https://docs.trychroma.com/docs/querying-collections/full-text-search

The way vector databases and Full-text search work are fundamentally different. While vector databases emphasize semantic similarity, full-text search databases emphasize lexical search.

So, the way we use vector databases is to use an embedding model to convert a string into vectors, and during retrieval, look for strings with semantically similar i.e. have similar meanings via metrics like cosine similarity.

However, a full-text search looks for lexical / spelling similarity . So phrases like "scoop" and "ice cream" can have high semantic similarity but low lexical similarity.

While this might make full-text search seem outdated, there are still use cases for it. One of the major drawbacks of vector similarity is the reliance on embedding models. There could be domain-specific keywords present in your phrase, which is not seen by embedding models much during training (especially if the technologies are more recent) and hence are not captured properly in the embedding vectors. In this case, full-text search outshines as it does word-to-word matches.

This drawback has led many vector databases to incorporate the Hybrid search functionality to combine the best of both worlds.

Full Text Search

In order to filter on document contents, you must supply a `where_document` filter dictionary to the query. We support two filtering keys: `$contains` and `$not_contains`. The dictionary must have the following structure:

```
# Filtering for a search_string
{
    "$contains": "search_string"
}

# Filtering for not contains
{
    "$not_contains": "search_string"
}
```

You can combine full-text search with Chroma's metadata filtering.

```
collection.query(
    query_texts=["doc10", "thus spake zarathustra", ...],
    n_results=10,
    where={"metadata_field": "is_equal_to_this"},
    where_document={"$contains":"search_string"}
)
```

Using logical operators

You can also use the logical operators `$and` and `$or` to combine multiple filters

```
{
    "$and": [
        {"$contains": "search_string_1"},
        {"$not_contains": "search_string_2"},
    ]
}
```

An `$or` operator will return results that match any of the filters in the list

```
{
    "$or": [
        {"$contains": "search_string_1"},
        {"$not_contains": "search_string_2"},
    ]
}
```