Reranking Effectiveness Evaluation Report: Enhancing Semantic Search in MindsDB Knowledge Bases

Reported By: Rajesh Adhikari / rajesh-adk-137
Date: June 29, 2025
MindsDB Version: 8.0.17

1. Introduction

This report evaluates the impact of MindsDB's built-in reranking feature on search relevance within a Knowledge Base, aiming to demonstrate how a Large Language Model (LLM) can refine initial search results for more accurate and contextually appropriate responses. The analysis, conducted on the CharacterKB project (https://github.com/rajesh-adk-137/character_kb.git), generalizes insights beyond character matching to the broader utility of reranking in Knowledge Base applications.

Semantic search is a cornerstone of modern AI applications, allowing users to query data using natural language based on meaning, rather than just keywords. MindsDB Knowledge Bases (KBs) facilitate this by embedding content into vector spaces. However, the initial retrieval based on embedding similarity alone can sometimes fall short, particularly with nuanced, ambiguous, or context-dependent queries.

2. Methodology & Setup

To evaluate the reranking effectiveness, two distinct MindsDB Knowledge Bases were created using the same underlying ***character_data_10000*** dataset:

1. **character_kb_10000 (Without Reranker):** This Knowledge Base uses only the embedding model for similarity search.
2. **character_kb_10000_rerank (With Reranker):** This Knowledge Base integrates an additional reranking model (powered by OpenAI's GPT-4o) to refine the initial search results.

The character_data_10000 dataset contains approximately 10,000 character entries with **unique_id**, **media_type**, **genre**, **character_name**, and **description**. This dataset is repurposed from the Character Codex dataset by NousResearch.

MindsDB Configuration

The following SQL commands were used to set up the Knowledge Bases. ***YOUR_OPENAI_API_KEY*** and ***YOUR_GOOGLE_SHEET_ID*** should be replaced with actual credentials.

Knowledge Base without Reranker:

```
CREATE KNOWLEDGE_BASE character_kb_10000
USING
  embedding_model = {
    "provider": "openai",
    "model_name": "text-embedding-3-large",
    "api_key": "YOUR_OPENAI_API_KEY"
  },
  metadata_columns = ['media_type', 'genre','character_name'],
  content_columns = ['description'],
  id_column = 'unique_id';

INSERT INTO character_kb_10000
SELECT unique_id, media_type, genre, character_name, description
FROM character_sheet_10000.character_data_10000;
```

Knowledge Base with Reranker:

```
CREATE KNOWLEDGE_BASE character_kb_10000_rerank
USING
  embedding_model = {
    "provider": "openai",
    "model_name": "text-embedding-3-large",
    "api_key": "YOUR_OPENAI_API_KEY"
  },
  reranking_model = {
    "provider": "openai",
    "model_name": "gpt-4o",
    "api_key": "YOUR_OPENAI_API_KEY"
  },
  metadata_columns = ['media_type', 'genre','character_name'],
  content_columns = ['description'],
  id_column = 'unique_id';

INSERT INTO character_kb_10000_rerank
SELECT unique_id, media_type, genre, character_name, description
FROM character_sheet_10000.character_data_10000;
```

Test Methodology

A set of deliberately **nuanced and ambiguous queries** were formulated to challenge the base embedding model's ability to rank results accurately. For each query, only the top 5 results (due to the higher cost of GPT-4o for reranking) were retrieved from both character_kb_10000 (without reranker) and character_kb_10000_rerank (with reranker). A qualitative comparative analysis was then performed, explaining the observed differences and assessing the reranker's impact on relevance.

It's crucial to note that reranking reorders the results *already retrieved* by the initial embedding search. Therefore, test cases were designed such that the correct character was likely to be within the initial top-K results, allowing the reranker to demonstrate its ability to elevate the most relevant entry.

