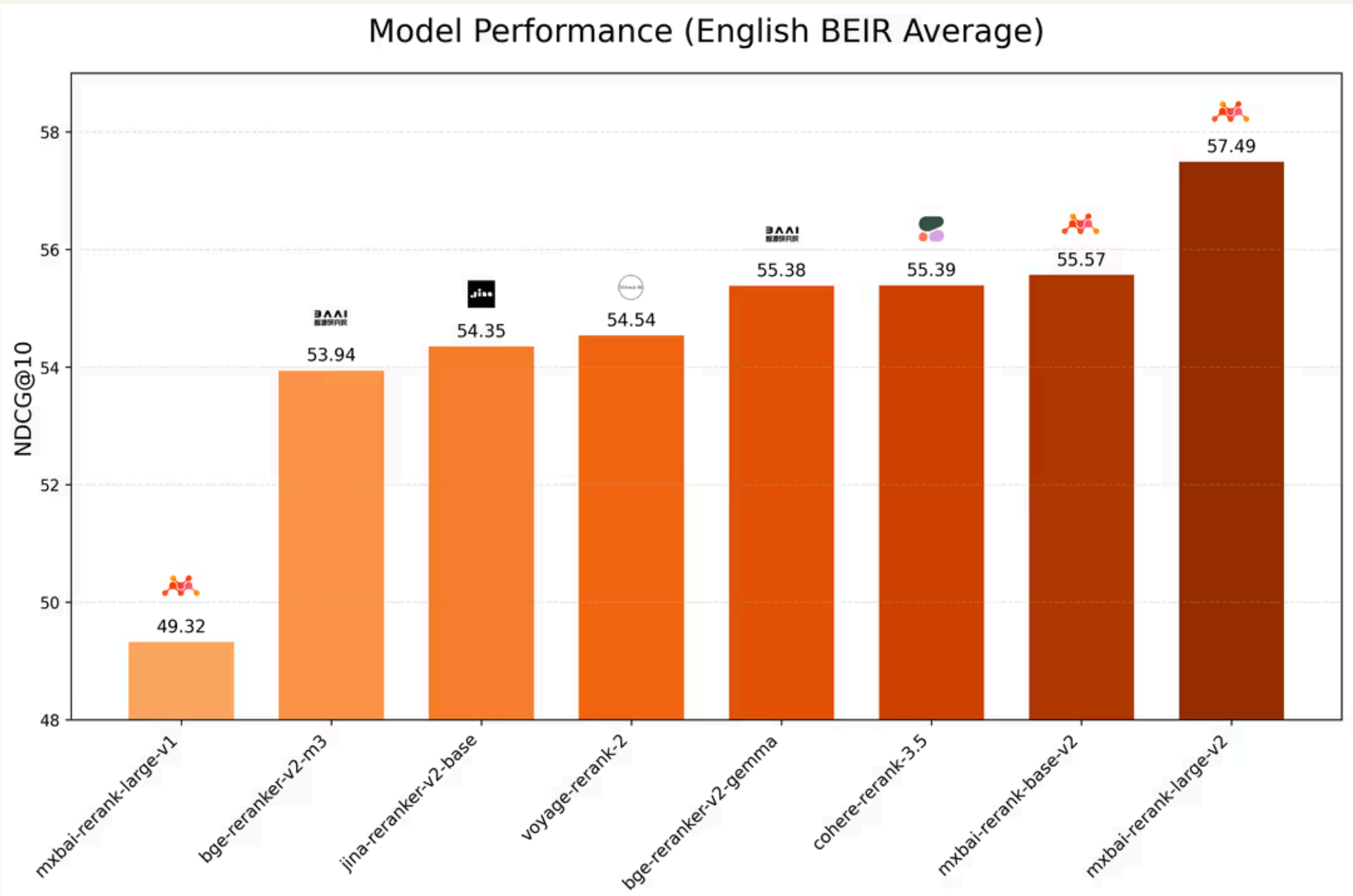
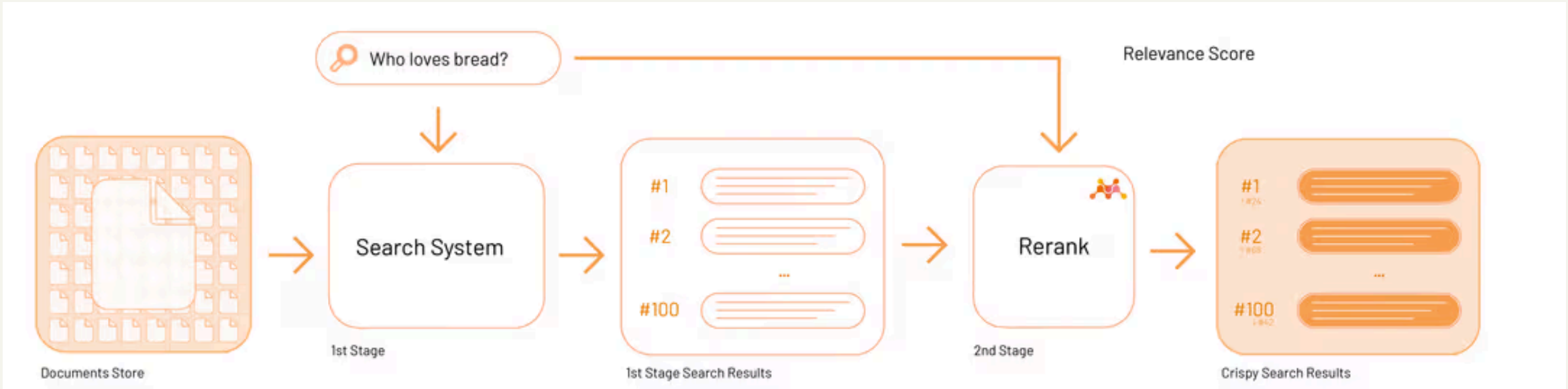
Top Rerankers for RAG Systems

