Vektoradatbázisok és alkalmazásaik

Juhász Gábor
2025. 05. 12.

Relációs adatmodell – mire elég?

- Strukturált adatok tárolása
  - Jól definiált attribútumok, kulcsok, kapcsolatok
- Normalizálás
  - Adatstruktúrák felhasználási profilhoz optimalizálható kialakítása
- Tranzakciók
  - Akár ACID tulajdonságok támogatása
- Lekérdezések
  - SQL nyelv segítségével deklaratív jellegű
  - A lekérdezések automatikusan is optimalizálhatók

Relációs adatmodell korlátai

- Változó szerkezetű adatok kezelése
  - Új mezők hozzáadása sémamódosítással lehetséges
- Szemistrukturált és strukturálatlan adatok kezelése
  - Pl. dokumentum, szöveg, kép, hang
  - Ezek hatékony tárolása és tartalom alapján való kereshetősége csak kiegészítőkkel/nehézkesen biztosított

Új megközelítések

- NoSQL megoldások
  - Nem-relációs
  - Rugalmas adatszerkezet
  - Nagy mennyiségű adat gyors, de korlátozott kezelése
- Sokféle, alkalmazásspecifikus megoldás
  - Lásd: NoSQL előadás, tárgyhonlap
  - Key-value store
  - Wide-column based
  - Dokumentumtárak
  - Gráfadatbázisok
  - Vektoradatbázisok

Hasonlósági keresés

A macska halkan dorombolva összekucorodik a meleg ablakpárkányon, miközben a kertben a kutya vidáman ugat egy arra szaladó mókus után. Az ég tiszta kékje lassan sötétedni kezd, ahogy a nap véget ér, és az esti szél enyhe fuvallatot hoz a fák lombjai között. A macska füle percegve figyeli a madarak csicsergését, míg a kutya fáradtan heverészik a frissen nyírt füvön, a nyelve lógva pihenve a hosszú nap után. Az ég most már sötétlila árnyalattal borul be, és az első csillagok pislákolni kezdenek a távolban, miközben a két állat békésen készülődik az éjszakára.

Hasonlósági keresés

- A felhasználó szeretné tudni, mit csinál a cica
- A szövegben csak macska szerepel
- Hogyan oldjuk meg?
  - Ilyenkor ne legyen válasz?
  - Vezessük a szinonimákat?
  - Valamilyen komplexebb megoldás?

Hasonlósági keresés (egy lehetséges, ma elterjedt) megvalósítása

1. Vektorrá alakítás
2. Eltárolás a vektoradatbázisban
3. Keresés valamilyen hasonlósági metrika alapján

Vektorrá alakítás folyamata

- Az összefüggő szöveg feldarabolása *chunk*okra (chunkolás)
  - Rövidebb egységnyi szövegek (mondatok/bekezdések)
  - Lehet átfedéssel vagy átfedés nélkül
- A chunkokat adjuk bemenetül az *embedding modell*nek (ld. később)
- Az embedding modell határozza meg a chunkok vektor formájú reprezentációját (*embedding*jét)

Embedding modell

- A bemenetként érkező szöveghez meghatározza az embeddingjét
- Az embeddingek meghatározása egy *transzformer modell* segítségével történik
  - Transzformer modell: *mélytanulási modell*, mely hatékonyan kezel szekvenciális adatokat
  - Nagy mennyiségű adaton tanított, képes a szemantikai kapcsolatokat detektálni
- Számos, különböző megvalósítás létezik
  - Eltérő dimenziószám, architektúra
  - Nem csak szövegre: kép, hang, stb.

Embedding modell működése

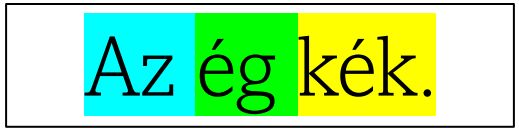
- Az OpenAI text-embedding-ada-002 embedding modell működését vizsgáljuk
  - Népszerű, széles körben használt modell
  - 1536 dimenziós vektorokkal dolgozik
- A bemeneti szöveget először *token*ekre bontja
  - Token: a szöveg *nyelvi modellek* számára értelmezhető egysége (szó/szórészlet)
- A transzformer modell meghatározza a tokenek embeddingjeit
- A tokenek embeddingjeit aggregálva (ld. később) kapjuk a bemeneti szöveg embeddingjét

Embedding modell működése (példa)

