WTSS: Benchmarking Expertise Retrieval from Unstructured Text in the Engineering Domain

Yuang Hou
Wade Trim
yhou@wadetrim.com

Rob Sinclair
Wade Trim
rsinclair@wadetrim.com

***Abstract*—Effective skill search is critical for engineering firms where managers must quickly locate qualified staff. Existing retrieval methods such as BM25 and embedding-based approaches have been widely studied on general-purpose benchmarks, but their performance on domain-specific technical content remains largely unexplored. We introduce WTSS (Wade Trim Skill Search), a synthetic benchmark dataset derived from queries and skill descriptions collected from our production-deployed enterprise skill search system that is available to approximately 800 internal users. With this data set, we investigate two central issues: (i) the optimal size of the candidate set, and (ii) the potential of a large language model (LLM) as a list-wise reranker where output format consistency is critical. We compare state-of-the-art sparse and dense models in the retrieval stage and test the limitation of an LLM as a list-wise reranker. While experiments are conducted on WTSS, our approach generalizes to other domain-specific free-form search tasks. Results show that dense retrieval consistently outperforms sparse retrieval, achieving 90% recall with a candidate set size of $1.2N$, where N is the average number of relevant documents per query. An LLM demonstrates stable list-wise reranking performance for candidate set sizes up to $k = 40$. This work provides the first systematic evaluation of free-form text search for engineering content and offers WTSS as a realistic benchmark dataset for future research.**

***Index Terms*—information retrieval, sparse retrieval, dense retrieval, candidate set size, LLM reranking, benchmark dataset, engineering expertise search**

I. Introduction

The problem of skill search has been a long-standing challenge in many professional organizations. Traditionally, such a search system is implemented through lookup tables, where employees score their own skills and managers query against predefined categories. While this approach is straightforward, it suffers from several key limitations: (1) logging a series of skills requires substantial effort from employees, (2) reviewing the filtered results is time-consuming for managers, and (3) the search itself lacks flexibility.

An alternative approach is to search directly on free-form text skill descriptions. This addresses the above limitations by reducing the burden of explicit logging, enabling more expressive search queries, and allowing for flexible matching between queries and employee profiles. Although our experiments focus on skill search in the engineering industry, this framework generalizes naturally to all other forms of metadata where the underlying data structure resembles ours.

Two primary retrieval technologies dominate in this space. The first is keyword-based search, most prominently BM25, which relies on lexical overlap between queries and documents. The second is semantic search, which leverages transformer-based encoders to map queries and documents into a shared embedding space. In practice, this retrieval step is often followed by a rerank step as illustrated in Figure 1. Understanding how these approaches perform in realistic conditions is critical for guiding the deployment of search systems.

Our contributions in this work are threefold:

1) We construct a realistic benchmark dataset for the engineering industry, derived from real data collected by Wade Trim's internal skill search tool [1].
2) We investigate the optimal size of the candidate set in the retrieval stage relative to the average number of ground truth matches in the corpus.
3) We test the limitations of an LLM as a list-wise one-step reranker, which is easier to apply than a cross-encoder. Specifically, we analyze the effect of candidate set size on the consistency of the LLM's output.

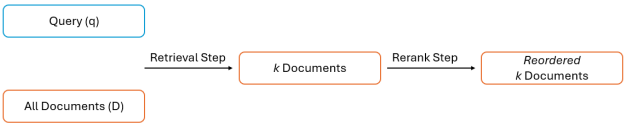


Fig. 1. Retrieval and rerank pipeline

II. Related work

Skill search in engineering firms can be framed as a specialized instance of the broader information retrieval (IR) problem, where free-form queries must be mapped to relevant records of staff skills, experiences, or project metadata. Unlike open-domain retrieval, enterprise contexts are characterized by domain-specific terminology and abbreviations, which pose a unique challenge for IR techniques.

A. Rule-based lookup tables

Early enterprise search systems often relied on lookup tables and predefined schema to match queries to documents. For example, a query for the "PE license" attribute could only be matched to profiles with the exact attribute "PE license" marked. While accurate and efficient at query time, these methods lack flexibility. With a fixed schema, these methods cannot handle abbreviations or semantic variation. Also, when

all search dimensions are predefined in a schema, the resulting long list of categories makes any data update cumbersome. As a result, we have seen a shift toward search over free-form text.

