Напредни Бази на Податоци

Тема: Векторска База на Податоци

Техничка документација

Подсистем за препорака на услуги

Платформа за резервации во салони за убавина

Python | PostgreSQL | SentenceTransformers | NumPy

Тим: BeautyBook

Членови: Марија Мисајловска | Марко Мијоски | Јован Стојковски

1. Преглед на системот

1.1 Цел и контекст

Овој подсистем за препорака е развиен во рамките на предметот **Напредни Бази на Податоци**, со фокус на практична примена на **векторски бази на податоци** и семантичко пребарување. Целта е да им се обезбедат на корисниците на платформата за резервации во салони за убавина персонализирани и семантички релевантни предлози за услуги.

Системот поддржува два режима:

- Семантичка препорака врз основа на текстуален промпт на корисникот
- Персонализирана препорака базирана на историјата на закажувања на клиент

1.2 Проблем и решение

Традиционалното keyword пребарување е нефлексибилно и не ги разбира намерите на корисникот. Системот го надминува ова преку векторски embeddings: секоја услуга е претставена со 384-димензионален вектор во семантички простор, а сличноста меѓу услуги и промптови се мери геометриски.

Дополнително, cold-start проблемот (нов корисник без историја) е решен преку fallback на директен текстуален промпт.

1.3 Архитектура

ОФЛАЈН: generate_embeddings.py
Еднократна предобработка: кодирање на услуги → складирање вектори во PostgreSQL
ОНЛАЈН: recommend_for_client.py
Барање во реално време: профил на клиент / текстуален промпт → хибридно рангирање → TOP 5

2. Компоненти на системот

Подсистемот се состои од **две скрипти** со јасно разграничени одговорности: офлајн предобработка и онлајн препорака.

2.1 generate_embeddings.py — Офлајн генерирање на вектори

Примарна намена е да генерира векторски репрезентации за секоја услуга и да ги зачува во базата. Скриптата се извршува еднаш (или при промени на каталогот на услуги).

Атрибут	Вредност	Белешка
Влез	Нема директен	Учитува од базата преку SELECT JOIN
Излез	UPDATE на service.embedding	384-dim float[] вектор по услуга
Модел	all-MiniLM-L6-v2	SentenceTransformer, 384 димензии
Зависности	sentence-transformers, psycopg2	

Тек на извршување

1. SELECT JOIN за учитување на сите услуги со категории
2. За секоја услуга: градење текстуален опис + domain-specific клучни зборови
3. model.encode(text) → 384-dim вектор
4. UPDATE service SET embedding = vector WHERE service_id = id
5. Commit и затворање на конекцијата

Логика за клучни зборови

```
АКО 'facial' е во името ИЛИ 'skin' е во категоријата:
    → skin care, dry skin, hydration, face treatment, healthy skin
АКО 'hair' е во категоријата:
    → hair styling, hair care, hair treatment
АКО 'nail' е во категоријата:
    → manicure, pedicure, nail care
```

2.2 recommend_for_client.py — Онлајн персонализирана препорака

Централна онлајн компонента. Прима client_id, го конструира профил-векторот на клиентот од неговата историја, потоа врши хибридно рангирање и враќа TOP 5 препораки. При отсуство на историја, автоматски преминува во режим на семантичка препорака по промпт (cold-start).

Атрибут	Вредност	Белешка
Влез (редовен)	client_id	ID на клиентот
Влез (cold-start)	Текстуален промпт	При отсуство на историја
Излез	TOP 5 услуги	name | category | price | score
Зависности	sentence-transformers, psycopg2, numpy	

Тек на извршување

6. SELECT embedding на сите закажани услуги за дадениот client_id (JOIN)
7. АКО нема историја → cold-start: client_vector = model.encode(user_query)
8. АКО постои историја → client_vector = numpy.mean(history_embeddings, axis=0)
9. SELECT сите услуги со embedding IS NOT NULL
10. Векторизирана косинусна сличност: similarities = (embeddings @ client_vector) / norms
11. Хибриден score: 0.75 * similarity + 0.15 * rating_score + 0.10 * popularity_score
12. Сортирање, дедупликација → испис на TOP 5

3. Алгоритми за препорака

3.1 Векторски embeddings

Клучниот концепт во основата на системот се **векторски embeddings**: нумерички репрезентации на текст во форма на вектори со фиксна должина. Моделот all-MiniLM-L6-v2 конвертира произволен текст во 384-димензионален вектор, при што семантички слични текстови добиваат геометриски блиски вектори.

3.2 Косинусна сличност

```
cos(theta) = (A . B) / (||A|| x ||B||)
каде: A . B  = Sum(Ai x Bi)     (скаларен производ)
||A||  = sqrt(Sum Ai^2)  (евклидова норма)
```

Евклидова норма (Големина)
Евклидовата норма претставува **геометриска должина** на еден вектор, која се добива како квадратен корен од сумата на квадратите на сите негови координати.
Скаларен производ (Интеракција)
Скаларен производ е математичка операција меѓу два вектора која дава **скаларна вредност** (број), што ја опишува нивната меѓусебна насоченост и „преклопување“ (проекција на еден врз друг).

