# Hybrid Lexical + Semantic Retrieval: A Combined BM25 and SBERT Framework for Scientific Document Retrieval

**Poojan Patel**
University of Illinois Chicago
CS 582: Information Retrieval
ppatel539@uic.edu

**Sanjith Jayasankar**
University of Illinois Chicago
CS 582: Information Retrieval
sjaya6@uic.edu

## Abstract

Information retrieval (IR) in scientific domains presents a particularly challenging setting for ranking models. Scientific queries often involve domain-specific terminology, paraphrased findings, and nuanced relationships that are not captured well by simple keyword matching. Traditional sparse retrieval models such as BM25 remain competitive because they reward exact term overlap and are easy to interpret and deploy. However, they suffer severely when authors use different surface forms to describe similar ideas. On the other hand, dense neural encoders such as Sentence-BERT (SBERT) capture semantic similarity in a continuous vector space and can match paraphrases but may underweight rare but crucial scientific terms or acronyms.

This project investigates whether a simple hybrid strategy—combining BM25 and SBERT through a weighted fusion of scores—is sufficient to improve retrieval quality on a realistic scientific IR task. We focus on the SciFact benchmark, where the goal is to retrieve scientific abstracts that provide evidence for or against a claim. We implement three systems: a BM25 baseline, an SBERT-based semantic retriever, and a hybrid model that integrates both signals. We evaluate them using Precision@k, Recall@k, and NDCG@k for $k \in \{5, 10, 20\}$. The hybrid model consistently outperforms both baselines, with especially notable gains in NDCG@10 and Recall@20. In addition to reporting metrics, we analyze qualitative examples, discuss typical failure modes, and reflect on the engineering challenges encountered during implementation. The results suggest that even a relatively simple hybrid design can offer a robust improvement over standalone sparse or dense retrieval for scientific evidence discovery.

## 1 Introduction

Modern scientific research produces an overwhelming volume of literature. For a single topic, hundreds or thousands of papers may be relevant, and manually searching through them is no longer feasible. Systems that can automatically retrieve relevant scientific abstracts in response to natural language questions or claims are therefore increasingly important. The SciFact dataset is one such benchmark: given a scientific claim, a retrieval model must surface abstracts that might support or refute the claim. This requirement mirrors real-world workflows where scientists, clinicians, or policymakers look for evidence to evaluate statements.

Traditional information retrieval has relied heavily on sparse representations of text, where each document is represented as a bag of words with associated weights. Techniques such as TF–IDF and BM25 (Robertson and Zaragoza, 2009) were engineered for this setting and remain widely used. In many cases, these models are surprisingly strong, especially when queries contain distinctive keywords or phrases that also appear verbatim in relevant documents. However, in scientific literature, the same concept can be expressed in many different ways: using synonyms, paraphrases, or abbreviations. Purely lexical models treat these surface forms as unrelated, which leads to recall failures.

The rise of transformer-based models such as BERT (Devlin et al., 2019), and their sentence-level variants like SBERT (Reimers and Gurevych, 2019), offers a complementary approach. Dense models map sentences or documents into a continuous vector space, where semantically similar texts are close under cosine similarity. Dense retrieval systems can successfully match paraphrased claims and are less sensitive to exact word overlap. However, they are not a silver bullet. In specialized scientific domains, pre-trained models may not fully capture domain-specific entities and acronyms, and dense representations can sometimes treat distinct processes as similar because they share general biomedical vocabulary.

These observations motivate the central idea of this project: instead of choosing between sparse and dense retrieval, we combine them. Intuitively, BM25 can ensure that exact scientific terms and abbreviations are respected in ranking, while SBERT can bridge vocabulary gaps and capture paraphrasing. A hybrid system that fuses both scores has the potential to be more robust than either model alone.

Concretely, this work addresses the following question:

> *Does a simple hybrid of BM25 and SBERT improve scientific evidence retrieval performance on the SciFact dataset compared to using BM25 or SBERT alone?*

To answer this question, we implemented all three retrieval modes, constructed an evaluation pipeline, and built a minimal web interface to explore query results interactively. Beyond raw metrics, the project provided valuable experience with practical IR engineering, score calibration, and the trade-offs between interpretability, performance, and complexity.

