Lab. 10: Buscador Semântico com Cosseno

Aléssio Miranda Júnior

Setembro - 2026/2

Lab. 10: Buscador Semântico com Cosseno

⋆ Objetivo do Laboratório

No Lab 09, você provou que palavras viram vetores e que vetores capturam semântica. Agora, você implementará um **motor de busca semântica do zero**, sem nenhuma abstração. Usando spaCy para gerar vetores e NumPy para álgebra linear pura, você construirá manualmente a função de Similaridade de Cosseno, criará um corpus de frases diversas, implementará o ranking Top-K completo, e provará que a busca semântica encontra documentos relevantes **mesmo sem nenhuma palavra-chave em comum**.

1 Parte 1: O Problema J — Estilo Maratona

Problema J: O Oráculo de Contratos da NeuroLex

Contexto: A NeuroLex (do Lab 09) gostou da sua PoC com vetores e agora quer o protótipo seguinte: um **buscador semântico** que receba uma query em linguagem natural e retorne os 3 documentos mais relevantes de um corpus de 10 cláusulas contratuais. O CTO impôs uma restrição educativa: *"Quero que o time implemente o cosseno na mão, sem importar nenhuma função pronta. Só NumPy puro."*
O diferencial é que as queries não compartilham **nenhuma palavra-chave** com os documentos — a busca funciona por *significado*, não por *casamento de strings*.

Modelo de Entrada: Query em texto livre → vetorização spaCy → cosseno manual.

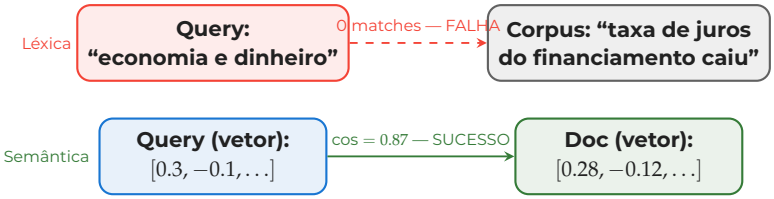
Modelo de Saída: Ranking Top-3 com scores de similaridade.

Critérios de Aprovação:

1. Cosseno de vetores idênticos $= 1.0$ (autoverificação)
2. Cosseno de vetores ortogonais $= 0.0$
3. Query "economia e dinheiro" retorna documentos financeiros no Top-2

2 Parte 2: Entendendo o Problema — Léxica vs Semântica

Para entender por que este lab é necessário, considere o seguinte cenário: um usuário digita "economia e dinheiro" no buscador. O corpus tem a frase "A taxa de juros do financiamento imobiliário caiu ontem". Um buscador léxico (Ctrl+F) não encontraria essa frase, pois nenhuma palavra da query aparece no documento. Já o buscador semântico entende que "juros" e "financiamento" estão no mesmo campo semântico que "economia" e "dinheiro".



A busca léxica compara **caracteres**; a busca semântica compara **vetores**. E como vimos no Lab 09, vetores capturam o *significado*, não a forma escrita.

3 Parte 3: Montando o Corpus (5 min)

Crie o arquivo `src/buscador_cosseno.py`. O primeiro passo é definir um "banco de dados" de frases diversas. Observe que o corpus foi construído intencionalmente com **zero sobreposição de palavras-chave** com as queries que testaremos depois:

```
import spacy
import numpy as np
from numpy.linalg import norm

# 1. Carrega modelo em portugues (com vetores embutidos)
nlp = spacy.load("pt_core_news_md")

# 2. Nosso "Banco de Dados" --- NENHUMA frase contem keywords!
banco_documentos = [
    "O felino descansava tranquilamente no sofa da sala.",
    "A taxa de juros do financiamento imobiliario caiu ontem.",
    "O presidente sancionou a nova lei orcamentaria hoje.",
    "Receitas com chocolate precisam de bastante acucar.",
    "O medico diagnosticou uma anomalia cardiaca grave.",
    "A vacina da gripe estara disponivel na UBS amanha.",
    "O professor explicou a multiplicacao de matrizes.",
    "O cachorro brincava com as criancas no parque.",
    "Investidores apostam na queda do dolar este mes.",
    "A inteligencia artificial transformou a industria.",
]
```

Leia as 10 frases com atenção. Note que nenhuma contém as palavras "economia", "dinheiro", "saúde" ou "animal" — as queries que usaremos adiante. Se o buscador funcionar, será prova irrefutável de que a busca é por *significado*, não por strings.