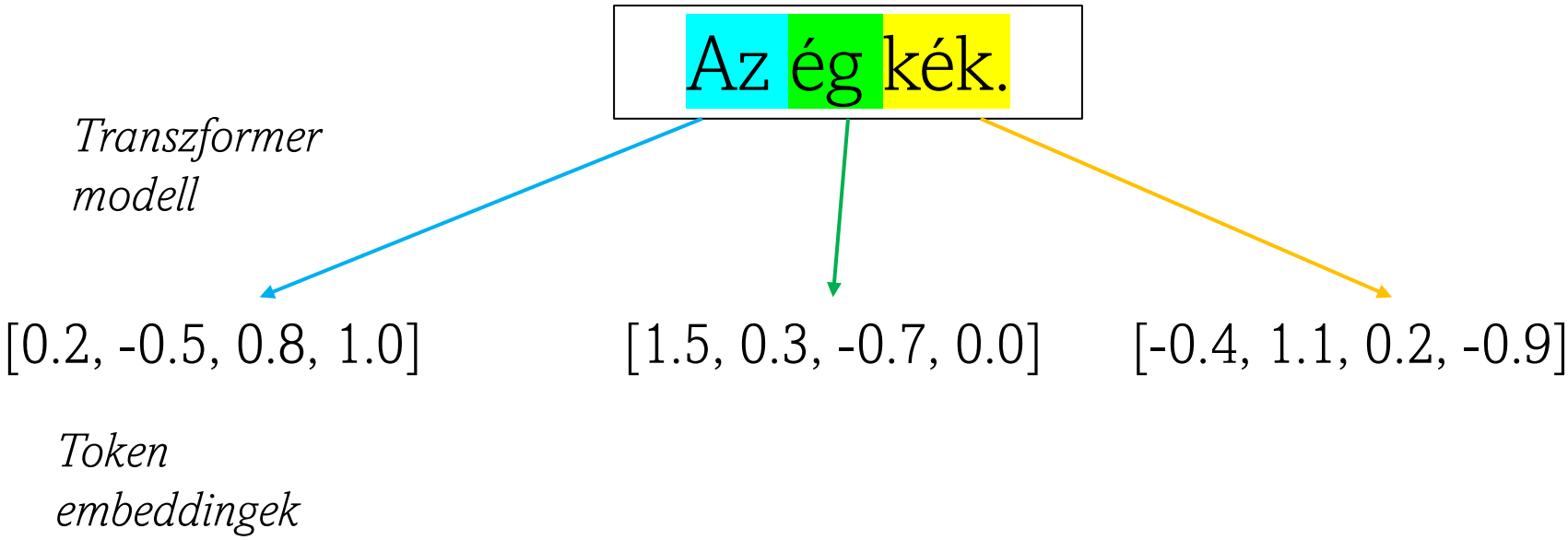
Az ég kék.

Embedding modell működése (példa)

