Branch: master ▾ Find file Copy path

elasticsearch / docs / reference / search / request / highlighting.asciidoc

jpountz Upgrade to lucene 8.4.0-snapshot-662c455. (#50016)

1329acc on Dec 11, 2019

24 contributors

Raw Blame History

929 lines (817 sloc) 29.7 KB

Highlighting

Highlighters enable you to get highlighted snippets from one or more fields in your search results so you can show users where the query matches are. When you request highlights, the response contains an additional `highlight` element for each search hit that includes the highlighted fields and the highlighted fragments.

Note	Highlighters don't reflect the boolean logic of a query when extracting terms to highlight. Thus, for some complex boolean queries (e.g nested boolean queries, queries using `minimum_should_match` etc.), parts of documents may be highlighted that don't correspond to query matches.

Highlighting requires the actual content of a field. If the field is not stored (the mapping does not set `store` to `true`), the actual `_source` is loaded and the relevant field is extracted from `_source` .

For example, to get highlights for the `content` field in each search hit using the default highlighter, include a `highlight` object in the request body that specifies the `content` field:

```
GET /_search
{
    "query" : {
        "match": { "content": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "content" : {}
        }
    }
}
```

{es} supports three highlighters: `unified`, `plain`, and `fvh` (fast vector highlighter). You can specify the highlighter `type` you want to use for each field.

Unified highlighter

The `unified` highlighter uses the Lucene Unified Highlighter. This highlighter breaks the text into sentences and uses the BM25 algorithm to score individual sentences as if they were documents in the corpus. It also supports accurate phrase and multi-term (fuzzy, prefix, regex) highlighting. This is the default highlighter.

Plain highlighter

The `plain` highlighter uses the standard Lucene highlighter. It attempts to reflect the query matching logic in terms of understanding word importance and any word positioning criteria in phrase queries.

Warning	The `plain` highlighter works best for highlighting simple query matches in a single field. To accurately reflect query logic, it creates a tiny in-memory index and re-runs the original query criteria through Lucene's query execution planner to get access to low-level match information for the current document. This is repeated for every field and every document that needs to be highlighted. If you want to highlight a lot of fields in a lot of documents with complex queries, we recommend using the `unified` highlighter on `postings` or `term_vector` fields.

Fast vector highlighter

The `fvh` highlighter uses the Lucene Fast Vector highlighter. This highlighter can be used on fields with `term_vector` set to `with_positions_offsets` in the mapping. The fast vector highlighter:

- Can be customized with a `boundary_scanner`.
- Requires setting `term_vector` to `with_positions_offsets` which increases the size of the index
- Can combine matches from multiple fields into one result. See `matched_fields`
- Can assign different weights to matches at different positions allowing for things like phrase matches being sorted above term matches when highlighting a Boosting Query that boosts phrase matches over term matches

Warning	The `fvh` highlighter does not support span queries. If you need support for span queries, try an alternative highlighter, such as the `unified` highlighter.

Offsets Strategy

To create meaningful search snippets from the terms being queried, the highlighter needs to know the start and end character offsets of each word in the original text. These offsets can be obtained from:

- The postings list. If `index_options` is set to `offsets` in the mapping, the `unified` highlighter uses this information to highlight documents without re-analyzing the text. It re-runs the original query directly on the postings and extracts the matching offsets from the index, limiting the collection to the highlighted documents. This is important if you have large fields because it doesn't require reanalyzing the text to be highlighted. It also requires less disk space than using `term_vectors`.
- Term vectors. If `term_vector` information is provided by setting `term_vector` to `with_positions_offsets` in the mapping, the `unified` highlighter automatically uses the `term_vector` to highlight the field. It's fast especially for large fields (> `1MB`) and for highlighting multi-term queries like `prefix` or `wildcard` because it can access the dictionary of terms for each document. The `fvh` highlighter always uses term vectors.
- Plain highlighting. This mode is used by the `unified` when there is no other alternative. It creates a tiny in-memory index and re-runs the original query criteria through Lucene's query execution planner to get access to low-level match information on the current document. This is repeated for every field and every document that needs highlighting. The `plain` highlighter always uses plain highlighting.

