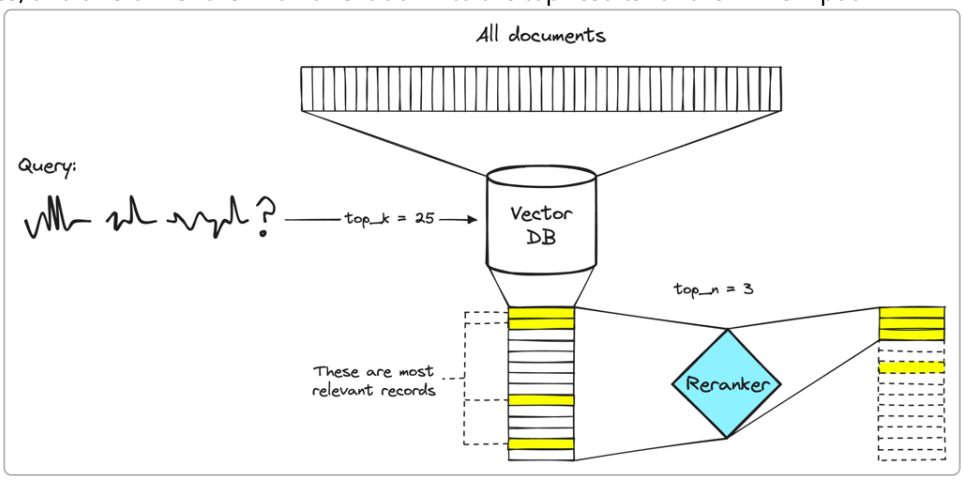
ChatGPT

# Comparison of Re-Ranking Tools for Retrieval-Augmented Generation (RAG)

**Retrieval-Augmented Generation (RAG)** pipelines typically employ a two-stage retrieval process: an initial retriever fetches candidate documents, and then a *re-ranker* refines the ordering to ensure the most relevant pieces of information are used for generation [1]. Rerankers act as a second-stage filter that **reassesses and reorders** initial search results, capturing more nuanced relevance signals and pushing the truly relevant items to the top [1]. This approach addresses the limitations of one-stage retrieval, which often prioritizes speed over precision – the reranker performs a deeper semantic analysis on the top *N* candidates to ensure the best answers are ranked highest [2]. In the context of RAG, this means only the most pertinent documents get fed into the large language model's context window, improving the accuracy and coherence of the final generated answer [3]. The diagram below illustrates a typical two-stage retrieval pipeline, where an initial vector search retrieves a set of candidates, and a reranker then narrows it down to the top results for the LLM's input




.

## BM25: Lexical Ranking Baseline

**BM25** (Best Match 25) is a **traditional lexical ranking** algorithm widely used as a first-stage retriever in information retrieval systems. It builds on TF-IDF principles and uses a probabilistic framework to score documents based on term frequency and document length normalization [4]. BM25 excels at exact keyword matching and is known for its robustness across many query types [4]. In practice, BM25 is **fast and effective** for retrieving a set of potentially relevant documents using literal term overlaps. However, as a purely lexical method, it may miss semantically relevant content expressed in different words (e.g. synonyms) and it doesn't incorporate deeper context understanding. This means BM25's ranking is limited by surface-level term matching – it might retrieve relevant documents, but not always rank them ideally by true relevance or answerability if the query's intent is nuanced [2]. Nonetheless, BM25 remains a **strong baseline** and is often the starting point in retrieval pipelines due to its speed and precision on straightforward keyword queries.

