

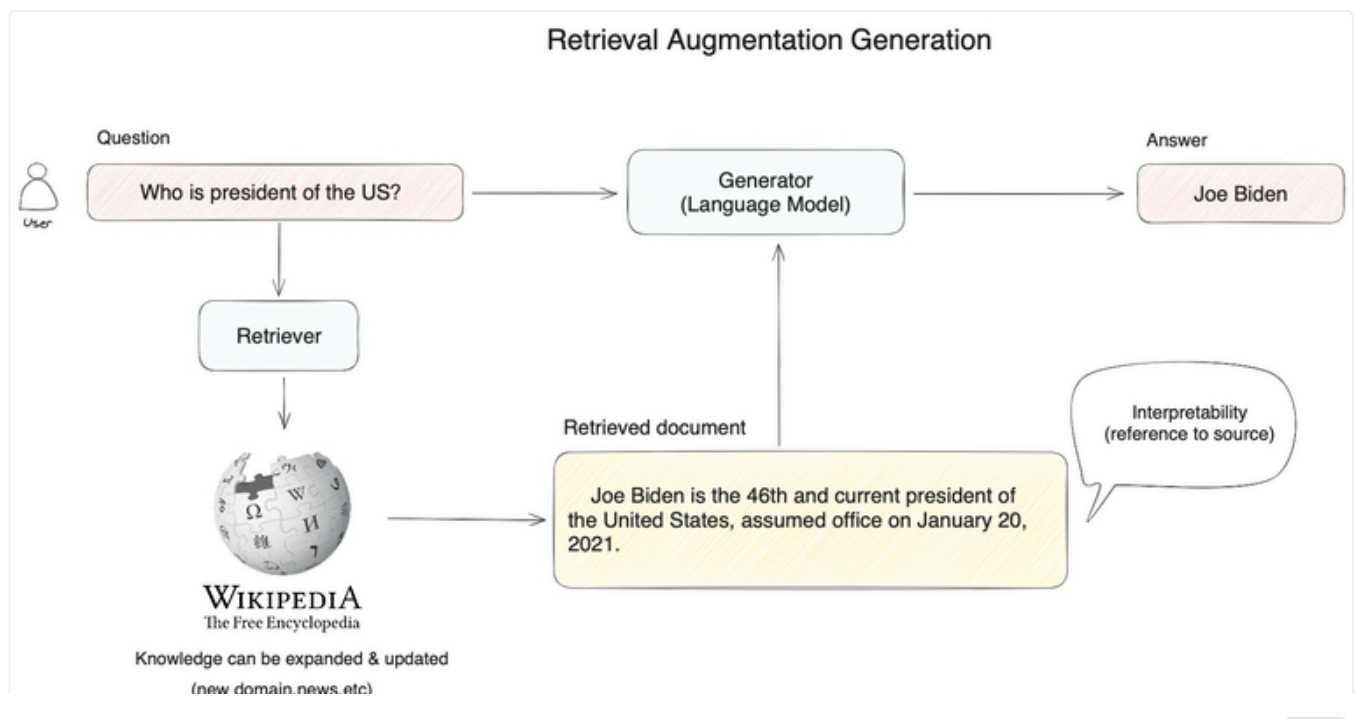
RAG (Retrieval Augmented Generation)

The concept of the RAG

The RAG architecture, with vector retrieval at its core, has become the leading technological framework for addressing two major challenges of large models: acquiring the latest external knowledge and mitigating issues of generating hallucinations. This architecture has been widely implemented in numerous practical application scenarios.

Developers can utilize this technology to cost-effectively build AI-powered customer service bots, corporate knowledge bases, AI search engines, etc. These systems interact with various forms of organized knowledge through natural language input. A representative example of a RAG application is as follows:

In the diagram below, when a user asks, "Who is the President of the United States?", the system doesn't directly relay the question to the large model for an answer. Instead, it first conducts a vector search in a knowledge base (like Wikipedia, as shown in the diagram) for the user's query. It finds relevant content through semantic similarity matching (for instance, "Biden is the current 46th President of the United States..."), and then provides the user's question along with the found knowledge to the large model. This enables the model to have sufficient and complete knowledge to answer the question, thereby yielding a more reliable response.



Why is this necessary?

We can liken a large model to a super-expert, knowledgeable in various human domains. However, this expert has its limitations; for example, it doesn't know your personal situation, as such information is private and not publicly available on the internet, and therefore, it hasn't had the opportunity to learn it beforehand.

When you want to hire this super-expert as your family financial advisor, you need to allow them to review your investment records, household expenses, and other relevant data before they can respond to your inquiries. This enables them to provide professional advice tailored to your personal circumstances.

This is what the RAG system does: it helps the large model temporarily acquire external knowledge it doesn't possess, allowing it to search for answers before responding to a question.

Based on this example, it's evident that the most critical aspect of the RAG system is the retrieval of external knowledge. The expert's ability to provide professional financial advice depends on accurately finding the necessary information. If the expert retrieves information unrelated to financial investments, like a family weight loss plan, even the most capable expert would be ineffective.

[Previous](#)
[Export/Import](#)

[Next](#)
[Hybrid Search](#)

Last updated 6 months ago



Hybrid Search

Why is Hybrid Search Necessary?