Warning	Plain highlighting for large texts may require substantial amount of time and memory. To protect against this, the maximum number of text characters that will be analyzed has been limited to 1000000. This default limit can be changed for a particular index with the index setting `index.highlight.max_analyzed_offset`.

Highlighting Settings

Highlighting settings can be set on a global level and overridden at the field level.

boundary_chars
A string that contains each boundary character. Defaults to `.,!? \t\n`.

boundary_max_scan

How far to scan for boundary characters. Defaults to `20` .

boundary_scanner

Specifies how to break the highlighted fragments: `chars` , `sentence` , or `word` . Only valid for the `unified` and `fvh` highlighters. Defaults to `sentence` for the `unified` highlighter. Defaults to `chars` for the `fvh` highlighter.

`chars`

Use the characters specified by `boundary_chars` as highlighting boundaries. The `boundary_max_scan` setting controls how far to scan for boundary characters. Only valid for the `fvh` highlighter.

`sentence`

Break highlighted fragments at the next sentence boundary, as determined by Java's BreakIterator. You can specify the locale to use with `boundary_scanner_locale` .

Note	When used with the `unified` highlighter, the `sentence` scanner splits sentences bigger than `fragment_size` at the first word boundary next to `fragment_size` . You can set `fragment_size` to 0 to never split any sentence.

`word`

Break highlighted fragments at the next word boundary, as determined by Java's BreakIterator. You can specify the locale to use with `boundary_scanner_locale` .

boundary_scanner_locale

Controls which locale is used to search for sentence and word boundaries. This parameter takes a form of a language tag, e.g. `"en-US"` , `"fr-FR"` , `"ja-JP"` . More info can be found in the Locale Language Tag documentation. The default value is Locale.ROOT.

encoder

Indicates if the snippet should be HTML encoded: `default` (no encoding) or `html` (HTML-escape the snippet text and then insert the highlighting tags)

fields

Specifies the fields to retrieve highlights for. You can use wildcards to specify fields. For example, you could specify `comment_*` to get highlights for all text and keyword fields that start with `comment_` .

Note	Only text and keyword fields are highlighted when you use wildcards. If you use a custom mapper and want to highlight on a field anyway, you must explicitly specify that field name.

force_source

Highlight based on the source even if the field is stored separately. Defaults to `false`.

fragmenter

Specifies how text should be broken up in highlight snippets: `simple` or `span`. Only valid for the `plain` highlighter. Defaults to `span`.

`simple`

Breaks up text into same-sized fragments.

`span`

Breaks up text into same-sized fragments, but tries to avoid breaking up text between highlighted terms. This is helpful when you're querying for phrases. Default.

fragment_offset

Controls the margin from which you want to start highlighting. Only valid when using the `fvh` highlighter.

fragment_size

The size of the highlighted fragment in characters. Defaults to 100.

highlight_query

Highlight matches for a query other than the search query. This is especially useful if you use a rescore query because those are not taken into account by highlighting by default.

Important	{es} does not validate that `highlight_query` contains the search query in any way so it is possible to define it so legitimate query results are not highlighted. Generally, you should include the search query as part of the `highlight_query`.

matched_fields

Combine matches on multiple fields to highlight a single field. This is most intuitive for multifields that analyze the same string in different ways. All `matched_fields` must have `term_vector` set to `with_positions_offsets`, but only the field to which the matches are combined is loaded so only that field benefits from having `store` set to `yes`. Only valid for the `fvh` highlighter.

no_match_size

The amount of text you want to return from the beginning of the field if there are no matching fragments to highlight. Defaults to 0 (nothing is returned).

number_of_fragments

The maximum number of fragments to return. If the number of fragments is set to 0, no fragments are returned. Instead, the entire field contents are highlighted and returned. This can be handy when you need to highlight short texts such as a title or address, but fragmentation is not required. If `number_of_fragments` is 0, `fragment_size` is ignored. Defaults to 5.