A Practical Guide



Dipanjan (DJ)

Why Rerankers for RAG Systems?



- **Most search engines and RAG systems rely on just a single retrieval strategy (e.g., keyword or vector search) to gather a set of similar context documents.**

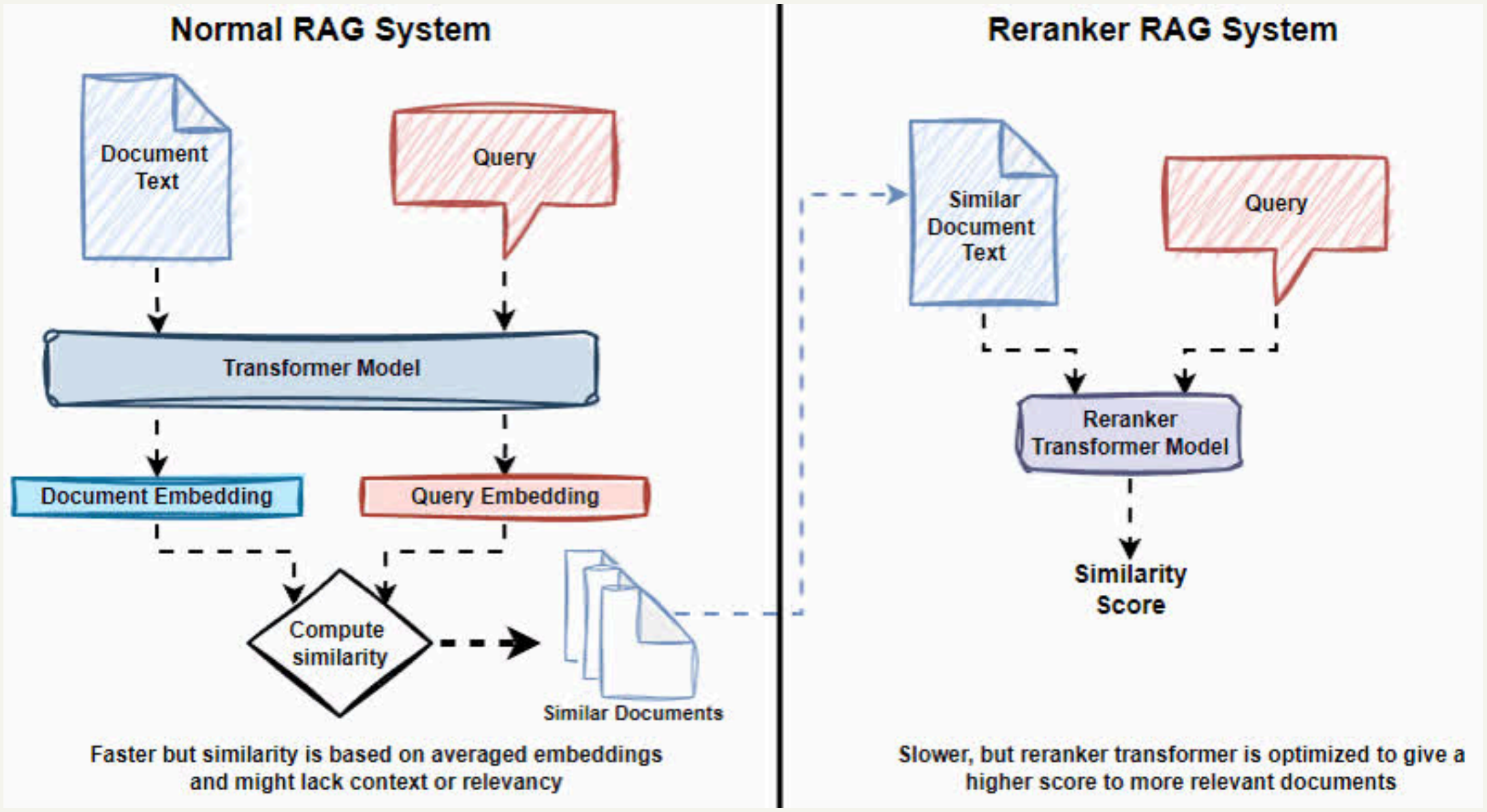
- **However, top-ranked candidates based on just similarity isn't always the most relevant.**

