Inteligência Artificial 2

Aula 10: Semântica e Similaridade Numérica
Da Busca Léxica à Revolução do RAG

Prof. Aléssio Miranda Júnior
alessio@cefetmg.br

CEFET-MG - Campus Timoteo
Dep. Engenharia de Computação

Setembro - 2026/2

Onde Paramos?

Na aula anterior, vocês dominaram o **vocabulário vetorial** da IA:

- **One-Hot Encoding** falha: dimensionalidade explosiva e zero semântica.
- **Hipótese Distribucional** de Firth: "o significado emerge do contexto".
- **Word2Vec**: CBOW e Skip-Gram criam vetores densos de dimensão fixa.
- **Aritmética Vetorial**: Rei $-$ Homem $+$ Mulher $\approx$ Rainha.
- **Similaridade de Cosseno**: $\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$.

A Pergunta de Hoje

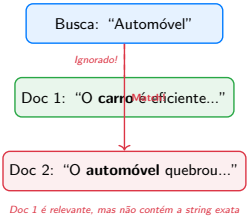
Agora que sabemos converter texto em vetores,
como **buscamos** o documento certo?

Objetivos da Sessão

- **Busca Léxica vs. Semântica:** Entender as falhas fundamentais do Ctrl+F e por que strings não bastam.
- **Sinonímia e Polissemia:** Os dois demônios que inviabilizam a busca por palavras-chave.
- **Busca Semântica:** O novo paradigma — buscar por *intenção* ao invés de *caracteres*.
- **Distância Euclidiana vs. Cosseno:** Por que a magnitude engana e o ângulo salva.
- **Normalização L2:** A projeção na hiperesfera unitária.
- **RAG (Retrieval-Augmented Generation):** A arquitetura que dominou a IA corporativa.
- **Busca Híbrida:** Semântica + Léxica para cenários reais.

O Limite do Ctrl+F

- Por décadas, a recuperação de informação dependeu da **Busca Léxica** (palavras-chave).
- SQL: `LIKE '%termo%'`. ElasticSearch: Índices Invertidos.
- **Suposição falha**: "A relevância é proporcional à frequência da palavra-chave pesquisada."



O Problema da Sinonímia e Polissemia

Sinonímia

Múltiplas palavras para o mesmo conceito.

Buscar "Automóvel" ignora documentos que usam "Carro", "Veículo" ou "Viatura".

Resultado: Recall baixíssimo.

Polissemia

Múltiplos conceitos para a mesma palavra.

Buscar "Banco" retorna: instituição financeira, assento de madeira, banco de dados, banco de areia...

Resultado: Precisão desastrosa.

A Lição

A busca léxica é **cega para o significado**. Ela compara *caracteres*, não *conceitos*. Para a IA "entender" a pergunta, precisamos operar no espaço vetorial.

Um Exemplo Real na Saúde

- Sistema de triagem hospitalar. Médico pesquisa: *"Pacientes com problemas no coração"*.

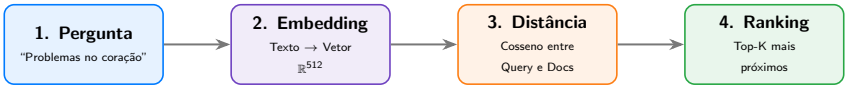
#	Prontuário	Léxica	Semântica
1	"Paciente com **anomalia cardíaca** grave."	Miss	Hit
2	"Dor na região **torácica** com suspeita de **infarto**."	Miss	Hit
3	"João teve problemas no trabalho e **quebrou o coração** após separação."	Hit!	Baixo

O Resultado da Busca Léxica

O buscador retorna o prontuário **3** (match de "problemas" + "coração") e ignora os **casos clínicos reais**. Um erro potencialmente fatal na saúde.