## BGE Re-Ranker (BAAI bge-reranker-v2-m3)

The **BGE re-ranker** (developed by the Beijing Academy of AI) is an open-source **neural re-ranking model** designed to work in tandem with bi-encoder embedding models. Unlike a bi-encoder that produces independent embeddings for query and document, BGE's re-ranker is a **cross-encoder**: it takes the query and a document together as input and directly outputs a relevance score [5]. This means the model "reads" the full context of the query-document pair, allowing it to capture nuanced semantic relationships and determine if the document truly answers the query. BGE-reranker-v2-m3 is a **multilingual** model (based on the BGE-M3 model) that supports 100+ languages and long inputs (up to 8192 tokens) [6] [7]. It was fine-tuned on large-scale retrieval data, and the BGE family has demonstrated state-of-the-art performance on retrieval benchmarks – for instance, BGE-M3 achieved top results in both English and multilingual tasks, even **surpassing OpenAI's embedding model** on the MIRACL benchmark [8]. In re-ranking specifically, the BGE re-rankers have proven extremely effective: the larger BGE reranker models often produce the highest or near-highest ranking accuracy, **rivaling proprietary models** like Cohere's reranker on standard tasks [9]. This high accuracy comes from the cross-encoder architecture's deep interaction between query and text, which **outperforms bi-encoder embeddings** in fine-grained relevance assessment [10].

**Advantages:** BGE's reranker offers **improved relevance scoring** due to the richer query–document interaction. It can catch subtle connections or context that a first-stage retriever might miss, thereby filtering out false positives from the initial results. In a RAG pipeline, using the BGE reranker after an embedding-based retrieval can significantly boost the quality of the top results [10]. Moreover, BGE-v2-m3 is designed to be relatively lightweight for a cross-encoder and "easy to deploy" with fast inference compared to larger models [11] (the base version uses an XLM-RoBERTa architecture). It is versatile for **multilingual applications**, making it suitable for global datasets. The BGE documentation explicitly recommends a pipeline of "**hybrid retrieval + re-ranking**": first use BGE-M3 (which can perform dense and sparse retrieval) to retrieve candidates, then apply the BGE re-ranker to further filter and reorder them [7]. This approach leverages BGE's strength in both stages to maximize accuracy.

**Trade-offs:** The main cost of using BGE (or any cross-encoder re-ranker) is **increased latency** and computation at query time. Scoring each query–document pair with a transformer is slower than a single vector similarity lookup. Thus, BGE rerankers are typically applied only on a limited number of top candidates (e.g. top 20 or 50) to keep the re-ranking stage tractable. Overall, BGE-reranker-v2-m3 is regarded as a **high-accuracy re-ranking tool** in RAG systems, especially when multilingual support is needed, and it pairs well with powerful embedding models (including BGE's own dual encoder). Its effectiveness in boosting retrieval performance has been validated on benchmarks, but users should be mindful of the computational overhead and ensure they have sufficient resources or use techniques like half-precision to speed it up [12].

## Jina Reranker v2 (Multilingual Cross-Encoder)

**Jina Reranker v2 (Base Multilingual)** is another **neural cross-encoder re-ranking model**, released by Jina AI (a company known for neural search frameworks). It is a 278M-parameter model built to **enhance search accuracy across languages and data types** [13]. Like BGE, Jina's reranker takes a query and document together and outputs a relevance score, evaluating the pair with full context. This model was trained through a sophisticated four-stage process: starting with English data, then adding cross-lingual and multilingual training, and finally fine-tuning with hard negatives for extra robustness [14]. The result is a **high-performance multilingual reranker** that supports over 100 languages and even has specialized capabilities for things like function call understanding and code search [13]. Notably, Jina Reranker v2 incorporates **FlashAttention 2** optimizations and a long-context architecture,

which allow it to handle extremely large input lengths (up to **524,288 tokens**) while maintaining high speed [14]. In practice, this means it can rerank very long documents or even entire code files in a single pass – an impressive feat for a transformer model.

