

Lab. 11 – Parte 2: Filtros e Trade-offs de Recuperação

Aléssio Miranda Júnior

Setembro - 2026/2

Lab. 11 – Parte 2: Filtros e Trade-offs de Recuperação

Capítulo em Revisão

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

★ Objetivo da sessão – 40 min

Combinar busca vetorial com filtros de metadados e comparar os resultados de uma referência exata com uma busca ANN. Ao final, você saberá que filtros, recall e latência precisam ser medidos no corpus, não inferidos de slogans sobre HNSW.

Problema K2: consulta auditável

A HistoriAI agora precisa responder apenas com fatos de uma categoria e período informados pelo usuário. Adicione metadados ao corpus, compare a mesma consulta com e sem filtro e construa uma pequena referência de busca exata para avaliar o conjunto retornado pelo índice.

1 Roteiro da sessão

Tempo	Entrega
10 min	Metadados e nova carga versionada
10 min	Filtros simples e compostos
12 min	Referência exata e comparação com ANN
5 min	Desafio: resultados com e sem filtro
3 min	Registro de evidências

2 1. Metadados filtráveis (10 min)

Crie `src/filtros_recuperacao.py` (ou atualize `src/banco_vetorial.py`) e acrescente campos estruturados aos documentos. Metadados não devem carregar informações que pertencem somente ao texto semântico; use-os para atributos verificáveis, como categoria e ano.

```

1 facts = [
2     ("fato_01", "A penicilina foi descoberta por Fleming em 1928.",
3      {"categoria": "ciencia", "ano": 1928}),
4     ("fato_02", "O ENIAC foi um computador pioneiro em 1945.",
5      {"categoria": "tecnologia", "ano": 1945}),
6     ("fato_03", "A vacina contra a varíola reduziu doença mortal em
7      1796.",
8      {"categoria": "ciencia", "ano": 1796}),
9     ("fato_04", "A República foi proclamada no Brasil em 1889.",
10     {"categoria": "politica", "ano": 1889}),
11     ("fato_05", "Marie Curie investigou a radioatividade em 1898.",
12     {"categoria": "ciencia", "ano": 1898}),
13     ("fato_06", "Ada Lovelace escreveu notas sobre computação em
14     1843.",
15     {"categoria": "tecnologia", "ano": 1843}),
16     ("fato_07", "O Muro de Berlim caiu marcando o fim da Guerra Fria
17     em 1989.",
18     {"categoria": "politica", "ano": 1989}),
19     ("fato_08", "A estrutura em dupla hélice do DNA foi descoberta em
20     1953.",
21     {"categoria": "ciencia", "ano": 1953}),
22 ]
23 collection.upsert(
24     ids=[f[0] for f in facts], documents=[f[1] for f in facts],
25     metadata=[{
26         **f[2], "source": "historiai-demo",
27         "source_version": "historiai-demo-v2"
28     } for f in facts],
29 )

```

3 2. Semântica mais restrição (10 min)

```

1 def buscar(query, top_k=3, where=None):
2     if top_k < 1:
3         raise ValueError("top_k deve ser positivo")
4     params = {
5         "query_texts": [query],
6         "n_results": min(top_k, collection.count()),
7         "include": ["documents", "metadata", "distances"],
8     }
9     if where:
10        params["where"] = where
11    return collection.query(**params)
12
13 sem_filtro = buscar("descoberta que protegeu a saúde", top_k=3)
14 so_ciencia = buscar(
15     "descoberta que protegeu a saúde", top_k=3,
16     where={"categoria": "ciencia"},
17 )
18 periodo = buscar(
19     "descobertas médicas do século XX", top_k=3,
20     where={"$and": [{"categoria": "ciencia"}, {"ano": {"$gte":
21         1900}}]}},

```

O motor pode aplicar filtros em etapas diferentes conforme a implementação e a seletividade; não assuma que o filtro sempre reduz custo ou que sempre é aplicado “antes” do índice. Se `n_results` exceder a quantidade elegível pelo filtro, o ChromaDB retorna com segurança apenas os documentos que satisfazem o predicado.

4 3. Exato versus ANN (12 min)

Para um corpus pequeno, crie uma referência exata com álgebra linear direta (força bruta). Ela serve como padrão-ouro para medir o que o índice aproximado recuperou.

```

1 import numpy as np
2
3 def top_k_exato(query_embedding, document_embeddings, ids, k):
4     # Normaliza explicitamente para que produto interno equivalha a
5     # cosseno
6     query = query_embedding / np.linalg.norm(query_embedding)
7     documentos = document_embeddings / np.linalg.norm(
8         document_embeddings, axis=1, keepdims=True)
9     scores = documentos @ query
10    ordem = np.argsort(scores)[::-1][:k]
11    return [ids[i] for i in ordem]

```

```
12 def recall_at_k(exatos, ann, k):
13     denominador = min(k, len(exatos))
14     if denominador == 0:
15         return 0.0
16     return len(set(exatos[:k]) & set(ann[:k])) / float(denominador)
17
18 def comparar_ann_vs_exato(query_texto, top_k=3, where=None):
19     dados = collection.get(where=where, include=["embeddings"])
20     doc_embeddings = np.array(dados["embeddings"])
21     doc_ids = dados["ids"]
22     query_emb = np.array(embedding_fn([query_texto])[0])
23
24     ids_exatos = top_k_exato(query_emb, doc_embeddings, doc_ids,
25                             top_k)
26     res_ann = buscar(query_texto, top_k=top_k, where=where)
27     ids_ann = res_ann["ids"][0]
28
29     recall = recall_at_k(ids_exatos, ids_ann, top_k)
30     return ids_exatos, ids_ann, recall
```

5 4. Desafio de comparação (5 min)

Para a consulta "descobertas médicas do século XX", execute os três modos e compare os resultados:

Modo	Top-3	Anos	Observação
Sem filtro	fato_01, fato_03	1928, 1796	Varíola (1796) foge ao séc. XX
Filtro ciência	fato_01, fato_03	1928, 1796	Mesmos (ambos são ciên- cia)
Ciência + ano ≥ 1900	fato_01, fato_08	1928, 1953	Varíola e Curie podadas!

△ Importante: Poda Estruturada vs Semântica Pura

Observe como a varíola (1796) possui proximidade semântica altíssima com “medicina”, mas viola a restrição cronológica do usuário. O filtro estruturado complementa o vetor ao eliminar candidatos semanticamente válidos que violam políticas de domínio.

6 5. Verificação e Testes (3 min)

Em `tests/test_filtros_recuperacao.py`, valide o comportamento dos filtros e a métrica de *Recall@K*:

```
1 def test_filtro_composto_temporal():
```

```
2 resultado = buscar("medicina", top_k=5,
3                     where={"$and": [{"categoria": "ciencia"}, {"
4                               ano": {"$gte": 1900}}]})
5 assert all(m["categoria"] == "ciencia" and m["ano"] >= 1900
6             for m in resultado["metadatas"][0])
7
8 def test_recall_at_k_perfeito():
9     assert recall_at_k(["a", "b", "c"], ["a", "b", "c"], k=3) == 1.0
10    assert recall_at_k(["a", "b", "c"], ["a", "b", "d"], k=3) == 2.0
11        / 3.0
```

✓ Takeaways

Você combinou relevância semântica com filtros estruturados e avaliou quantitativamente a busca aproximada (ANN) contra uma referência exata via *Recall@K*. Na Aula 12, aprofundaremos a topologia do grafo HNSW, investigando como os hiperparâmetros M e $efConstruction$ governam o trade-off entre latência, memória e precisão de busca.