order

Sorts highlighted fragments by score when set to `score`. By default, fragments will be output in the order they appear in the field (order: `none`). Setting this option to `score` will output the most relevant fragments first. Each highlighter applies its own logic to compute relevancy scores. See the document How highlighters work internally for more details how different highlighters find the best fragments.

phrase_limit

Controls the number of matching phrases in a document that are considered. Prevents the `fvh` highlighter from analyzing too many phrases and consuming too much memory. When using `matched_fields`, `phrase_limit` phrases per matched field are considered. Raising the limit increases query time and consumes more memory. Only supported by the `fvh` highlighter. Defaults to 256.

pre_tags

Use in conjunction with `post_tags` to define the HTML tags to use for the highlighted text. By default, highlighted text is wrapped in `<em>` and `</em>` tags. Specify as an array of strings.

post_tags

Use in conjunction with `pre_tags` to define the HTML tags to use for the highlighted text. By default, highlighted text is wrapped in `<em>` and `</em>` tags. Specify as an array of strings.

require_field_match

By default, only fields that contains a query match are highlighted. Set `require_field_match` to `false` to highlight all fields. Defaults to `true`.

tags_schema

Set to `styled` to use the built-in tag schema. The `styled` schema defines the following `pre_tags` and defines `post_tags` as `</em>`.

```
<em class="hlt1">, <em class="hlt2">, <em class="hlt3">,
<em class="hlt4">, <em class="hlt5">, <em class="hlt6">,
<em class="hlt7">, <em class="hlt8">, <em class="hlt9">,
<em class="hlt10">
```

type

The highlighter to use: `unified`, `plain`, or `fvh`. Defaults to `unified`.

Highlighting Examples

- Override global settings
- Specify a highlight query
- Set highlighter type
- Configure highlighting tags
- Highlight source
- Highlight all fields
- Combine matches on multiple fields
- Explicitly order highlighted fields
- Control highlighted fragments
- Highlight using the postings list
- Specify a fragmenter for the plain highlighter

Override global settings

You can specify highlighter settings globally and selectively override them for individual fields.

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "number_of_fragments" : 3,
        "fragment_size" : 150,
        "fields" : {
            "body" : { "pre_tags" : ["<em>"], "post_tags" : ["</em>"] },
            "blog.title" : { "number_of_fragments" : 0 },
            "blog.author" : { "number_of_fragments" : 0 },
            "blog.comment" : { "number_of_fragments" : 5, "order" : "score" }
        }
    }
}
```

Specify a highlight query

You can specify a `highlight_query` to take additional information into account when highlighting. For example, the following query includes both the search query and rescore query in the `highlight_query`. Without the `highlight_query`, highlighting would only take the search query into account.

```
GET /_search
{
    "stored_fields": [ "_id" ],
    "query" : {
        "match": {
            "comment": {
                "query": "foo bar"
            }
        }
    },
    "rescore": {
        "window_size": 50,
        "query": {
            "rescore_query" : {
                "match_phrase": {
                    "comment": {
                        "query": "foo bar",
                        "slop": 1
                    }
                }
            },
            "rescore_query_weight" : 10
        }
    },
    "highlight" : {
        "order" : "score",
        "fields" : {
            "comment" : {
                "fragment_size" : 150,
                "number_of_fragments" : 3,
                "highlight_query": {
                    "bool": {
                        "must": {
                            "match": {
                                "comment": {
                                    "query": "foo bar"
                                }
                            }
                        },
                        "should": {
                            "match_phrase": {
                                "comment": {
                                    "query": "foo bar",
                                    "slop": 1,
                                    "boost": 10.0
                                }
                            }
                        },
                        "minimum_should_match": 0
                    }
                }
            }
        }
    }
}
```

Set highlighter type

The `type` field allows to force a specific highlighter type. The allowed values are: `unified`, `plain` and `fvh`. The following is an example that forces the use of the plain highlighter:

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "comment" : {"type" : "plain"}
        }
    }
}
```

Configure highlighting tags

By default, the highlighting will wrap highlighted text in `<em>` and `</em>`. This can be controlled by setting `pre_tags` and `post_tags`, for example:

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "pre_tags" : ["<tag1>"],
        "post_tags" : ["</tag1>"],
        "fields" : {
            "body" : {}
        }
    }
}
```

When using the fast vector highlighter, you can specify additional tags and the "importance" is ordered.

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "pre_tags" : ["<tag1>", "<tag2>"],
        "post_tags" : ["</tag1>", "</tag2>"],
        "fields" : {
            "body" : {}
        }
    }
}
```

You can also use the built-in `styled` tag schema:

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "tags_schema" : "styled",
        "fields" : {
            "comment" : {}
        }
    }
}
```

Highlight on source

Forces the highlighting to highlight fields based on the source even if fields are stored separately. Defaults to `false` .

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "comment" : {"force_source" : true}
        }
    }
}
```

Highlight in all fields

By default, only fields that contains a query match are highlighted. Set `require_field_match` to `false` to highlight all fields.

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "require_field_match": false,
        "fields": {
                "body" : { "pre_tags" : ["<em>"], "post_tags" : ["</em>"] }
        }
    }
}
```

Combine matches on multiple fields

Warning	This is only supported by the `fvh` highlighter

The Fast Vector Highlighter can combine matches on multiple fields to highlight a single field. This is most intuitive for multifields that analyze the same string in different ways. All `matched_fields` must have `term_vector` set to `with_positions_offsets` but only the field to which the matches are combined is loaded so only that field would benefit from having `store` set to `yes` .

In the following examples, `comment` is analyzed by the `english` analyzer and `comment.plain` is analyzed by the `standard` analyzer.

```
GET /_search
{
    "query": {
        "query_string": {
            "query": "comment.plain:running scissors",
            "fields": ["comment"]
        }
    },
    "highlight": {
        "order": "score",
        "fields": {
            "comment": {
                "matched_fields": ["comment", "comment.plain"],
                "type" : "fvh"
            }
        }
    }
}
```

The above matches both "run with scissors" and "running with scissors" and would highlight "running" and "scissors" but not "run". If both phrases appear in a large document then "running with scissors" is sorted above "run with scissors" in the fragments list because there are more matches in that fragment.

```
GET /_search
{
    "query": {
        "query_string": {
            "query": "running scissors",
            "fields": ["comment", "comment.plain^10"]
        }
    },
    "highlight": {
        "order": "score",
        "fields": {
            "comment": {
                "matched_fields": ["comment", "comment.plain"],
                "type" : "fvh"
            }
        }
    }
}
```

The above highlights "run" as well as "running" and "scissors" but still sorts "running with scissors" above "run with scissors" because the plain match ("running") is boosted.

```
GET /_search
{
    "query": {
        "query_string": {
            "query": "running scissors",
            "fields": ["comment", "comment.plain^10"]
        }
    },
    "highlight": {
        "order": "score",
        "fields": {
            "comment": {
                "matched_fields": ["comment.plain"],
                "type" : "fvh"
            }
        }
    }
}
```

The above query wouldn't highlight "run" or "scissor" but shows that it is just fine not to list the field to which the matches are combined (`comment`) in the matched fields.

Note	Technically it is also fine to add fields to `matched_fields` that don't share the same underlying string as the field to which the matches are combined. The results might not make much sense and if one of the matches is off the end of the text then the whole query will fail.

Note

There is a small amount of overhead involved with setting `matched_fields` to a non-empty array so always prefer

```
"highlight": {
    "fields": {
        "comment": {}
    }
}
```

to

```
"highlight": {
    "fields": {
        "comment": {
            "matched_fields": ["comment"],
            "type" : "fvh"
        }
    }
}
```

Explicitly order highlighted fields

Elasticsearch highlights the fields in the order that they are sent, but per the JSON spec, objects are unordered. If you need to be explicit about the order in which fields are highlighted specify the `fields` as an array:

```
GET /_search
{
    "highlight": {
        "fields": [
            { "title": {} },
            { "text": {} }
        ]
    }
}
```

None of the highlighters built into Elasticsearch care about the order that the fields are highlighted but a plugin might.

Control highlighted fragments

Each field highlighted can control the size of the highlighted fragment in characters (defaults to `100`), and the maximum number of fragments to return (defaults to `5`). For example:

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "comment" : {"fragment_size" : 150, "number_of_fragments" : 3}
        }
    }
}
```