## 2 Background

This section briefly reviews key concepts in information retrieval, focusing on representations and evaluation metrics that are used throughout the project.

### 2.1 Sparse vs. Dense Representations

In sparse representations, documents are modeled as very high-dimensional vectors where each dimension corresponds to a vocabulary term. Most entries are zero, hence “sparse”. Methods like TF–IDF weight terms by frequency and rarity, while BM25 refines this by incorporating document length normalization and saturation effects for term frequency.

Dense representations instead embed text into a relatively low-dimensional continuous space (e.g., 384 or 768 dimensions). Models such as BERT and SBERT learn these embeddings from large corpora. Similarity is then computed using cosine similarity or dot product. Dense models capture semantics, but their behavior is less transparent, and they require more computational resources.

### 2.2 Evaluation Metrics

We use three standard IR metrics:

**Precision@k.** The fraction of the top-$k$ retrieved documents that are relevant. Higher values mean fewer irrelevant documents in the top-ranked results.

**Recall@k.** The fraction of all relevant documents that appear within the top-$k$ results. Recall is important when it is costly to miss relevant evidence.

**NDCG@k.** Normalized Discounted Cumulative Gain at rank $k$ takes into account the positions of relevant documents: highly relevant documents ranked near the top contribute more to the score. It is particularly suited to measuring overall ranking quality.

These metrics are computed using the SciFact relevance judgments (qrels), which map each query to one or more relevant documents.

## 3 Related Work

This project builds on three strands of prior work: traditional IR, neural semantic search, and hybrid retrieval.

### 3.1 Traditional IR and BM25

The probabilistic relevance framework underlying BM25 (Robertson and Zaragoza, 2009) has been a cornerstone of document retrieval for decades. BM25 is attractive because it is simple, efficient, and interpretable. It remains a very strong baseline, and in some cases still outperforms more complex neural models when queries are short and keyword-focused.

### 3.2 Neural Semantic Search

BERT (Devlin et al., 2019) enabled context-aware representations for text. Building on BERT, Sentence-BERT (SBERT) (Reimers and Gurevych, 2019) proposes a Siamese architecture to generate sentence embeddings that are directly usable with cosine similarity. This made dense semantic search more practical, as it avoids cross-encoding every query-document pair. Numerous works have since applied SBERT-like models to semantic search, question answering, and dialogue systems.

### 3.3 Hybrid Retrieval Approaches

Recent benchmarks such as BEIR have shown that hybrid methods, which combine sparse and dense features, often yield the best performance across a variety of datasets. Some approaches use dense models as re-rankers on top of BM25; others build more sophisticated late-interaction architectures. In contrast, our approach is intentionally simple: we normalize BM25 and SBERT scores and combine them via a weighted sum. This minimal design is easier to implement in a course project, while still capturing the main idea of hybrid retrieval.

## 4 Problem Definition

Formally, we are given:

- A corpus $C$ of scientific documents, where each document $d \in C$ is an abstract (with title).
- A set of queries $Q$, where each query $q \in Q$ is a scientific claim from SciFact.
- Relevance judgments $R$, where $R(q)$ is the set of documents labeled as relevant to query $q$.

For each query $q$, the retrieval system must produce an ordered list of documents $D(q) = [d_1, d_2, \ldots]$ such that relevant documents are ranked as high as possible. We evaluate the top-$k$ prefix of this list using the metrics described earlier.

We specifically compare performance across three retrieval configurations: BM25 (sparse-only), SBERT (dense-only), and Hybrid (BM25 + SBERT).

## 5 Methodology

### 5.1 BM25 Retrieval

The BM25 retriever tokenizes each document into lowercase alphanumeric tokens and builds an inverted index. Queries are processed using the same tokenization. For each query, BM25 computes a score for each document based on term frequency, inverse document frequency, and document length. Only the top-$k$ documents are kept for evaluation and fusion.

From an implementation perspective, BM25 was straightforward to set up using the `rank-bm25` library. One practical lesson is that tokenization choices (e.g., whether to split on hyphens or keep them) can affect performance, especially in scientific domains with hyphenated terms.