Despite its long context and broad functionality, Jina's reranker is designed to be **efficient in production**. It achieves **6× higher throughput** than Jina's previous reranker generation, and in real-world evaluations it processes documents **15× faster than BGE-reranker-v2-m3** – making it far more feasible for real-time applications [15]. At the same time, it does not sacrifice accuracy: Jina reports that this model reached **state-of-the-art performance on the AirBench leaderboard** (which measures RAG system effectiveness), and it showed strong results on multilingual QA benchmarks like MKQA (covering 26 languages) [15]. It also excelled in structured tasks (like function calling and SQL schema matching), demonstrating high recall for those specialized use cases [15]. In summary, Jina Reranker v2 is a **modern cross-encoder optimized for speed and scale**: it provides precise re-ranking across many languages, with added support for long documents and even multimodal (code) scenarios, all while significantly reducing inference latency compared to other large rerankers.

**Advantages:** The Jina reranker brings many of the same relevance improvements as other cross-encoders – **contextual understanding** of query and document, ability to capture subtle semantic matches, and thus improved ranking of truly relevant results. Its distinguishing advantage is **efficiency**: through model optimizations and possibly quantization ("Q8_0" indicates an 8-bit quantized version for smaller memory footprint), it manages to deliver high accuracy **with lower latency**. This makes it more practical to deploy reranking in production systems, where every millisecond counts. Jina v2's support for very long contexts is also a plus for use cases like re-ranking entire long documents or transcripts. Organizations dealing with multilingual data or code documentation may benefit from its specialized training in those areas, as it can interpret queries in one language and documents in another (cross-lingual search) and handle technical content. In testing, Jina's model has proven capable of improving retrieval performance significantly – for example, used alongside a strong embedding retriever, it can boost the Mean Reciprocal Rank (MRR) of the system to top-tier levels, comparable to or better than other rerankers on the market [9].

**Trade-offs:** While much faster than some alternatives, Jina's cross-encoder is still a deep neural network and thus carries *non-trivial* computational cost per query. It requires a GPU for best performance and careful deployment to handle high query volumes [16]. Its large context window capability, while impressive, might not always be needed – feeding hundreds of thousands of tokens is rare in practice and could be memory-intensive. However, having that headroom means the model won't easily truncate relevant information. Another consideration is the license (Jina's model is CC BY-NC 4.0, non-commercial use by default), which may affect enterprise adoption unless a commercial license is obtained [17]. Overall, Jina Reranker v2 is **cutting-edge in balancing performance and speed**. It represents the newer generation of rerankers aiming to make re-ranking feasible even on large-scale or real-time search systems, and it complements neural retrievers by substantially lifting the relevance of top results without an overwhelming latency penalty.

## Hybrid Retrieval (BM25 + Vector) in RAG

A **hybrid retrieval** approach combines **lexical search (e.g. BM25)** with **semantic vector search** to get the best of both worlds. The idea is that lexical methods like BM25 excel at precise term matching (catching exact keywords, names, etc.), while dense embedding models excel at semantic similarity (finding relevant texts that don't necessarily share keywords). By using them together, a RAG system can improve its recall and robustness. In practice, hybrid search can be done by retrieving from both methods and then merging results. For example, one might take the top *k* results from BM25 and the

top *k* from an embedding-based search, then **merge and deduplicate** them into a larger candidate set. This candidate set can then be re-ranked (either by a learned model or by some score fusion). Research has found that hybrid retrieval often yields **higher accuracy and better generalization** than either method alone [7]. This is because the two methods complement each other: BM25 might retrieve a document that contains an exact keyword match that a dense model overlooked, and the dense model might retrieve a paraphrased answer that BM25 wouldn't recall.

One simple hybrid strategy is **Reciprocal Rank Fusion (RRF)**, which merges ranked lists from multiple systems by giving each document a combined score based on its rank in each list [18]. For instance, if a document is ranked 2nd by BM25 and 5th by the vector search, it might receive a score like 1/2 + 1/5, and the documents can be re-sorted by these fused scores [19]. This method tends to improve recall without requiring any complex training. Another approach is a **weighted ensemble**: e.g., normalize the BM25 score and the embedding similarity score for each document, then add them with certain weights to get a final score [18]. Many modern vector databases and search frameworks (like Vespa, Milvus, etc.) support hybrid queries natively, allowing you to execute a combined dense+sparse search in one go [20]. Notably, the BGE-M3 model is built to support both dense and sparse retrieval modes – it can output token-wise importance (like IDF weights) alongside dense embeddings, enabling hybrid search without extra overhead [20].

