Технічна документація:
Telegram-бот для підбору парфумів

Огляд проєкту

Цей проєкт являє собою інтелектуального асистента у Telegram, який допомагає користувачам знаходити парфуми. Бот використовує сучасні технології штучного інтелекту для розуміння запитів користувачів та пошуку найбільш підходящих ароматів серед великої бази даних парфумів.

Основна ідея полягає в тому, що система перетворює текстові описи парфумів у математичні вектори (набори чисел), які дозволяють знаходити схожі аромати. Коли користувач описує, який парфум він шукає, система теж перетворює його запит у вектор і знаходить найближчі до нього парфуми з бази даних.

Архітектура системи

Технологічний стек

Проєкт побудовано на мові програмування Python версії 3.11 або новішої. Для роботи з Telegram використовується бібліотека aiogram версії 3.3.0, яка дозволяє створювати ботів з підтримкою складних сценаріїв спілкування. Обробка табличних даних виконується за допомогою pandas, а для перетворення тексту у вектори застосовується бібліотека sentence-transformers.

Векторна база даних Weaviate Cloud зберігає всі парфуми у вигляді векторів та дозволяє швидко шукати схожі аромати. Додатково використовується проста база даних SQLite для збереження профілів користувачів та їхніх уподобань.

Для розуміння складних запитів та генерації відповідей система звертається до мовної моделі Claude 3.5 Sonnet через API сервіс OpenRouter. Ця модель допомагає інтерпретувати нечіткі запити користувачів та формувати зрозумілі рекомендації.

Компоненти системи

1. Препроцесинг даних

Перший етап роботи системи полягає у підготовці даних про парфуми. Вихідні дані зберігаються у двох CSV файлах: fra_perfumes.csv та fra_cleaned.csv. Ці файли містять

інформацію про тисячі парфумів, включаючи їхні назви, бренди, ноти та інші характеристики.

Система завантажує ці файли та виконує їх очищення. По-перше, всі URL-адреси парфумів приводяться до нижнього регістру для уніфікації. По-друге, ароматичні аккорди форматуються у стандартний вигляд "main accords: accord1, accord2, accord3". Застарілі колонки з окремими аккордами видаляються, оскільки вся ця інформація тепер знаходиться в одному полі.

На наступному кроці система генерує текстові описи для кожного парфуму. Опис будується за шаблоном і включає назву парфуму, його призначення (для жінок, чоловіків або унісекс), інформацію про бренд, країну та рік створення. Далі перераховуються верхні, серединні та базові ноти, а також основні аккорди. Таким чином, кожен парфум отримує повний текстовий опис, який потім буде використовуватися для пошуку.

Результат цієї роботи зберігається у файлі fra_final.csv, який містить тільки два поля: URL парфуму та його повний опис. Цей файл є основою для подальшої векторизації.

2. Векторизація

Векторизація це процес перетворення текстового опису парфуму у набір чисел (вектор), який комп'ютер може використовувати для пошуку схожих ароматів. Для цього використовується спеціальна модель sentence-transformers/all-MiniLM-L6-v2, яка вже навчена розуміти значення тексту.

Ця модель перетворює кожен опис парфуму у вектор з 384 числами. Цікаво, що парфуми зі схожими описами отримують схожі вектори, а різні парфуми мають вектори, які відрізняються один від одного. Саме це дозволяє системі знаходити схожі аромати, порівнюючи їхні вектори.

Модель оптимізована для швидкої роботи, що важливо для обробки великої бази парфумів. Вона може векторизувати сотні описів за лічені секунди. Для вимірювання схожості між векторами використовується косинусна подібність, математична метрика, яка показує, наскільки два вектори спрямовані в один бік у багатовимірному просторі.

3. Векторна база даних Weaviate

Всі вектори парфумів зберігаються у векторній базі даних Weaviate Cloud. На відміну від звичайних баз даних, які зберігають таблиці з текстом та числами, векторна база спеціалізується на роботі з векторами та швидкому пошуку схожих елементів.

У базі створюється колекція під назвою "Perfume". Кожен запис у цій колекції містить URL парфуму, його текстовий опис та вектор з 384 чисел, який представляє цей опис. База використовує спеціальний індекс HNSW (Hierarchical Navigable Small World), який дозволяє дуже швидко шукати схожі вектори навіть серед мільйонів записів.