### 5.2 SBERT Retrieval

For dense retrieval, we use the `all-MiniLM-L6-v2` Sentence-BERT model. Each document is encoded into a 384-dimensional embedding and stored. At query time, we encode the query into the same space and compute cosine similarity with each document embedding. We then rank documents by similarity and take the top-$k$ candidates.

Encoding the SciFact corpus took a few minutes on CPU, which is acceptable for an offline precomputation step. Query time retrieval involves computing cosine similarities over the precomputed embeddings. For this project, we used a simple brute-force implementation rather than an approximate nearest neighbor index, because the corpus is relatively small (about 5,000 documents).

### 5.3 Hybrid Fusion

Hybrid scoring is performed by taking the union of candidate documents from BM25 and SBERT, normalizing their scores, and computing a weighted average. We use min–max normalization to rescale scores into the $[0, 1]$ range. Let $s_{\text{BM25}}(d)$ and $s_{\text{SBERT}}(d)$ be the raw scores for document $d$. After normalization we obtain $s'_{\text{BM25}}(d)$ and $s'_{\text{SBERT}}(d)$, and define:

$$\text{HybridScore}(d) = \alpha \cdot s'_{\text{BM25}}(d) + (1-\alpha) \cdot s'_{\text{SBERT}}(d).$$

In our experiments we set $\alpha = 0.5$, giving equal weight to both signals. Informally, we also tried values such as $0.3$ and $0.7$ and observed qualitatively similar behavior, suggesting that the method is relatively robust to small changes in $\alpha$.

### 5.4 System Architecture

The overall retrieval pipeline is intentionally designed to be modular, so that each retrieval component can be developed, tested, and reasoned about independently. At a high level, the system follows a multi-stage ranking process: first generating candidate documents from separate sparse and dense retrievers, then normalizing their scores, and finally merging them through a lightweight fusion mechanism. This modular structure proved extremely helpful during implementation because it allowed issues in one component (e.g., tokenization bugs in BM25, or embedding shape mismatches in SBERT) to be diagnosed without affecting the rest of the pipeline.

We chose a simple late-fusion architecture rather than a complex learned interaction model because the goal of the project was not to develop the most sophisticated hybrid retriever possible, but rather to understand and compare the complementary retrieval strengths of lexical and semantic scoring. By keeping BM25 and SBERT as independent modules, it becomes much easier to inspect intermediate outputs, such as the top-k candidates from each model, and to interpret why certain documents rise or fall in the final ranking.

The following ASCII diagram summarizes the flow of information through the system. It is intentionally narrow so that it fits naturally in an ACL column, and it served as a useful conceptual reference during development.

```
Query (scientific claim)
          |
          v
 +-----------------------+
 |     BM25 Retriever    |
 +-----------------------+
          |
          |  top-k sparse results
          |
          v
 +-----------------------+
 |    SBERT Retriever    |
 +-----------------------+
          |
          |  top-k dense results
          |
          v
   Score Normalization
          |
          v
      Weighted Fusion
          |
          v
     Hybrid Ranking
```

Beyond its simplicity, this architecture also makes the system flexible for future extensions. For example, one could plug in a SciBERT encoder in place of SBERT, switch BM25 for a learned sparse retriever (e.g., SPLADE), or add an optional re-ranking stage using a cross-encoder—all without redesigning the pipeline. This modularity was a deliberate design choice, as it mirrors how real IR systems often evolve over time through incremental improvements rather than wholesale rewrites.

## 6 Experimental Setup

### 6.1 Dataset: SciFact

SciFact contains scientific abstracts paired with claims that need to be verified. We use the provided test split, which includes 5,183 documents and 300 query claims. For each document, we concatenate the title and abstract into a single text field. The relevance judgments specify which abstracts contain evidence relevant to each claim.

Table 1 summarizes key statistics.

| **Statistic** | **Value** |
|---|---|
| Number of documents | 5,183 |
| Number of queries | 300 |
| Average abstract length (tokens) | 172 |
| Approximate vocabulary size | 28,000 |
| Domain | Biomedical / scientific |
| Field composition | Title + abstract |

Table 1: Summary statistics for the SciFact dataset.