B. Sparse retrieval: BM25

Classical IR approaches formalized the retrieval task as a probabilistic ranking problem. Among the earliest weighting schemes, Term Frequency-Inverse Document Frequency (TF–IDF) assigns document scores by balancing a term's frequency within a document against its prevalence across the corpus. [2]. BM25, a widely adopted successor, refines TF–IDF by introducing term frequency saturation and document length normalization [3], [4]. Taking term frequency, inverse document frequency, and document length into account, the BM25 relevance score between a query q and a document d is defined as:

$$\text{score}(q,d) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{\text{freq}(t,d) \cdot (k_1 + 1)}{\text{freq}(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)}$$

where t is a term in query q which is tokenized as a collection of terms. $\text{freq}(t,d)$ is the frequency of the term t appearing document d, $|d|$ is this document's length, avgdl is the average document length in the collection of all documents. The constants k_1 and b serve to balance the influence of term frequency and document length so that relevance scores remain meaningful across queries and documents. The inverse document frequency term is commonly defined as follows:

$$\text{IDF}(t) \;=\; \log\left(\frac{N - n(t) + 0.5}{n(t) + 0.5}\right)$$

where N is the total number of documents and $n(t)$ is the number of documents containing term t. In general, the IDF component weights terms by their discriminative power across the corpus, while the score component scores how frequently that term shows up in the document. The more frequently a term appears overall, the less informative it is when two documents overlap on that term.

Sparse models like BM25 remain strong baselines for free-form text retrieval, though they depend on exact term matches.

C. Semantic retrieval with dense models

The limitations of sparse methods motivated the rise of dense retrieval, where both queries and documents are mapped into a shared embedding space that facilitates the search. Open-source models such as E5 [5] and Sentence-BERT [6], and closed-source ones such as OpenAI's text-embedding models [7], allow semantic matching by encoding all samples, queries and documents alike, into a shared vector space using Transformer-based encoders. Dense models can offer comparable accuracy to sparse models while exact performances depend on the dataset [8], [9]. The Transformer-based architecture, though similar to that used for generation, is trained with a contrastive loss over cosine similarity, encouraging matching pairs to move closer while driving non-matching pairs farther apart on embedding space. For a single query, contrastive loss is defined as follows [10]:

$$\mathcal{L} = -\log \frac{\exp(\cos(e_q, e_{d^+})/\tau)}{\sum_{d' \in D} \exp(\cos(e_q, e_{d'})/\tau)}$$

where e_q is the query embedding, e_{d^+} refers to the positive document's embedding, D is the set of all documents, and τ is a temperature parameter. The similarity function used above, $\cos()$, is cosine similarity.

While dense models capture semantic similarity well, they are prone to producing false positives if not calibrated carefully. Cosine similarity values exist on a continuous scale, and the point at which two embeddings should be considered "reasonably relevant" is not absolute. Human judgments of relevance are inherently subjective and context dependent, making them hard to quantify. As a result, determining an appropriate threshold requires extensive calibration and often depends on the data [11], [12].

D. The role of reranking

How do you identify bomb threats in a city? First, the entire police force gathers all citizen reports of suspected bombs, creating a large but noisy pool of candidates objects. Second, a specialized but small bomb squad examines these objects in detail to identify the actual threats. This analogy aligns with the standard two-stage IR pipeline widely studied in retrieval literature. Since initial retrieval must prioritize recall, candidate sets are often noisy. reranking addresses this issue by applying a more expressive model on the top-k candidates. Traditional rerankers are cross-encoders that jointly encode the query and each candidate document to compute a direct relevance score [13], [6]. More recently, large language models (LLMs) have been adapted for reranking, either by using them as cross-encoders to assign pairwise relevance scores or by treating list-wise reranking directly as a natural language inference task [14], [15]. This rerank step significantly improves precision, though at the cost of latency and compute.

