CS-410 Text Information Systems

Final Course Project

Team

Hasham Ul Haq

NetID: huhaq2

Email: huhaq2@illinois.edu

Table of Contents

1. Introduction

Clinical information retrieval systems are increasingly used to help clinicians and coders find evidence for specific medical conditions in large collections of clinical notes. In many production systems, retrieval is still dominated by keyword or fuzzy matching: documents are ranked primarily by lexical overlap with the query.

While simple and fast, these approaches struggle with important clinical nuances:

- Synonymy and concept variants (e.g., *hyperlipidemia* vs. *hypercholesterolemia*).
- Negation and exclusion criteria (e.g., *hyperlipidemia* ***without*** *diabetes*).
- Section-specific evidence (problem list vs. family history vs. plan).

This project, **DocuSherlock**, is a small retrieval workbench designed to explore these trade-offs in a controlled setting. The goal is to **compare multiple retrieval strategies** on a clinical-style corpus:

- Lexical methods: **TF-IDF**, **BM25**, **BM25+RM3** (pseudo-relevance feedback).
- Semantic method: **dense retrieval** with **BioBERT** embeddings and FAISS.
- **Hybrid** method: rank fusion of BM25 and dense scores via **Reciprocal Rank Fusion (RRF)**.

DocuSherlock provides:

1. A Cranfield-style evaluation pipeline over a manually curated set of queries and relevance judgments.
2. A Streamlit workbench that lets users enter queries, inspect results side-by-side across methods, and view evaluation metrics interactively (P@k, nDCG@k, Recall@k).

The central question is: **for realistic clinical queries on this corpus, when do dense and hybrid methods help over strong lexical baselines, and when do they hurt?**

2. Dataset

2.1 Source

The corpus consists of de-identified clinical-style notes scraped from **MTSamples.com**, a public repository of medical transcription samples. These notes resemble real clinical documentation, which is ideal for this project.

In total, the final corpus contains:

- **4,996 notes**
- Spanning multiple specialties (e.g., cardiology, orthopedics, neurology, gastroenterology, general surgery).

2.2 Preprocessing

Preprocessing was intentionally minimal to keep the focus on retrieval models rather than heavy NLP pipelines:

- Removed HTML artifacts and boilerplate.
- Normalized whitespace and line breaks.
- Retained case and punctuation for indexing (Pyserini / Lucene handles tokenization and normalization internally).
- Each note was assigned a unique doc_id (e.g., doc_4818).

The same corpus is used for all methods (TF-IDF, BM25, RM3, dense, hybrid).

3. Retrieval Methods and Indexing

3.1 TF-IDF baseline

As the simplest baseline, I implemented a **TF-IDF** retriever over the full corpus:

- Scored documents using cosine similarity between the query vector and each document vector.
- Returned the top-K documents sorted by decreasing TF-IDF score.

This serves as a sanity check that the pipeline is wired correctly and as a reference point for more advanced models.

3.2 BM25

For a stronger lexical baseline, I used BM25 implemented via Pyserini/Lucene:

- Built a Lucene index over the corpus (one document per note).
- Applied standard BM25 scoring with Pyserini's default parameters (e.g., typical values around $k_1 \sim 0.9$, $b \sim 0.4$; I did not hand-tune these).
- Queried the index with the raw query text, retrieving the top-K documents.

BM25 is expected to outperform TF-IDF on long, heterogeneous documents like clinical notes, due to better length normalization.

3.3 BM25 + RM3 (pseudo-relevance feedback)

To test whether pseudo-relevance feedback helps in this domain, I implemented **BM25 + RM3**:

- First, run BM25 and take the top N documents (pseudo-relevant set).
- Build a feedback language model from these documents.
- Interpolate the feedback model with the original query terms.
- Re-issue the expanded query and re-rank documents.

Parameters (feedback depth, number of feedback terms, interpolation weight) were kept close to Pyserini defaults; no heavy tuning was performed. The expectation was that RM3 might capture additional terminology and synonyms beyond the initial query.

3.4 Dense retrieval with BioBERT + FAISS

For semantic retrieval, I used **BioBERT** to generate dense representations for documents and queries:

- Model: **BioBERT** (clinical / biomedical domain tuned).
- Input truncation: first **512 tokens** of each document.
- Embedding: mean-pooling over the last hidden layer to produce a single vector per document.
- Numeric type: 32-bit floats to preserve quality.
- Index: **FAISS** flat index over all document embeddings, using inner product / cosine similarity.