Вредноста е во [-1, 1]; вредноста 1.0 означува максимална сличност, 0.0 отсуство на сличност. Во имплементацијата се користи NumPy за векторизирана пресметка:

```
embeddings  [N x 384] — матрица на сите услуги
client_vec  [384]     — профил вектор на клиентот
norms       = ||embeddings|| x ||client_vec||
similarities = (embeddings @ client_vec) / norms   → [N]
```

3.3 Хибридно рангирање

```
final_score = similarity x 0.75 + (avg_rating/5) x 0.15 +
              min(popularity/50, 1) x 0.10
```

Семантичката сличност (0.75) е доминантен фактор, додека оценките и популарноста (0.15 + 0.10) ги рафинираат резултатите.

3.4 Персонализиран профил-вектор

```
client_vector = (1/N) x Sum[ embedding(service_i) ]   за i = 1..N
```

Аритметичката средина на историските embedding вектори ја претставува центроидата на преференциите на клиентот. Услугите геометриски блиску до оваа центроида добиваат висок similarity score.

3.5 Cold-start

Кога клиентот нема историја, client_vector се генерира директно преку внесување на промпт: Внесуваш категорија за која сакаш да добиеш препорака.

4. Интеграција со базата на податоци

4.1 Користени табели

Табела	Улога
Service	Главна табела; содржи embedding (float[]), avg_rating, popularity
service_category	Категоризација на услуги (hair, nail, skin...)
Appointment	Поврзува клиент со термин
appointment_service	M:N врска: термин ↔ услуги

Полињата **avg_rating** и **popularity** мора да се претходно додадени во табелата service.

4.2 Складирање на вектори — Векторска база

Системот го имплементира концептот на **векторска база** преку складирање на 384-димензионалните embedding вектори директно во PostgreSQL (тип float8[]). Сличноста се пресметува на апликациско ниво со NumPy.

5. Анализа на перформанси

5.1 Офлајн предобработка на вектори

Генерирањето на embedding за еден текст трае ~5-50ms. Со однапред пресметани и складирани вектори, онлајн препораката изведува само едно енкодирање (на промптот/профилот) и матрична операција — вкупно под 200ms.

5.2 NumPy векторизација

NumPy ги извршува матричните операции преку C/Fortran рутини со SIMD инструкции, елиминирајќи го Python interpreter overhead. За N=1000 услуги и 384 димензии, NumPy е типично 50-200× побрз од Python for-loop.

5.3 Временска сложеност

Операција	Сложеност
Генерирање embeddings (офлајн)	O(N x D), N = услуги, D = 384
Косинусна сличност (онлајн)	O(N x D) — матрична операција
Сортирање на резултати	O(N log N)

6. Заклучок

Подсистемот за препорака претставува практична примена на **векторски бази на податоци** во реален проблем. Системот успешно ги решава клучните предизвици: семантичко разбирање, персонализација преку историски профил-вектор и cold-start fallback. NumPy векторизацијата обезбедува ефикасни пресметки без потреба од специјализирана векторска инфраструктура, а архитектурата е поставена за лесно проширување кон pgvector при пораст на обемот.

Прилог А – Библиотеки и зависности

Библиотека	Верзија	Намена
sentence-transformers	>=2.2.0	Генерирање на 384-dim embedding вектори
psycopg2	>=2.9.0	PostgreSQL конектор за Python
numpy	>=1.21.0	Матрични операции и косинусна сличност
torch (transitive)	>=1.9.0	Backend за sentence-transformers

Инсталација:

```
pip install sentence-transformers psycopg2-binary numpy
```

Прилог Б – Конфигурација и предуслови

Предуслови

- Python 3.8+, PostgreSQL 12+
- Пополнета табела service со полиња: embedding float8[], avg_rating, popularity
- Пополнета табела service_category и appointment/appointment_service

Додавање на полиња доколку ги нема и update на потребни вредности

```
ALTER TABLE service ADD COLUMN IF NOT EXISTS embedding float8[];
ALTER TABLE service ADD COLUMN IF NOT EXISTS avg_rating REAL
DEFAULT 0;
ALTER TABLE service ADD COLUMN IF NOT EXISTS popularity INT;

UPDATE service s
SET avg_rating = sub.avg_rating,
    popularity = sub.popularity
FROM (
    SELECT
        s.service_id,
        COALESCE(AVG(r.rating),0) AS avg_rating,
        COUNT(DISTINCT a.appointment_id) AS popularity
    FROM service s
    LEFT JOIN appointment_service aps ON s.service_id = aps.service_id
    LEFT JOIN appointment a ON aps.appointment_id = a.appointment_id
    LEFT JOIN review r ON a.appointment_id = r.appointment_id
    GROUP BY s.service_id
) sub
WHERE s.service_id = sub.service_id;
```

Прилог В – Поставување и стартување

Инсталација

Создај виртуелна средина: python -m venv venv && source venv/bin/activate

Инсталирај зависности: pip install sentence-transformers psycopg2-binary numpy

Постави ги конекционите параметри (host, port, database, user, password) во двете скрипти

Редослед на извршување

ЧЕКОР 1 — само еднаш (или при промена на каталогот):
python generate_embeddings.py
ЧЕКОР 2 — при секое барање за препорака:
python recommend_for_client.py
→ Внеси client_id (или текстуален промпт при cold-start)