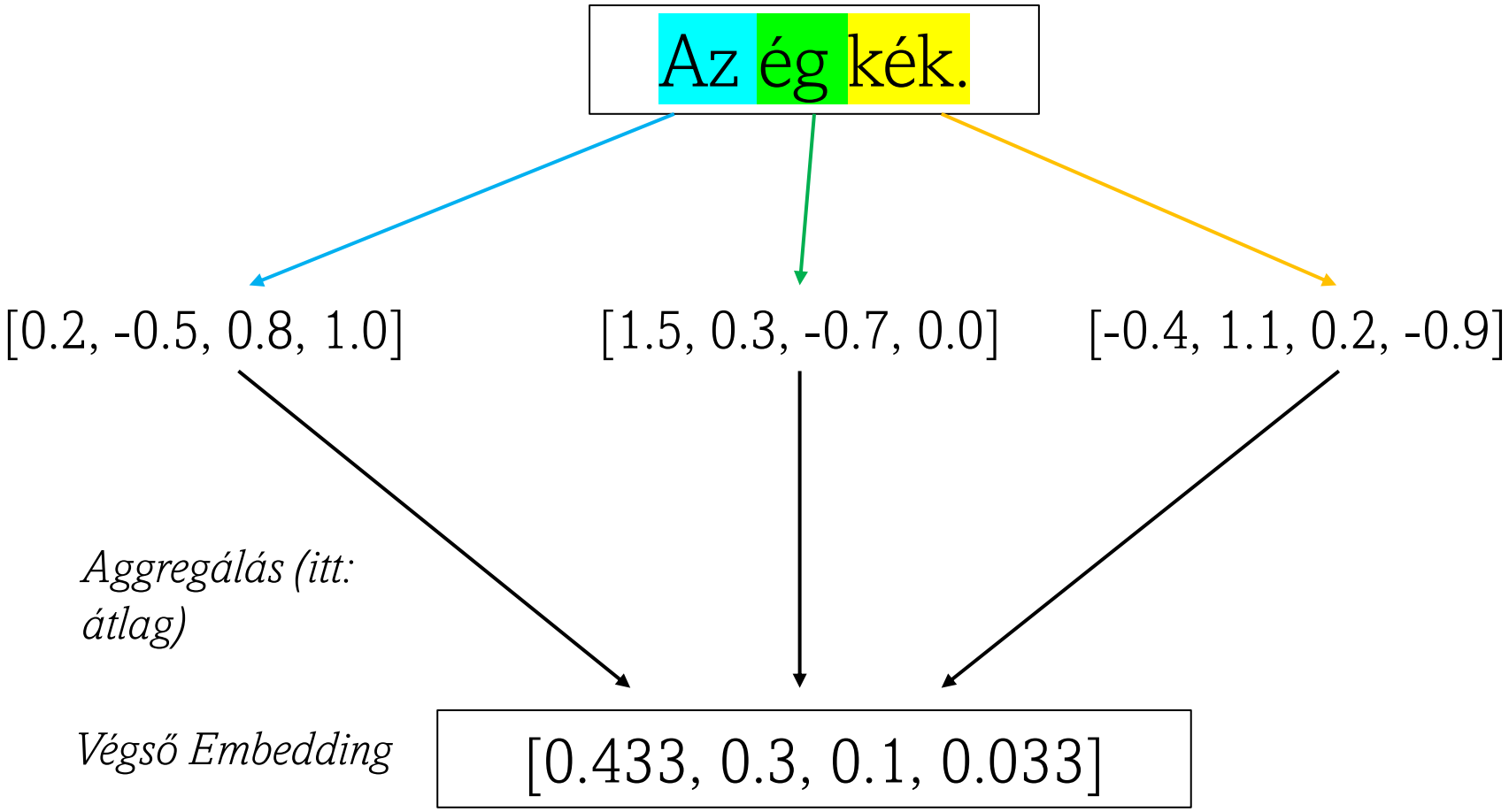
Tokenizálás



Embedding modell működése (példa)



Embedding modell működése (példa)

