ЛЕКЦІЯ 2

Retrieval: векторизація та семантичний пошук

Embedding • Cosine Similarity • ChromaDB • Chunking • HNSW

Семантичний пошук — знаходить не слова, а смисл

Що таке Retrieval?

Перший і критично важливий етап у системі RAG

Задача Retrieval

Знайти у великому масиві зовнішніх даних лише ті фрагменти, що містять відповідь на запитання користувача.

Традиційний пошук

WHERE title = 'Inception'
Точний збіг слів — не розуміє зміст

Семантичний пошук (RAG)

Знаходить документи за смислом, навіть без спільних слів із запитом

«Хочу фільм про космос»

→ знаходить 'Interstellar', 'Gravity'

«Серіал про сімейну кризу»

→ знаходить 'Marriage Story'

«Драма із сильними емоціями»

→ знаходить 'Manchester by the Sea'

Retrieval забезпечує LLM «свіжими» фактами з власної бази — уникає галюцинацій

Векторизація тексту (Embeddings)

Текст → щільний вектор дійсних чисел, де зашифрований його смисл

Формальна модель

$$f : T \to \mathbb{R}^d$$

Модель перетворює текст T у щільний вектор розмірності d (зазвичай 384–1536 вимірів)

Популярні моделі (2026)

- BGE (BAAI/bge-m3)
- all-MiniLM-L6-v2
- OpenAI text-embedding-3
- Архітектура: Transformer (BERT / RoBERTa)

Приклад: простір d = 3

«Фільм про космос»

[0.91, 0.12, 0.03]

«Астронавт у вакуумі»

[0.88, 0.15, 0.05]

«Рецепт борщу»

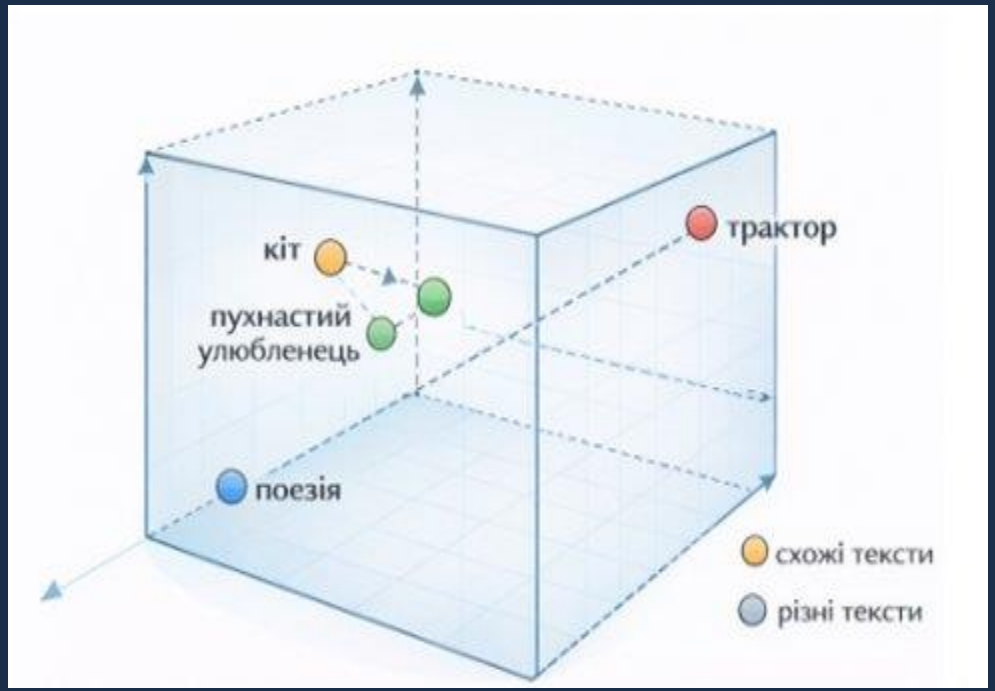
[-0.10, 0.82, 0.45]

Перші два вектори майже збігаються — вони семантично близькі до теми космосу.

Схожий текст → схожий вектор → близьке розташування у просторі

Векторний простір

Рисунок 2 — Представлення векторного простору



Що таке векторний простір?