The mainstream method in the RAG retrieval phase is Vector Search, which is based on semantic relevance matching. The technical principle involves initially dividing documents from external knowledge bases into semantically complete paragraphs or sentences, and then converting them through a process known as embedding into a series of numerical expressions (multidimensional vectors) that computers can understand. The user's query undergoes a similar conversion.

Computers can detect subtle semantic correlations between user queries and sentences. For example, the semantic relevance between "a cat chases a mouse" and "a kitten hunting a mouse" is higher than between "a cat chases a mouse" and "I like to eat ham." After identifying the text with the highest relevance, the RAG system provides it as context alongside the user's query to the large model, aiding in answering the question.

Besides complex semantic text searches, Vector Search has other advantages:

- Understanding of similar semantics (e.g., mouse/mousetrap/cheese, Google/Bing/search engine)
- Multilingual comprehension (e.g., matching Chinese input with English content)
- Multimodal understanding (supports matching text, images, audio, and video)
- Fault tolerance (handles spelling mistakes, vague descriptions)

However, Vector Search might underperform in certain scenarios, like:

- Searching names of people or objects (e.g., Elon Musk, iPhone 15)
- Searching acronyms or short phrases (e.g., RAG, RLHF)
- Searching IDs (e.g., `gpt-3.5-turbo`, `titan-xlarge-v1.01`)

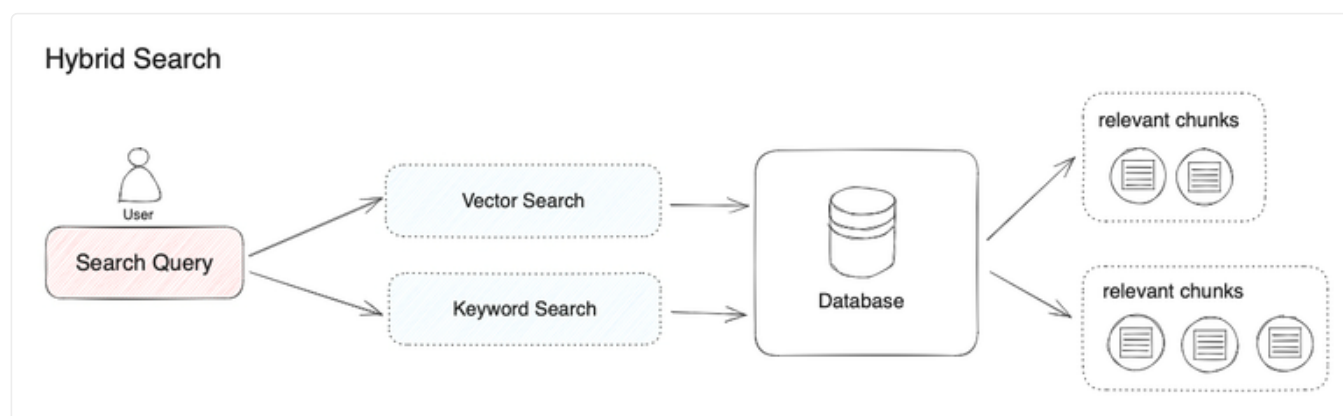
These limitations are precisely where traditional keyword search excels, being adept at:

- Precise matching (e.g., product names, personal names, product numbers)
- Matching a small number of characters (vector search performs poorly with few characters, but users often input just a few keywords)
-

Matching low-frequency vocabulary (which often carries more significant meanings, like in “Do you want to go for a coffee with me?”, words like “drink” and “coffee” carry more weight than “you”, “want”, “me”)

In most text search scenarios, it's crucial to ensure that the most relevant results appear in the candidates. Vector and keyword searches each have their strengths in the search domain. Hybrid Search combines the advantages of both techniques while compensating for their respective shortcomings.

In Hybrid Search, vector and keyword indices are pre-established in the database. Upon user query input, the system searches for the most relevant text in documents using both search methods.



Hybrid Search

"Hybrid Search" doesn't have a definitive definition; this article exemplifies it as a combination of Vector Search and Keyword Search. However, the term can also apply to other combinations of search algorithms. For instance, we could combine knowledge graph technology, used for retrieving entity relationships, with Vector Search.

Different search systems each excel at uncovering various subtle connections within texts (paragraphs, sentences, words), including precise relationships, semantic relationships, thematic relationships, structural relationships, entity relationships, temporal relationships, and event relationships. It's safe to say that no single search mode is suitable for all scenarios. **Hybrid Search, by integrating multiple search systems, achieves a complementarity among various search technologies.**

Vector Search

Definition: Vector Search involves generating query embeddings and then searching for text chunks that most closely match these embeddings in terms of vector representation.

Vector Search
Generate query embeddings and search for the text chunk most similar to its vector representation.

Rerank Model ?

rerank-english-v2.0

Top K ? **Score Threshold** ?

3 0.5

Settings for Vector Search


TopK: This setting is used to filter text chunks that have the highest similarity to the user's query. The system also dynamically adjusts the number of chunks based on the context window size of the selected model. The default value for this setting is 3.

Score Threshold: This setting is used to establish a similarity threshold for the selection of text chunks. It means that only text chunks exceeding the set score are recalled. By default, this setting is turned off, meaning that the system does not filter the similarity values of the recalled text chunks. When activated, the default value is set to 0.5.