### 6.2 Implementation Environment

All experiments were run on a standard laptop environment: a quad-core CPU, 16GB RAM, and no dedicated GPU. Python 3.10 was used, along with `sentence-transformers` for SBERT and `rank-bm25` for BM25. The codebase is structured into separate modules for data loading, retrieval models, hybrid fusion, evaluation, and the Streamlit interface.

### 6.3 Evaluation Protocol

We evaluate retrieval performance on the SciFact test split. For each query, each model produces a ranked list of documents. We compute Precision@5, Precision@10, and Precision@20, as well as Recall@5, Recall@10, Recall@20, and the corresponding NDCG values. We then average these metrics across all queries.

It is important to note that we are not training any models; both BM25 and SBERT are used as-is, without additional fine-tuning. The hybrid model only combines scores at test time and does not introduce additional trainable parameters. This makes the comparison cleaner: any difference in performance is due to the retrieval strategy, not differences in training.

## 7 Results

### 7.1 Overall Performance

Table 2 shows the performance of BM25, SBERT, and the hybrid model at different cutoffs. For each $k$, the hybrid model achieves the highest NDCG, as well as higher recall.

### 7.2 Discussion of Quantitative Results

The differences in NDCG are particularly noteworthy. At $k = 10$, the hybrid model achieves an NDCG@10 of 0.7122, compared to 0.6519 for BM25 and 0.6484 for SBERT. This suggests that the hybrid model does a better job ranking highly relevant documents near the top of the list. Recall also consistently improves, meaning that hybrid

| Model | P@5 | R@5 | NDCG@5 |
|---|---|---|---|
| BM25 | 0.1580 | 0.7284 | 0.6360 |
| SBERT | 0.1647 | 0.7413 | 0.6321 |
| Hybrid | 0.1700 | 0.7706 | 0.6890 |

| Model | P@10 | R@10 | NDCG@10 |
|---|---|---|---|
| BM25 | 0.0850 | 0.7740 | 0.6519 |
| SBERT | 0.0890 | 0.7883 | 0.6484 |
| Hybrid | 0.0943 | 0.8377 | 0.7122 |

| Model | P@20 | R@20 | NDCG@20 |
|---|---|---|---|
| BM25 | 0.0460 | 0.8323 | 0.6672 |
| SBERT | 0.0478 | 0.8440 | 0.6630 |
| Hybrid | 0.0497 | 0.8833 | 0.7240 |

Table 2: Retrieval performance on the SciFact test set. The hybrid model achieves the best performance across all metrics and cutoffs.

retrieval recovers a larger fraction of relevant documents within the top-$k$ results.

The gains are modest but consistent, which is realistic for combining two strong baselines. The results support the intuition that BM25 and SBERT provide complementary signals: when one model fails, the other often compensates.

## 8 Qualitative Analysis

While aggregate metrics are important, they do not fully explain why a hybrid model is better. To build intuition, we examined individual queries and their top-ranked results.

### 8.1 Case Where BM25 Misses

For a query about "lineage-specific regulatory behavior in stem cells", BM25 heavily favored documents that happened to contain the exact token "lineage" or "regulatory", but several of these documents were only loosely related. SBERT, on the other hand, surfaced abstracts that discussed differentiation and regulatory pathways in stem cells, even when they did not use the exact same phrasing. The hybrid model placed the SBERT-retrieved abstracts higher than BM25-alone, reflecting the benefit of semantic matching in this context.

### 8.2 Case Where SBERT Misses

In a query involving "nanoparticle-enhanced cellular uptake", SBERT retrieved some documents that mentioned general cellular signaling but did not mention nanoparticles at all. BM25, by contrast, placed papers explicitly mentioning "nanoparticles" among the top results. The hybrid model effectively balanced these signals: documents that mentioned nanoparticles and were semantically aligned with the query were ranked higher than those that were only loosely related.