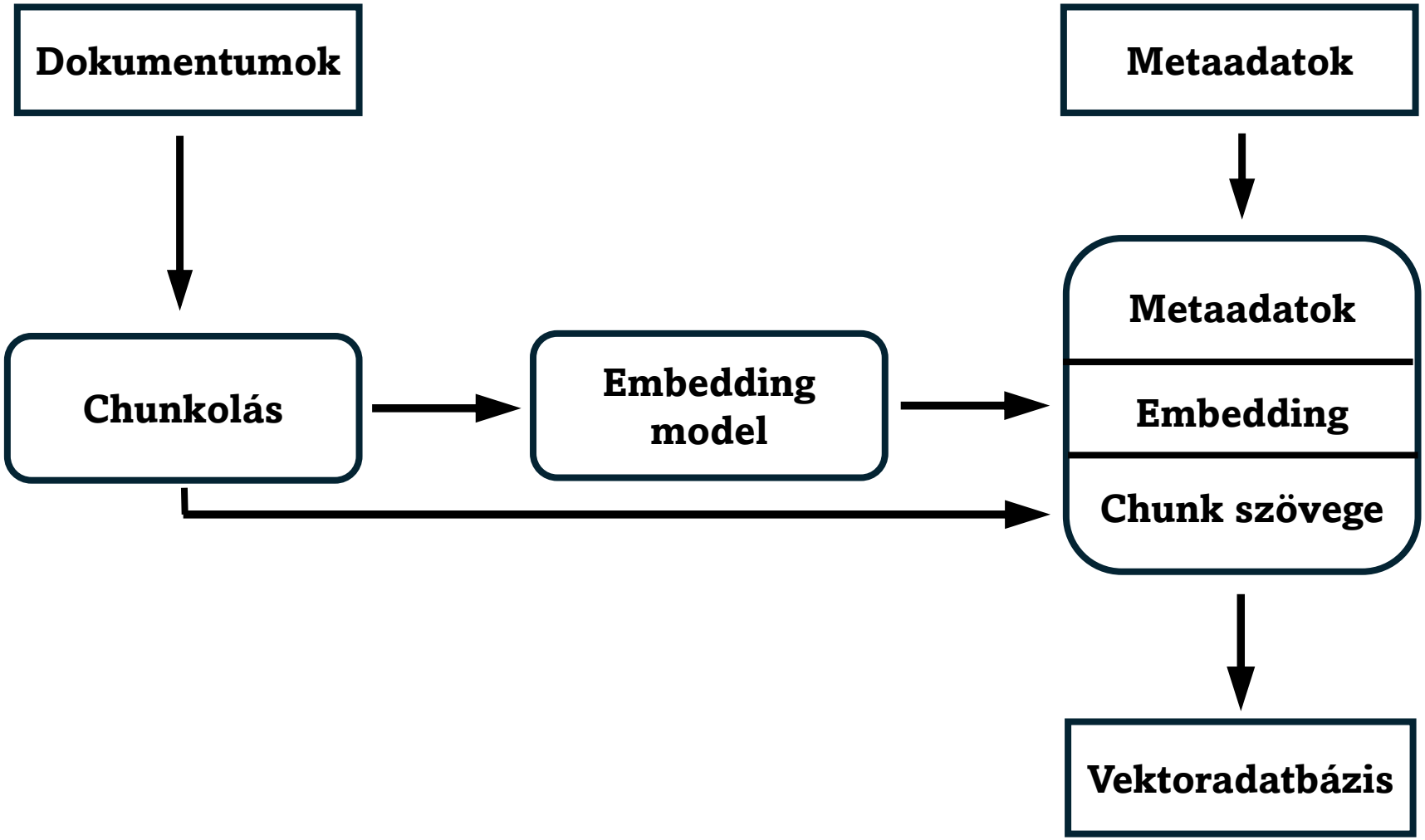


Vektoradatbázis

- Magas dimenziós vektorok (embeddingek) tárolására specializált adatbázis
- Vektorok közötti hasonlósági keresés támogatott
- Speciális indexelési algoritmusok
- Vektorokhoz rendelt metaadatok tárolása is lehetséges
  - Pl. dokumentum címe, szerzője, oldalszám
- Számos implementáció létezik
  - Open-source és fizetős szolgáltatások is

Vektoradatbázis implementációk

Vektoradatbázis építése



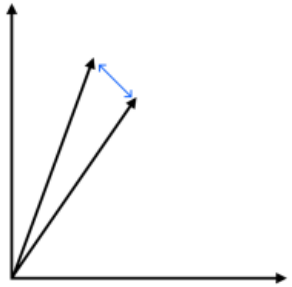
Keresés a vektoradatbázisban

- Keresési vektor: a felhasználói kérdés vektorrá alakítva
  - Fontos: ugyanazt az embedding modellt használjuk a kérdés vektorrá alakítására, mint amit a vektoradatbázis építésekor használtunk
- A keresés eredménye egy halmaz
  - A keresési vektorhoz képest az n leghasonlóbb vektor az adatbázisból
- A hasonlóság mérése több metrika szerint lehetséges
  - Euklideszi távolság
  - Skalárszorzat
  - Koszinusz-távolság