Rerank Model: After configuring the Rerank model's API key on the "Model Provider" page, you can enable the "Rerank Model" in the search settings. The system then performs a semantic re-ranking of the document results that have been recalled after semantic search, optimizing the order of these results. Once the Rerank model is set up, the TopK and Score threshold settings are only effective in the Rerank step.

Full-Text Search


Definition: Full-Text Search involves indexing all the words in a document, enabling users to query any term and retrieve text chunks that contain these terms.



Full-Text Search

Index all terms in the document, allowing users to search any term and retrieve relevant text chunk containing those terms.

Rerank Model ?

 rerank-english-v2.0

Top K ?

3

Settings for Full-Text Search

TopK: This setting is utilized to select text chunks that most closely match the user's query in terms of similarity. The system also dynamically adjusts the number of chunks based on the context window size of the chosen model. The default value for TopK is set at 3.

Rerank Model: After configuring the API key for the Rerank model on the "Model Provider" page, you can activate the "Rerank Model" in the search settings. The system will then perform a semantic re-ranking of the document results retrieved through full-text search, optimizing the order of these results. Once the Rerank model is configured, the TopK and any Score threshold settings will only be effective during the Rerank step.

Hybrid Search

Hybrid Search operates by concurrently executing Full-Text Search and Vector Search. It then applies a re-ranking step to choose the best results that match the user's query from both types of search results. To effectively use this feature, it is necessary to configure the Rerank Model API.

Hybrid Search [Recommend](#)

Execute full-text search and vector searches simultaneously, re-rank to select the best match for the user's query. Configuration of the Rerank model APIs is necessary.

Rerank Model ?

rerank-english-v2.0

Top K ? **Score Threshold** ?

3 0.5

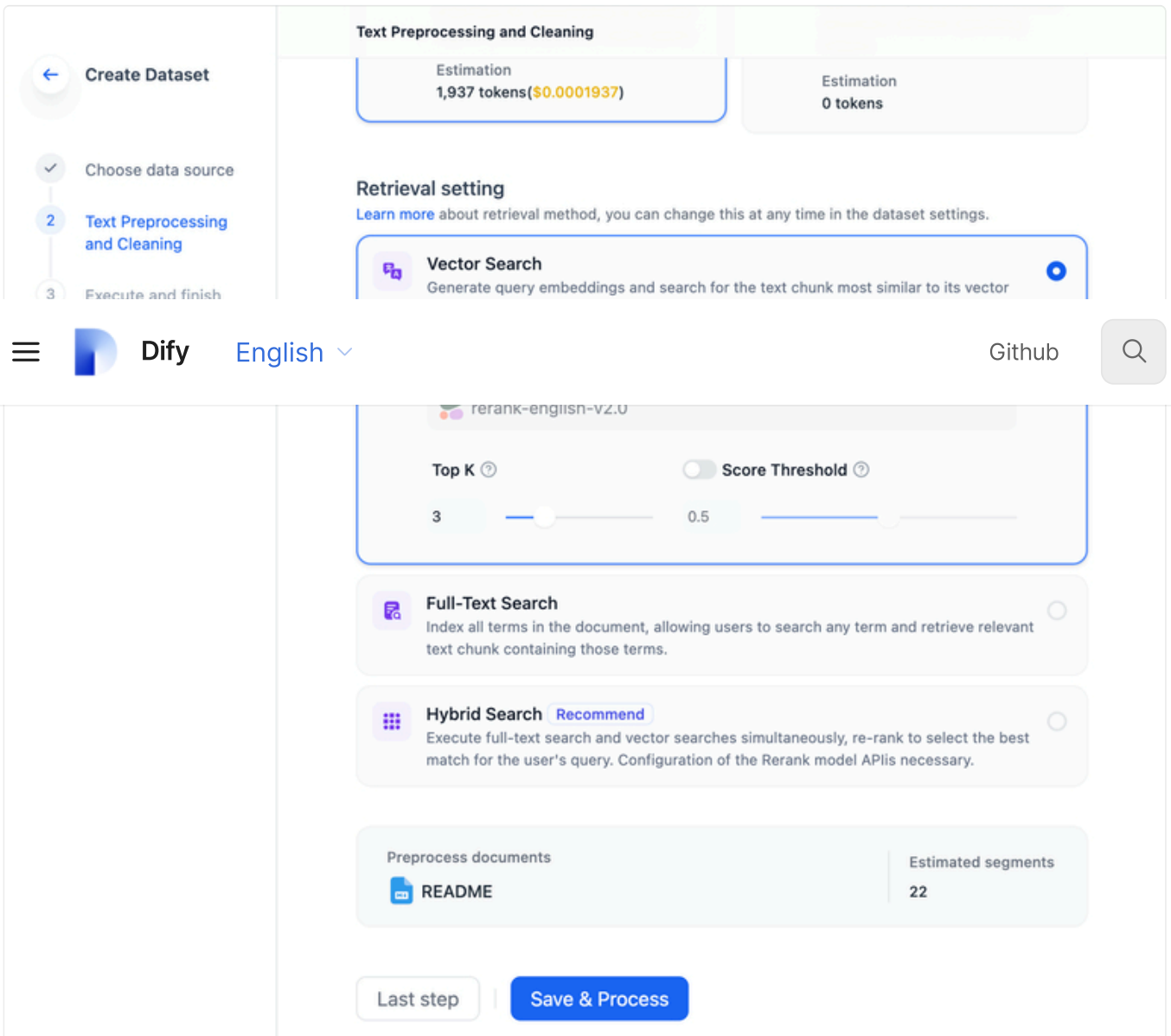
Settings for Hybrid Search

TopK: This setting is used for filtering text chunks that have the highest similarity to the user's query. The system will dynamically adjust the number of chunks based on the context window size of the model in use. The default value for TopK is set at 3.