- **Reranking addresses this by performing a second pass that "reorders" those candidates based on deeper semantic relevance.**

- **Put simply, the reranker reads each candidate result alongside the query, scoring and sorting them so that truly relevant items rise to the top.**

Source: Baked-in Brilliance: Reranking Meets RL with mxbai-rerank-v2

What is a Reranker?



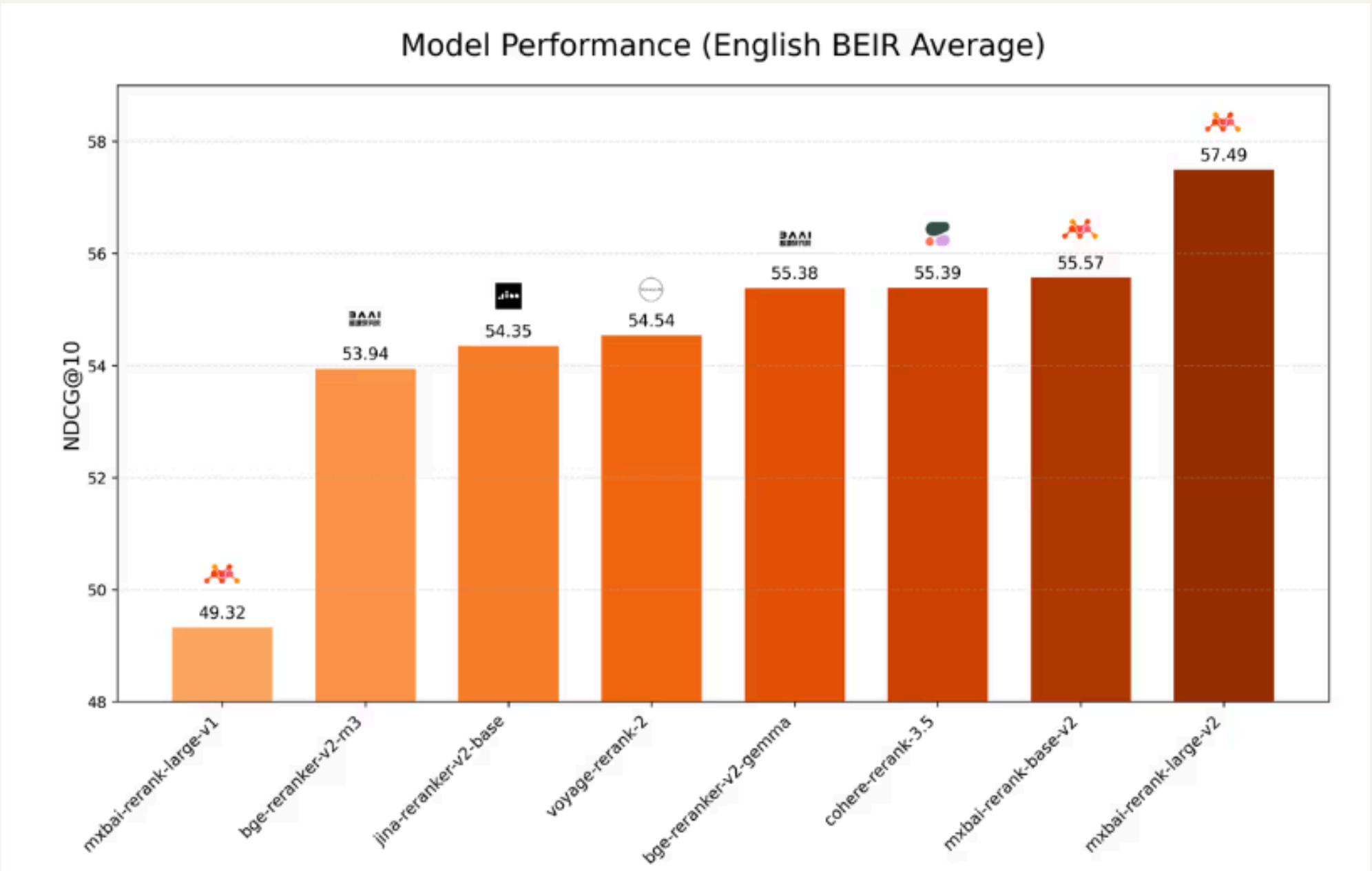
- **A Reranker is typically a cross-encoder transformer model which has been fine-tuned (trained) using supervised learning on labeled data of** (query, context document) -> relevance_score format