A Mudança de Paradigma: Busca Semântica

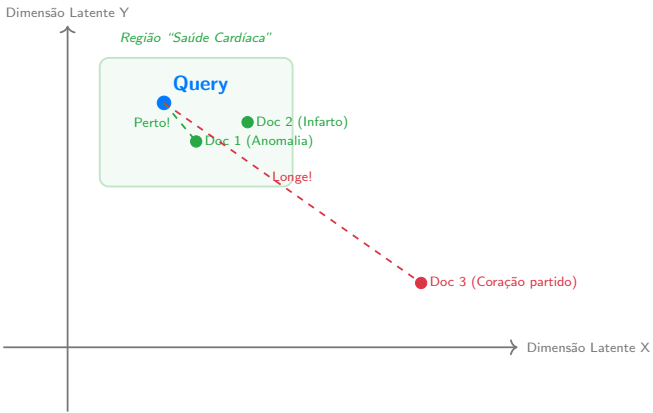
- Abandonar as *strings* e focar na **intenção** (o sentido por trás da pergunta).
- O novo fluxo é puramente **geométrico**:



A Grande Sacada

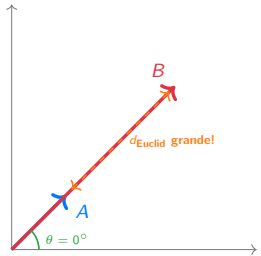
"Problemas no coração" e "Anomalia cardíaca" são codificados como tensores **quase idênticos**, apesar de não compartilharem nenhuma palavra em comum. O buscador é **agnóstico à ortografia**.

Pergunta e Documento no Espaço Latente



Como Medir a Distância de um Sentido?

- Se Query e Documento são vetores, precisamos de uma **métrica de distância**.
- **Distância Euclidiana:** $d = \sqrt{\sum (A_i - B_i)^2}$
- **Falha:** Textos longos geram vetores de maior magnitude. Um parágrafo e uma frase sobre o mesmo tema ficam "distantes" euclidianamente — apenas pelo tamanho.



Mesma direção, tamanhos diferentes

A Lição

A distância Euclidiana penaliza a **magnitude** (comprimento do vetor), não a **direção** (o significado). Precisamos de algo melhor.

A Solução Definitiva: Similaridade de Cosseno

- Ignora a magnitude e foca exclusivamente na **direção** do vetor.
- Mede o **ângulo** θ entre os dois vetores a partir da origem:

Derivação a partir do Produto Escalar

Da definição geométrica do dot product (Álgebra Linear):

$$A \cdot B = \|A\|\|B\| \cos(\theta) \quad \Rightarrow \quad \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Equação Expandida

$$\text{Cosine Similarity} = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Exemplo Numérico: Cosseno Passo-a-Passo

Dados:
Query $A = [3, 4]$ ("problemas cardíacos")
Doc $B = [6, 8]$ ("anomalia cardíaca")
Doc $C = [4, -3]$ ("coração partido")

$\cos(A, B)$**:**
$A \cdot B = 3 \times 6 + 4 \times 8 = 50$
$\|A\| = \sqrt{9 + 16} = 5$
$\|B\| = \sqrt{36 + 64} = 10$
$\cos\theta = \frac{50}{5 \times 10} = \mathbf{1.0}$

$\cos(A, C)$**:**
$A \cdot C = 3 \times 4 + 4 \times (-3) = 0$
$\|C\| = \sqrt{16 + 9} = 5$
$\cos\theta = \frac{0}{5 \times 5} = \mathbf{0.0}$

Interpretação

B aponta na **mesma direção** que A ($\cos = 1$).
C é **ortogonal** a A ($\cos = 0$).
Apesar de B ser mais "longo" que A, o cosseno não se importa!

Interpretação Trigonométrica: 1, 0 e -1

Valor	Ângulo	Significado
$+1.0$	$0°$	Vetores apontam na mesma direção. Semântica idêntica.
$+0.5$	$60°$	Relacionamento parcial. Alguma sobreposição semântica.
0.0	$90°$	Ortogonais. Sem qualquer correlação semântica.
-0.5	$120°$	Sentidos parcialmente opostos (raro em NLP).
-1.0	$180°$	Diametralmente opostos. Contradição direta.