E. IR in an engineering context

Despite the maturity of information retrieval research, applications in engineering contexts remain underexplored. Enterprise search systems often focus exclusively on structured attributes, such as project codes or software proficiencies, leaving valuable data as freeform text underutilized. Prior studies have investigated the importance of domain context and granularity when retrieving project knowledge from repositories [16]. Work on retrieving documents related to past engineering projects underscores the challenges of handling free-form descriptions, where synonyms, abbreviations, and technical codes complicate retrieval [17]. Similarly, research on hazard identification frameworks emphasizes the challenge of retrieving relevant knowledge from large and unstructured case repositories [18]. To our knowledge, no existing studies have systematically evaluated the performance of sparse

retrieval, dense retrieval, and LLM reranking on engineering-specific queries and documents. Addressing this gap requires a benchmark data set that is domain-specific and reflective of the data quality expected in a real-world industry setting.

III. Methodology

A. Experiments

Experiments are structured around two main objectives. First, we aim to determine the optimal size of the candidate set, k, as a multiple of N, the average number of true matches for each query. Scaling k up as a function of N provides insight into how deeply retrieval must search to reliably capture most true matches. Second, we evaluate the potential of incorporating an LLM as a list-wise reranker by varying the candidate set size k from 10 to 150 in increments of 10 and recording the proportion of failed reranking attempts. An attempt is considered failed if the output is not a strict reordering of the original candidate list (i.e. indices are altered, created, or removed). This setup allows us to identify the practical limitations of using an LLM as a list-wise reranker.

Through few-shot prompting with real queries and documents, we construct a realistic dataset intended to facilitate our experiments and also serve as a benchmark for the broader research community.

For sparse retrieval, we used the Okapi BM25 algorithm as implemented in the **rank_bm25** package [19], which provides a well-established probabilistic baseline. For dense retrieval, we used OpenAI's smallest embedding model, **text-embedding-3-small** [7], to generate vector representations of documents and queries that enable a semantic search. Both retrieval approaches were evaluated under identical conditions with an optional rerank step performed by GPT-4o.

B. Evaluation metrics

We evaluate the effectiveness of both retrieval setups, with and without reranking, with the following metrics:

- **Recall**: fraction of true matches retrieved within top-k.
- **Precision**: fraction of retrieved documents in top-k that are true matches.
- **Mean Reciprocal Rank (MRR)**: the reciprocal rank of the first true match.
- **Mean Average Precision (MAP)**: the average of precision values at all ranks where a true match is retrieved. MAP reflects both the ability to retrieve relevant documents and that to rank them highly.
- **Normalized Discounted Cumulative Gain (nDCG)**: A fine-grained representation of ranking quality that considers the relevance of all retrieved documents up to position k, with higher-ranked true matches receiving exponentially larger weights.

These metrics provide a comprehensive view of search performance and are widely used in IR literature. Note that macro averaging across queries is implied for all metrics so they are explained above on a *per query* basis [20].

IV. Dataset

We introduce **WTSS** (Wade Trim Skill Search Dataset), a benchmark for evaluating expertise retrieval in engineering firms. WTSS contains 911 professional expertise descriptions (documents), 20 natural language queries, and a pre-built binary relevance matrix, where rows correspond to documents and columns to queries. WTSS is publicly available on Hugging Face under the CC-BY-4.0 license [1].

This dataset is grounded in Wade Trim's internal skill search tool, which project managers and staff actively use to log and discover expertise across the company. It was synthetically generated in a few-shot manner using real examples of queries and documents, ensuring that it mirrors the types of queries and records one would expect in a professional engineering environment. This dataset consists of two components, queries and documents: $\mathcal{Q}_{\text{initial}} = \{q_1, q_2, \ldots, q_{30}\}$ and $\mathcal{D}_{\text{initial}} = \{d_1, d_2, \ldots, d_{1000}\}$.

First, we identified a set of 30 skills. Using a few-shot prompting approach with real samples, we generated one representative query for each skill. These queries emulate how managers or staff members phrase search requests in practice. Second, we generated 1,000 synthetic but realistic resume style documents. While most documents correspond to exactly one skill in our ground truth matrix, roughly 2% of them include multiple skills.