- **Typically is usually a classification or regression model which outputs a score or class indicating if a document is relevant to a given query or not**

- **Newer rerankers have often more sophisticated training methods built-in (check later for more details)**

Source: Author

Top Rerankers (BEIR Avg.)



BEIR Benchmark Performance.

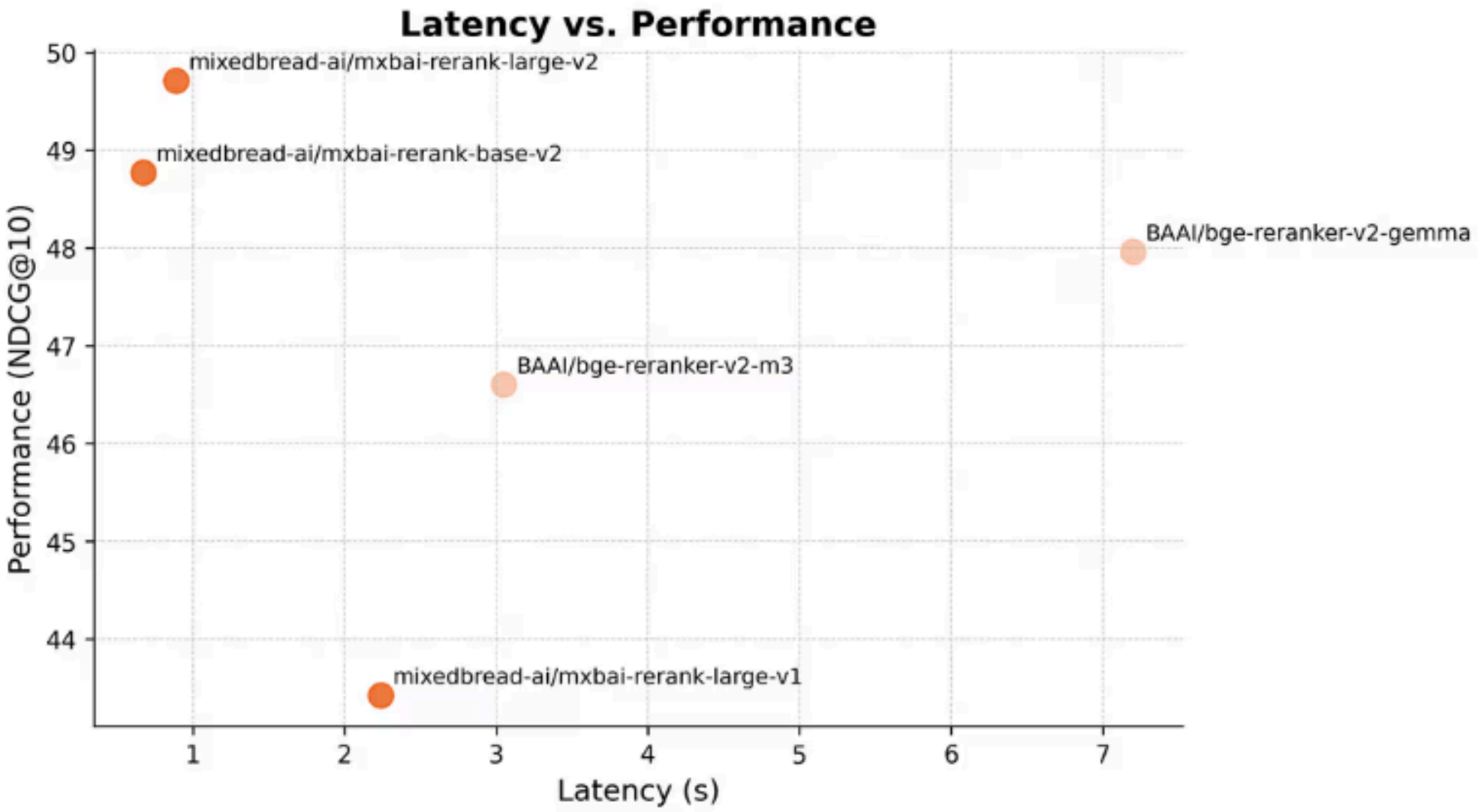
Model	NDCG@10
mixedbread-ai/mxbai-rerank-large-v2 (1.5B)	**57.49**
mixedbread-ai/mxbai-rerank-base-v2 (0.5B)	**55.57**
cohere-rerank-3.5	55.39
BAAI/bge-reranker-v2-gemma (2.5B)	55.38
voyage-rerank-2	54.54
jinaai/jina-reranker-v2-base-multilingual	54.35
BAAI/bge-reranker-v2-m3 (568M)	53.94
mixedbread-ai/mxbai-rerank-large-v1 (435M)	49.32

