nSUCESSO: 'rainha' e o vizinho mais proximo!")
else:
    print("\nAVISO: O modelo nao captou a analogia perfeitamente.")
```

CAPÍTULO EM REVISÃO (18 de agosto de 2025) - O conteúdo pode sofrer alterações.

O valor de `sim_rainha` deve ser significativamente maior que `sim_cadeira`. Se "rainha" tem score 0.85 e "cadeira" tem 0.30, a prova está feita: o espaço vetorial realmente codifica relações semânticas que sobrevivem a operações algébricas.

▷ Exemplo: title=Desafio: Teste Mais Analogias

Adicione pelo menos 2 analogias extras ao seu código e verifique os resultados:

- **Capital:** $\vec{\text{Paris}} - \vec{\text{França}} + \vec{\text{Itália}} \approx ?$ (Roma?)
- **Gênero:** $\vec{\text{irmão}} - \vec{\text{homem}} + \vec{\text{mulher}} \approx ?$ (irmã?)

Nem todas as analogias funcionam perfeitamente com o modelo médio — modelos maiores (ou Word2Vec treinados em corpus gigantes) são mais precisos.

15 Parte 7: Garantia de Qualidade — Testes CI/CD (8 min)

Os testes automatizados validam as três propriedades essenciais que demonstramos hoje. Crie o arquivo `tests/test_extrator.py`:

```python
import numpy as np
import spacy

nlp = spacy.load("pt_core_news_md")

def cosseno(a, b):
    return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))

def test_vetor_300_dimensoes():
    """Verifica que o modelo retorna vetores de 300D."""
    token = nlp("gato")[0]
    assert token.vector.shape == (300,), \
        f"Shape errado: {token.vector.shape}"

def test_similaridade_semantica():
    """Palavras do mesmo campo devem ter cosseno > 0.5."""
    sim = nlp("gato").similarity(nlp("cachorro"))
    assert sim > 0.5, f"Similaridade muito baixa: {sim:.3f}"

def test_aritmetica_vetorial():
    """Rei - Homem + Mulher deve ser mais proximo de Rainha
    do que de Cadeira."""
    resultado = (nlp("rei").vector - nlp("homem").vector
                 + nlp("mulher").vector)
```

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2026) — O conteúdo pode sofrer alterações.

```
    sim_rainha = cosseno(resultado, nlp("rainha").vector)
    sim_cadeira = cosseno(resultado, nlp("cadeira").vector)
    assert sim_rainha > sim_cadeira, \
        f"Analogia falhou: rainha={sim_rainha:.3f} < cadeira={
            sim_cadeira:.3f}"
```

Cada teste mapeia diretamente para um dos três critérios de aprovação do Problema I: vetores de 300D, similaridade semântica coerente e aritmética vetorial funcional. Execute `pytest tests/test_extrator.py -v` e confirme os **3 passed** em verde.

16 Parte 8: Commit e Push

```
git add src/extrator_vetores.py tests/test_extrator.py
git commit -m "Lab 09: Embeddings spaCy + aritmética vetorial + CI/CD
    "
git push origin main
```

Conclusão: O que Você Construiu Hoje

✔ Takeaways

- Você extraiu **vetores de 300 dimensões** de palavras em Português usando o spaCy — a mesma tecnologia usada em pipelines NLP industriais.
- Construiu uma **Matriz de Similaridade Semântica** completa e verificou que palavras do mesmo campo semântico têm cosseno alto.
- Provou empiricamente a **Aritmética Vetorial**: $\vec{\text{Rei}} - \vec{\text{Homem}} + \vec{\text{Mulher}} \approx \vec{\text{Rainha}}$ — a IA navega conceitos como coordenadas num mapa.
- Os 3 testes CI/CD garantem: vetores de 300D, similaridade semântica coerente e aritmética vetorial funcional.

◇ Informação

Gancho para a Próxima Aula: Você agora sabe converter qualquer texto em vetores e calcular similaridade. Mas e se a busca retornar um texto que *contém* a palavra "coração" mas fala de **metáfora emocional** — não de cardiologia? Na próxima aula, formalizaremos a **Busca Semântica** com cosseno e veremos por que ela supera a busca léxica. Você implementará um **buscador semântico completo** do zero, usando o cosseno na mão com NumPy.

Acesse a versão completa em: https://alessiojr.com/disciplines/ia2/conteudo/00-Livro/

CAPÍTULO EM REVISÃO (18 de agosto de 2025 – 2026). O conteúdo pode sofrer alterações.