Rerank Model: After configuring the Rerank model's API key on the "Model Supplier" page, you can enable the "Rerank Model" in the search settings. The system will perform a semantic re-ranking of the document results retrieved through hybrid search, thereby optimizing the order of these results. Once the Rerank model is set up, the TopK and any Score threshold settings are only applicable during the Rerank step.

Setting the Search Mode When Creating a Knowledge

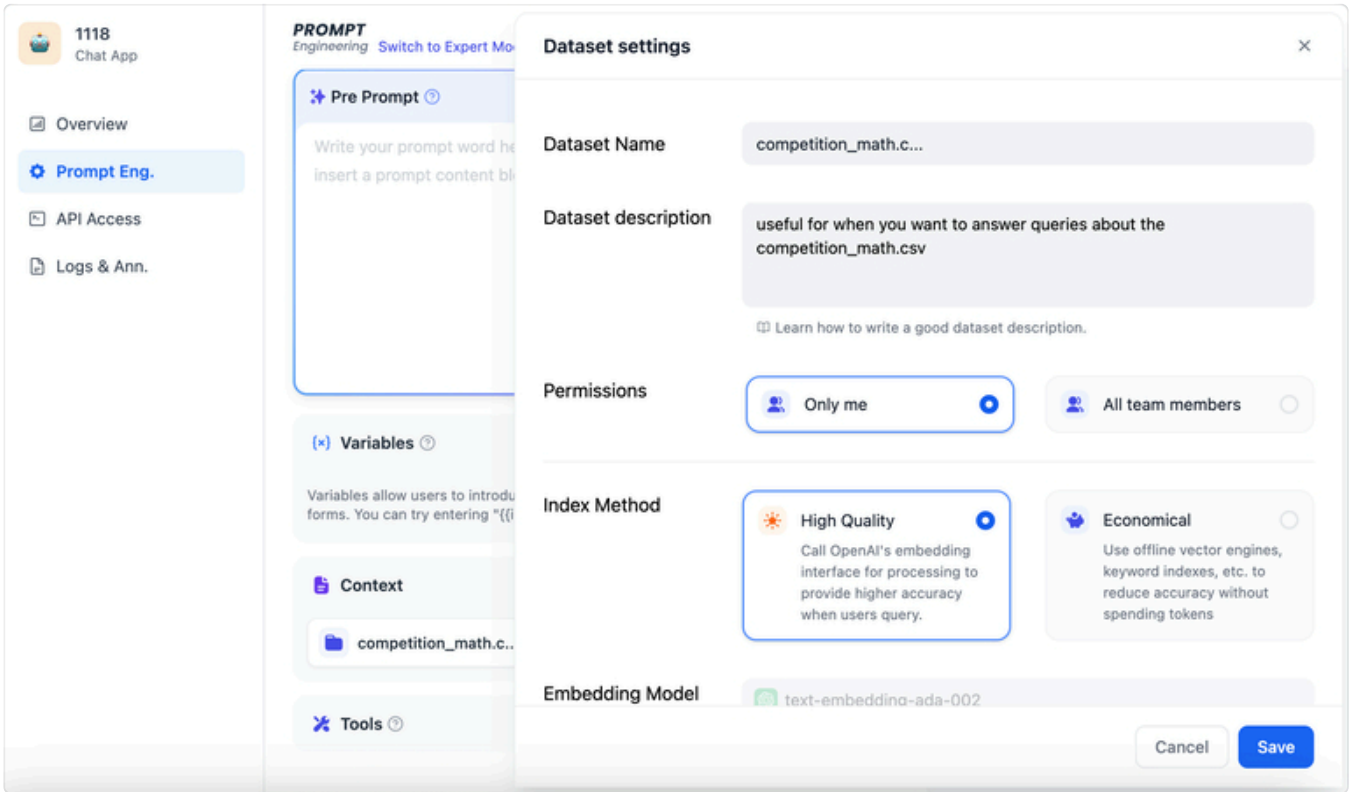
To set the search mode when creating a knowledge base, navigate to the "Knowledge → Create Knowledge" page. There, you can configure different search modes in the retrieval settings section.



Setting the Search Mode When Creating a Knowledge base

Modifying the Search Mode in Prompt Engineering

You can modify the search mode during application creation by navigating to the "Prompt Engineering → Context → Select Knowledge → Settings" page. This allows for adjustments to different search modes within the prompt arrangement phase.