## 9 Error Analysis and Challenges

### 9.1 Observed Failure Modes

Through manual inspection, several patterns emerged:

- SBERT sometimes grouped together abstracts that shared general biomedical terms (e.g., "pathway", "regulation") even when the specific processes were unrelated.
- BM25 occasionally over-scored very long abstracts that repeated the query terms multiple times without being truly central to the claim.
- Hybrid retrieval could still fail when both BM25 and SBERT were misled by noisy or ambiguous queries.

These issues highlight that hybrid retrieval is not a cure-all; it reduces the weaknesses of each component but does not eliminate them entirely.

### 9.2 Practical Engineering Challenges

From an implementation standpoint, several practical challenges arose:

- Efficiently encoding thousands of documents using SBERT without a GPU required some attention to batching and memory usage.
- Ensuring that IDs, embeddings, and BM25 indices all aligned correctly was non-trivial; small bugs could silently corrupt evaluation.
- Designing an evaluation pipeline that could easily switch between BM25, SBERT, and hybrid modes helped in debugging and comparison.

These challenges were valuable learning experiences in building IR systems beyond toy examples.

## 10 Conclusions and Future Work

This project set out to test whether a simple hybrid combination of BM25 and SBERT could improve scientific document retrieval on the SciFact dataset. The answer appears to be yes: across all metrics and cutoffs, the hybrid model outperformed both BM25 and SBERT alone. The improvements are not dramatic but are consistent and meaningful, especially considering how simple the fusion method is.

There are several clear directions for future work:

- **Domain-Specific Encoders:** Using models such as SciBERT or PubMedBERT could improve dense representations for biomedical text.
- **Learned Fusion:** Instead of manually choosing $\alpha$, one could train a small model to learn query-dependent fusion weights.
- **Re-Ranking with Cross-Encoders:** A cross-encoder that jointly processes queries and candidate documents could be used to rerank the top results, potentially improving NDCG further.
- **Scalability:** For much larger corpora, approximate nearest neighbor search (e.g., with FAISS) would be necessary for efficient dense retrieval.

Overall, the project demonstrates that even within the constraints of a semester-long course, it is possible to implement and empirically evaluate meaningful IR design choices using real-world scientific datasets.

## 11 Team Work Division and Reflections

This project was completed collaboratively by two team members, Poojan Patel and Sanjith Jayasankar. Both members contributed roughly equally to all major phases of the work, including project planning, implementation, and writing.

On the implementation side, we worked together on setting up the SciFact dataset, implementing BM25 and SBERT retrieval pipelines, and integrating the hybrid fusion module. We jointly debugged issues related to tokenization, score normalization, and alignment between document IDs, embeddings, and relevance judgments. Both members also participated in designing the evaluation pipeline and interpreting the resulting metrics.

For the Streamlit interface, we collaborated on the overall design and shared the work of wiring the retrieval back-end to the user interface. The final report was also a shared effort: we discussed the structure together, and both contributed text, edits, and revisions across sections, including the introduction, methodology, results, and conclusions.

Overall, the project felt like a genuine joint effort. We both gained a deeper understanding of the trade-offs between lexical and semantic retrieval, as well as practical experience building and evaluating an end-to-end IR system.

## A Appendix: Example Query and Ranked Results

To make the behavior of the hybrid system more concrete, we include here a simplified example based on one of the SciFact claims. For a claim about “0-dimensional biomaterials” and their inductive properties, the BM25 and SBERT models surfaced different but overlapping sets of candidate documents. The hybrid model ranked documents that mentioned both biomaterials and inductive properties, and whose abstracts semantically aligned with the query, higher than documents that matched only one of these aspects. This behavior aligns with our intuition that hybrid retrieval should combine precise term matching with semantic understanding.

## B Appendix: Streamlit Interface

In addition to the local version of the system, we deployed a lightweight Streamlit web interface that exposes BM25, SBERT, and Hybrid retrieval modes for interactive exploration of the SciFact corpus. The deployed demo is available at:

`https://poojan-9603-cs582project-appmain-t5npoi.streamlit.app/`

This interface was useful during development for sanity-checking retrieval behavior and for visually comparing the different ranking modes.

## References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*.