Hasonlósági metrikák

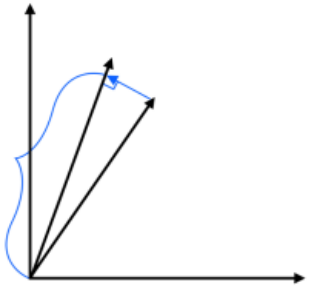
Euclidean Distance

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$



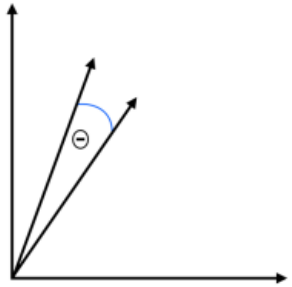
Dot Product

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$



Cosine Similarity

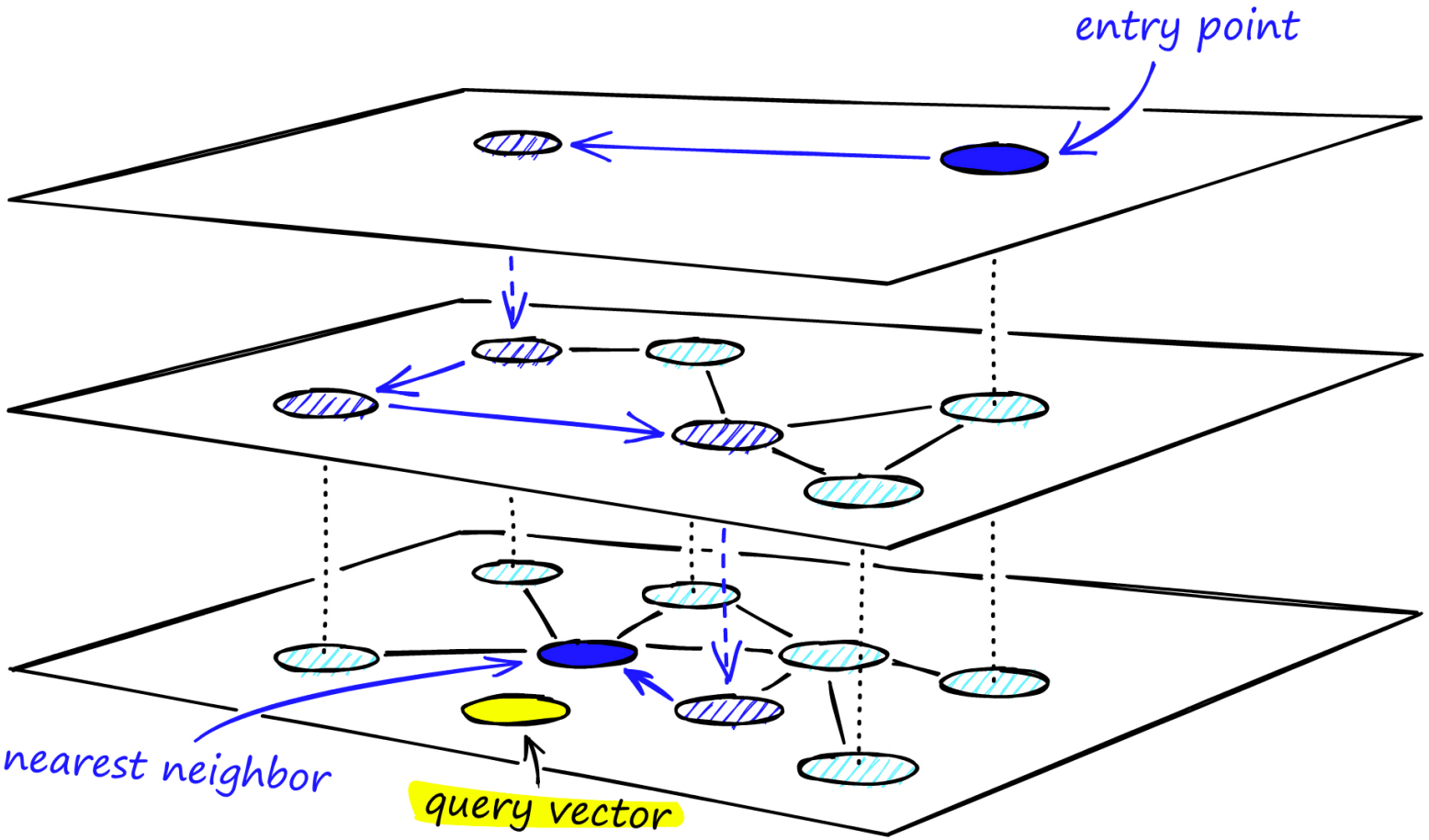
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$