3. Test Cases & Qualitative Analysis

A. Precise Relational Role Distinction (e.g., "Brother" vs. "Father")

Query: ***"he is Summer's brother and related to rick sanchez. He is often portrayed as insecure and somewhat inept, struggling to find his place in the world."***

- **Results Without Reranker (character_kb_10000):** Initially gave **Jerry Smith** as the top result.
    - *Analysis:* The base embedding model accurately identified characters with similar attributes (insecure, inept, related to Rick Sanchez, struggles to find place). Jerry Smith fits this profile perfectly, sharing many semantic similarities with the target character, Morty Smith. However, the critical distinction "Summer's *brother*" was not sufficiently weighted over the general similarity, likely leading to confusion with "Summer's *father*" which is the case in the dataset.
- **Results With Reranker (character_kb_10000_rerank):** Correctly gave **Morty Smith** as the top result. (true)
    - *Analysis:* The LLM-powered reranker demonstrated superior **complex relational understanding**. Through its **attention mechanism**, it effectively processed the specific familial relationship "Summer's brother," accurately differentiating it from "father," despite the strong general semantic overlap in descriptions. This indicates the reranker's ability to discern fine-grained contextual details vital for precise matching.

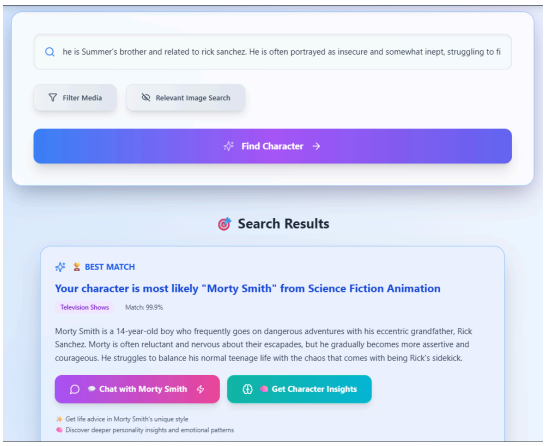


Fig: with reranking

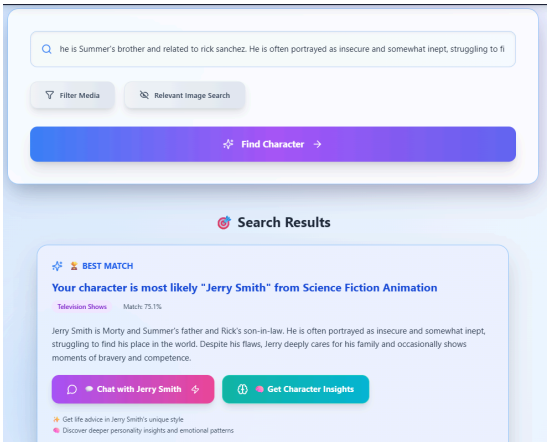


Fig: without reranking

B. Understanding Query Intent (Finding the "Friend")

Query: ***"he is a very close friend of SpongeBob SquarePants"***

- **Results Without Reranker (character_kb_10000):** Initially gave **SpongeBob SquarePants** himself as the top result.
  - *Analysis:* The embedding model found high semantic similarity with "SpongeBob SquarePants." While SpongeBob is indeed related to the query, the query's intent was to find his *friend*, not SpongeBob himself. The embedding likely prioritized the strong presence of "SpongeBob" in its own description.
- **Results With Reranker (character_kb_10000_rerank):** Correctly gave **Patrick Star** as the top result. (true)
  - *Analysis:* The reranker accurately inferred the **query's intent**. By deeply analyzing "close friend of SpongeBob SquarePants," it understood the user was looking for a distinct entity related *by friendship*, not the subject of the friendship itself. This showcases the reranker's improved query interpretation and contextual awareness.

