- Goal: Build a search engine for BPL using RAG.
- Purpose: Make digital resources more accessible with natural language queries.
- Solution: Use RAG pipeline to retrieve documents based on metadata + text, generate AI responses.

Database

- Source: Digital Commonwealth database.
- Size:
    - 1.3M items (text, video, audio, images).
    - 147K full-text OCR documents.
    - Metadata JSON files (up to 135 fields per document).
    - 600K+ still images, fewer than 100K non-text files.
    - Most images have textual descriptions (metadata used for retrieval).
- Data pulled from Digital Commonwealth API.
- Metadata extracted with load_script.py.
- Outputs JSON files (out<BEGIN>_<END>.json).
- Chunking used for large documents to improve search granularity.

Embedding Process

- Uses all-MiniLM-L6-v2 (Hugging Face model).
- Vector size: 384.
- Stored in Pinecone vector store.
- Only key metadata fields embedded due to storage limits.

RAG

- Query Processing:
    - User query converted into embeddings.
    - Compared to stored document vectors.
- Document Retrieval:
    - Uses cosine similarity.
    - Retrieves top 100 closest documents.
    - Metadata fetched from API (slow process).
- Reranking & Content Integration:
    - Uses BM25 for metadata-based reranking.
    - Example: If query mentions "1919," prioritizes matching metadata.
    - Improves response quality.
- Response Generation:
    - Sends retrieved text to GPT-4o-mini.
    - Uses structured prompt to avoid hallucination.
    - UI shows:
        - Generated response.

    - Retrieved snippets.
    - Links to sources.

Vector Store

- Full dataset embeddings (~140GB) too large for local storage.
- Used Pinecone, but cost is high.
- Only a subset of metadata vectors stored.
- API calls for metadata are slow (25-70 sec response time).
- Pinecone storage limits prevent full metadata association.
- Fix: Cache metadata in a local database.
- Vague queries (e.g., "What happened in 1919?") perform poorly.
- Fix: Improve metadata-based pre-filtering.

UI

- Streamlit UI used (easy integration but limited customization).
- Slow retrieval affects user experience.

Deployment

- Hosted on Hugging Face Spaces.
- Uses Pinecone API for vector retrieval.
- GPT-4o-mini for response generation.
- API keys stored in .env.
- Metadata partially embedded, limiting retrieval effectiveness.
- Retrieval uses vector similarity + BM25 reranking.
- Latency issues from API calls.
- Next steps: Focus on better metadata integration and faster retrieval.
-