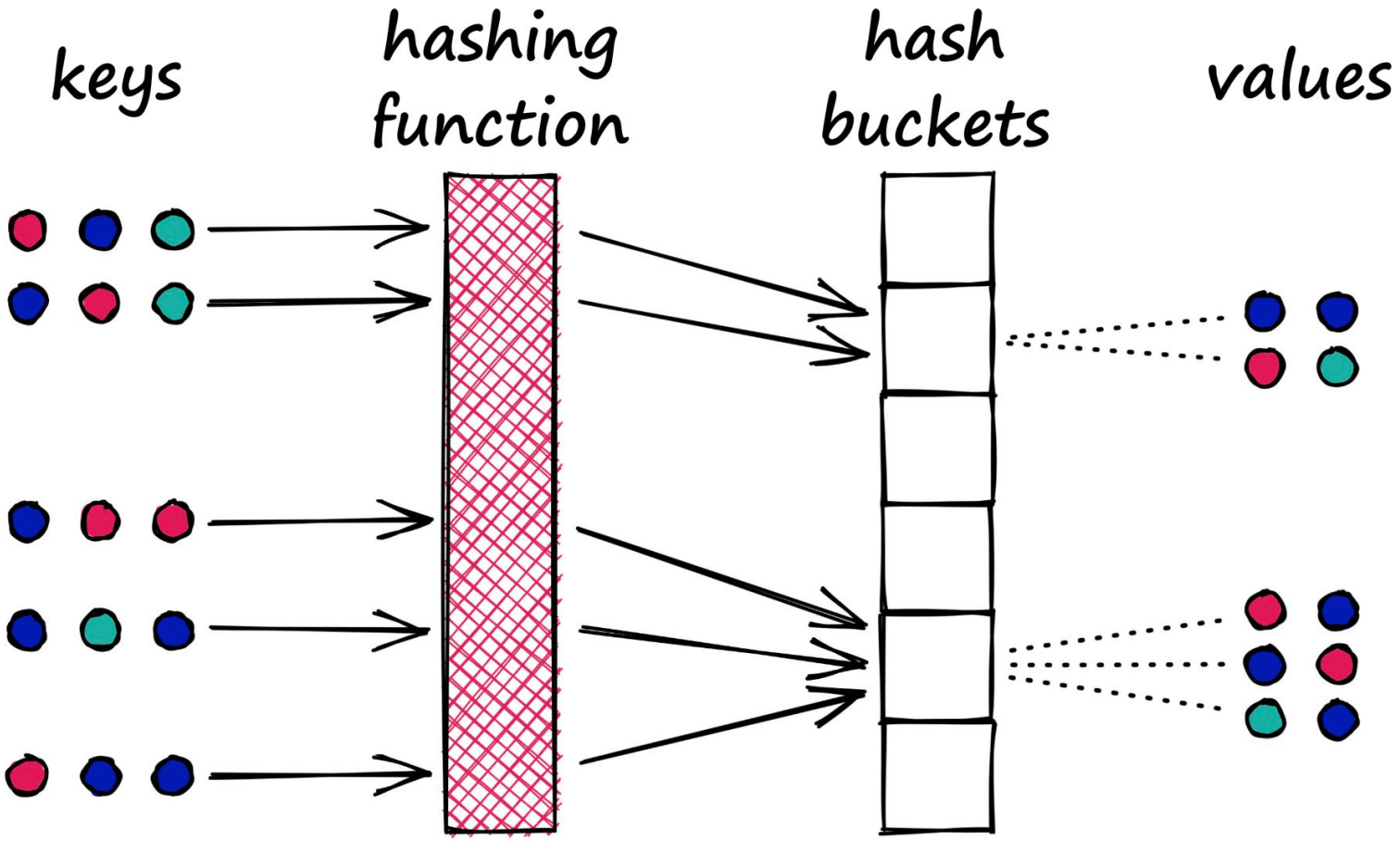
Indexek a vektoradatbázisban

- Ugyanaz a szerepük, mint egy hagyományos adatbázisban
- Három lényeges indexelési algoritmus:
  - HNSW: Hierarchical Navigable Small World
  - LSH: Locality Sensitive Hashing
  - IVF: Inverted File Index
- Brute-force keresés is támogatott
  - Kis adathalmazon hatékony
  - Skálázódási problémák

