Refining Text Queries

Axel Sirota
MACHINE LEARNING ENGINEER
@AxelSirota

Order Those Results: $meta Operator and textScore

Ordering the Results

```
> db.rent.find({$text : {$search : "tribeca loft"}}, {_id:0, name: 1})
{ "name" : "Loft" }
{ "name" : "Loft" }
{ "name" : "Lofted Bedroom in Awesome Greenpoint Loft!" }
{ "name" : "King Loft Bed in Bushwick Loft" }
{ "name" : "Lofted Bed in a Funky Family Loft!!" }
{ "name" : "Lofted Bedroom in Cool Sundrenched Brooklyn Loft" }
{ "name" : "The Loft" }
{ "name" : "Sunny Loft in heart of williamsburg - entire loft!" },
...,
{ "name" : "Loft 230" }
{ "name" : "Brooklyn Loft" }
Type "it" for more
```

The $meta Operator

MongoDB recalculates the relevance of the term in a text via the *textScore*

We can leverage that metafield with the **$meta** operator

To do it we need to add **score: {$meta: "textScore"}**

Adding the textScore

```
> db.rent.find({$text : {$search : "tribeca loft"}},
{_id:0, name: 1, score: {$meta:
"textScore"}}).limit(5)
{ "name" : "❤️Huge 7Bd Loft in Trendy
Location!!", "score" : 0.6 }
{ "name" : "SPACIOUS Floor-Through Loft in
Flatiron, NYC", "score" : 0.6 }
{ "name" : "Heart of Tribeca next to everything",
"score" : 0.625 }
{ "name" : "Artist LOFT in Bushwick - 15min
from Manhattan", "score" : 0.6 }
{ "name" : "Charming Loft in the East Village",
"score" : 0.625 }
```

- **# We search for "tribeca loft"**
- **# We project a new field containing the textScore**

- **# Now we see it in every result!**

Not to be confused with the score in Atlas cloud solution, based on Lucene!

Sorting by Score

```
> db.rent.find({$text : {$search : "tribeca loft"}},
{_id:0, name: 1, score: {$meta:
"textScore"}}).sort({score: {$meta:
"textScore"}})
{ "name" : "Tribeca Loft", "score" : 1.5 }
{ "name" : "large Tribeca loft", "score" :
1.3333333333333333 }
{ "name" : "Beautiful Tribeca Loft", "score" :
1.3333333333333333 }
{ "name" : "Prime Tribeca Loft", "score" :
1.3333333333333333 }
```

- **# Note the sorting!!**
- **# We project by textScore.**
- **# “Tribeca Loft” is first! Makes sense...**

Sorting Errors

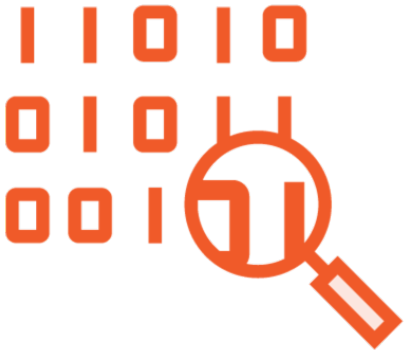
If we sort without projecting...

```
> db.rent.find({$text : {$search : "tribeca
loft"}}, {_id:0, name: 1}).sort({score: {$meta:
"textScore"}})
Error: error: {
    "ok" : 0,
    "errmsg" : "must have $meta projection for
all $meta sort keys",
    "codeName" : "BadValue"
}
```

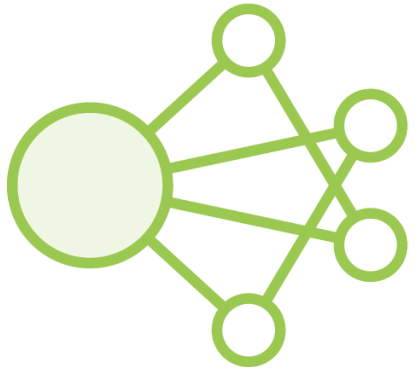
Or sort on a meta field named differently...

```
> db.rent.find({$text : {$search : "tribeca
loft"}}, {_id:0, name: 1, score2: {$meta:
"textScore"}}).sort({score: {$meta:
"textScore"}})
Error: error: {
    "ok" : 0,
    "errmsg" : "must have $meta
projection for all $meta sort keys",
    "codeName" : "BadValue"
}
```

What Happens under the Hood?

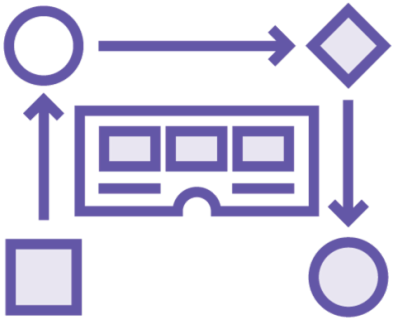


FIND DOCUMENTS



SORTS THE RESULTS

What Happens under the Hood?



CREATE PLANS

EXECUTE SEARCH

PROJECT RESULTS

Understanding the Failures

If we sort without projecting.... We don't have the metafield available at sorting in the pipeline

If we sort on a different field name.... Similarly the metaled won't exist

If we set .**sort({score:1})** as it was a normal field... It would try to do a normal sort on a metafield!