Коли користувач робить запит, система теж перетворює його у вектор і просить Weaviate знайти найближчі до нього вектори з бази. База повертає топ-5 найбільш схожих парфумів за лічені мілісекунди. Завантаження даних у базу відбувається пакетами по 100 записів для оптимізації швидкості.

4. Система пошуку

Система підтримує кілька режимів пошуку парфумів, кожен з яких працює трохи по-різному залежно від того, що саме шукає користувач.

Пошук за описом

Коли користувач описує бажаний аромат словами (наприклад, "хочу щось ванільне та квіткове"), система проходить кілька етапів обробки запиту. Спочатку запит надсилається до Claude API для перевірки. Модель аналізує, чи достатньо інформації у запиті для пошуку, чи потрібно задати уточнюючі питання.

Якщо запит містить конкретні ароматичні ноти або характеристики, система переходить до нормалізації. Claude перетворює всі згадані ноти на стандартні англійські назви, які використовуються у базі даних. Наприклад, якщо користувач написав "вишневий" або "черрі", система перетворить це на "cherry".

Далі нормалізований запит перетворюється у вектор за допомогою тієї ж embedding моделі. Цей вектор надсилається до Weaviate, яка знаходить топ-5 найближчих векторів парфумів. Але це ще не фінальний результат.

Система виконує додатковий крок, який називається reranking. Вона заново обчислює косинусну подібність між вектором запиту та векторами знайдених парфумів, використовуючи повні описи. Це дозволяє підвищити точність результатів. Парфуми з подібністю нижче 0.5 відфільтровуються як нерелевантні.

Фінальні результати надсилаються до Claude разом з запитом користувача. Модель генерує зрозумілу відповідь українською мовою, пояснюючи, чому кожен парфум підходить до запиту, та додає посилання на сторінки парфумів.

Пошук за схожістю до парфуму

Цей режим використовується, коли користувач знає конкретний парфум і хоче знайти щось схоже на нього. Наприклад, "хочу щось схоже на Tom Ford Lost Cherry".

Спочатку система надсилає запит до Claude з проханням витягти назву бренду та парфуму. Модель достатньо розумна, щоб розпізнати навіть неправильно написані назви або назви з опечатками. Якщо назва занадто нечітка, Claude поверне помилку з проханням уточнити.

Далі система шукає цей парфум у своїй базі даних. Якщо знаходить, витягує з його опису верхні та серединні ноти. Якщо парфум відсутній у базі, система знову звертається до Claude з проханням надати інформацію про ноти цього аромату зі своїх знань.

Знайдені ноти об'єднуються в один текст і перетворюються у вектор. Цей вектор використовується для пошуку схожих парфумів, але з додатковою фільтрацією. Система витягає інформацію про стать аромату (жіночий, чоловічий, унісекс) і шукає тільки серед парфумів відповідної категорії або унісекс.

Для цього режиму використовується нижчий поріг схожості (0.39 замість 0.5), оскільки пошук за нотами менш точний, ніж пошук за повним описом. Система шукає серед 20 кандидатів і після reranking залишає найкращі результати.

Claude генерує відповідь, в якій пояснює для кожного знайденого парфуму, чому він схожий на базовий аромат, описуючи стиль, характер та настрій парфумів простими словами без технічних деталей про ноти.

5. Інтеграція з мовною моделлю

Мовна модель Claude 3.5 Sonnet відіграє ключову роль у роботі системи. Вона допомагає розуміти нечіткі запити користувачів та генерувати природні відповіді. Доступ до моделі здійснюється через сервіс OpenRouter, який надає зручний API для роботи з різними мовними моделями.

Система використовує Claude для трьох основних завдань. По-перше, перевірка запитів користувачів. Коли хтось пише боту, модель аналізує, чи містить повідомлення опис ароматів або конкретні ноти. Якщо запит занадто загальний (наприклад, "хочу парфум"), Claude формулює уточнююче питання типу "Можеш описати аромат або назвати ноти?". Якщо запит містить достатньо інформації, модель просто відповідає "ok".

По-друге, нормалізація ароматичних термінів. Користувачі можуть описувати аромати різними словами українською або англійською, з опечатками або синонімами. Claude перетворює всі ці варіанти на стандартні англійські назви нот, які використовуються в базі Fragrantica. Наприклад, "вишня", "вишневий", "черрі" перетворюються на "cherry". Це критично важливо для точності пошуку.