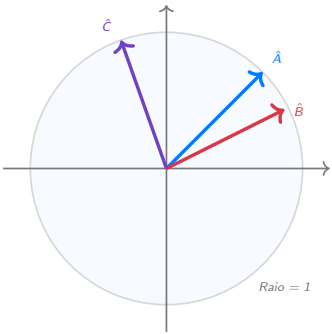
Na Prática

Em NLP, valores acima de 0.85 indicam alta similaridade semântica. Entre 0.6 e 0.85, há relação

O Efeito da Normalização L2

- Ao dividir pelo produto das normas, transformamos todos os vetores para **comprimento = 1**.
- As pontas dos vetores repousam sobre uma **Hiperesfera Unitária**.
- Após normalização, o **produto escalar** entre vetores unitários *já é* o cosseno:

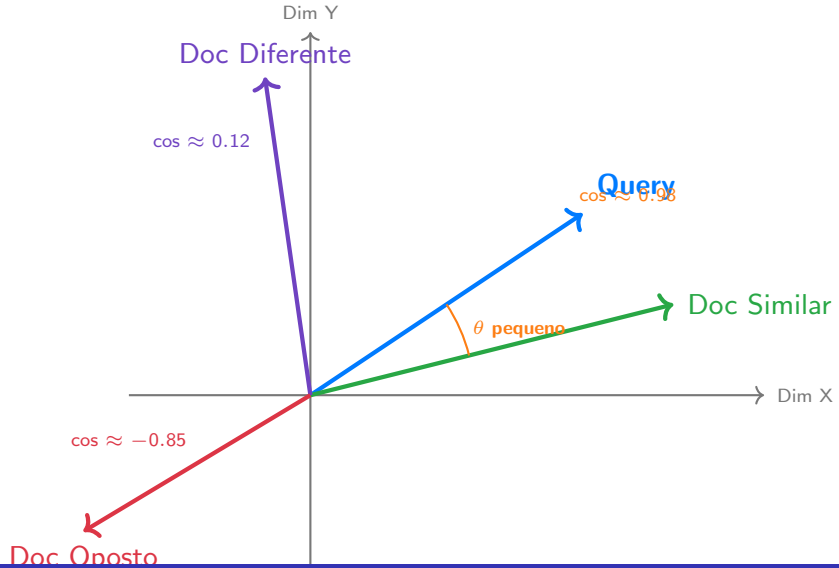
$$\hat{A} = \frac{A}{\|A\|}, \quad \hat{A} \cdot \hat{B} = \cos\theta$$



Implicação Prática

Muitos Vector DBs pré-normalizam os vetores no `INSERT`. Assim, a busca se resume a um simples `dot product` — a operação mais rápida em hardware GPU.

Representação Visual Completa



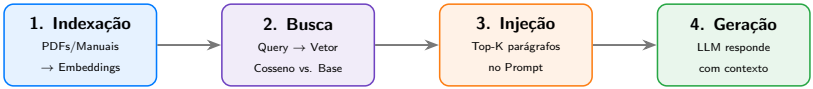
O Coração da IA Corporativa: RAG

- **RAG (Retrieval-Augmented Generation)** é o padrão ouro da indústria para chatbots corporativos.
- A IA (LLM) por si só "alucina" fatos e desconhece dados internos da empresa.
- RAG resolve isso: **busca semântica** em documentos privados + injeção de contexto no prompt do LLM.

Por que RAG existe?

Re-treinar um LLM com dados corporativos custa **milhões de dólares** e semanas. RAG atinge resultados comparáveis por uma fração do custo, atualizando apenas os documentos indexados.

O Ciclo de Vida do RAG



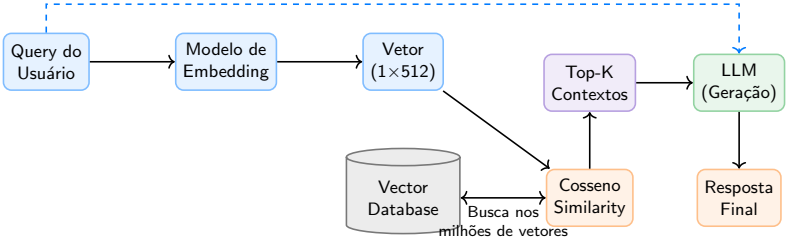
Exemplo

Usuário: "Como aprovar reembolso?"
RAG busca Top-3 parágrafos (cosseno: 0.92, 0.89, 0.88) e injeta no prompt do LLM.