Refining Our Search: language, caseSensitive, and diacriticSensitive

Setting Case Sensitiveness

Sometimes we need to distinguish capitalized phrases

The $text operator has a $caseSensitive field

It is used this way **$text : {$search : TEXT, $caseSensitive: true}**

Sorting by Score

```
> db.rent.find({$text : {$search : "tribeca",
$caseSensitive:true}}, {_id:0, name: 1})
{ "name" : "Spacious light tribeca loft!" }


> db.rent.find({$text : {$search : "tribeca"}},
{_id:0, name: 1}).count()
185
```

- **# With $caseSensitive: true we got only one result**
- **# But turning it off we get 185**

A Corner Case

```
> db.books.find({$text : {$search: "wass"}})
{ "_id" : ObjectId("5e7a77ecd24af14b11f616cc"),
"title" : "WaSs" }

> db.books.find({$text : {$search: "wass",
$caseSensitive: true}})

> db.books.find({$text : {$search: "waSs",
$caseSensitive: true}})

> db.books.find({$text : {$search: "WaS",
$caseSensitive: true}})

> db.books.find({$text : {$search: "WaSs",
$caseSensitive: true}})
{ "_id" : ObjectId("5e7a77ecd24af14b11f616cc"),
"title" : "WaSs" }
```

- # Without case sensitive, we see that the stemmed suffix has “S”
- # With caseSensitive:true, “wass” as well as “waSs” return nothing. Lacks the “W”?
- # But we see that “WaS” gets nothing too!
- caseSensitive became exact match!!

What Is Diacritic Sensitivity?

Papa
(Pope or potato)

≠

Papá
(Dad)

```
{$text : {$search : "TEXT", $diacriticSensitive: true}}
```

An Example

```
> db.books.find({$text: {$search: "papa"}})
{ "_id" : ObjectId("5e7a7ad0d24af14b11f616cd"),
"title" : "El Papa de Roma" }
{ "_id" : ObjectId("5e7a7ae2d24af14b11f616ce"),
"title" : "El Papá  de Camila" }

> db.books.find({$text: {$search: "papa",
$diacriticSensitive: true}})
{ "_id" : ObjectId("5e7a7ad0d24af14b11f616cd"),
"title" : "El Papa de Roma" }

> db.books.find({$text: {$search: "Papá",
$diacriticSensitive: true}})
{ "_id" : ObjectId("5e7a7ae2d24af14b11f616ce"),
"title" : "El Papá  de Camila" }
```

- # With diacriticSensitive:false we get both results
- # With diacriticSensitive:true, “papa” only returns the Pope
- **diacriticSensitive became exact match!!**

Non-default Languages

```
>db.articles.find( { $text: { $search: "leche",
$language: "es" } })

{ "_id" : 5, "subject" : "Café Con Leche",
"author" : "abc", "views" : 200 }
{ "_id" : 8, "subject" : "Cafe con Leche",
"author" : "xyz", "views" : 10 }
```

◀ **# We query for results in spanish!**

◀ **# Although the index is in english**

Handling Multiple Languages

```
> db.quotes.find({}, {_id: 0, original: 1, "translation.quote" : 1}).pretty()
{
    "original" : "A sorte protege os audazes.",
    "translation" : [
        {
            "quote" : "Fortune favors the bold."
        },
        {
            "quote" : "La suerte protege a los audaces."
        }
    ]
},
...
{
    "original" : "is this a dagger which I see before me.",
    "translation" : {
        "quote" : "Es este un puñal que veo delante de mí."
    }
}
```

Adding a Language Field

```
> db.quotes.findOne({}, {_id: 0}).pretty()
{
    "language" : "portuguese",
    "original" : "A sorte protege os audazes.",
    "translation" : [
        {
            "language" : "english",
            "quote" : "Fortune favors the bold."
        },
        {
            "language" : "spanish",
            "quote" : "La suerte protege a los audaces."
        }
    ]
}
```

Non-default Languages

```
> db.quotes.find({$text: {$search: "delante",
$language: "es"}})
{ "_id" : 3, "original" : "is this a dagger which I
see before me.", "translation" : { "language" :
"spanish", "quote" : "Es este un puñal que veo
delante de mí." } }

> db.quotes.find({$text: {$search: "favors",
$language: "en"}})
{ "_id" : 1, "language" : "portuguese", "original" :
"A sorte protege os audazes.", "translation" :
[ { "language" : "english", "quote" : "Fortune
favors the bold." }, { "language" : "spanish",
"quote" : "La suerte protege a los audaces." } ] }
```

◀ **# We query for results in spanish!**

◀ **# And here we query in english**

But the index has no language set!

Rules for Language

Language field of document

Inherit enclosing document's language field

Language from text index

Default language

Demo

Create compound indexes

Use caseSensitive both with capitalization in last suffix and not

Use diacriticSensitive

Summary

The $meta projector orders our search results by the textScore

The caseSensitive and diacriticSensitive $text fields help refining our searches

How to query multi-language collections!