CANCER LITERATURE SEARCH ENGINE

Information Retrieval Project

BY: Menna Mohsn
Aisha samir
Rola Hany

Problem & Motivation

Problem: Finding relevant cancer research articles in PubMed is challenging due to the large number of abstracts.

Motivation: Enable researchers to quickly access important studies and support scientific research.

Key points:

- Thousands of abstracts in PubMed
- Hard to find relevant studies quickly
- Need for ranked and Boolean search

Project Overview

System Architecture

Workflow Diagram

DATA COLLECTION → PREPROCESSING → INDEXING → RETRIEVAL → USER INTERFACE → EVALUATION

- Simple pipeline to collect, clean, index, and search cancer abstracts.
- Evaluation loop ensures search quality.

Data Collection

- Source: PubMed database via BioPython Entrez API
- Domain: Cancer
- Query: cancer[Title/Abstract]
- Date Filter: 2018–2025
- Dataset Size: 1,000 abstracts
- Metadata collected: Abstract text, disease label

Fetched abstracts in batches (50 per batch) with 0.4s delay to comply with NCBI polite usage. Saved data in both CSV and JSON formats.

Text Preprocessing

STEPS APPLIED:

1. Lowercasing
2. Punctuation removal
3. Tokenization
4. Stop word removal
5. Stemming (Porter Stemmer)

Original

"Cancer immunotherapy shows promising results."

Processed

"cancer immunotherapi show promis result"

Each abstract is cleaned and stemmed to prepare for indexing and retrieval.

INDEXING

Inverted Index

- Data Structure: Python dictionary mapping words → list of document IDs
- Purpose: Enables fast retrieval of abstracts containing query terms
- Storage: Serialized CSV or pickle file

Example structure:

```
inverted_index = {
   "cancer": [1, 2, 5, 10],
   "immunotherapy": [“2", “5",” 7"],
   ...
}
```

Each word points to the abstracts it appears in. This is the core of search efficiency.

Retrieval & Ranking

- Boolean Search: AND, OR, NOT queries
- Vector Space Model (TF-IDF):
- TF-IDF computes importance of terms in each abstract
- Cosine similarity ranks abstracts by relevance

USERS GET A RANKED LIST OF ABSTRACTS RELEVANT TO THEIR QUERY.

Boolean Search

Operations: AND, OR, NOT

```
RES_AND = BOOLEAN_AND(["CANCER",
"THERAPY"], INV_INDEX, ALL_DOCS)
RES_OR = BOOLEAN_OR(["CANCER",
"THERAPY"], INV_INDEX)
RES_NOT = BOOLEAN_NOT(["CANCER"],
INV_INDEX, ALL_DOCS)
```

- **RETRIEVES DOCUMENTS SATISFYING THE QUERY CONDITIONS.**

TF-IDF RANKING

- VECTOR SPACE MODEL: TF-IDF + COSINE SIMILARITY
- PURPOSE: RANK ABSTRACTS BY RELEVANCE

EXAMPLE:

```
RANKER = TFIDFRANKER(DOCUMENTS)
RANKED = RANKER.RANK("CANCER THERAPY", TOP_K=10)
```

- **RETURNS TOP 10 RELEVANT DOCUMENTS WITH SCORES.**

Snippet Generation

- HIGHLIGHTS FIRST OCCURRENCE OF QUERY TERMS IN DOCUMENT
- EXAMPLE FUNCTION: MAKE_SNIPPET(DOCUMENT_TEXT, QUERY_TERMS)
- SHOWS SHORT CONTEXT FOR USER-FRIENDLY DISPLAY

EVALUATION

- Metrics: Precision, Recall, F1, Average Precision, MAP

Example:

```
precision_at_k(retrieved_docs, relevant_docs, k=10)
mean_average_precision(results, relevance_judgments)
```

EVALUATE SYSTEM USING KNOWN RELEVANT ABSTRACTS FOR QUERIES LIKE "BREAST CANCER", "LUNG CANCER".

GUI

- Built using Tkinter

Features:

- Query input
- Boolean AND results
- TF-IDF ranked results
- Clear button
- User-friendly interface for interacting with the system

SCREENSHOT EXAMPLE: DISPLAY SEARCH INPUT, BOOLEAN RESULTS BOX, TF-IDF RESULTS BOX

Demo / Example Queries

- Query "breast cancer" → shows top 10 TF-IDF ranked abstracts
- Boolean AND query "cancer therapy" → shows matching abstracts
- Allows exploring multiple queries interactively

Conclusion

- Successfully built a PubMed IR system
- Supports Boolean search and TF-IDF ranking
- Allows evaluation using IR metrics
- Provides a GUI for user-friendly search

Thank you!