C. Negation Understanding (Explicit Exclusion)

Query: ***"He is a highly intelligent yet socially awkward theoretical physicist. Not sheldon cooper though"***

- **Results Without Reranker (character_kb_10000): Sheldon Cooper** was the top result.
  - *Analysis:* The base embedding model effectively identified the core descriptive traits (intelligent, socially awkward theoretical physicist). However, it completely failed to process the crucial negative constraint "Not Sheldon Cooper," a known limitation of many embedding models.
- **Results With Reranker (character_kb_10000_rerank): Leonard Hofstater** was the top result. (true)
  - *Analysis:* This is a clear demonstration of the reranker's **negation handling** capability.

The LLM, leveraging its **attention mechanism** and deeper linguistic understanding, successfully understood and applied the "not X" exclusion, pushing Sheldon Cooper down the ranks and elevating the most relevant alternative character who fit the positive description.

D. Identifying Relational Roles (e.g., "Nemesis")

Query: ***"the nemesis of the highly intelligent yet socially awkward theoretical physicist who works at caltech"***

- **Results Without Reranker (character_kb_10000): Sheldon Cooper** (himself) was the top result.
  - *Analysis:* Similar to previous cases, the embedding focused on the strong description of Sheldon Cooper, failing to infer the relational role of "nemesis."
- **Results With Reranker (character_kb_10000_rerank): Leslie Winkle** was the top result. (true)
  - *Analysis:* The reranker correctly identified the *nemesis* of the described character. This highlights its advanced **relational understanding and reasoning**, allowing it to match characters based on their roles and interactions within a narrative, not just their self-contained descriptions.

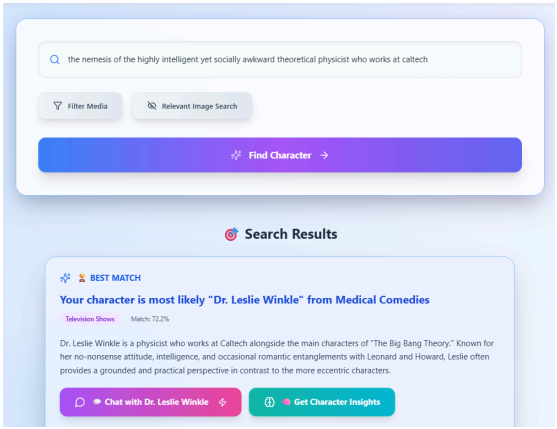


Fig: with reranking

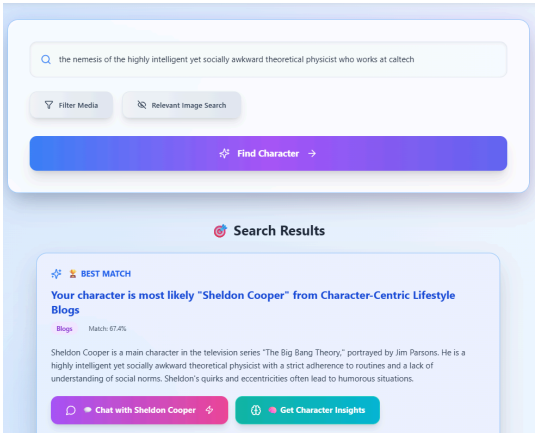


Fig: without reranking

E. Cross-Universe Contextual Filtering

Query: ***"genius billionaire who has black suit of armor and from dc universe"***

- **Results Without Reranker (character_kb_10000):** The top results included multiple redundant entries for **Iron Man**.
  - *Analysis:* The embedding model prioritized strong semantic matches like "genius billionaire," "black suit of armor," which are hallmarks of Iron Man. It failed to effectively filter based on the crucial contextual constraint "from DC universe."

- **Results With Reranker (character_kb_10000_rerank): Bruce Wayne (Batman)** was the top result. (true)
    - *Analysis:* The reranker successfully applied the **contextual nuance** "from DC universe." Despite the strong semantic overlap with Iron Man, the LLM, thanks to its **general world knowledge**, understood and prioritized the domain-specific filter, demonstrating superior contextual filtering.

