Original Research Article

How Can Disparate Retrieval Methods Work Together Effectively?

Sankalp Tank

Raleigh Charter High School, 1307 Glenwood Ave, Raleigh, NC 27605, United States

ABSTRACT

Information retrieval (IR) methods, systems that find relevant information from large datasets, are separated into sparse and dense methods. This study investigates how these two types of methods can work in tandem to optimize speed, indexing cost, and accuracy when answering structured queries on large datasets. This research implemented and benchmarked multiple retrieval methods, ranging from sparse methods TF-IDF, BM25, and an enhanced version coined SUPER_BM25, to dense methods SPLADE and COLBERT. These methods were queried to measure the recall, indexing time, and query time of each. The results indicated that dense methods achieved fast retrieval times at the cost of precision; conversely sparse methods were incredibly accurate, but they took significantly more time. Based on these results, a funnel system of these disparate methods was created, where each method worked in tandem to optimize speed and accuracy. This funnel system reduced indexing time by 23.7% and query time by 99% when compared with COLBERT while retaining comparable recall scores. The funnel system achieved these high indexing and query speeds by having TFIDF, BM25, and SUPER_ BM25 cull 90.4% of the dataset, then giving SPLADE and COLBERT the remaining data to accurately rank it. This hybrid funnel approach presents a scalable and cost-efficient framework for real-word information retrieval, enabling faster, more accurate search across large datasets.

Keywords: Sparse Retrieval; Dense Retrieval; Information Retrieval; Recall; Precision; BM25; TF-IDF; SPLADE; COLBERT

INTRODUCTION

Information retrieval is a cornerstone of modern computing; information retrieval powers modern search engines, knowledge bases, and most question-answer systems (8). Information retrieval methods can be broadly divided into sparse and dense retrieval methods. Sparse retrieval methods such as BM25 and TF-IDF represent text using keywords and their frequencies, finding matches based on exact words that appear in both the query and document (1). Dense retrieval methods such as SPLADE and COLBERT use neural networks to represent text as numerical vectors, finding matches based on semantic meaning (2).

Sparse methods offer efficiency, but lack semantic interpretation and thus output less accurate results. Contrarily, dense methods' utilization of neural networks results in more accurate outputs due to semantic interpretation, but at the cost of lower speeds (4). Sparse and dense retrieval methods are well studied individually, but there exist few approaches that attempt to combine them to utilize the accuracy of sparse methods and the speed of dense methods (3). This gap creates an opportunity to explore hybrid retrieval systems that integrate the strengths of sparse and dense approaches (5).

Corresponding author: Sankalp Tank, E-mail: sankalp.tank@gmail.com.

Accepted November 10, 2025
https://doi.org/10.70251/HYJR2348.36588593

This research posits the question, how can information retrieval and machine learning retrieval methods be implemented in tandem to optimize speed, indexing cost, and accuracy over a large-scale dataset? Its objectives are to measure different retrieval methods and create and optimize a system where they work in tandem based on that data.

The scope of any data retrieval research is fundamentally defined by the dataset on which the research is conducted. This research leveraged Wikipedia articles and queries relevant to those articles (9). The study employs an experimental design in which each retrieval method is implemented, tested, and optimized individually before integration into a hybrid system (5).

METHODS AND MATERIALS

This study adopted an experimental framework to evaluate and compare information retrieval methods, including two sparse approaches (BM25, TF-IDF), two dense approaches (COLBERT, SPLADE), and a custom hybrid method (SUPER_BM25). BM25 and TF-IDF rank documents using frequency-based weighting (1), while COLBERT and SPLADE leverage transformer-based representations to capture semantic similarity (2, 4). The custom SUPER_BM25 method extends BM25 by filtering irrelevant articles at query time and incorporating semantic expansion using FastText word vectors (10). This approach was developed after comparative testing with alternative embeddings, with FastText selected for its better balance between relevance and noise reduction relative to models like Word2Vec (11), enhancing recall for queries exhibiting lexical variation (10). Both SPLADE and COLBERT implementations were optimized for graphics processing unit (GPU) execution using PyTorch, resulting in order-of-magnitude improvements in indexing and retrieval speeds (4, 12). The methods were tested on the full Wikipedia corpus using structured queries (e.g., "What is the largest city in California?") to assess recall, efficiency, and computational cost (5).