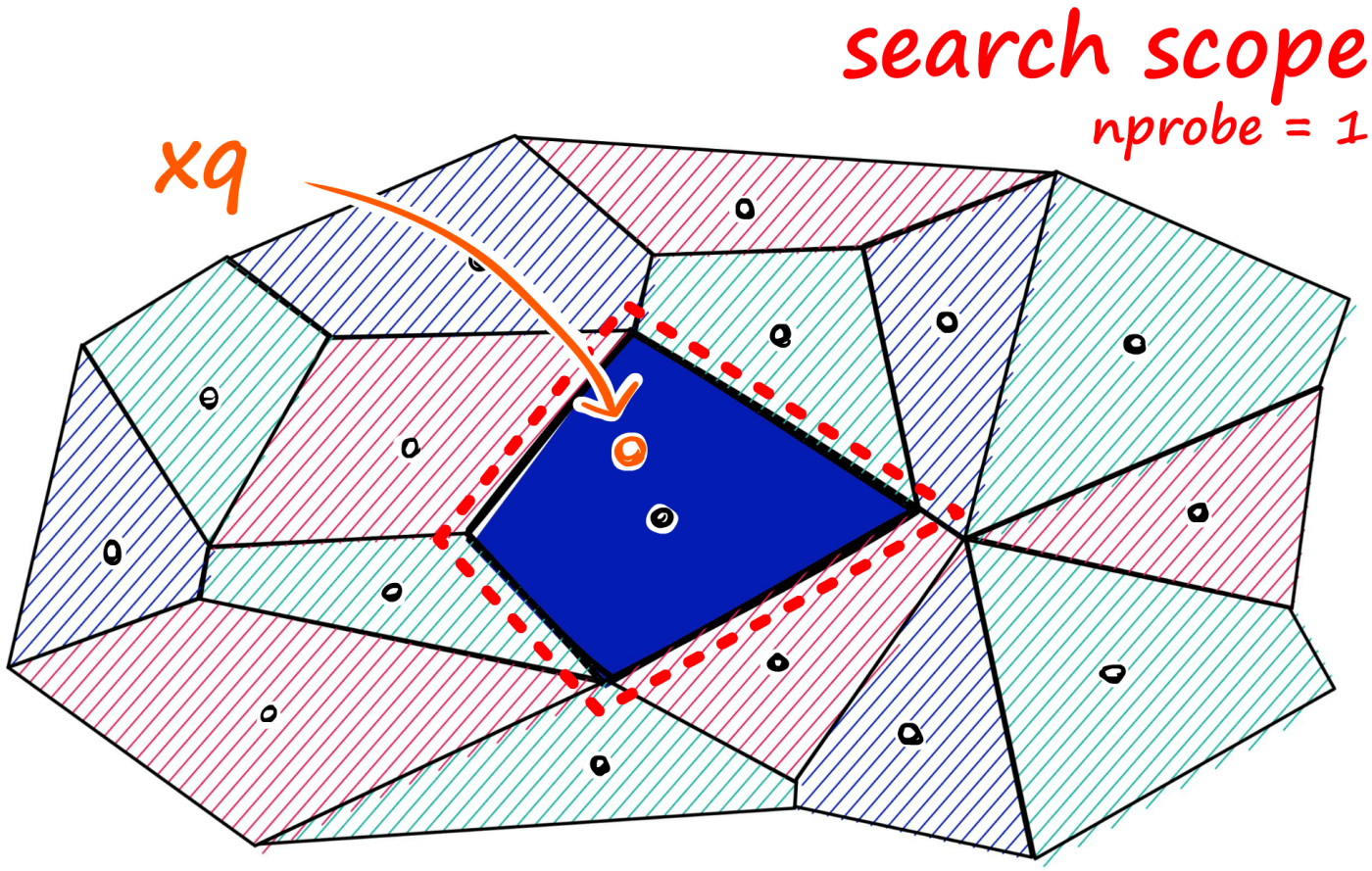
HNSW algoritmus



LSH algoritmus



IVF algoritmus



Chroma használata

```
from langchain.vectorstores import Chroma
```

```
# Vektoradatbázis felépítése
vector_store = Chroma.from_documents(documents=chunks, embedding=custom_embeddings)

retriever = vector_store.as_retriever(
    search_type="similarity",
    search_kwargs={"k": 5}
)
```

```
docs = retriever.invoke(query)
```

Retrieval Augmented Generation (RAG)