In summary, hybrid retrieval **increases the chance of finding all relevant information** by casting a wider net. This is especially useful in RAG when missing a crucial document can lead to a wrong or incomplete answer. Once a hybrid set of candidates is obtained, a reranker (like BGE or Jina) can be applied on that set. Using a hybrid retrieval as the first stage followed by a neural re-ranker often yields excellent results: the hybrid retrieval ensures high recall (more likely to include the answer-bearing document), and the re-ranker then ensures high precision (surfacing the best answers to the top). This combination has been recommended by experts to build reliable RAG pipelines [7]. The trade-off is more complexity and slightly more compute (since you perform two types of searches), but the improvement in retrieval quality can be significant for complex queries.

## Why "Reader" Models Improve Re-Ranking

Re-ranking models that *"read"* the document (i.e. cross-encoders or other **reader-based rerankers**) dramatically improve ranking quality by leveraging deeper comprehension of the text. There are several key reasons why these reader models help:

- **Full Context Matching:** A cross-encoder reranker like BGE or Jina actually looks at the entire query and document together, allowing it to evaluate relevance in context. In contrast, first-stage retrievers (whether BM25 or bi-encoder embeddings) treat query and document separately – BM25 just matches terms, and a bi-encoder compresses each into a vector independently. The cross-encoder's joint analysis **avoids the information loss** that occurs when condensing a document into a single vector [21]. It can pick up on subtle cues like the query's intent and whether the document's content truly satisfies that intent. For example, if the query is "What is the capital of Australia?" a bi-encoder might give high score to a document mentioning Australia multiple times (even if it never mentions the capital), whereas a reader model will notice whether "Canberra" is present as the answer. This leads to **much more accurate relevance scoring** [22].

- **Query-Specific Understanding:** Because cross-encoders operate at query time, they incorporate the latest query context. Bi-encoders have no knowledge of the query when encoding documents (embeddings are precomputed), so they produce a generic representation of a document's overall meaning. A reranker, on the other hand, can focus on the **query-specific**

**aspects** of the document – essentially answering "does this document answer *this* query?" [23]. This is especially important for ambiguous or complex queries. If a query has multiple facets or requires disambiguation, a reader model can pick up those nuances in ways a one-stage retrieval often cannot [24]. This query-aware re-scoring is why rerankers consistently boost the performance of retrieval pipelines [22].

- **Thorough Analysis of Top *N* Candidates:** Initial retrieval often brings back a mix of truly relevant documents and somewhat relevant or noisy ones. The reranker effectively performs a **deeper second pass** on, say, the top 50 candidates: it can weed out those that were only tenuously related and promote those that are directly on-point [2] [25]. This reduces noise and focuses the final set on documents that are not just marginally relevant, but *exactly* relevant to the query. In RAG, where we typically can only feed a limited number of documents into the LLM (due to context window size), it's crucial that these are the *best* documents. The reranker ensures that the **LLM sees the most pertinent information first**, which greatly improves the odds that the generated answer will be correct and supported by the source text [3]. Empirically, using a reranker has been shown to improve answer accuracy and reduce hallucinations, because the LLM is less likely to be distracted by irrelevant context.

- **Enabling Smaller Retrievers:** Another benefit of having a strong reranker is that you can often use a lighter or faster first-stage retriever without losing too much end performance. The reranker will recover some of the lost precision by reordering the results. Research indicates that multi-stage retrieval with a reranker can achieve high accuracy **even if the embedding model is smaller or less expensive**, which in turn **reduces indexing cost and latency for the first stage** [26]. Essentially, the reranker "fixes" mistakes of the initial retrieval, so you don't need that stage to be perfect – just good enough to get the candidate in the pile. This can be cost-effective: you might index a billion documents with a fast, medium-quality vector model, then rely on a reranker to fine-tune the ranking on the fly.