Since we use a few-shot prompting approach to generate samples with a large language model, it is difficult to precisely control the label distribution over all documents. To achieve a flat skill label distribution as shown in Figure 2, we perform a label cutoff and restrict the skill indices to $\mathcal{Q} = \{q_0, q_1, \ldots, q_{19}\}$. After this filtering, the total number of records used in the experiments is 911, that is, $\mathcal{D} = \{d_1, d_2, \ldots, d_{911}\}$. For the remainder of this paper, $\mathcal{Q}$ and $\mathcal{D}$ refer to these filtered sets.

Combining both dimensions, we can construct a binary ground truth matrix $\mathbf{Y} \in \{0,1\}^{911\times 20}$, where each row corresponds to a document (d_i) and each column corresponds to a query (q_j). In this matrix, an entry of 1 indicates a relevant match and 0 indicates non-relevance. Skills appear at approximately the same frequency across the dataset, though not uniformly. On average, each skill query corresponds to $N = 46.45$ records. This value will be useful as we investigate the optimal candidate set size for this retrieval task. A sample of the first 30 rows of the ground truth matrix $\mathbf{Y}$ is shown in Figure 3 where a colored cell is 1 and a blank cell is 0.

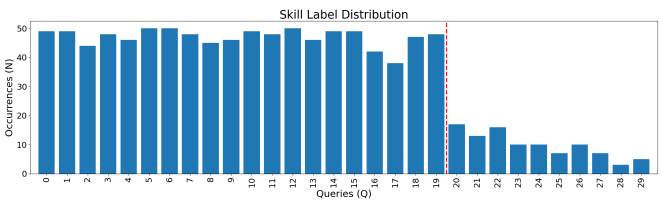


Fig. 2. Skill label distribution across documents

An example query and a corresponding document are shown below:

Query example:

{
"label": [5],
"content": *"Looking for a structural engineer to check beam loads."*
}

This query (q_i) corresponds to multiple documents $d \in \mathcal{D}$, one of which is shown below:

Document example:

{
"label": [5],
"content": *"Experienced structural engineer focused on the design of reinforced concrete structures."*
}

where

{ "5": "Structural Engineering" } .

Overall, this design yields a balanced and realistic dataset that supports evaluation of retrieval and ranking methods under conditions closely reflecting production use.

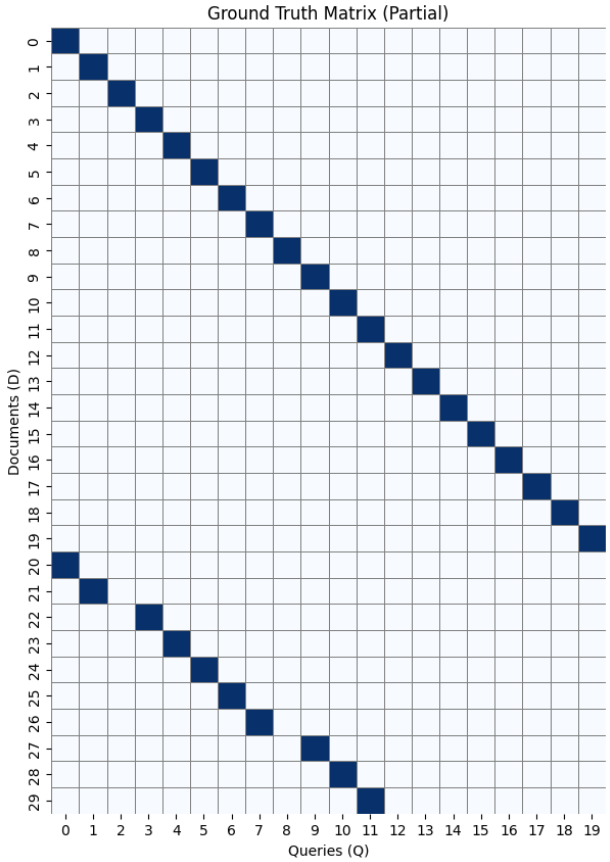


Fig. 3. Ground Truth Matrix (First 30 rows)

V. Results and Discussion

This section presents all experiment results for both sparse and dense approaches, across a range of candidate set sizes, parameterized by k. For each retrieval setup, we report two configurations: a retrieval-only pipeline and a retrieval-rerank pipeline. This setup allows us to explore both optimal candidate set size and the effect of LLM reranking. Tables II and III document all results for the sparse BM25 approach and the dense embedding-based approach, respectively. Rows in bold indicate setups that include a rerank step. Figure 4 provides a graphical summary of these results.