Уявіть 3D-куб, де кожен текст — це точка. Схожі тексти опиняються поруч. У реальних моделях — тисячі вимірів.

Схожі тексти

«кіт» і «пухнастий улюбленець» — семантично близькі: схожий зміст, схожі координати у просторі.

Різні тексти

«трактор» і «поезія» — протилежні концепції: далеко одне від одного у векторному просторі.

Формула векторизації: $f : T \to \mathbb{R}^d$ | Схожий текст → схожий вектор → близьке розташування у просторі

ChromaDB та Векторний простір

Як зберігаються та шукаються вектори на практиці

1 Embedding

Кожен опис проходить крізь Embedding-модель (BGE, MiniLM тощо) та перетворюється на вектор.

2 Vector Store

Вектори зберігаються у спеціальній структурі разом із метаданими: назва, рік, рейтинг.

3 HNSW-граф

ChromaDB будує граф, де близькі вектори з'єднані. Пошук — «стрибки» по вузлах за O(log n).

Алгоритм HNSW

Hierarchical Navigable Small World

Шар 0 (верхній) — ~3 вузли

Грубий пошук — швидкий перехід між кластерами даних

Шар 1 — ~12 вузлів

Середня деталізація пошуку

Шар 2 (базовий) — всі N

Точний пошук серед найближчих сусідів

Складність: O(log n) замість O(N·n)

Семантична близькість: Cosine Similarity

Основний математичний інструмент пошуку в RAG

$$\cos(\theta) = (q \cdot d) / (\|q\| \cdot \|d\|)$$

q·d — скалярний добуток (напрямок) | ‖q‖·‖d‖ — довжини (нормалізація)

cos ≈ 1

Максимальна схожість

Вектори дивляться в одному напрямку. «Фільм про космос» ↔ «Астронавт у вакуумі»

cos ≈ 0

Не пов'язані теми

Вектори перпендикулярні. Спільного між темами немає зовсім.

cos ≈ –1

Протилежні значення

Антоніми або протилежні концепції. Рідкість у практичних задачах.

Distance = 1 − Similarity | Чим менша дистанція → кращий результат пошуку в ChromaDB | Косинус ігнорує довжину тексту — лише напрямок думки

Приклад семантичного пошуку

Крок 1. Запит користувача
"I want to watch a heart-wrenching story about family loss."

Крок 2.

Векторизація запиту

. Модель перетворює цей запит у вектор.

Крок 3.

Пошук у ChromaDB.

Система знаходить документи з найменшою дистанцією:
1. "Manchester by the Sea" (Distance: 0.12), де опис: "A man returns to his hometown after his brother dies..."
2. "Marriage Story" (Distance: 0.25), де опис: "A look at a marriage breaking up and a family staying together..."
3. "Fast & Furious" (Distance: 0.85), де опис: "Action movie about street racing..." (Відкинуто).

Результат. Система повертає перші два фільми, хоча в їхньому описі немає слова "heart-wrenching".

Chunking — розбиття документів

Великий текст → менші, логічно завершені фрагменти перед векторизацією

Ліміт моделей

Embedding-моделі обробляють обмежену кількість токенів за один запит.

Точність пошуку

Ціла книга як один вектор → розмитий сенс. Дрібні чанки → прицільний результат.

Економія контексту

LLM отримує лише релевантний шматок, зберігається місце для інших даних.

Методи чанкінгу

Fixed-size

Розбиття строго по N символів. Речення обривається посередині — контекст втрачається.

Overlap

Кожен чанк захоплює кінець попереднього (10–20%). Стандарт 2026 — контекст не розривається.

Semantic

Розбиття між думками (паузи, крапки). Найкраща якість, але складніша реалізація.

Chunk Size: 512–1024 токени | Chunk Overlap: 10–20% | Великі чанки → ↓ релевантність | Малі чанки → ↓ контекст

Retrieval як оптимізаційна задача

Знайти k найближчих точок у високовимірному просторі $\mathbb{R}^d$

$$D_k = \text{TopK}\ \text{argmax}\ \text{sim}(\ f(q),\ f(d_i)\ )$$

$D_k = \{d_1, d_2, \ldots, d_k\}$ — k найбільш релевантних фрагментів, впорядкованих за спаданням схожості

Brute-force O(N · n)

Порівняння запиту з кожним документом.