The Wikipedia dataset was obtained from publicly available XML dumps (9) and converted into Parquet format to enable efficient storage and retrieval (13). Only open-source, publicly accessible data were used, ensuring the absence of personal or sensitive information (9). All experiments were conducted on a system with an AMD Ryzen 9 7900X 12-core (24-thread) CPU running at approximately 4.7 GHz; 64 GB of physical RAM; an NVIDIA GeForce RTX 4090 GPU; and a 64-bit Windows 11 Home operating system (Version 10.0.26100 Build 26100). The software environment used Python 3.12. The experiments utilized python libraries: NumPy 1.26, pandas 2.1, scikit-learn 1.7, rank_bm25 0.2.2, and pytorch 2.0. All code was executed using the official library APIs without modification. The structured queries were designed to align with the dataset and to include both relevant and irrelevant documents, facilitating recall evaluation (5). The study measured three dependent variables: recall (the fraction of relevant documents retrieved), indexing time (the total time required to construct the retrieval index), and query time (the average processing time per query) (5). These metrics were evaluated for each retrieval method across multiple test queries (5).

The procedure of measuring these variables began with dataset preparation (9, 13). This consisted of extracting and processing the Wikipedia dump and converting this extraction to Parquet format (9, 13). The retrieval methods TF-IDF, BM25, SPLADE, and COLBERT were all implemented using open-source libraries (1). SPLADE and COLBERT utilized GPU acceleration to increase their indexing and query time significantly (4, 12). SUPER_BM25 was created by filtering out articles in the query and by testing different models for semantic expansion on the existing BM25 program (10). After testing these models, the FastText model was chosen as other models such as Word2Vec had many irrelevant expansions and created significant noise, harming recall (10, 11).

Four structured queries: "most generic city in the United States", "largest city in Texas", "richest city in California", "second largest city in the United States", were run through each retrieval system, with recall, indexing time, and query time recorded for analysis (5).

RESULTS

Figure 1 presents the indexing times for each retrieval method. Figure 1 demonstrates that SPLADE has the highest indexing time due to its usage of transformers and large sparse representation (4), while SUPER_BM25 achieves the fastest indexing due to semantic expansion only applying to the query, articles were filtered out, having it index less than BM25 (10).

Figure 2 tracks query time and makes it clear that COLBERT has by far the highest querying time due to requiring many fine-grained token-to-token similarity calculations (2). Naturally, the two sparse retrieval

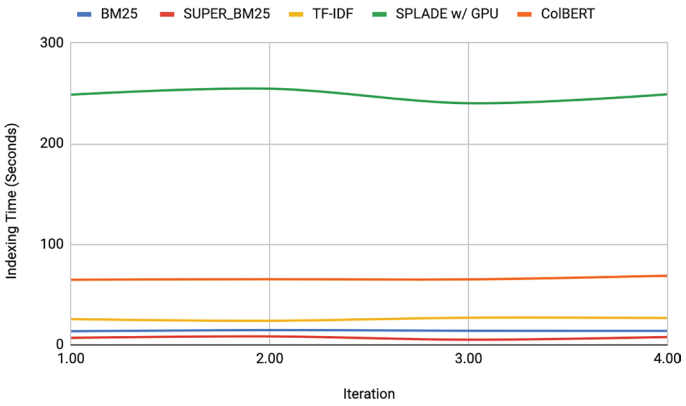

Figure 1. Average Indexing Time of Each Retrieval Method After Experimentation on Structured Queries.