A. Sparse vs. dense retrieval

For retrieval without reranking, as shown in Figures 4(a) and 4(b), the dense approach consistently outperforms the sparse baseline. At $k = N$, the dense model achieves a recall of 0.82 compared to 0.66 for the sparse model, and a precision of 0.84 compared to 0.68. The three position-sensitive metrics show an even more pronounced advantage for the dense model, reflecting its ability to rank relevant documents higher in the candidate list. These observations hold consistently across all values of k, establishing dense retrieval as the stronger approach for this task.

B. Effect of reranking

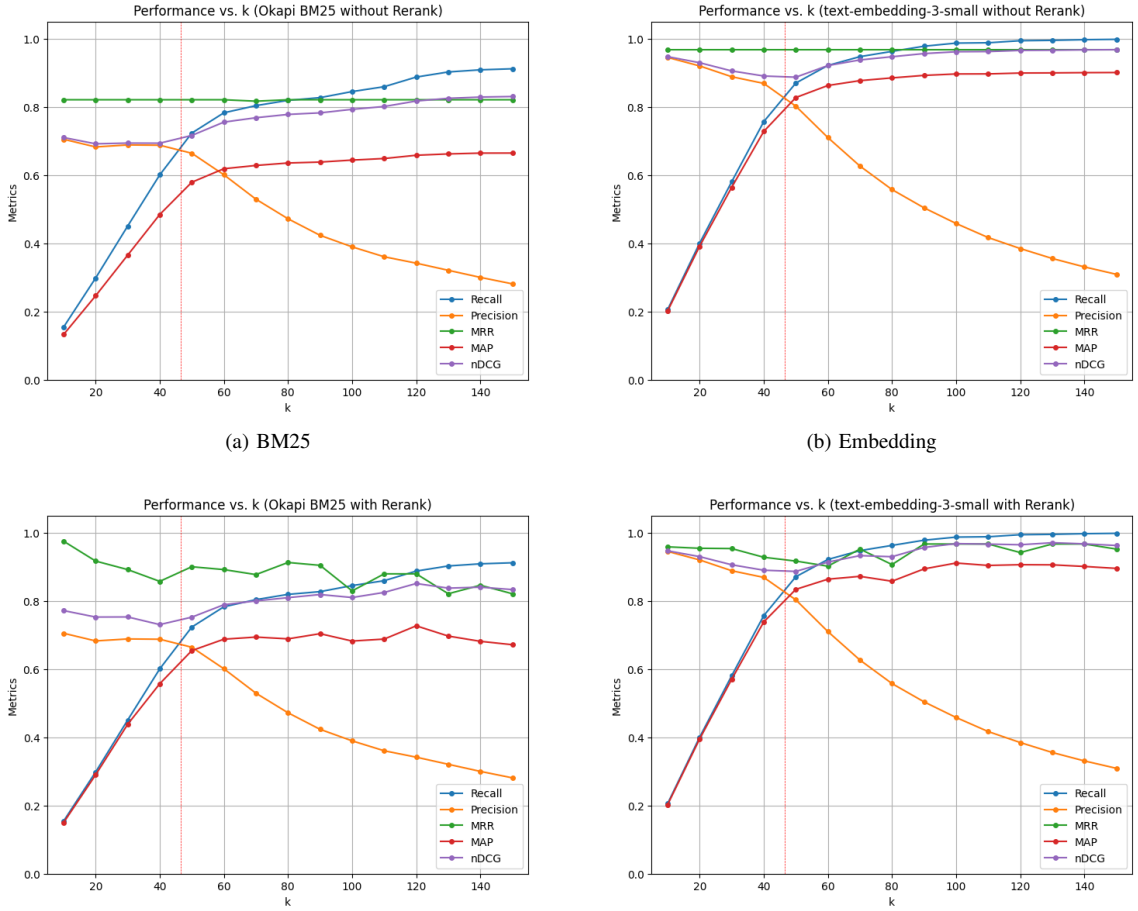
A comparison between Figures 4(a) and 4(c) demonstrate the effect of the rerank step. Since reranking only reorders candidates within the top-k, it does not affect precision or recall, but it has a direct impact on the rank-sensitive metrics (MRR, MAP, nDCG). For the sparse BM25 baseline, reranking yields noticeable improvements, especially in the reasonably low k range ($k < 2N$), where many relevant documents are initially retrieved but not ranked highly. In contrast, the dense model is largely unaffected by reranking, suggesting that it already provides sufficiently fine-grained rankings for this task.