Modifying the Search Mode in Prompt Engineering

Previous
RAG (Retrieval Augmented Generation)

Next
Rerank

Last updated 6 months ago

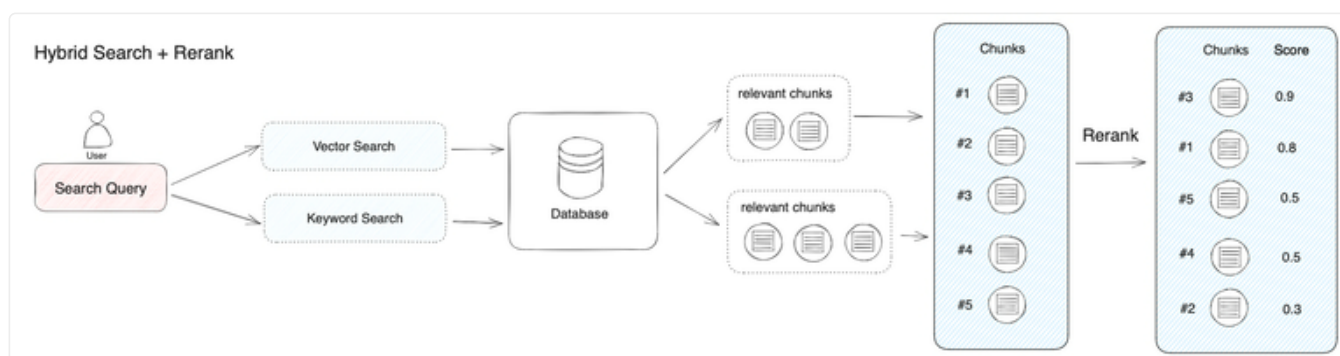


Rerank

Why is Rerank Necessary?

Hybrid Search combines the advantages of various search technologies to achieve better recall results. However, results from different search modes need to be merged and normalized (converting data into a uniform standard range or distribution for better comparison, analysis, and processing) before being collectively provided to the large model. This necessitates the introduction of a scoring system: Rerank Model.

The Rerank Model works by reordering the list of candidate documents based on their semantic match with the user's question, thus improving the results of semantic sorting. It does this by calculating a relevance score between the user's question and each candidate document, returning a list of documents sorted by relevance from high to low. Common Rerank models include Cohere rerank, bge-reranker, and others.



Hybrid Search + Rerank

In most cases, there is an initial search before rerank because calculating the relevance score between a query and millions of documents is highly inefficient. **Therefore, rerank is typically placed at the end of the search process, making it very suitable for merging and sorting results from different search systems.**

However, rerank is not only applicable to merging results from different search systems. Even in a single search mode, introducing a rerank step can effectively improve the recall of documents, such as adding semantic rerank after keyword search.

In practice, apart from normalizing results from multiple queries, we usually limit the number of text chunks passed to the large model before providing the relevant text chunks (i.e., TopK, which can be set in the rerank model parameters). This is done because the input

window of the large model has size limitations (generally 4K, 8K, 16K, 128K Token counts), and you need to select an appropriate segmentation strategy and TopK value based on the size limitation of the chosen model's input window.

It should be noted that even if the model's context window is sufficiently large, too many recalled chunks may introduce content with lower relevance, thus degrading the quality of the answer. Therefore, the TopK parameter for rerank is not necessarily better when larger.

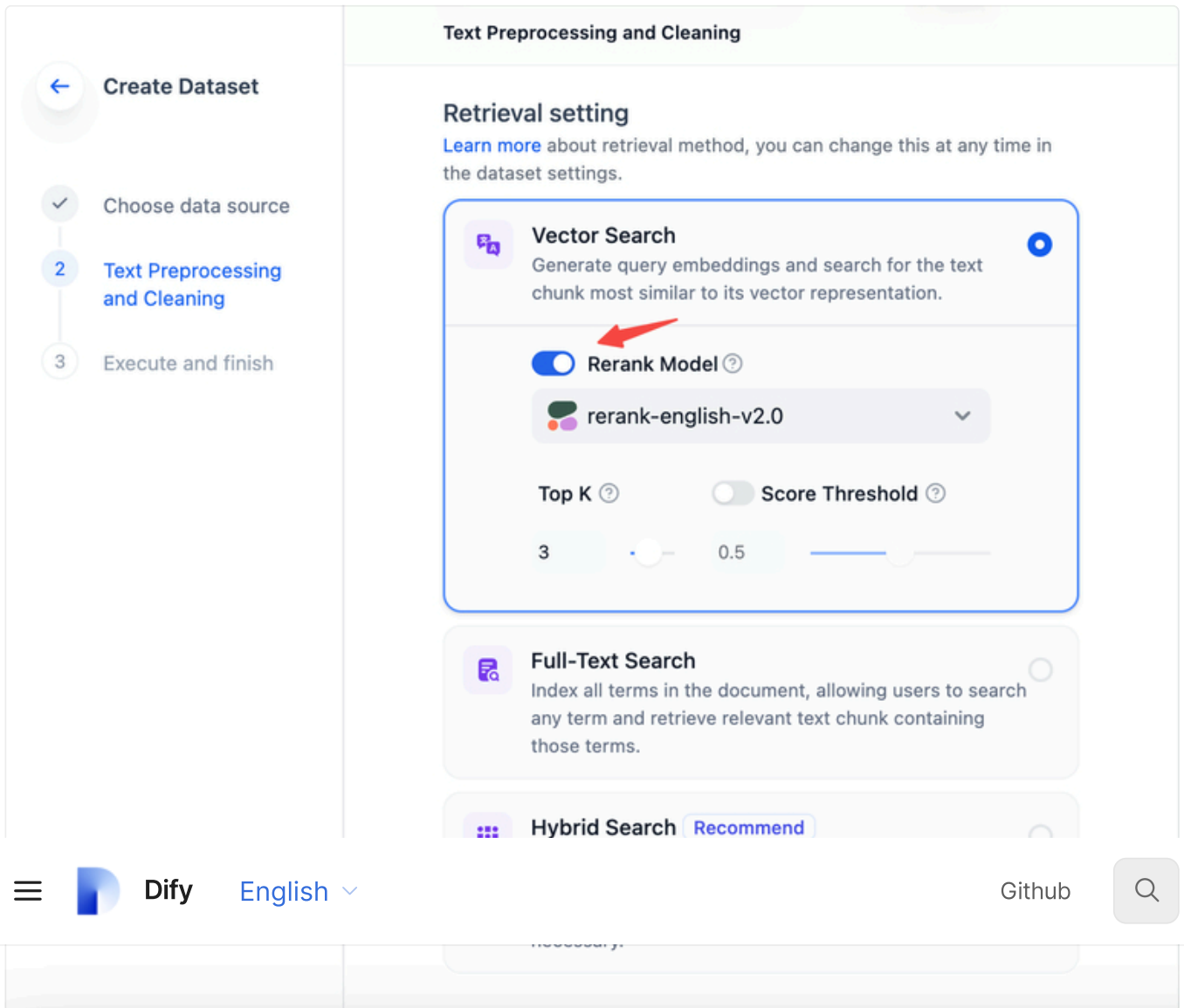
Rerank is not a substitute for search technology but an auxiliary tool to enhance existing search systems. **Its greatest advantage is that it not only offers a simple and low-complexity method to improve search results but also allows users to integrate semantic relevance into existing search systems without the need for significant infrastructure modifications.**