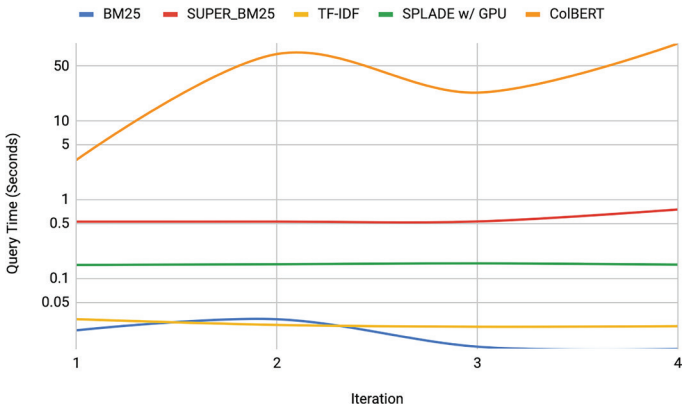

Figure 2. Average Query Time of Each Retrieval Method After Experimentation on Structured Queries.

methods have incredibly small query times, due to their simplicity (1, 4).

Figure 3 quantitatively assesses recall by comparing the top 200 ranked Wikipedia articles retrieved by each method for four structured queries against a curated set of ground-truth relevant articles for each query. The results indicate that COLBERT achieves the highest recall, substantially outperforming the other methods, followed by SPLADE and then the custom hybrid SUPER_BM25 (2, 4, 10). The sparse methods, BM25 and TF-IDF, demonstrate considerably lower recall, with TF-IDF yielding a notably low recall score of 0.323 (1, 5).

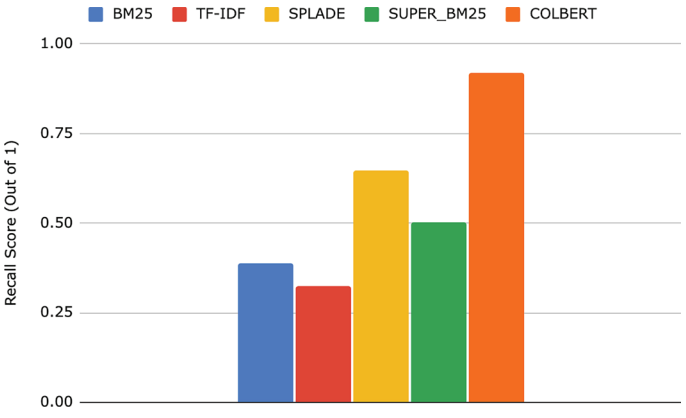

Figure 3. Average Recall Score of Each Retrieval Method After Experimentation on Structured Queries.

Evaluation of system benchmarks reveals a tradeoff between recall and efficiency (5). COLBERT, despite delivering the greatest number of relevant documents within the top 200 results, incurs the highest computational cost and the longest indexing and query times due to dense semantic encoding and (2, 4). In contrast, TF-IDF, which ranks solely by term frequency–inverse document frequency statistics, achieves the lowest recall but offers the fastest runtime (1). BM25, while introducing normalization and saturation factors to TF-IDF, attains moderate recall with a corresponding increase in computational time (1).

SUPER_BM25, augmented with FastText-driven semantic expansion, demonstrates improved recall over BM25 but at the expense of much slower indexing and query times (10). SPLADE balances efficiency and effectiveness by leveraging sparse transformer-generated term expansions, securing recall higher than sparse methods but slightly lower than COLBERT, while also maintaining greater computational efficiency due to GPU optimization (4, 12).

Based on these comparative findings, a retrieval funnel architecture was developed in which each method progressively reduces the dataset size, passing a filtered candidate set to subsequent, more computationally intensive retrieval models for refined re-ranking and recall maximization (5).