C. Limitation of an LLM as a list-wise reranker

A key limitation of using an LLM as a list-wise reranker lies in its stability. A comparison of Figures 4a and 4c shows that the LLM is indeed capable of reranking candidates effectively. However, an output is only valid if the model strictly reorders the given indices without creating new ones, deleting existing ones, or altering any of them. A rerank is therefore considered successful only if all above constraints are satisfied. In production, we mitigate this issue by falling back to the original (unreranked) list whenever the reranker fails, making a low failure rate acceptable. On a dataset whose complexity is on par with ours, an LLM is stable up to approximately $k = 40$, as shown in Table I. The fluctuations observed in Figures 4c and 4d are consistent with this observation.

D. Candidate set size

Candidate set size plays a central role in the retrieval-rerank pipeline. First, for any pipeline that includes a rerank step, recall in the retrieval stage is more critical than precision, since relevant documents must be present in the candidate set before reranking can be effective. As k increases, performance gradually plateaus, so we need to identify the elbow point for



(a) BM25

(b) Embedding

(c) BM25 + Rerank

(d) Embedding + Rerank

Fig. 4. Comparison of sparse and dense approaches

the optimal balance between effectiveness and efficiency. Targeting a recall of 90% requires $k = 55$, that is approximately $1.2N$, in the dense retrieval without rerank configuration as shown in Figure 4b. At this point, precision drops to 75%. Note that this behavior of the precision curve is expected when you push k past N because there is simply not enough true matches available.

Cosine similarity values are hard to threshold, so how do we deal with this delimma where we want a large k to ensure a high recall but also need a high precision? This issue highlights a side benefit of an LLM as a list-wise reranker beyond reordering top k records. It can exercise judgment to remove reasonably irrelevant candidates from a candidate set, thereby improving precision at the end.

VI. Conclusion and Future Work

In our experiments, dense retrieval consistently outperforms sparse retrieval across all evaluation metrics. On our task, An optimal candidate size of approximately $k = 1.2N$ is required to achieve 90% recall and the optimal pipeline setup is semantic retrieval without reranking, Figures 4b. Under this optimal setup, with $k = 55$, the system achieves a recall of 0.90, precision of 0.76, MRR of 0.97, MAP of 0.85, and nDCG of 0.91 (linearly interpolated from Table III and rounded to two decimal places).

For any search task of a comparable difficulty level, our results indicate that semantic retrieval alone is sufficient for ranking purposes, as additional reranking does not further improve ranking precision within the candidate set. However, a rerank step remains valuable for *removing* irrelevant records to improve precision. An LLM serves this role effectively, although its stability as a list-wise reranker is limited to candidate set sizes of up to $k = 40$.

We release our full dataset on Hugging Face [1], with the goal of providing a reproducible resource for the community and fostering future research on information retrieval in the engineering domain.

Finally, the team would like to highlight an opportunity for further study. Fine-tuning an LLM specifically for the task of list-wise reranking presents a promising direction. Current results suggest that while off-the-shelf LLMs are capable rerankers, their stability deteriorates as k grows. An LLM that is instruction fine-tuned for list-wise ranking could potentially behave more consistently across larger candidates set sizes.