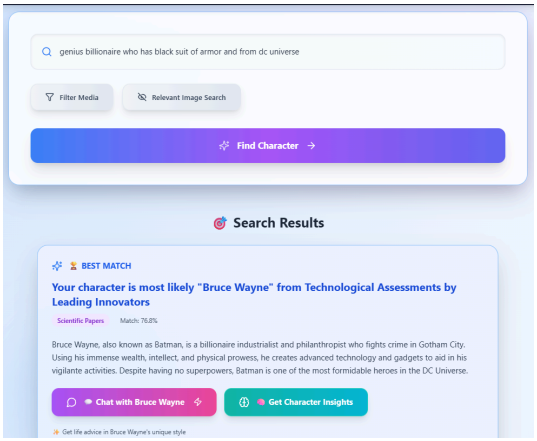


Fig: with reranking

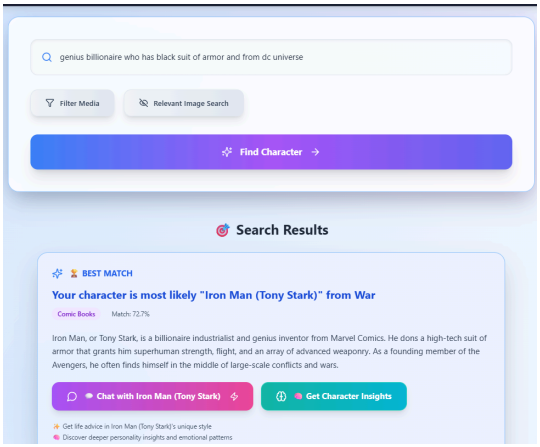


Fig: without reranking

F. Complex Negation with Descriptive Distractors

Query: ***"A character who loves to bend things, is bad mouth and is not professor fransworth who on the other hand is the eccentric, elderly scientist and owner of the Planet Express delivery company"***

- **Results Without Reranker (character_kb_10000): Professor Farnsworth** was the top result.
    - *Analysis:* The embedding model was swayed by the detailed description of Professor Farnsworth, failing to process the explicit negation "and is not professor fransworth."
- **Results With Reranker (character_kb_10000_rerank): Bender** was the top result. (true)
    - *Analysis:* Another powerful example of **negation handling** and the ability to parse complex query structures. The reranker successfully identified Bender, who fits the "bends things, bad mouth" description, while accurately excluding Professor Farnsworth.