По-третє, генерація фінальних відповідей користувачам. Після того, як система знайшла підходящі парфуми, Claude отримує їхні описи та запит користувача. Модель генерує зрозумілу українською мовою відповідь, де коротко пояснює, чому кожен парфум підходить, описуючи його характер та стиль. До кожного парфуму додається пряме посилання на Fragrantica.

Для різних завдань використовуються різні налаштування температури моделі. Для перевірки запитів та нормалізації нот температура встановлюється на 0.1, що робить відповіді максимально передбачуваними та точними. Для генерації рекомендацій

температура підвищується до 0.4, що дозволяє моделі бути трохи більш креативною у формулюваннях.

Всі запити до Claude мають таймаут 15 секунд. Якщо модель не відповіла за цей час, система повертає користувачу повідомлення про помилку з проханням спробувати ще раз.

6. База користувачів

Для зберігання інформації про користувачів використовується проста база даних SQLite. Це легка база даних, яка зберігається в одному файлі на диску і не вимагає окремого сервера.

База містить таблицю user_profile з чотирма полями: унікальний ідентифікатор користувача Telegram, обрана стать (жіноча або чоловіча), список улюблених парфумів та список улюблених нот. Списки зберігаються як текст, де елементи розділені комами.

Коли користувач вперше запускає бота командою /start, система створює для нього запис у базі даних. Користувач обирає свою стать через кнопки в інтерфейсі Telegram, і ця інформація зберігається у профілі.

Після того, як бот показує користувачеві рекомендації, той може натиснути кнопку "Сподобалось" або "Не сподобалось". Якщо користувач лайкає рекомендації, система зберігає назви всіх показаних парфумів у його профіль. Крім того, система автоматично витягує всі ароматичні ноти з цих парфумів (через Claude API або з описів у базі) і додає їх до списку улюблених нот користувача.

З часом у профілі користувача накопичується інформація про його вподобання. Ця інформація використовується для персоналізованих рекомендацій. Коли користувач обирає режим "Підбір за власними вподобаннями", система формує запит до пошукової системи на основі всіх збережених улюблених нот. Це дозволяє знаходити парфуми, які найкраще відповідають смакам конкретного користувача.

Користувач може в будь-який момент очистити свої дані, якщо захоче почати з чистого аркуша. Функція clear_user_data видаляє всі збережені парфуми та ноти, але залишає сам профіль з вказаною статтю.

7. Telegram бот

Інтерфейс користувача реалізовано у вигляді Telegram бота з використанням бібліотеки aiogram версії 3.3.0. Ця бібліотека дозволяє створювати ботів з підтримкою складних сценаріїв спілкування, кнопок та збереженням стану діалогу.

Бот використовує концепцію FSM (Finite State Machine) для управління діалогами. Це означає, що бот завжди знає, на якому етапі спілкування він знаходиться з користувачем.

Наприклад, чи очікує він, що користувач введе стать, опис бажаного аромату, або назву парфуму для пошуку схожих.

Коли користувач вперше запускає бота командою /start, система перевіряє, чи є він вже зареєстрованим. Якщо ні, бот просить обрати стать через дві кнопки: "Жіноча" або "Чоловіча". Після вибору створюється профіль користувача і показується головне меню з трьома основними опціями.

Перша опція "Пошук за назвою парфуму" призначена для випадків, коли користувач знає конкретний аромат і хоче знайти щось схоже. Після натискання цієї кнопки бот переходить у стан очікування назви. Користувач пише щось на кшталт "хочу щось схоже на Tom Ford Lost Cherry", і система запускає відповідний алгоритм пошуку за схожістю.

Друга опція "Підбір аромату за описом" використовується, коли користувач не знає конкретних назв, але може описати бажаний аромат. Наприклад, "хочу щось ванільне та квіткове для літа". Бот переходить у стан очікування опису, а потім запускає пошук за векторною базою.

Третя опція "Підбір за власними вподобаннями" працює тільки для користувачів, які вже лайкали якісь рекомендації раніше. Система бере всі збережені улюблені ноти з профілю користувача, формує з них запит і шукає парфуми, які містять ці ноти.

Після показу рекомендацій бот завжди додає дві кнопки: "Сподобалось" та "Не сподобалось". Якщо користувач лайкає результати, система автоматично додає показані парфуми та їхні ноти до профілю. Це дозволяє з часом будувати все більш точну картину вподобань користувача.