On top of this it is possible to specify that highlighted fragments need to be sorted by score:

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "order" : "score",
        "fields" : {
            "comment" : {"fragment_size" : 150, "number_of_fragments" : 3}
        }
    }
}
```

If the `number_of_fragments` value is set to `0` then no fragments are produced, instead the whole content of the field is returned, and of course it is highlighted. This can be very handy if short texts (like document title or address) need to be highlighted but no fragmentation is required. Note that `fragment_size` is ignored in this case.

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "body" : {},
            "blog.title" : {"number_of_fragments" : 0}
        }
    }
}
```

When using `fvh` one can use `fragment_offset` parameter to control the margin to start highlighting from.

In the case where there is no matching fragment to highlight, the default is to not return anything. Instead, we can return a snippet of text from the beginning of the field by setting `no_match_size` (default `0`) to the length of the text that you want returned. The actual length may be shorter or longer than specified as it tries to break on a word boundary.

```
GET /_search
{
    "query" : {
        "match": { "user": "kimchy" }
    },
    "highlight" : {
        "fields" : {
            "comment" : {
                "fragment_size" : 150,
                "number_of_fragments" : 3,
                "no_match_size": 150
            }
        }
    }
}
```

Highlight using the postings list

Here is an example of setting the `comment` field in the index mapping to allow for highlighting using the postings:

```
PUT /example
{
  "mappings": {
    "properties": {
      "comment" : {
        "type": "text",
        "index_options" : "offsets"
      }
    }
  }
}
```

Here is an example of setting the `comment` field to allow for highlighting using the `term_vectors` (this will cause the index to be bigger):

```
PUT /example
{
  "mappings": {
    "properties": {
      "comment" : {
        "type": "text",
        "term_vector" : "with_positions_offsets"
      }
    }
  }
}
```

Specify a fragmenter for the plain highlighter

When using the `plain` highlighter, you can choose between the `simple` and `span` fragmenters:

```
GET twitter/_search
{
    "query" : {
        "match_phrase": { "message": "number 1" }
    },
    "highlight" : {
        "fields" : {
            "message" : {
                "type": "plain",
                "fragment_size" : 15,
                "number_of_fragments" : 3,
                "fragmenter": "simple"
            }
        }
    }
}
```

Response:

```
{
    ...
    "hits": {
        "total" : {
            "value": 1,
            "relation": "eq"
        },
        "max_score": 1.6011951,
        "hits": [
            {
                "_index": "twitter",
                "_id": "1",
                "_score": 1.6011951,
                "_source": {
                    "user": "test",
                    "message": "some message with the number 1",
                    "date": "2009-11-15T14:12:12",
                    "likes": 1
                },
                "highlight": {
                    "message": [
                        " with the <em>number</em>",
                        " <em>1</em>"
                    ]
                }
            }
        ]
    }
}
```

```
GET twitter/_search
{
    "query" : {
        "match_phrase": { "message": "number 1" }
    },
    "highlight" : {
        "fields" : {
            "message" : {
                "type": "plain",
                "fragment_size" : 15,
                "number_of_fragments" : 3,
                "fragmenter": "span"
            }
        }
    }
}
```

Response:

```
{
    ...
    "hits": {
        "total" : {
            "value": 1,
            "relation": "eq"
        },
        "max_score": 1.6011951,
        "hits": [
            {
                "_index": "twitter",
                "_id": "1",
                "_score": 1.6011951,
                "_source": {
                    "user": "test",
                    "message": "some message with the number 1",
                    "date": "2009-11-15T14:12:12",
                    "likes": 1
                },
                "highlight": {
                    "message": [
                        " with the <em>number</em> <em>1</em>"
                    ]
                }
            }
        ]
    }
}
```

If the `number_of_fragments` option is set to `0`, `NullFragmenter` is used which does not fragment the text at all. This is useful for highlighting the entire contents of a document or field.

highlighters-internal.asciidoc