Reranking

Building Custom Rerankers

Learn how to create custom rerankers in LanceDB by extending the base Reranker class.

You can build your own custom reranker in LanceDB by subclassing the `Reranker` class and implementing the `rerank_hybrid()` method. Optionally, you can also implement the `rerank_vector()` and `rerank_fts()` methods if you want to support reranking for vector and FTS search separately.

Interface

The `Reranker` base interface comes with a `merge_results()` method that can be used to combine the results of semantic and full-text search. This is a vanilla merging algorithm that simply concatenates the results and removes the duplicates without taking the scores into consideration. It only keeps the first copy of the row encountered. This works well in cases that don’t require the scores of semantic and full-text search to combine the results. If you want to use the scores or want to support `return_score="all"` , you’ll need to implement your own merging algorithm.

Below, we show the pseudocode of a custom reranker that combines the results of semantic and full-text search using a linear combination of the scores:

```
    def __init__(self, param1, param2, ..., return_score="relevance"):
        super().__init__(return_score)
        self.param1 = param1
        self.param2 = param2

    def rerank_hybrid(self, query: str, vector_results: pa.Table, fts_result
        # Use the built-in merging function
        combined_result = self.merge_results(vector_results, fts_results)

        # Do something with the combined results
        # ...

        # Return the combined results
        return combined_result

    def rerank_vector(self, query: str, vector_results: pa.Table):
        # Do something with the vector results
        # ...

        # Return the vector results
        return vector_results

    def rerank_fts(self, query: str, fts_results: pa.Table):
        # Do something with the FTS results
        # ...

        # Return the FTS results
        return fts_results
```

Example

As an example, let's build custom reranker that enhances the Cohere Reranker by accepting a filter query, and accepts any other `CohereReranker` params as `kwargs` .

```
class ModifiedCohereReranker(CohereReranker):
    def __init__(self, filters: Union[str, List[str]], **kwargs):
        super().__init__(**kwargs)
        filters = filters if isinstance(filters, list) else [filters]
        self.filters = filters

    def rerank_hybrid(self, query: str, vector_results: pa.Table, fts_result
        combined_result = super().rerank_hybrid(query, vector_results, fts_r
        df = combined_result.to_pandas()
        for filter in self.filters:
            df = df.query("not text.str.contains(@filter)")

        return pa.Table.from_pandas(df)

    def rerank_vector(self, query: str, vector_results: pa.Table)-> pa.Table
        vector_results = super().rerank_vector(query, vector_results)
        df = vector_results.to_pandas()
        for filter in self.filters:
            df = df.query("not text.str.contains(@filter)")

        return pa.Table.from_pandas(df)

    def rerank_fts(self, query: str, fts_results: pa.Table)-> pa.Table:
        fts_results = super().rerank_fts(query, fts_results)
        df = fts_results.to_pandas()
        for filter in self.filters:
            df = df.query("not text.str.contains(@filter)")

        return pa.Table.from_pandas(df)
```

Under the hood, `vector_results` and `fts_results` are PyArrow tables. You can learn more about PyArrow tables **here**. The advantage of PyArrow tables is their interoperability — you can easily convert them to Pandas/Polars the example above.

Was this page helpful? Yes No Suggest edits Raise issue

‹ Overview Evaluation ›

Ask a question... Ctrl+I