O Resultado

O LLM responde baseado **exclusivamente** nos documentos reais da empresa — sem alucinar procedimentos inventados.

Arquitetura de Recuperação Vetorial



Bancos de Dados Vetoriais: O Gargalo

- Um `for` em Python: Query vs. todos os docs? Funciona para 100 documentos.
- E se a empresa tiver **500 milhões de parágrafos**? $500M \times 512D$ multiplicações por pergunta.
- É por isso que existem **Vector Databases** (ChromaDB, Pinecone, Milvus, FAISS).

Docs	Operações/Query
1.000	512K
100.000	51M
10M	5,1B
500M	256B

Tempo

256 bilhões de multiplicações para **uma** pergunta de chat. Inviável.

RAG vs Busca Híbrida (Hybrid Search)

O Problema

Busca Semântica é péssima para "nomes exatos" e "códigos".

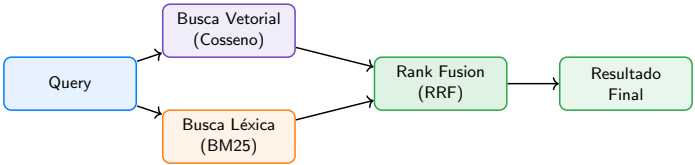
Pesquisar "Peça AX-990" falha se não houver contexto semântico.

A Solução: Hybrid Search

Duas buscas paralelas:

1. Vetorial (Semântica)
2. Léxica (BM25/Keyword)

Fusão via *Reciprocal Rank Fusion (RRF)*.



Implementação: Cosseno em NumPy

A Similaridade de Cosseno na base — sem abstrações:

```python
import numpy as np

def cosine_similarity(vec_a, vec_b):
    # Produto Escalar (Numerador)
    dot_product = np.dot(vec_a, vec_b)

    # Normas Vetoriais L2 (Denominador)
    norm_a = np.linalg.norm(vec_a)
    norm_b = np.linalg.norm(vec_b)

    # Previne divisao por zero (vetor nulo)
    if norm_a == 0 or norm_b == 0:
        return 0.0

    # Calculo final do Cosseno
    return dot_product / (norm_a * norm_b)
```

O Laço de Busca: Varrendo o Banco

O algoritmo k-NN Exaustivo (força bruta) de um RAG simples:

```
query_vetor = get_embedding("Como pedir ferias?")

melhor_score = -1.0
melhor_documento = None

for documento in banco_de_dados_corporativo:
    # Obtem a representacao vetorial do documento atual
    doc_vetor = documento.vetor

    # Calcula quao alinhados eles estao
    score_cosseno = cosine_similarity(query_vetor, doc_vetor)

    # Atualiza se encontrou um vizinho mais proximo
    if score_cosseno > melhor_score:
        melhor_score = score_cosseno
        melhor_documento = documento
```

A Matemática da Normalização L2

- Quando aplicamos a norma L2, estamos forçando o vetor a ter comprimento 1.
- Isso remove a "força" (frequência) das palavras e deixa apenas a "direção" (tema).

Demonstração

Seja $v = [3, 4]$. A norma L2 é $\|v\| = \sqrt{3^2 + 4^2} = 5$.
Vetor normalizado $\hat{v} = [3/5, 4/5] = [0.6, 0.8]$.
Nova norma: $\sqrt{0.6^2 + 0.8^2} = \sqrt{0.36 + 0.64} = 1.0$.

RAG: O Paradoxo do Contexto

- Por que não jogar **todos** os documentos da empresa no prompt do LLM?
- **1. Limite de Tokens:** Modelos aceitam 128k tokens, não gigabytes.
- **2. Lost in the Middle:** Se você mandar 100 páginas de texto, a IA se "esquece" do que estava no meio e prioriza o início e o fim.

Solução

O RAG filtra o ruído! Só os Top-K (geralmente 3 a 5 fragmentos) com **maior similaridade de cosseno** entram no prompt. Qualidade $>$ Quantidade.