How to Obtain the Cohere Rerank Model?

Visit <https://cohere.com/rerank>, register on the page, and apply for usage rights for the Rerank model to obtain the API key.

Setting the Rerank Model in Knowledge Search Mode

Access the Rerank settings by navigating to “Knowledge → Create Knowledge → Retrieval Settings”. Besides setting Rerank during knowledge creation, you can also modify the Rerank configuration in the settings of an already created knowledge base, and change the Rerank configuration in the knowledge recall mode settings of application arrangement.



Setting the Rerank Model in Knowledge Search Mode

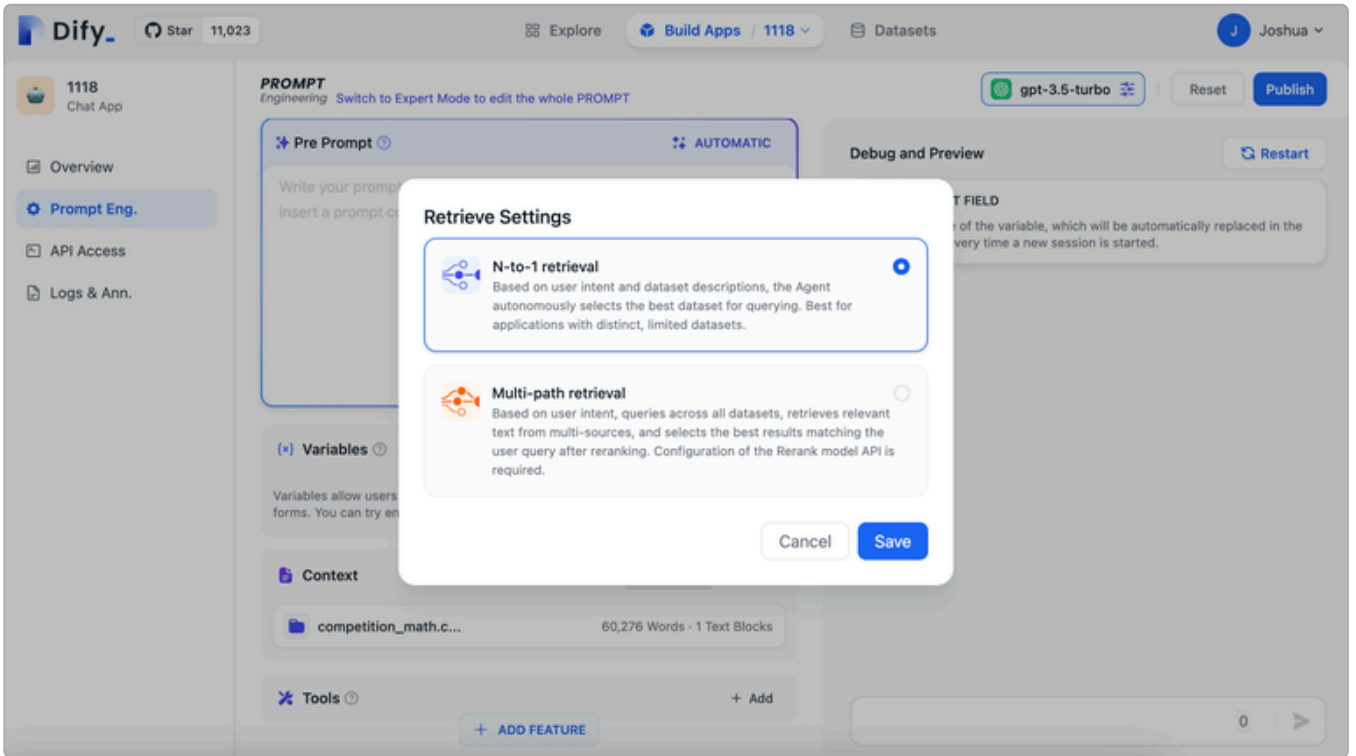
TopK: Used to set the number of relevant documents returned after Rerank.