4 Parte 4: Implementando o Cosseno na Mão (10 min)

A. A Fórmula

A Similaridade de Cosseno mede o ângulo θ entre dois vetores. Se ambos apontam na mesma direção, $\cos(\theta) = 1$; se são ortogonais, $\cos(\theta) = 0$:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}|| \times ||\mathbf{B}||} = \frac{\sum a_i \cdot b_i}{\sqrt{\sum a_i^2} \times \sqrt{\sum b_i^2}}$$

O numerador é o **produto escalar** (`np.dot`) que vocês já viram em Álgebra Linear; o denominador é o produto das **normas L2** (`np.linalg.norm`).

B. Traduzindo para Código

```
# 3. Funcao de Similaridade de Cosseno --- SEM ABSTRACOES!
def cosine_similarity(vec_a, vec_b):
    """
    Calcula a Similaridade de Cosseno entre dois vetores.
    Formula: cos(theta) = (A . B) / (||A|| * ||B||)
    """
    dot_product = np.dot(vec_a, vec_b)   # Numerador
    norm_a = norm(vec_a)                  # ||A||
    norm_b = norm(vec_b)                  # ||B||
    if norm_a == 0 or norm_b == 0:        # Previne /0
        return 0.0
    return dot_product / (norm_a * norm_b)
```

A guard clause `if norm_a == 0` previne divisão por zero caso alguma palavra esteja fora do vocabulário (vetor nulo). Na prática, isso raramente acontece com o modelo `_md`.

△ Importante: Conexão com Labs Anteriores

O `np.dot()` é o mesmo produto escalar do Perceptron (Lab 01) e da MLP (Lab 02). A única novidade aqui é dividi-lo pelas normas para normalizar o resultado entre $[-1, 1]$. Você está reutilizando a mesma matemática em contextos cada vez mais sofisticados.

5 Parte 5: Ranking Top-K — Buscador Completo (15 min)

A. O Motor de Busca

Com o cosseno implementado, agora construímos o motor de busca que varre todo o banco e retorna os K documentos mais relevantes. O algoritmo é simples: vetorizar a query, calcular o cosseno contra cada documento, ordenar em ordem decrescente e retornar os K primeiros:

```
# 4. Motor de Busca Semantica com Ranking Top-K
def buscar_semanticamente(query_texto, banco, top_k=3):
    """
    Busca os top_k documentos mais semanticamente
    similares a query.
    """
    vetor_query = nlp(query_texto).vector
    resultados = []
    for idx, doc_texto in enumerate(banco):
        vetor_doc = nlp(doc_texto).vector
        score = cosine_similarity(vetor_query, vetor_doc)
        resultados.append((score, idx, doc_texto))
```

```
    resultados.sort(key=lambda x: x[0], reverse=True)
    return resultados[:top_k]
```

B. Testando com Queries Diversas

Agora vem o momento da verdade. Execute as 4 queries abaixo e observe quais documentos são retornados:

```
# 5. Testando com queries diversas
queries_teste = [
    "Me fale sobre economia e dinheiro.",
    "Problemas de saude e doencas.",
    "Animais domesticos e seus habitos.",
    "Tecnologia e computacao moderna.",
]

if __name__ == "__main__":
    for query in queries_teste:
        print(f"\n{'='*60}")
        print(f"QUERY: '{query}'")
        print(f"{'='*60}")
        top = buscar_semanticamente(query, banco_documentos)
        for rank, (score, idx, texto) in enumerate(top, 1):
            print(f"  #{rank} [Score: {score:.4f}] {texto}")
```

Observe: “economia e dinheiro” deve trazer “taxa de juros” e “investidores” como Top-2, mesmo sem palavra em comum. “Problemas de saúde” deve retornar “anomalia cardíaca” e “vacina da gripe”. Se “chocolate” aparecer no Top-3 de “economia”, algo está errado.