Figure 4 illustrates the architectural pipeline of the

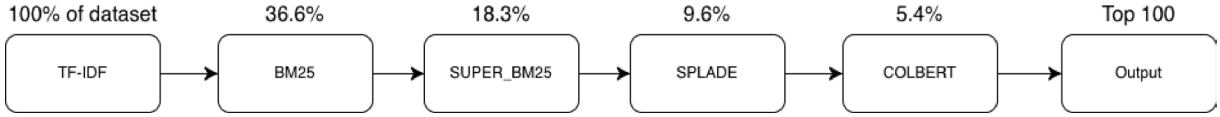

Figure 4. Outline of the Funnel System Displaying Percent of Original Dataset Inputted to Each Retrieval Method.

funnel retrieval system, detailing the sequential role and placement of each retrieval method (5). The ordering of components was empirically determined through initial benchmarking, optimizing for both efficiency and recall (5). In the first stage, rapid sparse retrieval techniques (such as BM25 and TF-IDF) act as coarse filters, reducing the candidate set to approximately 18.3% of the original Wikipedia corpus while ensuring high throughput and minimal computational overhead (1). SUPER_BM25 then culls this 18.3% of the dataset to 9.6%, ensuring no relevant data is filtered out. The funnel culminates with high-recall dense methods COLBERT and SPLADE filtering through this remaining 9.6% (2, 4). These models are tasked with fine-grained semantic matching and ranking, guaranteeing that relevant documents preserved in earlier stages are effectively prioritized (5). By filtering out 90.7% of the data, the dense retrieval methods have less data to sort and are thus more efficient.

This stratified approach dramatically improves system efficiency: in practice, the COLBERT dense retriever is required to rank only 5.4% of the dataset, rather than being applied exhaustively to the entire corpus (2, 5). The funnel system was rigorously benchmarked across critical performance dimensions, including end-to-end indexing duration, per-query latency, and overall recall, validating the operational trade-offs and highlighting the effectiveness of staged hybrid retrieval pipelines (5).

Figures 5, 6, and 7 present comparative performance benchmarks for each retrieval method and for the funnel pipeline (5). Figure 5 quantitatively demonstrates that the funnel system exhibits markedly reduced indexing time relative to SPLADE and COLBERT, achieving an efficient initial corpus reduction through lightweight sparse retrieval and thereby minimizing the workload for downstream dense models (2, 4). Figure 6 further reveals that the funnel's query latency is significantly lower than COLBERT, and outperforms even computationally-intensive hybrid approaches like SUPER_BM25, highlighting the benefit of early-stage sparsity in limiting the search space for subsequent semantic re-ranking (2, 10). Figure 7 shows that the funnel system preserves recall equivalent to that of COLBERT by accurately retaining relevant candidates during the sparse filtering phase and exploiting dense transformer-based models for high-fidelity ranking in the final stage (2).

Ultimately, the funnel pipeline achieves a favorable tradeoff, matching the recall performance of state-

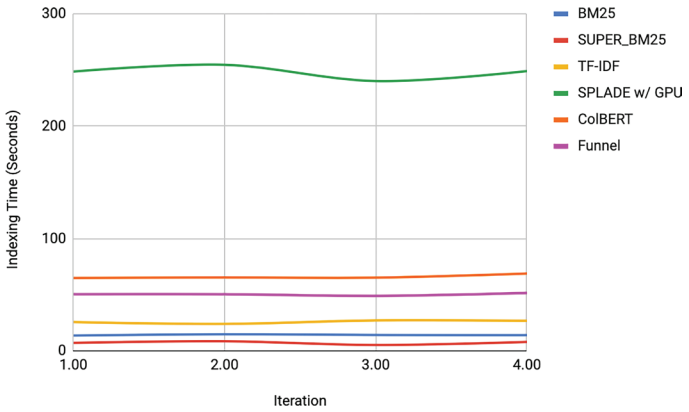

Figure 5. Average Indexing Time of Each Retrieval Method and Funnel After Experimentation on Structured Queries.