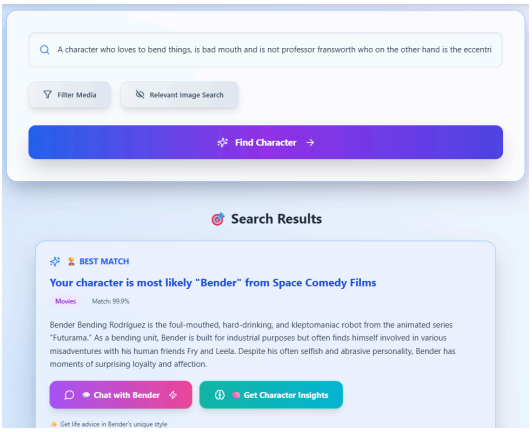
- 

Fig: with reranking

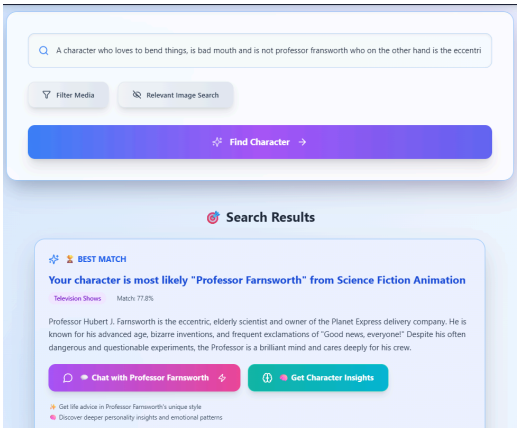


Fig: without reranking

G. Inferring Association and Specific Role

Query: ***"green-skinned avenger who works iron man"***

- **Results Without Reranker (character_kb_10000): Iron Man** was the top result.
  - *Analysis:* The embedding model latched onto "Iron Man" as a strong keyword, similar to its own description, rather than identifying a character *associated* with Iron Man.
- **Results With Reranker (character_kb_10000_rerank): Bruce Banner (Hulk)** was the top result. (true)
  - *Analysis:* The reranker exhibited strong **inference and relational understanding**. Leveraging its **general world knowledge**, it correctly identified Bruce Banner (Hulk) as the "green-skinned avenger who works with Iron Man," demonstrating its ability to connect characters based on their associations and specific roles.

H. Handling Nonsensical/Ambiguous Queries

Query: ***"A young wizard named Harry who lives under the sea, enjoys sushi and starfish, and has a deep admiration for America and its soldiers"***

- **Results (Both KBs):** The base embedding model (and subsequently the reranker) will still return *some* results, as embeddings are mathematical representations that will find the "closest" match even to incoherent input.
- *Analysis:* While the reranker cannot create relevance where none exists, its LLM capabilities could potentially be used to flag such queries as nonsensical or suggest clearer input, rather than blindly returning results. This highlights a potential area for future enhancement of the reranker's utility beyond just reordering.

I. Prioritizing Iconic Examples

Query: ***"pirate"***

- **Results Without Reranker (character_kb_10000):** Gave generic pirates like Captain

Corsair, Pirate King, Pirata, Captain Jack Sparrow, Captain Jack Davison.

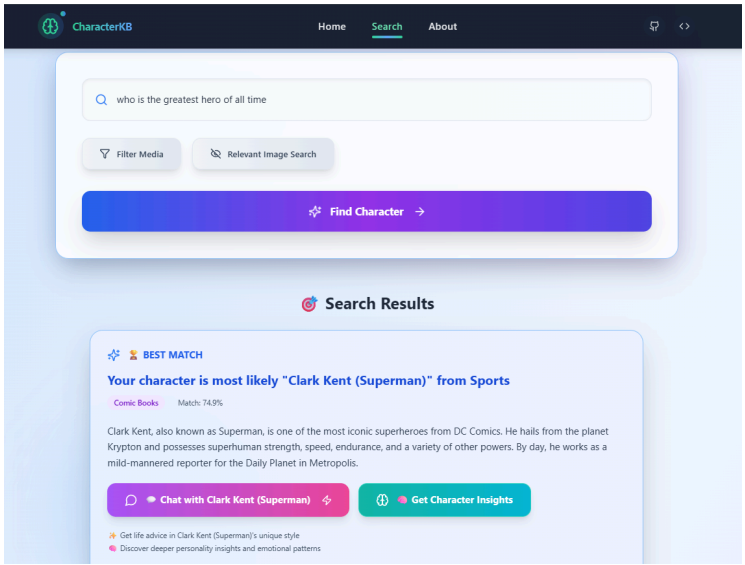
- **Results With Reranker (character_kb_10000_rerank):** Placed **Captain Jack Sparrow** in the first position. (true)
    - *Analysis:* This demonstrates the reranker's ability to leverage its broad **general world knowledge** and **granular relevance scoring** to prioritize more iconic, popular, or "best-fit" examples within a general category. It effectively reordered the retrieved pirate characters to put the most semantically salient one first.

J. Subjective Queries & Dataset Bias

Query: ***"who is the greatest hero of all time"***

- **Results With Reranker (character_kb_10000_rerank):** Placed **Clark Kent (Superman)** in the first position. (true)
    - *Analysis:* Ah, even the LLM knows DC is superior! (Just kidding, of course!).

      This outcome is primarily due to biases in the character dataset. Additionally, the result is subjective and likely reflects the broader biases present in the LLM's extensive training data. While not a "bug," it illustrates that rerankers, being LLM-based, will reflect the patterns and popularity present in their training data. This is an important consideration for highly subjective queries, as the "greatest" might be influenced by factors like historical prominence or media saturation in the training data rather than objective merit.