Score Threshold: Used to set the minimum score for relevant documents to be returned after Rerank. After setting the Rerank model, the TopK and Score threshold settings are only effective in the Rerank step.

Setting the Rerank Model in Multi-path retrieval

Recall Mode Enable the Rerank model by setting it to Multi-path retrieval mode in the “Prompt Engineering → Context → Settings” page.

Explanation of Multi-path retrieval Mode: [🔗](#)



Setting the Rerank Model in Multi-path retrieval

[Previous Hybrid Search](#)

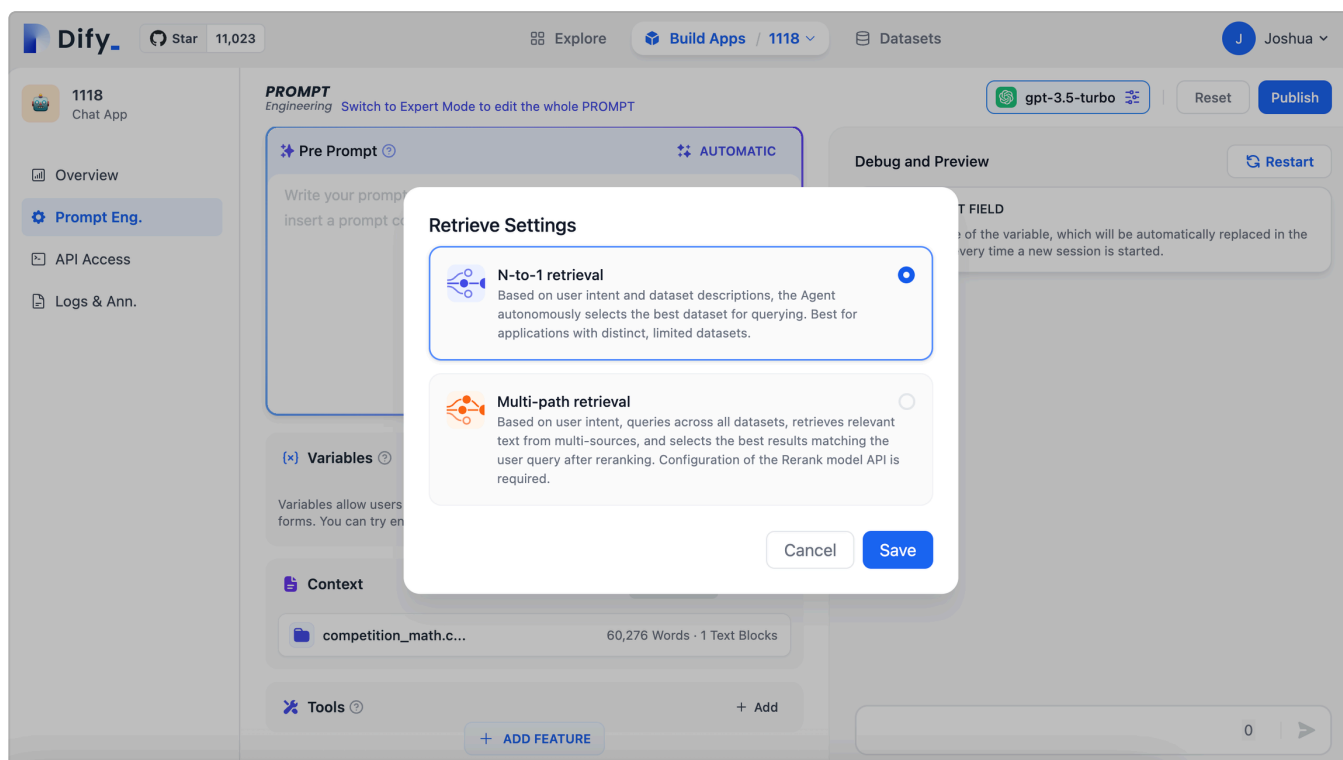
[Next Retrieval](#)

Last updated 6 months ago



Retrieval

When users build knowledge base Q&A AI applications, if multiple knowledge bases are associated within the application, Dify supports two retrieval modes: N-to-1 retrieval and Multi-path retrieval.