Бот також має кнопку "Про бота" з короткою інформацією та команду /help, яка показує список доступних команд. Весь інтерфейс реалізовано українською мовою для зручності користувачів.

Важлива особливість реалізації полягає в тому, що після показу рекомендацій бот не просто чекає наступного запиту. Він зберігає список показаних парфумів у стані діалогу, і коли користувач натискає "Сподобалось", система точно знає, які саме парфуми потрібно зберегти. Це робить взаємодію плавною та інтуїтивною.

Ключові алгоритми

Reranking

Reranking це процес уточнення результатів пошуку після того, як Weaviate вже знайшла кандидатів. Хоча векторна база дуже швидка, її результати базуються на попередньо обчислених векторах, які можуть бути не ідеально точними для кожного конкретного запиту.

Алгоритм працює так: для кожного парфуму з списку кандидатів система заново перетворює його повний опис у вектор за допомогою embedding моделі. Потім обчислюється косинусна подібність між цим свіжим вектором та вектором запиту користувача. Це дає більш точну оцінку релевантності.

Всі парфуми з їхніми оцінками схожості сортуються від найбільшої до найменшої. Парфуми з низькою схожістю (нижче встановленого порогу) відкидаються. Фінальний список повертається користувачеві через Claude для генерації відповіді.

Цей додатковий крок займає трохи більше часу, але значно підвищує точність результатів. Користувачі отримують дійсно релевантні рекомендації, а не просто найближчі вектори з бази.

Фільтрація за гендером

При пошуку схожих парфумів система враховує стать аромату, щоб не пропонувати жіночі парфуми тим, хто шукає чоловічі, і навпаки. Алгоритм працює з трьома категоріями: female (жіночі), male (чоловічі) та unisex (універсальні).

Спочатку система витягує інформацію про стать базового парфуму з його опису. Потім формує список дозволених категорій. Якщо базовий парфум жіночий, дозволені категорії це female та unisex. Якщо чоловічий то male та unisex. Універсальні парфуми можуть шукатися серед всіх трьох категорій.

Weaviate шукає топ-20 найближчих парфумів без фільтрації. Після цього система перевіряє опис кожного знайденого парфуму і залишає тільки ті, що належать до дозволених категорій. Це дозволяє максимально використати можливості векторного пошуку, але при цьому дотримуватися логічних обмежень.

Цікаво, що система не покладається на заздалегідь збережене поле зі статтю парфуму. Замість цього вона кожен раз аналізує текстовий опис, шукаючи там ключові фрази типу "for women", "for men" або "unisex". Це робить систему більш гнучкою і стійкою до неповних даних.

Конфігурація та розгортання

Для роботи системи потрібно налаштувати кілька зовнішніх сервісів та створити конфігураційний файл з секретними ключами.

По-перше, потрібен акаунт у Weaviate Cloud. Після створення кластера ви отримаєте URL адресу кластера та API ключ. Ці дані необхідно зберегти у файлі .env як WEAVIATE_URL та WEAVIATE_API_KEY.

По-друге, потрібен доступ до Claude API через OpenRouter. Для цього створюється акаунт на openrouter.ai, поповнюється баланс (модель платна) та генерується API ключ. Цей ключ зберігається як OPENROUTER_KEY.

По-третє, створюється Telegram бот через @BotFather. Бот отримує унікальний токен, який зберігається як TOKEN у файлі .env.

Встановлення залежностей виконується через pip. Основні пакети: aiogram для Telegram, sentence-transformers для векторизації, weaviate-client для роботі з базою, pandas для обробки CSV файлів, scikit-learn для математичних операцій, python-dotenv для читання конфігурації та requests для HTTP запитів.

Перед першим запуском потрібно виконати препроцесинг даних та завантаження у Weaviate. Це робиться один раз шляхом запуску відповідних блоків коду з Jupyter notebook. Процес включає читання CSV файлів, генерацію описів, векторизацію та батч-завантаження у базу.

Після підготовки даних запускається сам бот. Система ініціалізує SQLite базу для користувачів, завантажує embedding модель у пам'ять, підключається до Weaviate та запускає aiogram dispatcher для обробки повідомлень від Telegram.

Важливо, що embedding модель завантажується в пам'ять один раз при старті. Це дозволяє швидко векторизувати запити користувачів без затримок. Підключення до Weaviate також встановлюється один раз і підтримується протягом всієї роботи бота.