In summary, **reader-style rerankers improve RAG by marrying the strengths of retrieval and comprehension**. They scrutinize candidate documents with the same kind of understanding that an actual reader would, scoring relevance in a way that simple term frequency or vector similarity cannot. The payoff is a more accurate and reliable retrieval pipeline: higher likelihood of retrieving the correct answer and properly ranking it at the top. The primary downside is computational – these models are heavier and add latency (they "read" multiple documents for each query) [27]. To mitigate this, systems only rerank a small set of top candidates, and newer rerankers like Jina's are optimized for speed. Despite the cost, the consensus in the industry is that rerankers are often *worth it*: the **boost in relevance and answer quality** is crucial for building RAG systems that truly work well [22] [28]. As a result, incorporating a re-ranking stage (using tools like BM25+embedding hybrid retrieval followed by a cross-encoder, or even an LLM-based verifier) has become a best practice for advanced RAG applications. Each of the tools discussed – BM25, BGE reranker, Jina reranker, and hybrid combinations – can play a role in this pipeline, and choosing the right mix depends on the use case (e.g. language needs, speed requirements, data domain). By combining fast retrieval with intelligent re-ranking, one can significantly improve the **precision of retrieved context**, which in turn leads to more accurate and informative generative responses.

**Sources:**


- Denis Kuria. *"What Are Rerankers and How Do They Enhance Information Retrieval?"* Zilliz (Aug 13, 2024) – Overview of rerankers' role in IR and RAG [1] [2] [3].
- BAAI FlagEmbedding Team. *BGE-M3 and BGE Reranker Documentation* (2024) – Details on BGE models and recommended RAG pipeline [7] [10] [8].

- LlamaIndex Blog. *"Boosting RAG: Picking the Best Embedding & Reranker models"* (Nov 3, 2023) – Empirical comparison showing BGE reranker's strong performance [9].
- Jina AI. *Jina Reranker v2 Model Card* (June 25, 2024) – Model description and performance claims for jina-reranker-v2-base-multilingual [13] [15].
- Pinecone. *"Rerankers and Two-Stage Retrieval"* (2023) – Explanation of why rerankers improve accuracy (cross-encoder vs bi-encoder) and how to use them in RAG [22] [21].
- Marco Tulio Ribeiro et al. *"Enhancing Q&A Text Retrieval with Ranking Models"* (arXiv Sept 2024) – Benchmarking rerankers, noting improved accuracy and ability to use smaller retrievers [26].

---


[1] [2] [3] [4] [18] [19] [24] [25] [27] [28] What Are Rerankers and How They Enhance Information Retrieval - Zilliz Learn
https://zilliz.com/learn/what-are-rerankers-enhance-information-retrieval

[5] [11] [12] BAAI/bge-reranker-v2-m3 · Hugging Face
https://huggingface.co/BAAI/bge-reranker-v2-m3

[6] [7] [8] [10] [20] BAAI/bge-m3 · Hugging Face
https://huggingface.co/BAAI/bge-m3

[9] Boosting RAG: Picking the Best Embedding & Reranker models — LlamaIndex - Build Knowledge Assistants over your Enterprise Data
https://www.llamaindex.ai/blog/boosting-rag-picking-the-best-embedding-reranker-models-42d079022e83

[13] [14] [15] [16] [17] jina-reranker-v2-base-multilingual - Search Foundation Models
https://jina.ai/models/jina-reranker-v2-base-multilingual/

[21] [22] [23] Rerankers and Two-Stage Retrieval | Pinecone
https://www.pinecone.io/learn/series/rag/rerankers/

[26] Enhancing Q&A Text Retrieval with Ranking Models: Benchmarking, fine-tuning and deploying Rerankers for RAG
https://arxiv.org/pdf/2409.07691