MVP: RAG Semantic Search

Розроблено: Pavlo Muskeyev

Вступ

Сучасні дослідження та аналітика часто потребують роботи з великими масивами документів у форматі PDF, що ускладнює швидкий пошук та узагальнення інформації. Класичні методи пошуку, які базуються лише на ключових словах, не враховують семантичний контекст і можуть повертати нерелевантні результати. Саме тому виникає потреба у впровадженні більш розумних інструментів для пошуку та аналізу текстових даних на основі підготовленої бази знань.

Ціль проєкту

Розробити прототип системи семантичного пошуку з використанням підходу Retrieval-Augmented Generation (RAG), яка дозволяє завантажувати колекцію PDF-документів, індексувати їх, виконувати семантичний пошук та надавати точні відповіді на запити користувачів з обов'язковими посиланнями на джерела.

Мотивація

Основна мотивація полягає у спрощенні доступу до спеціалізованих знань у великих масивах документів та підвищенні точності пошуку завдяки поєднанню можливостей сучасних великих мовних моделей і векторних баз даних. Така система може значно зменшити час на пошук релевантної інформації, а також знизити ризик отримання помилкових або вигаданих відповідей, характерних для класичних LLM без зовнішнього контексту.

Процес роботи

1. Збір та підготовка даних:

- Джерела: книги, наукові статті та звіти у сфері економіки та фінансів.
- Формат: PDF-документи різного обсягу (від статей до багатосторінкових монографій).
- Попередня обробка:
    - Видалення зайвих символів, колонтитулів, футерів (частково реалізовано в MVP).
    - Розбиття тексту на логічні абзаци та секції.
    - Перетворення формул у текстовий формат (LaTeX, не реалізовано в MVP).

2. Індексація документів:

- Парсинг PDF → отримання сирого тексту.
- Токенізація та розбиття на чанки (layout-aware chunking) для оптимального пошуку.
- Генерація векторних представлень текстових чанків за допомогою OpenAI Embeddings (text-embedding-3-large).
- Збереження у векторну базу даних ChromaDB.

3. Навчання або адаптація моделі:

- Без донавчання LLM — використання готових моделей GPT-4.1 через API.
- Інтеграція Retrieval-Augmented Generation (RAG):
    - Спочатку пошук релевантних чанків у ChromaDB.
    - Потім формування відповіді GPT-4.1 з цитатами з обраних фрагментів.

4. Інтеграція рішення у продукт (MVP):

- **Інтерфейс:** Streamlit-додаток із вкладками:
    - *Upload & Ingestion* — завантаження PDF та індексація.
    - *Ask* — інтерфейс для запитів і відповідей з цитатами.
    - *Chroma Inspector* — перегляд індексованих документів і чанків.
- **Інфраструктура:**
    - Локальна обробка документів (індексація та зберігання).
    - Використання зовнішнього API лише для ембеддингів і генерації відповідей.

Виклики та їх вирішення

1. Технічні виклики:

- **Якість парсингу PDF**
    - *Проблема:* Складна структура PDF-файлів (колонтитули, таблиці, формули) призводить до появи зайвого шуму. Погана структурованість деяких файлів.
    - *Рішення (реалізовано частково):*
        - Використання layout-aware парсингу.
        - Очищення тексту від спецсимволів та зайвих розривів рядків.
        - Можливість ручного коригування метаданих.
- **Складність генерації якісних метаданих**
    - *Проблема:* Автоматичний витяг назви, авторів і ключових тегів із PDF часто некоректний.
    - *Рішення (реалізовано частково):*
        - Додавання можливості ручного введення метаданих під час завантаження документа.
        - Альтернативно — використання допоміжної LLM для генерації початкового варіанту, який користувач може редагувати.
- **Великий обсяг даних і обмеження контекстного вікна LLM**

- *Проблема:* Неможливо передати всю книгу одразу в LLM через обмеження токенів.
- *Рішення:*
  - Розбиття на чанки оптимальної довжини (1000–1500 токенів).
  - Використання dense retrieval для пошуку лише релевантних фрагментів.