BEIR is the industry-standard benchmark for evaluating English-language information retrieval models. Mixedbread's **rerank-v2** leads the BEIR leaderboard, outperforming all competing models and approaching the effectiveness of state-of-the-art embedding models, while using BM25 as the first stage retriever.

Source: Baked-in Brilliance: Reranking Meets RL with mxbai-rerank-v2

Top Rerankers (Latency)

To understand how quickly each model processes queries in real-world settings, we measured **average latency per query** (seconds) on the NFCorpus dataset using an A100 (80GB) GPU:



Model	Latency (s)
mixedbread-ai/mxbai-rerank-xsmall-v1	0.32
mixedbread-ai/mxbai-rerank-base-v2	**0.67**
mixedbread-ai/mxbai-rerank-base-v1	0.76
mixedbread-ai/mxbai-rerank-large-v2	**0.89**
mixedbread-ai/mxbai-rerank-large-v1	2.24
BAAI/bge-reranker-v2-m3	3.05
BAAI/bge-reranker-v2-gemma	7.20

Our 1.5B model is **8x faster** than bge-reranker-v2-gemma while delivering higher accuracy. This speed advantage means you can process more queries per second without sacrificing quality, making our models ideal for high-volume production environments where both performance and cost-efficiency matter.

Source: Baked-in Brilliance: Reranking Meets RL with mxbai-rerank-v2

How to use mxbai-v2

Below is a basic Python snippet that sends a query and multiple candidate passages to the mxbai-rerank-v2 model. The model scores each passage, helping you rank the most relevant content at the top:

Python | Python (API) | TypeScript (API)

```python
from mxbai_rerank import MxbaiRerankV2

# Load the model, here we use our base sized model
model = MxbaiRerankV2("mixedbread-ai/mxbai-rerank-base-v2")

# Example query and documents
query = "Who wrote To Kill a Mockingbird?"
documents = [
    "To Kill a Mockingbird is a novel by Harper Lee published in 1960. It was immediately successf
    "The novel Moby-Dick was written by Herman Melville and first published in 1851. It is conside
    "Harper Lee, an American novelist widely known for her novel To Kill a Mockingbird, was born i
    "Jane Austen was an English novelist known primarily for her six major novels, which interpret
    "The Harry Potter series, which consists of seven fantasy novels written by British author J.K
    "The Great Gatsby, a novel written by American author F. Scott Fitzgerald, was published in 19
]

# Calculate the scores
results = model.rank(query, documents)
```

Once you have the scores, you can reorder your documents based on their ranking. This approach quickly upgrades the “second pass” in your search pipeline, making results more relevant without overhauling your entire search infrastructure.

Source: Baked-in Brilliance: Reranking Meets RL with mxbai-rerank-v2