Retrieval Settings

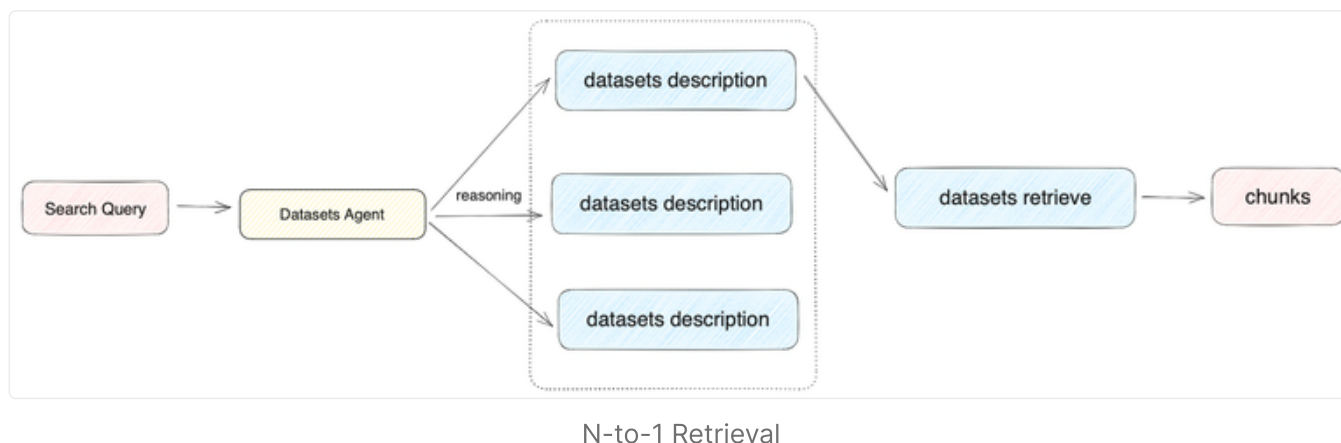
Retrieval Settings

N-to-1 Retrieval

Based on user intent and knowledge description, the Agent independently determines and selects the most matching single knowledge base for querying relevant text. This mode is suitable for applications with distinct knowledge and a smaller number of knowledge bases. N-to-1 retrieval relies on the model's inference capability to choose the most relevant knowledge base based on user intent. When inferring the knowledge, the knowledge serves as a tool for the Agent, chosen through intent inference; the tool description is essentially the knowledge description.

When users upload knowledge, the system automatically creates a summary description of each knowledge base. To achieve the best retrieval results in this mode, you can view the system-generated summary description under “Knowledge → Settings → Knowledge Description” and check if this content clearly summarizes the knowledge's content.

Here is the technical flowchart for N-to-1 retrieval:



Therefore, this mode's recall effectiveness can be impacted when there are too many knowledge bases or when the knowledge descriptions lack sufficient distinction. This mode is more suitable for applications with fewer knowledge bases.

Tip: OpenAI Function Call already supports multiple tool calls, and Dify plans to upgrade this mode to "N-to-M retrieval" in future versions.

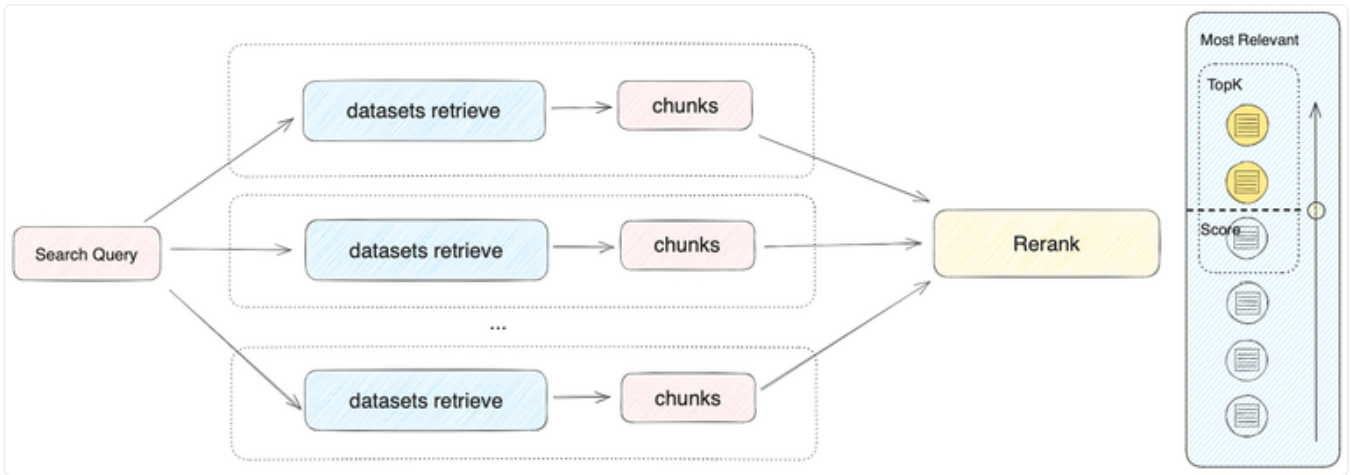
Multi-path Retrieval

Based on user intent, this mode matches all knowledge bases simultaneously, queries relevant text chunks from multiple knowledge bases, and after a re-ranking step, selects the best results matching the user's question from the multi-path query results. Configuring the Rerank model API is required. In Multi-path retrieval mode, the search engine retrieves text



using the rerank model. [How to configure the Rerank model:](#)

Here is the technical flowchart for Multi-path retrieval:



Multi-path retrieval

As Multi-path retrieval does not rely on the model's inferencing capability or knowledge descriptions, this mode can achieve higher quality recall results in multi-knowledge searches. Additionally, incorporating the Rerank step can effectively improve document recall. Therefore, when creating a knowledge base Q&A application associated with multiple knowledge bases, we recommend configuring the retrieval mode as Multi-path retrieval.

Previous
Rerank

Next
Knowledge Import

Last updated 6 months ago