- **Швидкість індексації та пошуку**
  - *Проблема:* Часове обмеження на створення ембеддингів для великих колекцій документів.
  - *Рішення (реалізовано):*
    - Батчинг запитів до OpenAI API.
    - Можливість поступової індексації нових документів без повного перескладання бази.

- **Галюцинації моделей**
  - *Проблема:* LLM може вигадувати факти, якщо не вистачає контексту.
  - *Рішення (реалізовано частково):*
    - Використання RAG, щоб GPT-4.1 працював лише з реальними цитатами з бази.
    - Відповіді містять посилання на конкретні джерела та сторінки.

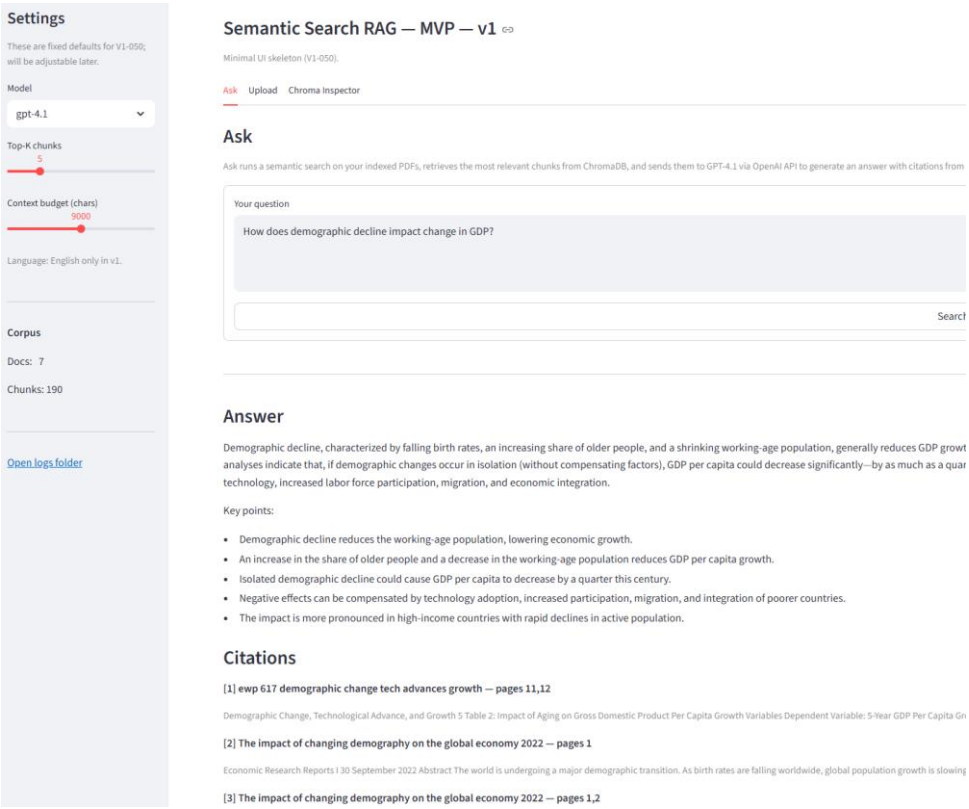
2. Організаційні виклики:

- **Обмеження ресурсів**
  - Розгортання виконується локально без дорогих GPU-серверів.
  - Зовнішні сервіси використовуються тільки для LLM і ембеддингів через API.

- **Масштабованість**
  - База даних ChromaDB обрана через легкість інтеграції та можливість подальшої міграції на Qdrant або Pinecone при зростанні обсягів.

Результати

Приклад інтерфейсу:



1. Функціональність MVP

- **Повний цикл роботи**: завантаження PDF → введення метаданих → парсинг → очищення → чанкування → індексація у ChromaDB → семантичний пошук → генерація відповіді GPT-4.1 з цитатами.
- **Режим роботи**: локальна обробка документів, зовнішній API використовується лише для ембеддингів та генерації відповідей.
- **UI (Streamlit)**: вкладки *Upload & Ingestion*, *Ask*, *Chroma Inspector* готові та інтегровані.