Indexing the full dataset once with BioBERT takes roughly **20 minutes** on CPU, after forcing PyTorch to a single thread due to a Python 3.13 + Torch compatibility issue. This is a one-time cost and acceptable for this project.

At query time:

1. Encode the query with BioBERT the same way (512 tokens, mean-pool).
2. Search the FAISS index for the nearest neighbors.
3. Return the top-K document IDs and scores.

3.5 Hybrid retrieval (BM25 + Dense via RRF)

Finally, I combined lexical and dense retrieval with **Reciprocal Rank Fusion (RRF)**:

- Run BM25 and Dense retrieval separately for a query, each returning a ranked list.
- For a document at rank r in a given list, assign an RRF score 1 / (k + r) (with a small constant k).
- Sum RRF scores from both lists per document.
- Sort by the fused score and take the top-K.

RRF is attractive because it is simple, robust to differing score scales, and often performs well even without careful tuning.

4. Evaluation Setup

4.1 Queries and relevance judgments

For evaluation, I constructed a small **Cranfield-style benchmark**:

- **10 clinical-style queries**, e.g.:
    - “leg pain without swelling”
    - “hyperlipidemia without diabetes”
    - “chest pain with shortness of breath”
- For each query:
    - Pooled candidates from multiple methods (TF-IDF, BM25, BM25+RM3, Dense, Hybrid).
    - Manually inspected the top candidates and assigned **graded relevance labels**:
        - **2** - clearly relevant, good match to the condition and context.
        - **1** - partially relevant or contains tangential evidence.
        - **0** - non-relevant.

Judgments are stored in qrels.jsonl as (query_id, doc_id, rel) and used for all methods.

4.2 Metrics

I used standard IR metrics computed over the 10 queries:

- **P@10**: Precision at rank 10, treating any rel > 0 as relevant.
- **nDCG@10**: Normalized Discounted Cumulative Gain at 10, using graded relevance (0/1/2).
- **Recall@10**: Fraction of all relevant documents (with rel > 0) retrieved in the top 10.

A shared Python module (eval_metrics.py) computes these metrics.
These metrics are exposed both in a command-line script (src/evaluate_all_methods.py) and in the Streamlit UI, where the user can vary k interactively.

5. Results

Method	P@10	nDCG@10	Recall@10
BM25	0.3700	0.5568	0.5972
BM25+RM3	0.3200	0.4700	0.5294
TF-IDF	0.3400	0.5453	0.5704
Dense (BioBERT)	0.0600	0.0888	0.0694
Hybrid (BM25 + Dense, RRF)	0.3400	0.4029	0.5585

A few immediate observations:

- **BM25** is the best overall performer on this benchmark by **nDCG@10** and **Recall@10**.
- **TF-IDF** is surprisingly competitive with BM25, especially on nDCG@10.
- **BM25+RM3** actually **hurts performance** compared to plain BM25 across all metrics.
- **Dense (BioBERT)** performs poorly in this configuration.
- **Hybrid fusion** improves recall compared to Dense alone and BM25+RM3, but **does not beat BM25** on nDCG@10.

6. Analysis and Discussion

6.1 Lexical methods (TF-IDF, BM25, BM25+RM3)

On this dataset, **lexical methods clearly dominate**.

- **BM25 vs. TF-IDF.**
  BM25 gives a small boost in nDCG@10 and Recall@10 over TF-IDF, but the gap is modest. This suggests that (1) queries usually share **exact terms** with relevant notes, and (2) document length variation isn't extreme enough for BM25's length normalization to matter a lot.
- **BM25+RM3 underperforms.**

RM3 pseudo-relevance feedback **hurts** both precision and recall. With short, specific queries and a noisy feedback pool, expansion tends to add off-topic terms and dilute the core intent. With only 10 queries, a few bad expansions also drag down the average.

Overall, for this benchmark, **plain BM25 is a strong, stable baseline**, and naive RM3 makes things worse rather than better.

6.2 Dense retrieval with BioBERT

Dense retrieval with BioBERT was meant to add **semantic matching** and robustness to synonyms, but in this setup it performs **very poorly** (P@10 ≈ 0.06, similarly low nDCG@10 and Recall@10).

