

Lab. 10 (Parte 1): Buscador Semântico com Cosseno

Aléssio Miranda Júnior

Setembro - 2026/2

Lab. 10 (Parte 1): Buscador Semântico com Cosseno

★ Objetivo da Sessão 1

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.

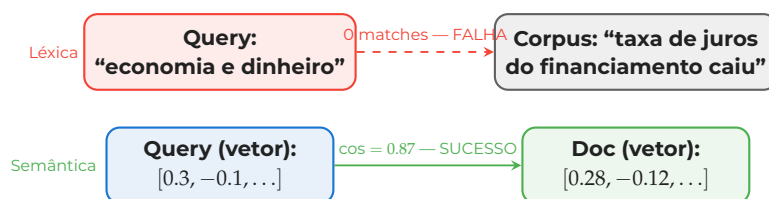
Modelo de Saída: Ranking Top-3 com scores de similaridade.

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:

```

1 import spacy
2 import numpy as np
3 from numpy.linalg import norm
4
5 # 1. Carrega modelo em portugues (com vetores embutidos)
6 nlp = spacy.load("pt_core_news_md")
7
8 # 2. Nosso "Banco de Dados" --- NENHUMA frase contem keywords!
9 banco_documentos = [
10     "O felino descansava tranquilamente no sofa da sala.",
11     "A taxa de juros do financiamento imobiliario caiu ontem.",
12     "O presidente sancionou a nova lei orcamentaria hoje.",
13     "Receitas com chocolate precisam de bastante acucar.",
14     "O medico diagnosticou uma anomalia cardiaca grave.",
15     "A vacina da gripe estara disponivel na UBS amanha.",
16     "O professor explicou a multiplicacao de matrizes.",
17     "O cachorro brincava com as criancas no parque.",
18     "Investidores apostam na queda do dolar este mes.",
19     "A inteligencia artificial transformou a industria.",
20 ]

```

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

```

1 # 3. Funcao de Similaridade de Cosseno --- SEM ABSTRACOES!
2 def cosine_similarity(vec_a, vec_b):
3     """
4     Calcula a Similaridade de Cosseno entre dois vetores.
5     Formula: cos(theta) = (A . B) / (||A|| * ||B||)
6     """
7     dot_product = np.dot(vec_a, vec_b)    # Numerador
8     norm_a = norm(vec_a)                 # ||A||
9     norm_b = norm(vec_b)                 # ||B||
10    if norm_a == 0 or norm_b == 0:        # Previne /0
11        return 0.0
12    return dot_product / (norm_a * norm_b)

```

A guard clause `if norm_a == 0 or norm_b == 0:` previne divisão por zero caso alguma palavra esteja fora do vocabulário (vetor nulo).

△ 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:

```

1 # 4. Motor de Busca Semantica com Ranking Top-K
2 def buscar_semanticamente(query_texto, banco, top_k=3):
3     """
4     Busca os top_k documentos mais semanticamente
5     similares a query.
6     """
7     vetor_query = nlp(query_texto).vector
8     resultados = []
9     for idx, doc_texto in enumerate(banco):
10        vetor_doc = nlp(doc_texto).vector
11        score = cosine_similarity(vetor_query, vetor_doc)
12        resultados.append((score, idx, doc_texto))
13    resultados.sort(key=lambda x: x[0], reverse=True)
14    return resultados[:top_k]

```

B. Testando com Queries Diversas

```

1 # 5. Testando com queries diversas
2 queries_teste = [
3     "Me fale sobre economia e dinheiro.",
4     "Problemas de saude e doencas.",
5     "Animais domesticos e seus habitos.",
6     "Tecnologia e computacao moderna.",
7 ]
8
9 if __name__ == "__main__":
10     for query in queries_teste:
11         print(f"\n{'='*60}")
12         print(f"QUERY: '{query}'")
13         print(f"{'='*60}")
14         top = buscar_semanticamente(query, banco_documentos)
15         for rank, (score, idx, texto) in enumerate(top, 1):
16             print(f"    #{rank} [Score: {score:.4f}] {texto}")

```

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

```

1 # 6. Busca Lexica (baseline ingenua) para comparacao
2 def buscar_lexicamente(query_texto, banco):
3     """Busca documentos que contenham ALGUMA palavra da query."""
4     palavras_query = set(query_texto.lower().split())
5     resultados = []
6     for doc in banco:
7         palavras_doc = set(doc.lower().split())
8         intersecao = palavras_query & palavras_doc
9         resultados.append((len(intersecao), doc))
10    resultados.sort(key=lambda x: x[0], reverse=True)
11    return resultados[:3]
12
13 # 7. Comparacao direta
14 if __name__ == "__main__":
15     query_teste = "Me fale sobre economia e dinheiro."
16     print("\n== COMPARACAO: LEXICA vs. SEMANTICA ==")
17     print(f"Query: '{query_teste}'\n")
18
19     print("--- Busca LEXICA (Ctrl+F) ---")
20     for score, doc in buscar_lexicamente(
21         query_teste, banco_documentos):
22         print(f"    [Matches: {score}] {doc}")
23
24     print("\n--- Busca SEMANTICA (Cosseno) ---")
25     for score, idx, doc in buscar_semanticamente(
26         query_teste, banco_documentos):

```

```
27 print(f" [Score: {score:.4f}] {doc}")
```

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

Crie o arquivo `tests/test_buscador.py`:

```
1 import numpy as np
2 from src.buscador_cosseno import (
3     cosine_similarity, buscar_semanticamente, banco_documentos
4 )
5
6 def test_cosseno_vetores_identicos():
7     """Cosseno de um vetor consigo mesmo deve ser 1.0."""
8     v = np.array([1.0, 2.0, 3.0])
9     assert abs(cosine_similarity(v, v) - 1.0) < 1e-6
10
11 def test_cosseno_vetores_ortogonais():
12     """Cosseno de vetores ortogonais deve ser 0.0."""
13     v1 = np.array([1.0, 0.0])
14     v2 = np.array([0.0, 1.0])
15     assert abs(cosine_similarity(v1, v2)) < 1e-6
16
17 def test_busca_economia_retorna_financas():
18     """Query sobre economia deve retornar docs financeiros."""
19     resultados = buscar_semanticamente(
20         "economia e dinheiro", banco_documentos, top_k=2)
21     textos_top2 = [r[2] for r in resultados]
22     financeiro = any(
23         "juros" in t or "investid" in t or "dolar" in t
24         for t in textos_top2
25     )
26     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

```
1 git add src/buscador_cosseno.py tests/test_buscador.py
2 git commit -m "Lab 10 (Parte 1): Buscador semantico + cosseno manual
   + CI/CD"
3 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 Parte 2: Seu buscador funciona e impressiona no terminal, mas uma demonstração visual não é métrica de engenharia! Na Parte 2, implementaremos métricas formais de avaliação (Precision@K e Recall@K com Ground Truth), diagnosticaremos as falhas onde os embeddings perdem para a busca léxica (termos exatos e códigos técnicos), e criaremos um buscador híbrido com política de abstenção.