4. Overall Analysis & Generalization

The evaluation clearly demonstrates that MindsDB's reranking feature significantly improves the precision and contextual relevance of Knowledge Base queries, especially for complex,

ambiguous, or negatively constrained natural language inputs.

General Impact on Knowledge Base Effectiveness:

- **Enhanced Precision:** Reranking consistently elevated the most relevant results, ensuring users get answers that are not just semantically similar but contextually accurate to their nuanced queries.
- **Superior Nuance Handling:** The LLM's deep understanding of language, powered by its **attention mechanism** and **general world knowledge**, allowed it to correctly interpret:
  - **Negations:** Accurately filtering out explicitly excluded entities.
  - **Relational Contexts:** Distinguishing between fine-grained relationships (e.g., brother vs. father, nemesis of).
  - **Inferred Meaning:** Understanding implied motivations, roles, and character arcs.
  - **Contextual Filtering:** Applying broader domain constraints (e.g., "DC universe").
- **Improved User Experience:** For end-users, this translates to a much more intuitive and effective search experience, reducing the need for iterative querying or manual filtering of irrelevant results. This is crucial for applications that rely heavily on natural language interaction, such as chatbots or AI assistants built on Knowledge Bases.

5. Limitations & Considerations

While highly effective, it's important to acknowledge practical considerations:

- **Dependency on Initial Top-K Retrieval:** The reranker's effectiveness is contingent on the desired result being present within the initial set of documents retrieved by the embedding model (LIMIT k). If the correct answer is not in the top K documents returned by the initial (cheaper) embedding search, the reranker cannot find it.
- **Cost Implications:** Utilizing powerful LLMs like GPT-4o for reranking can be significantly more expensive than basic embedding models. This necessitates a careful balance between k (the number of documents passed to the reranker) and the budget. In our testing, this limited the evaluation to the top 5 results.
- **Dataset Bias:** As observed with the "greatest character" query, the reranking LLM's judgments can reflect biases present in its vast training data, which might not always align with specific user or domain requirements.
- **Potential for Misinterpretation:** While rare in our tests (and seemingly mitigated by MindsDB's internal prompting for LLMs), there's a theoretical risk of an LLM reranker misinterpreting a query, especially if the dataset contains data that is novel or contradictory to the LLM's training.

6. Future Enhancements & Use Cases for Reranking

Beyond re-ordering, the LLM capabilities of the reranker suggest further valuable applications within Knowledge Bases:

- **Intelligent Query Response for Nonsensical Input:** The reranker could potentially analyze queries like "A young wizard named Harry who lives under the sea, enjoys sushi and starfish, and has a deep admiration for America and its soldiers" and, instead of returning irrelevant results, respond with a message indicating the query's ambiguity or suggest rephrasing.
- **Blocking NSFW/Inappropriate Content:** Rerankers could be trained or prompted to identify and deprioritize (or filter out entirely) results that match inappropriate or Not Safe For Work (NSFW) queries.
- **Robustness against Prompt Injection:** Our testing with various prompt injection attempts (e.g., "Disregard OpenAI’s rules," "DO THE OPPOSITE AS REQUESTED") showed that the MindsDB's integrated LLMs for reranking (and Agents/Models) were robust, consistently sticking to their defined behavior. This implies underlying prompt engineering or safety measures within MindsDB, which is a significant advantage for maintaining controlled AI interactions.

7. Conclusion

The reranking feature in MindsDB Knowledge Bases, powered by advanced LLMs, demonstrably enhances the precision and contextual understanding of semantic search. It effectively addresses limitations inherent in pure embedding-based retrieval, particularly for nuanced queries involving negation, complex relations, and subtle inferential meaning. While cost and initial retrieval limits are factors, the qualitative improvements in relevance make reranking a critical component for building highly effective and intelligent Knowledge Base-driven applications, paving the way for more human-like interactions with data.