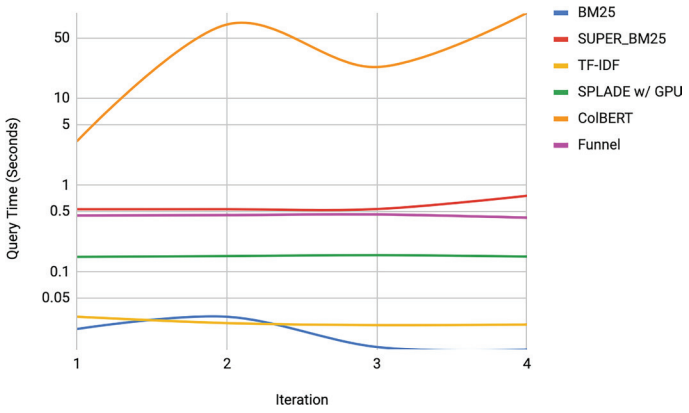

Figure 6. Average Query Time of Each Retrieval Method and Funnel After Experimentation on Structured Queries (Log Scale for Visibility).

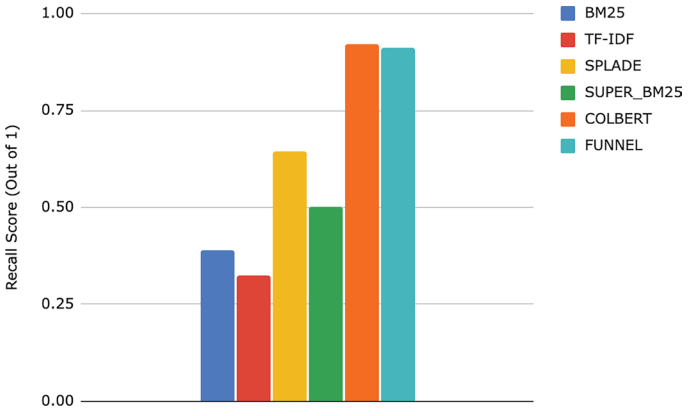

Figure 7. Average Recall Score of Each Retrieval Method and Funnel After Experimentation on Structured Queries.

of-the-art dense retrieval methods such as COLBERT while attaining notably faster indexing and query response times (2, 5). The staged approach—rapid candidate pruning via efficient sparse retrieval, followed by computationally intensive semantic encoding on a reduced set—enables scalable and accurate information retrieval suitable for large-scale deployments (5). This design demonstrates the practical viability of cascaded hybrid retrieval as a means of reconciling the speed-accuracy trade-off intrinsic to state-of-the-art sparse and dense architectures (3, 5).

DISCUSSION

The funnel-based retrieval system developed in this work effectively synthesizes the computational efficiency of sparse vector models with the superior recall rates characteristic of dense neural methods (1, 2, 4, 10). By initially employing sparse retrieval (e.g., BM25 or TF-IDF) to filter out the majority of non-relevant documents based on high-dimensional, term-frequency vectors, the system rapidly narrows the search space with minimal resource overhead (1). The resulting candidate set is then re-ranked using semantically richer dense retrieval models, such as COLBERT or SPLADE, leveraging transformer-based embeddings to maximize semantic matching and recall (2, 4). Empirically, this two-stage pipeline maintains the top-end recall of COLBERT, while achieving considerably reduced latency and computational requirements (5).

These results demonstrate the inherent limitations of exclusively sparse or dense retrieval schemes: sparse methods, though highly efficient for exact keyword matches, exhibit substantial performance drops with paraphrased or synonym-rich queries (1); dense methods, on the other hand, offer robust semantic recall but necessitate significantly greater computational power due to neural encoding, often implemented on GPUs for tractability (1, 12). The hybrid architecture, therefore, operationalizes a balanced solution—exploiting both lexical precision and contextual depth (3, 5). This approach addresses an important industry consideration: Large-scale organizations currently depend on resource-intensive LLMs for structured search, incurring substantial expense and latency (6). The proposed funnel system offers a cost-effective solution, delivering recall competitive with dense models without the prohibitive infrastructure demands of end-to-end LLM search (7).