The main reasons are architectural rather than model quality:

- Each note is collapsed into a **single 512-token embedding**, which is too coarse for long, multi-topic clinical notes.
- There is **no passage-level retrieval or chunking**; the model must decide, with one vector, whether a whole document is "about" the query.
- The evaluation set is **tiny** (10 queries), so a few failures make the average look especially bad.
- There is **no cross-encoder reranker** to clean up the top candidates.

In short, dense retrieval here is handicapped by the pipeline: **a naive single-vector-per-note BioBERT setup is not competitive with strong lexical baselines**.

6.3 Hybrid retrieval (BM25 + Dense via RRF)

The hybrid RRF approach partially behaves as intended:

- **Recall@10** (0.5585) is higher than Dense alone and close to BM25.
- It occasionally recovers relevant notes that lexical methods bury lower.

But:

- **nDCG@10** (0.4029) is noticeably worse than BM25 (0.5568) and TF-IDF (0.5453).
- **P@10** (0.34) only matches TF-IDF and lags BM25.

Because the dense retriever is weak, RRF ends up **injecting noise into BM25's ranking**. Hybrid methods only help when each component is reasonably strong and complementary; here, dense retrieval isn't yet a good partner.

6.4 Qualitative behavior

Qualitative inspection in the Streamlit app aligns with the metrics:

- **BM25 / TF-IDF** typically surface notes where query terms appear in the right local context (problem list, HPI, assessment).
- **Dense retrieval** often finds topically related notes (e.g., other chronic conditions) but misses specific constraints like "without diabetes".

- **Hybrid fusion** occasionally pulls these semantically related but irrelevant notes into higher ranks, which explains its lower nDCG.

7. Limitations and Future Work

This project deliberately focused on building a small but realistic workbench rather than a production system. Key limitations:

1. **Tiny evaluation set (10 queries) & insufficient coverage to test all complexities**
   The metrics are noisy and not statistically robust. The main value is **directional insight** and qualitative understanding, not final conclusions.
2. **Document-level dense embeddings**
   As noted, embedding each entire clinical note (or just the first 512 tokens) into a single vector is a poor approximation for complex, multi-topic notes. A better pipeline would:
   - Chunk documents into passages or section-aware segments.
   - Use passage-level dense retrieval.
   - Aggregate scores at the document level.
3. **No explicit negation or section modeling yet**
   The original proposal included **feature-aware reranking** (negation, section titles, synonyms). In this report, the focus is on lexical vs dense vs hybrid methods; feature-aware reranking remains future work. A natural next step would be:
   - Rule-based negation detection (e.g., NegEx-style patterns).
   - Section-aware scoring (boost matches in "Assessment & Plan" over "Family History").
   - Synonym expansion using clinical ontologies (UMLS, SNOMED, etc.).

8. Streamlit Workbench

Alongside the offline evaluation pipeline, I implemented a **Streamlit app** that plays the role of an "IR lab notebook":

- **Search tab**:
  - User enters a clinical-style query.
  - The app runs TF-IDF, BM25, BM25+RM3, Dense (BioBERT), and Hybrid (RRF) in parallel.
  - Results are shown side-by-side in columns with:
    - rank, doc_id, score, snippet
    - expandable full note view for the top hits.
- **Evaluation tab**:
  - Uses the same retrievers and the qrels.jsonl benchmark.
  - Lets the user choose k for P@k / nDCG@k and Recall@k.
  - Runs evaluate_all_methods and displays a metrics table (one row per method).
  - Serves as a quick way to re-run experiments and inspect how metric values change with k.

This interactive workbench made it much easier to debug the retrieval behavior and to understand *why* certain methods underperform.

9. Conclusion

DocuSherlock demonstrates that on a small, realistic clinical corpus:

- **Strong lexical methods (BM25, TF-IDF)** remain very competitive baselines.
- **Naive dense retrieval** (one BioBERT embedding per note) performs poorly and can even harm hybrid methods when fused blindly.
- **Pseudo-relevance feedback (RM3)** is not guaranteed to help; in this setting it broadly hurts performance.

The main takeaway is not that dense retrieval is "bad", but that **clinical IR requires careful modeling of document structure, context (negation, section, temporality), and evaluation**. With a more refined dense pipeline (chunking, cross-encoders) and feature-aware reranking, there is still significant headroom for improvement over pure keyword methods.

For this course project, the combination of a **Cranfield-lite benchmark**, **multi-strategy retrieval pipeline**, and **interactive Streamlit UI** provides a solid foundation for further exploration and a concrete demonstration of how different IR models behave on clinical evidence retrieval tasks.