Лабораторна робота №1

Naive RAG (Retrieval-Augmented Generation)

Мета роботи: опанувати навички обробки структурованих даних, створення векторних ембедингів та реалізації семантичного пошуку.

Стек технологій:

Python / Pandas *для обробки структурованих даних.*

ChromaDB *- векторна база даних (зберігає дані локально) з підтримкою індексації HNSW.*

Sentence-Transformers *нейромережева модель для генерації щільних векторів (створення ембедингів - перетворення тексту в цифрові вектори).*

Зміст роботи

Завдання. Дослідити ефективність застосування векторних ембедингів для аналізу текстового контенту шляхом створення інтелектуального сховища даних та проведення його дескриптивного і семантичного оцінювання.

Етапи роботи:

1. *Підготовка даних:* імпорт та первинне опрацювання датасету *netflix_titles.csv*.

2. *Статистичне обґрунтування*

– Розрахунок дескриптивних статистик для числових атрибутів.

– Інтерпретація ключових метрик: обсяг вибірки, заходи центральної тенденції та діапазони значень (min/max).

3. *Розгортання векторного сховища*

– Інтеграція бібліотеки ChromaDB у програмне середовище.

– Екстракція семантичного ядра (колонка description) для подальшої обробки.

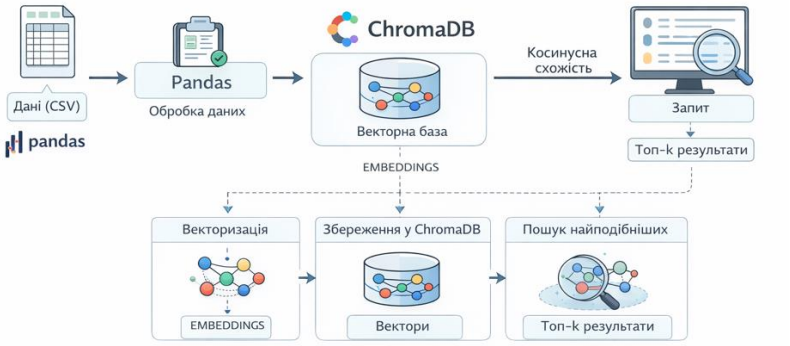
– Ініціалізація бази даних та наповнення векторного простору текстовими ембедингами.

– Валідація цілісності бази шляхом порівняння кількості завантажених об'єктів.

4. *Оцінка ефективності.* Верифікація системи шляхом виконання серії контрольних семантичних запитів.

Методичні рекомендації

Загальна схема семантичного пошуку та аналізу даних з використанням ChromaDB та Pandas.



Підготовка та статистичний скринінг. Завантаження датасету та огляд його статистичних характеристик.

```
import pandas as pd

df = pd.read_csv('netflix_titles.csv')
print(df.describe(include='all'))

df = df.dropna(subset=['description']).reset_index(drop=True)
```

Ініціалізація векторного сховища. Використовуємо *PersistentClient* для того, щоб база даних зберігалася на диску, а не зникала після вимкнення скрипту.

```
import chromadb
from chromadb.utils import embedding_functions

chroma_client = chromadb.PersistentClient(path="./netflix_db")
# Використовуємо модель за замовчуванням (або sentence-transformers)
emb_fn = embedding_functions.DefaultEmbeddingFunction()

collection = chroma_client.get_or_create_collection(
    name="netflix_catalog",
    embedding_function=emb_fn
)
```

Векторизація та завантаження (Batching). Оскільки в датасеті Netflix близько 8000 записів, найкраще завантажувати їх порціями (batching), щоб не перевантажувати пам'ять.

```
batch_size = 5000
for i in range(0, len(df), batch_size):
    batch = df.iloc[i : i + batch_size]
```

```
    collection.add(
        documents=batch['description'].tolist(),
        ids=[str(idx) for idx in batch.index],
        metadatas=[{"title": t, "year": int(y)} for t, y in
zip(batch['title'], batch['release_year'])]
    )
```

Семантичний запит та аналіз метрик. Виконуємо запит та аналізуємо відстані (distances). Чим менша відстань, тим вища релевантність**.**

```
query = "Dramatic story about high school friendship and secrets"
results = collection.query(query_texts=[query], n_results=3)

for i in range(len(results['documents'][0])):
    print(f"Документ: {results['metadatas'][0][i]['title']}")
    print(f"Відстань (Distance): {results['distances'][0][i]:.4f}")
    print(f"Опис: {results['documents'][0][i][:100]}...\n")
```

Обґрунтування обраних методів реалізації:

Статистичний аналіз. Використання методу df.describe() забезпечує первинне розуміння структури датасету, зокрема розподілу категоріальних даних (співвідношення фільмів та серіалів).

Семантична векторизація. Застосування ChromaDB дозволяє автоматично трансформувати текстові описи у багатовимірні вектори, відображаючи їхній зміст у математичному просторі.

Векторний пошук. Механізм запитів базується на обчисленні косинусної схожості.

Система ідентифікує найбільш релевантні фрагменти тексту, знаходячи вектори, що мають мінімальну відстань до вектора запиту користувача.

Контрольні запитання

1. Які задачі можна вирішувати за допомогою *ChromaDB*?
2. Як впливає наявність `NaN` значень у полі *description* на процес векторизації?
3. Поясніть різницю між пошуком за ключовим словом (*SQLLike*) та семантичним пошуком (*ChromaDB Query*).

4. Чому важливо використовувати *PersistentClient* замість звичайного клієнта в ChromaDB?

5. Як змінюється затримка (*latency*) пошуку при збільшенні обсягу даних з 8 тисяч до 1 мільйона записів?

6. Що таке Semantic Drift (семантичний зсув) і як він може вплинути на результати пошуку в медіа-бібліотеках?

7. Поясніть концепцію In-context Learning у контексті RAG-систем (Retrieval-Augmented Generation).

8. Як впливає нормалізація векторів на обчислення косинусної схожості?

9. Поясніть принцип роботи алгоритму HNSW у ChromaDB. Чому він швидший за повний перебір векторів?

10. Як розмірність вектора (наприклад, 384 проти 768) впливає на точність пошуку та обсяг споживаної пам'яті?

11. Що таке лексичний розрив і як семантичний пошук його вирішує?