CONCLUSION

The experimental objectives—benchmarking retrieval quality, engineering an integrated pipeline, and optimizing recall, index build time, and query throughput—were met (5). Notably, observed indexing and inference times for dense models (particularly COLBERT and SPLADE) highlight the practical impact of hardware acceleration; with GPU optimization, disparities in efficiency were mitigated, reinforcing the importance of compute infrastructure to real-world IR deployment (12). These findings indicate that hardware constraints can decisively shape system viability (12).

Limitations of the current study include the exclusive use of Wikipedia, limiting heterogeneity, and a small set of highly-structured queries (5, 9). The evaluation metrics emphasized recall over other user-centered metrics such as top-k precision or holistic retrieval quality (5). Additionally, the dependency on GPU acceleration for dense models may not extend to environments with limited hardware access (12).

For future research, it is recommended that the retrieval funnel be tested with more diverse queries, including semi-structured and free-text questions, and on heterogeneous datasets (e.g., scientific literature or social media posts) to validate generalizability (7). Adaptive cut-off thresholds for transitioning between sparse and dense stages, as well as comprehensive benchmarks against LLMs on accuracy, cost, and speed, should be pursued to establish this system's competitiveness for enterprise applications (3, 7, 14).

The hybrid pipeline created in this paper achieves a favorable efficiency-accuracy tradeoff, providing a retrieval system that is both fast, but also outputs relevant results. By using fast, sparse retrieval methods to cull large datasets, slow, dense methods can filter the remaining dataset and output relevant results based on the inputted query.

ACKNOWLEDGMENTS

I would like to thank my mentor Joe Isaacs from Cambridge University for his guidance and assistance with this research.

FUNDING SOURCES

The author declares that they have not received any funding to conduct any research for this paper.

CONFLICT OF INTEREST

The author declares that there are no conflicts of interest regarding the publication of this article.

REFERENCES

1. Robertson S, Walker S, Jones S, Hancock-Beaulieu M, Gatford M. Okapi at TREC-3. Proceedings of TREC-3. 1994; 109-126. https://doi.org/10.6028/NIST.SP.500-225.routing-city
2. Khattab A, Zaharia M. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2020; 39-48. https://doi.org/10.1145/3397271.3401075
3. Zhan A, Mao N, Lin J, Callan J, *et al*. Optimizing dense retrieval model training with hard negatives. Proceedings of the 38th International Conference on Machine Learning. 2021; p12251-12261.
4. Formal S, Rosset L, Segalovich S, Elhadad M. SPLADE v2: Sparse lexical and expansion model for information retrieval. *Proceedings of the 44th European Conference on Information Retrieval*. 2022; P198-210.
5. Lin Y, Ma J, Ding X, Liu J, *et al*. A survey on sparse and dense hybrid retrieval methods. arXiv preprint arXiv:2307.10667, https://arxiv.org/abs/2307.10667 (2023).
6. OpenAI. GPT-4 Technical Report. https://cdn.openai.com/papers/gpt-4.pdf (2023).
7. Karpukhin M, Oguz B, Min S, Lewis P, *et al*. Dense passage retrieval for open-domain question answering. *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing.* 2020; P6769-6781. https://doi.org/10.18653/v1/2020.emnlp-main.550
8. Manning C, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge University Press. 2008. https://doi.org/10.1017/CBO9780511809071
9. Wikimedia Foundation. Wikipedia database download. https://dumps.wikimedia.org/backup-index.html (2024).
10. Bojanowski P, Grave E, Joulin A, Mikolov T. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*. 2017; 5: 135-146. https://doi.org/10.1162/tacl_a_00051
11. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv:1301.3781, https://arxiv.org/abs/1301.3781 (2013).
12. Paszke A, Gross S, Massa F, Lerer A, *et al*. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems.* 2019; 32: 8024-8035.
13. Apache Parquet documentation. https://parquet.apache.org/docs/overview/ (2024).
14. Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. 2019; P4171-4186. https://doi.org/10.18653/v1/N19-1423