Documentation: Approach, Architecture, and Trade-Offs

1. Synthetic Data Artifacts (PDF)

Data yang dihasilkan merupakan dokumen sintetis berupa petunjuk penggunaan produk fiktif *Robot Pembersih Lantai* yang dibuat menggunakan Python dan disimpan dalam format PDF.

Struktur Dokumen

- File PDF ini terdiri dari beberapa bagian utama:
- Cover → Halaman depan dengan desain produk.
- Kata Sambutan → Perkenalan dari tim atau brand pembuat produk.
- Pendahuluan → Gambaran umum tentang *Robot Pembersih Lantai*.
- Fitur Produk → Penjelasan spesifikasi dan fitur utama.
- Bagian Penutup → Ringkasan dan informasi tambahan.
- Tabel, Chart, dan Grafik → Hasil dari data sintetis yang disimpan dalam database SQL.

Architecture

PDF ini dibuat dengan:

- Library `reportlab` untuk menyusun teks dan gambar ke dalam format PDF.
- Library `matplotlib` & `pandas` untuk membuat grafik dan tabel berdasarkan data sintetis dari SQL.
- Python script otomatis yang menghasilkan teks secara programatik.

Trade-Offs

- Kelemahan: Format PDF tidak selalu mudah diproses oleh model AI tanpa teknik khusus seperti OCR atau structured extraction.
- Kelebihan: Memungkinkan pembuatan dokumentasi realistis dengan elemen visual dan teks yang lebih natural.

2. Synthetic Data Artifacts (SQL)

Data yang dihasilkan adalah dataset sintetis yang memuat informasi produk dalam bentuk tabel SQL.

Struktur Data

Dataset ini terdiri dari 10 kolom dan 1000 baris, mencakup:

- product_id → ID unik produk.
- product_name → Nama produk.
- harga_barang → Harga produk dalam satuan mata uang tertentu.

- battery_life → Durasi pemakaian baterai (jam).
- berat → Berat produk (kg).
- cleaning_mode → Mode pembersihan yang tersedia.
- tahun_garansi → Lama garansi produk.
- ketersediaan_produk → Status ketersediaan (tersedia/tidak).
- rating_pengguna → Skor ulasan dari pengguna.
- tanggal_perilisan → Tanggal rilis produk.

Sebagian data dari SQL ini juga dimasukkan ke dalam file PDF untuk memberikan gambaran lebih jelas kepada pengguna.

Architecture

- Library `sqlite3` digunakan untuk membuat database lokal.
- Library `faker` & `random` digunakan untuk menghasilkan data sintetis yang bervariasi.
- Library `pandas` untuk menyimpan dan memanipulasi data sebelum dimasukkan ke SQL.

Trade-Offs

- Kelemahan: Data sintetis mungkin tidak sepenuhnya mencerminkan pola data dunia nyata.
- Kelebihan: Memungkinkan pembuatan dataset yang aman untuk pengujian tanpa risiko kebocoran data sensitif.

3. RAG Chatbot Interface

Sistem Retrieval-Augmented Generation (RAG) dikembangkan untuk menjawab pertanyaan pengguna berdasarkan isi PDF yang telah diekstraksi.

Proses Implementasi

- Ekstraksi teks dari PDF menggunakan PyMuPDF (`fitz`) dan pdfplumber.
- Chunking Data → Teks dari PDF dibagi ke dalam segmen kecil agar lebih mudah dicari.
- Vector Store → Data teks dikonversi menjadi vektor menggunakan LLaMA embeddings (Groq API).
- Retrieval & Query Matching → Pertanyaan pengguna dibandingkan dengan teks yang sudah dikonversi ke dalam vektor.
- Jawaban diberikan berdasarkan teks yang paling relevan.
- Testing dengan API → Dilakukan uji coba menggunakan berbagai pertanyaan untuk mengukur akurasi dan relevansi.
- Deployment di Hugging Face → Chatbot diunggah ke Hugging Face untuk digunakan secara publik.

Architecture

- PyMuPDF & pdfplumber untuk ekstraksi teks.
- LLaMA embeddings (Groq API) untuk membuat vektor teks.
- FAISS / ChromaDB sebagai penyimpanan vektor.
- LangChain untuk manajemen retrieval dan query.

- Hugging Face API untuk deployment chatbot.

Trade-Offs

- Kelemahan: Model RAG membutuhkan penyimpanan vektor yang besar dan optimasi query retrieval.
- Kelebihan: Jawaban lebih kontekstual dibandingkan chatbot berbasis LLM murni.

4. Formatted Synthetic PDF

Setelah file PDF dibuat dan chatbot diuji, dilakukan proses formatting ulang untuk memastikan data dapat digunakan dalam fine-tuning model AI.

Proses Pengolahan

- Ekstraksi teks & gambar dari PDF untuk memahami struktur dokumen.
- Pembersihan data → Menghapus noise seperti header, footer, dan format yang tidak diperlukan.
- Konversi ke format JSONL agar bisa digunakan untuk fine-tuning LLM seperti
- LLaMA, Gemma, Mistral, dan Phi.
  Penambahan data uji coba → Jawaban chatbot dikumpulkan untuk dibandingkan dengan output dari LLM tanpa RAG.

Architecture

- pdfplumber & PyMuPDF untuk ekstraksi teks.
- pytesseract untuk OCR jika ada teks berbasis gambar.
- JSONL processing dengan Python untuk membuat format dataset yang kompatibel dengan fine-tuning.

Trade-Offs

- Kelemahan: Format JSONL lebih sulit dibaca manusia dibandingkan CSV atau Excel.
- Kelebihan: JSONL sangat cocok untuk fine-tuning model AI karena mendukung format *instruction tuning*.