6 Parte 6: Comparando Léxica vs. Semântica (10 min)

Para completar a PoC, implemente um buscador léxico (equivalente ao Ctrl+F) e compare lado a lado com o semântico:

```
# 6. Busca Lexica (baseline ingenue) para comparacao
def buscar_lexicamente(query_texto, banco):
    """Busca documentos que contenham ALGUMA palavra da query."""
    palavras_query = set(query_texto.lower().split())
    resultados = []
    for doc in banco:
        palavras_doc = set(doc.lower().split())
        intersecao = palavras_query & palavras_doc
        resultados.append((len(intersecao), doc))
    resultados.sort(key=lambda x: x[0], reverse=True)
    return resultados[:3]
```

```
# 7. Comparacao direta
if __name__ == "__main__":
    query_teste = "Me fale sobre economia e dinheiro."
    print("\n=== COMPARACAO: LEXICA vs. SEMANTICA ===")
    print(f"Query: '{query_teste}'\n")

    print("--- Busca LEXICA (Ctrl+F) ---")
    for score, doc in buscar_lexicamente(
            query_teste, banco_documentos):
        print(f"  [Matches: {score}] {doc}")

    print("\n--- Busca SEMANTICA (Cosseno) ---")
    for score, idx, doc in buscar_semanticamente(
            query_teste, banco_documentos):
        print(f"  [Score: {score:.4f}] {doc}")
```

A busca léxica retornará frases com palavras genéricas como “e” ou “o” (que aparecem em quase todas as frases), sem nenhuma relevância temática. A semântica retornará documentos sobre finanças. A diferença é gritante — e essa comparação é o argumento mais poderoso da sua PoC para o CTO.

7 Parte 7: Garantia de Qualidade — Testes CI/CD (5 min)

Crie o arquivo `tests/test_buscador.py`. Cada teste valida uma propriedade matemática ou semântica:

```
import numpy as np
from src.buscador_cosseno import (
    cosine_similarity, buscar_semanticamente, banco_documentos
)

def test_cosseno_vetores_identicos():
    """Cosseno de um vetor consigo mesmo deve ser 1.0."""
    v = np.array([1.0, 2.0, 3.0])
    assert abs(cosine_similarity(v, v) - 1.0) < 1e-6

def test_cosseno_vetores_ortogonais():
    """Cosseno de vetores ortogonais deve ser 0.0."""
    v1 = np.array([1.0, 0.0])
    v2 = np.array([0.0, 1.0])
    assert abs(cosine_similarity(v1, v2)) < 1e-6

def test_busca_economia_retorna_financas():
    """Query sobre economia deve retornar docs financeiros."""
    resultados = buscar_semanticamente(
        "economia e dinheiro", banco_documentos, top_k=2)
    textos_top2 = [r[2] for r in resultados]
    financeiro = any(
```

```
            "juros" in t or "investid" in t or "dolar" in t
            for t in textos_top2
        )
        assert financeiro, f"Top-2 nao contem financas: {textos_top2}"
```

Execute `pytest tests/test_buscador.py -v`. Os **3 testes** devem passar.

8 Parte 8: Commit e Push

```
git add src/buscador_cosseno.py tests/test_buscador.py
git commit -m "Lab 10: Buscador semantico + cosseno manual + CI/CD"
git push origin main
```

Conclusão: O que Você Construiu Hoje

✔ Takeaways

- Você implementou a **Similaridade de Cosseno do zero** com NumPy puro — usando a mesma fórmula de Álgebra Linear que vocês viram na graduação.
- Construiu um **buscador semântico completo** com ranking Top-K que encontra documentos por *significado*, não por palavras-chave.
- Provou empiricamente que a **busca semântica supera a léxica**: "economia e dinheiro" encontra "taxa de juros" sem nenhuma palavra em comum.
- Os 3 testes CI/CD garantem: cosseno correto para vetores unitários, ortogonalidade e relevância semântica.

◇ Informação

Gancho para a Próxima Aula: Seu buscador funciona, mas usa um laço `for` que varre N documentos: $O(N \times D)$. Se o banco tiver 500 milhões de parágrafos, essa busca levaria **minutos** por query. Na próxima aula, você substituirá esse laço por um **Banco de Dados Vetorial (ChromaDB)** que usa o algoritmo HNSW para buscar em $O(\log N)$ — milissegundos ao invés de minutos. Seu código artesanal de hoje é o "motor" que o ChromaDB executa internamente.