TABLE I
LLM RERANK FAILURE RATE

	BM25 Indices		**Embedding Indices**	
k	Failures out of 20		Failures out of 20	
10	0	**0%**	0	**0%**
20	0	**0%**	0	**0%**
30	0	**0%**	0	**0%**
40	2	**10%**	3	**15%**
50	7	**35%**	9	**45%**
60	11	**55%**	9	**45%**
70	6	**30%**	11	**55%**
80	12	**60%**	11	**55%**
90	12	**60%**	15	**75%**
100	10	**50%**	11	**55%**
110	15	**75%**	10	**50%**
120	15	**75%**	14	**70%**
130	16	**80%**	12	**60%**
140	16	**80%**	14	**70%**
150	18	**90%**	17	**85%**

TABLE II
SPARSE (OKAPI BM25)

k	Recall	Precision	MRR	MAP	nDCG
10	0.15	0.71	0.82	0.13	0.71
	0.15	**0.71**	**0.97**	**0.15**	**0.77**
20	0.30	0.68	0.82	0.25	0.69
	0.30	**0.68**	**0.92**	**0.29**	**0.75**
30	0.45	0.69	0.82	0.37	0.69
	0.45	**0.69**	**0.89**	**0.44**	**0.75**
40	0.60	0.69	0.82	0.48	0.69
	0.60	**0.69**	**0.86**	**0.56**	**0.73**
50	0.72	0.66	0.82	0.58	0.72
	0.72	**0.66**	**0.90**	**0.65**	**0.75**
60	0.78	0.60	0.82	0.62	0.76
	0.78	**0.60**	**0.89**	**0.69**	**0.79**
70	0.80	0.53	0.82	0.63	0.77
	0.80	**0.53**	**0.88**	**0.69**	**0.80**
80	0.82	0.47	0.82	0.64	0.78
	0.82	**0.47**	**0.91**	**0.69**	**0.81**
90	0.83	0.42	0.82	0.64	0.78
	0.83	**0.42**	**0.90**	**0.70**	**0.82**
100	0.84	0.39	0.82	0.64	0.79
	0.84	**0.39**	**0.83**	**0.68**	**0.81**
110	0.86	0.36	0.82	0.65	0.80
	0.86	**0.36**	**0.88**	**0.69**	**0.82**
120	0.89	0.34	0.82	0.66	0.82
	0.89	**0.34**	**0.88**	**0.73**	**0.85**
130	0.90	0.32	0.82	0.66	0.83
	0.90	**0.32**	**0.82**	**0.70**	**0.84**
140	0.91	0.30	0.82	0.66	0.83
	0.91	**0.30**	**0.85**	**0.68**	**0.84**
150	0.91	0.28	0.82	0.66	0.83
	0.91	**0.28**	**0.82**	**0.67**	**0.83**

TABLE III
DENSE (TEXT-EMBEDDING-3-SMALL)

k	Recall	Precision	MRR	MAP	nDCG
10	0.21	0.94	0.97	0.20	0.95
	0.21	**0.94**	**0.96**	**0.20**	**0.95**
20	0.40	0.92	0.97	0.39	0.93
	0.40	**0.92**	**0.95**	**0.40**	**0.93**
30	0.58	0.89	0.97	0.56	0.91
	0.58	**0.89**	**0.95**	**0.57**	**0.91**
40	0.76	0.87	0.97	0.73	0.89
	0.76	**0.87**	**0.93**	**0.74**	**0.89**
50	0.87	0.80	0.97	0.83	0.89
	0.87	**0.80**	**0.92**	**0.83**	**0.89**
60	0.92	0.71	0.97	0.86	0.92
	0.92	**0.71**	**0.90**	**0.86**	**0.91**
70	0.95	0.63	0.97	0.88	0.94
	0.95	**0.63**	**0.95**	**0.87**	**0.93**
80	0.96	0.56	0.97	0.89	0.95
	0.96	**0.56**	**0.91**	**0.86**	**0.93**
90	0.98	0.50	0.97	0.89	0.96
	0.98	**0.50**	**0.97**	**0.89**	**0.96**
100	0.99	0.46	0.97	0.90	0.96
	0.99	**0.46**	**0.97**	**0.91**	**0.97**
110	0.99	0.42	0.97	0.90	0.96
	0.99	**0.42**	**0.97**	**0.90**	**0.97**
120	0.99	0.38	0.97	0.90	0.97
	0.99	**0.38**	**0.94**	**0.91**	**0.96**
130	0.99	0.36	0.97	0.90	0.97
	0.99	**0.36**	**0.97**	**0.91**	**0.97**
140	1.00	0.33	0.97	0.90	0.97
	1.00	**0.33**	**0.97**	**0.90**	**0.97**
150	1.00	0.31	0.97	0.90	0.97
	1.00	**0.31**	**0.95**	**0.90**	**0.96**