При N = 1536 вимірів (OpenAI) та n = мільйон документів час стає неприйнятним.

Не підходить для продуктивних систем.

HNSW — ANN O(log n)

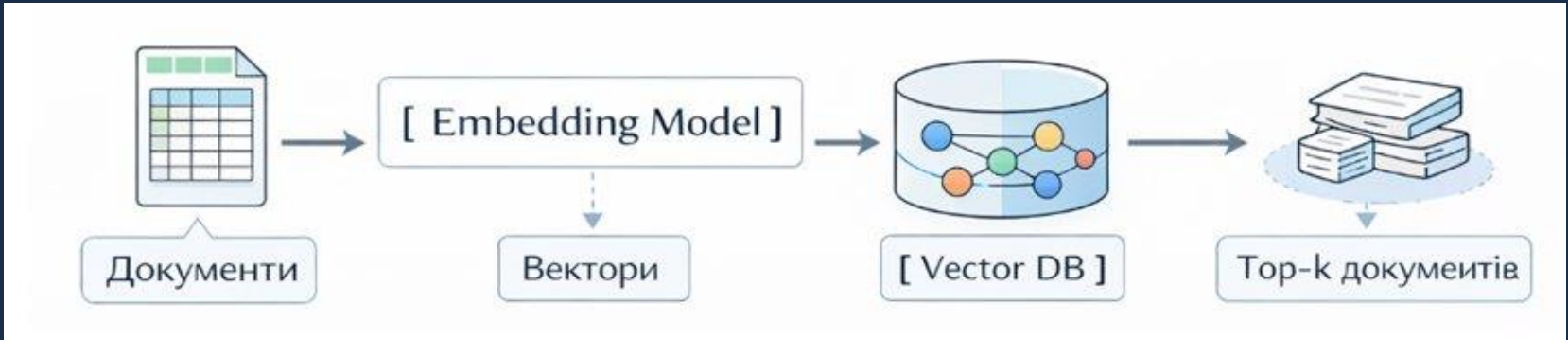
Approximate Nearest Neighbors — незначна жертва точністю заради колосального приросту швидкості.

ChromaDB будує ієрархічний граф, де близькі вектори з'єднані між собою. Пошук ведеться «стрибками» по шарах.

Також застосовуються: IVF (Inverted File Index) • PQ (Product Quantization) • Мета: maximize sim при мінімальному часі пошуку

Архітектура Retrieval

Архітектура Dense Retrieval у RAG (стандарт 2026)



01 Документи

Вхідні дані: описи фільмів, тексти, статті будь-якого формату.

02 Embedding Model

Нейромережа (BERT, BGE) перетворює текст у щільний вектор $\mathbb{R}^d$.

03 Vector DB

ChromaDB зберігає вектори з метаданими і будує HNSW-граф для пошуку.

04 Top-k документів

Система повертає k найближчих фрагментів як контекст для LLM.

Точки відмови та ключові висновки

1 Погана Embedding-модель

Якщо модель не знає мови датасету — векторизація буде хибною, пошук не знайде нічого релевантного.

2 Невдалий Chunking

Опис, розбитий посеред речення, дає «брудний» вектор. Частина змісту назавжди втрачається.

3 Шум у даних

HTML-теги та зайві символи в описах псують векторне представлення. Потрібен preprocessing!

Метрики якості Retrieval

Precision (Точність)

Чи дійсно знайдені документи стосуються теми запиту? Скільки «сміття» повернула система?

Recall (Повнота)

Чи не пропустила система важливу інформацію, що була в базі? Повнота покриття результатів.

- Retrieval = оптимізаційна задача: знайти k семантично найближчих точок у просторі $\mathbb{R}^d$
- Cosine Similarity вимірює кут (напрямок думки), а не відстань — ідеально для тексту
- HNSW у ChromaDB дає O(log n) — пошук серед мільйонів векторів за мілісекунди