Reciprocal Rank Fusion (RRF)

- Como juntamos Busca Semântica (Cosseno) e Léxica (BM25)?
- Se o Doc A é 1° lugar na semântica e 10° na léxica...
- O RRF calcula um novo score somando os inversos dos ranks:

$$\text{Score}_{\text{RRF}} = \frac{1}{k + \text{Rank}_{\text{Semântico}}} + \frac{1}{k + \text{Rank}_{\text{Léxico}}}$$

O Poder do RRF

Não importa a escala original (0 a 1 do Cosseno vs. 0 a ∞ do BM25). O rank é agnóstico. O RRF eleva documentos que são consistentemente bons em ambos os métodos.

Recapitulação — O que Levar Desta Aula

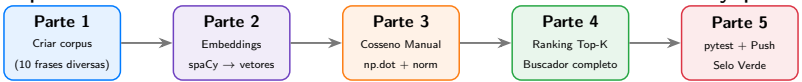
1. **Busca Léxica falha**: Sinonímia e polissemia destroem relevância.
2. **Busca Semântica**: Opera no espaço vetorial, buscando por *intenção*.
3. **Euclidiana vs. Cosseno**: Magnitude engana; o **ângulo** captura significado.
4. **Normalização L2**: Projeta vetores na hiperesfera unitária (comprimento $= 1$).
5. **Cosseno na prática**: $\frac{A \cdot B}{\|A\|\|B\|}$, valor de -1 a 1.
6. **RAG**: Indexação $\rightarrow$ Busca $\rightarrow$ Injeção $\rightarrow$ Geração.
7. **Hybrid Search**: Semântica $+$ Léxica para cenários industriais.

Próxima Aula

O loop `for` sobre todos os documentos é $O(N)$ — inviável para milhões de vetores. Na próxima aula, **Vector Databases** e o algoritmo HNSW que faz busca em $O(\log N)$.

Roteiro do Laboratório 10

Missão: Implementar um buscador semântico "na mão" com NumPy puro + spaCy.



O Diferencial

Nenhuma abstração! Vocês implementarão o **núcleo algébrico** de um buscador semântico — cosseno na mão, ranking na mão — para entender exatamente o que o ChromaDB faz por baixo.

Lab Passo 1: O Corpus

- O primeiro passo será montar um corpus "na mão".
- Uma lista Python contendo 10 frases de domínios diferentes (ex: saúde, tecnologia, esportes).

```
corpus = [
    "A anomalia cardiaca exige tratamento rapido",
    "O novo processador atinge frequencias extremas",
    "A selecao de futebol venceu o campeonato mundial",
    # ...
]
```

Lab Passo 2: Extração em Lote

- Vocês passarão todo o corpus pelo spaCy para gerar a Base de Vetores.
- Cada documento será representado por um vetor de 300 dimensões (se usarem o modelo "md").

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

# Extrair vetores para o banco de dados simulado
banco_vetores = [nlp(doc).vector for doc in corpus]
```

Lab Passo 3: A Consulta e o Cosseno

- O usuário fará uma busca, ex: "problemas no coracao".
- A consulta deve ser vetorizada exatamente pelo **mesmo modelo** (spaCy).

```
query_vec = nlp("medico cardiologista").vector

# Funcao de cosseno que voces criaram na Aula 09
def cosseno(a, b):
    return np.dot(a, b) / (np.linalg.norm(a) * np.linalg.norm(b))
```

Lab Passo 4: O Motor de Busca Simulado

- Vocês varrerão o `banco_vetores` e calcularão o cosseno de cada um com a `query_vec`.
- Guardarão as pontuações e ordenarão de forma decrescente para achar o Top-K!

```
resultados = []
for idx, doc_vec in enumerate(banco_vetores):
    score = cosseno(query_vec, doc_vec)
    resultados.append((score, corpus[idx]))

resultados.sort(reverse=True, key=lambda x: x[0])
print(resultados[:3])  # Top-3!
```

Lab Passo 5: Testes e Avaliação

- O pipeline de CI irá testar a sua implementação.
- Ele fará buscas com termos que não estão no texto (ex: "CPU" para tentar achar o processador).
- Se o seu código usar Busca Léxica, ele falhará; se a Similaridade de Cosseno estiver bem implementada, a semântica resolverá e vocês ganharão o Selo Verde.

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: Vector DBs e ChromaDB

100% da Aula 10 Concluída!