References

[1] Y. Hou and R. Sinclair, "Wtss: Wade trim skill search dataset," https://huggingface.co/datasets/Kevin1219/WTSS, Hugging Face, 2025, cC-BY-4.0 License.

[2] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of documentation*, vol. 28, no. 1, pp. 11–21, 1972.

[3] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[4] C. Kamphuis, A. P. De Vries, L. Boytsov, and J. Lin, "Which bm25 do you mean? a large-scale reproducibility study of scoring variants," in *European Conference on Information Retrieval*. Springer, 2020, pp. 28–34.

[5] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[6] G. Rosa, L. Bonifacio, V. Jeronymo, H. Abonizio, M. Fadaee, R. Lotufo, and R. Nogueira, "In defense of cross-encoders for zero-shot retrieval," *arXiv preprint arXiv:2212.06121*, 2022.

[7] A. Neelakantan, T. Xu, R. Puri, A. Radford, J. M. Han, J. Tworek, Q. Yuan, N. Tezak, J. W. Kim, C. Hallacy *et al.*, "Text and code embeddings by contrastive pre-training," *arXiv preprint arXiv:2201.10005*, 2022.

[8] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. Wu, Y. Ding, H. Wu, H. Wang, and J.-R. Wen, "A thorough examination on zero-shot dense retrieval," *arXiv preprint arXiv:2204.12755*, 2022.

[9] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," *arXiv preprint arXiv:2112.09118*, 2021.

[10] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[11] N. Rekabsaz, M. Lupu, and A. Hanbury, "Exploration of a threshold for similarity based on uncertainty in word embedding," in *European conference on information retrieval*. Springer, 2017, pp. 396–409.

[12] N. Rossi, J. Lin, F. Liu, Z. Yang, T. Lee, A. Magnani, and C. Liao, "Relevance filtering for embedding-based retrieval," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 4828–4835.

[13] R. Nogueira and K. Cho, "Passage re-ranking with bert," *arXiv preprint arXiv:1901.04085*, 2019.

[14] M. Rathee, S. MacAvaney, and A. Anand, "Guiding retrieval using llm-based listwise rankers," in *European Conference on Information Retrieval*. Springer, 2025, pp. 230–246.

[15] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, H. Chen, Z. Liu, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv preprint arXiv:2308.07107*, 2023.

[16] P. Demian and P. Balatsoukas, "Information retrieval from civil engineering repositories: importance of context and granularity," *Journal of Computing in Civil Engineering*, vol. 26, no. 6, pp. 727–740, 2012.

[17] C. Esposito and O. Tamburis, "An effective retrieval approach for documents related to past civil engineering projects," in *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2019, pp. 295–300.

[18] H. Kim, H.-S. Lee, M. Park, B. Chung, and S. Hwang, "Information retrieval framework for hazard identification in construction," *Journal of Computing in Civil Engineering*, vol. 29, no. 3, p. 04014052, 2015.

[19] A. Trotman, A. Puurula, and B. Burgess, "Improvements to bm25 and language models examined," in *Proceedings of the 19th Australasian Document Computing Symposium*, 2014, pp. 58–65.

[20] A. Jadon and A. Patil, "A comprehensive survey of evaluation techniques for recommendation systems," in *International Conference on Computation of Artificial Intelligence & Machine Learning*. Springer, 2024, pp. 281–304.