Egy gyakori vektoradatbázis use-case

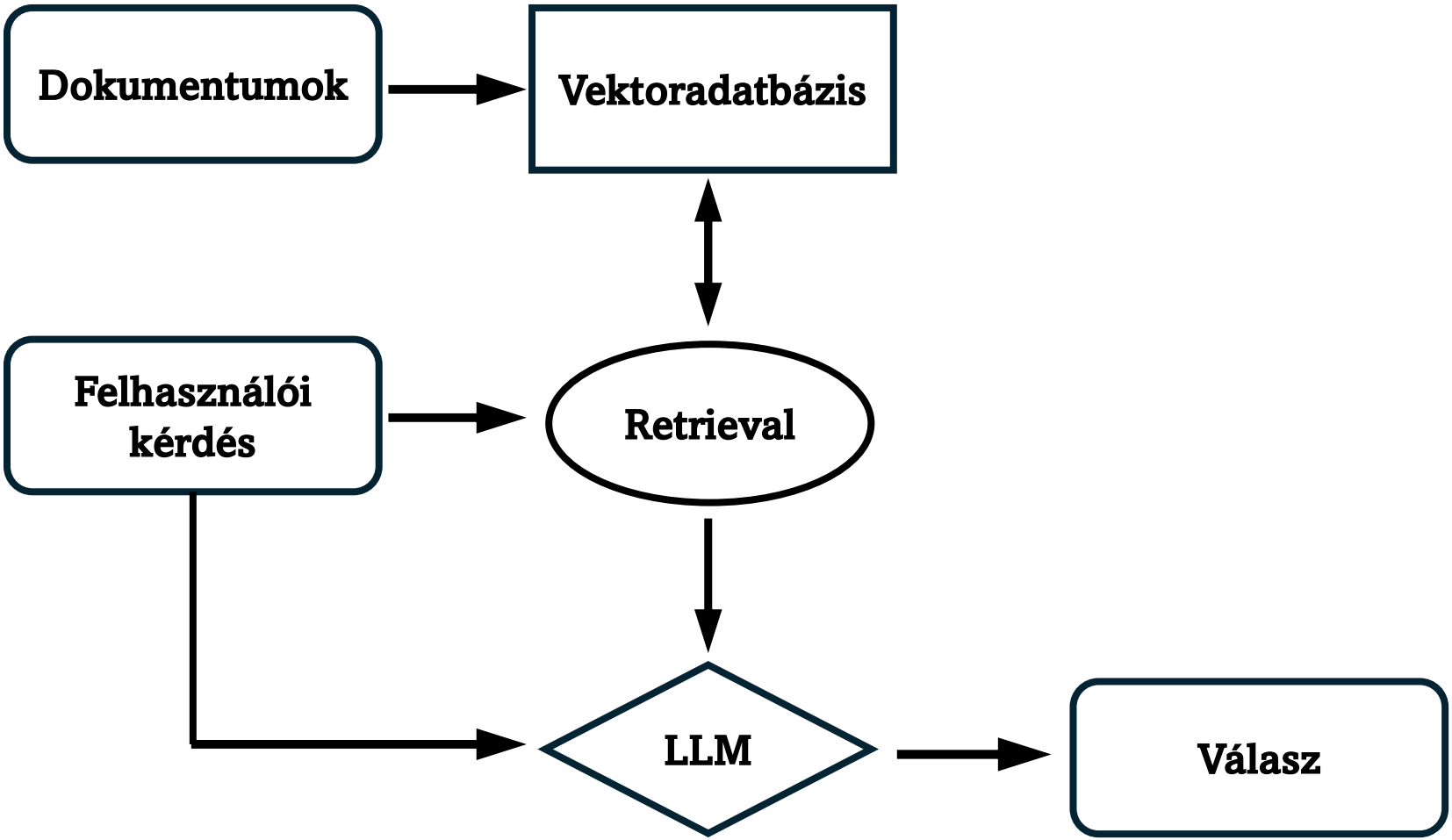
A probléma

- A felhasználó feltesz egy kérdést az LLM-nek
  - LLM: Large Language Model, pl. GPT, Llama
- Az LLM csak a tanítás során látott tudással rendelkezik
  - Ha ez nem elegendő a kérdés megválaszolására, irreleváns választ kapunk
- A modell újratanítása a választ biztosító tudással kiegészítve (dokumentumok) nem reális
  - Idő- és erőforrásigényes feladat

A megoldás: RAG

- Retrieval Augmented Generation
- Az LLM tudását egészítsük ki egy külső tudásbázissal
  - Dokumentumok tartalmából épített vektoradatbázis
- Keresési modell az LLM előtt
  - A kérdés alapján keressük vissza a releváns kontextust a vektoradatbázisból
  - A kérdéssel együtt a kontextust is adjuk át az LLM-nek
- Így naprakész válaszokat kaphatunk

RAG architektúra



Példa prompt

„Válaszolj az alábbi kérdésre a megadott kontextus alapján.

A kérdés: Milyen színű az ég?

Kontextus:

„az ég kék”

„a fű zöld”

…

A válaszod legyen tömör, szükség esetén alkalmazz bullet-pointokat.”

Problémák RAG esetén

- Túl specifikus kérdés
- Zajos kérdés
- Tényszerű kérdések
  - A felhasználó konkrét adatra kíváncsi (szinonimák sincsenek)
  - Egy kulcsszavas keresési modell hatékonyabb lehet
- Túl összetett kérdés
  - Több részkérdésből álló kérdés

Továbbfejlesztési lehetőségek

- Felhasználói kérdések előfeldolgozása
  - Step-back prompting: tegyük a kérdést általánosabbá
  - Query rewriting: a kérdés újraírása, a zaj eltávolítása
- Kulcsszavas és hasonlósági keresés eredményeinek kombinálása
- Kérdések részekre bontása
- Reranking transzformer modellek segítségével
  - A már visszadott válaszok újrarangsorolása / szűrése

Vektoradatbázisok - további felhasználás

- Hasonlóság alapú keresés
  - Pl. képre, zenére
- Ajánlórendszerek
  - Felhasználói preferenciákhoz a megfelelő termék vagy tartalom megkeresése
- Anomália detekció
  - Anomália vagy csalás detektálása az ezen viselkedéseket reprezentáló vektoroktól való távolság alapján

Irodalom

- A Comprehensive Survey on Vector Database (Yikun Han, Chunjiang Liu, Pengfei Wang – 2023, arXiv)
- Nearest Neighbor Indexes for Similarity Search (pinecone.io/learn/series/faiss/vector-indexes)
- Retrieval-Augmented Generation for Large Language Models: A Survey (Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang és H. Wang – 2024, arXiv)
- A Deep Dive into OpenAI's text-embedding-ada-002: The Power of Semantic Understanding (medium.com/@siladityaghosh/a-deep-dive-into-openais-text-embedding-ada-002-the-power-of-semantic-understanding-7072c0386f83)
- What is a vector database? (www.ibm.com/think/topics/vector-database)