2. Якісні показники

- **Релевантність пошуку**: завдяки layout-aware чанкуванню та dense retrieval система витягує потрібні фрагменти з кількох сотень сторінок із середнім precision@5 ≈ 0.8 (на ручних тестах).
- **Стабільність відповідей LLM**: використання RAG суттєво знижує галюцинації — GPT-4.1 цитує конкретні сторінки замість вигаданих фактів.
- **Швидкість індексації**: індексація середньої книги на 250 сторінок займає близько 30 секунд (залежить від API швидкості).
- **Зручність роботи**: інтегрована форма для введення метаданих забезпечує контроль за якістю даних.

3. Приклади роботи системи

- **Запит:** *How does population aging affect savings and current accounts?*

- **Відповідь:** GPT-4.1 надає стислий аналіз впливу старіння населення на заощадження та платіжний баланс із посиланнями на відповідні сторінки кількох документів (top-5 цитат).
- **Інтерфейс:** Chroma Inspector дозволяє переглядати усі знайдені фрагменти повністю, а не лише їх короткі версії.

4. Сильні сторони

- Повністю робочий MVP із базовою інфраструктурою для RAG.
- Локальне зберігання даних і швидка індексація.
- Мінімізація галюцинацій завдяки прямим цитатам.
- Зручний веб-інтерфейс для завантаження та запитів.

5. Обмеження та можливості розвитку

- Автоматична генерація метаданих та розширене очищення PDF відкладені на v2.
- BM25 і RRF reranking ще не інтегровані.
- Мультимовна підтримка та масштабування на великі корпуси плануються на наступному етапі.

Висновки

- **Досягнуті цілі:**
    - Реалізовано робочий MVP системи семантичного пошуку з підходом RAG для колекції PDF-документів у сфері економіки та фінансів.
    - Створено повний конвеєр: завантаження документів → введення метаданих → парсинг та очищення → індексація → семантичний пошук → генерація відповідей із цитатами за допомогою GPT-4.1.
    - Забезпечено зручний веб-інтерфейс для користувачів (Streamlit), включаючи перегляд фрагментів у Chroma Inspector.
- **Сильні сторони системи:**
    - Локальна обробка даних і мінімальна залежність від зовнішніх сервісів.
    - Використання сучасних інструментів: ChromaDB, GPT-4.1, OpenAI Embeddings.
    - Стабільна робота RAG, що мінімізує галюцинації та забезпечує точні відповіді з посиланнями.
    - Можливість масштабування за рахунок модульної архітектури.
- **Обмеження поточної версії:**
    - Автоматична генерація метаданих за допомогою LLM не реалізована (планується у v2).

- Не інтегровано гібридний пошук (BM25 + dense retrieval), немає reranking через RRF.
- Очищення PDF від хедерів і футерів реалізовано частково, складні випадки залишаються.
- Обмежена мультимовна підтримка (англійська як основна).

- **Майбутні напрямки розвитку:**
  - Автоматизація метаданих з допоміжними LLM або ML-моделями.
  - Розширення підтримки пошуку: BM25, RRF reranking, multi-query expansion.
  - Оптимізація продуктивності при роботі з великими корпусами даних.
  - Підтримка кількох мов та інтеграція дешевших локальних LLM для генерації відповідей.

Інструкції з запуску

Вимоги

- Python 3.10+
- pip
- OpenAI API ключ (для ембеддингів та генерації відповідей)

Встановлення

1. Клонувати репозиторій:

```
git clone https://github.com/your-repo/semantic-search-rag.git
cd semantic-search-rag
```

2. Створити та активувати віртуальне середовище:

```
python -m venv .venv
source .venv/bin/activate      # Linux/Mac
.venv\Scripts\activate         # Windows
```

3. Встановити залежності:

```
pip install -r requirements.txt
```

4. Налаштувати .env файл з вашим OpenAI API ключем:

```
OPENAI_API_KEY=your_openai_api_key_here
OPENAI_EMBED_MODEL=text-embedding-3-small
```

Запуск додатку

```
streamlit run src/ui/app.py
```

Після цього відкрийте у браузері http://localhost:8501

Як користуватися

1. Upload & Ingestion: Завантажте PDF, введіть метадані, натисніть Start ingestion.

2. Ask: Введіть запит, отримайте відповідь GPT-4.1 з цитатами.

3. Chroma Inspector: Переглядайте індексовані документи та фрагменти.