MAC - Labo 3 : Indexing and Search with Elasticsearch

Olivier D'Ancona & Hugo Huart & Nelson Jeanrenaud

Contents

2.2 Indexing

Using the following pipeline:

```
PUT _ingest/pipeline/my_pipeline
{
  "processors": [
    {
      "csv": {
        "field": "_row",
        "target_fields": [
          "id",
          "author",
          "title",
          "date",
          "summary"
        ],
        "separator": "\t",
        "quote": "§"
      }
    },
    {
      "split": {
        "field": "author",
        "separator": ";",
        "ignore_missing": true
      }
    },
    {
      "remove": {
        "field": "_row"
      }
    }
  ]
}
```

D.1

API requests to create **cacm_standard**:

Mappings:

```
PUT /cacm_standard
{
  "mappings": {
    "properties": {
      "author": {
        "type": "keyword"
      },
      "date": {
        "type": "date"
      },
      "id": {
        "type": "unsigned_long"
      },
      "summary": {
        "type": "text",
        "fielddata": true
      },
      "title": {
        "type": "text",
        "fielddata": true
      }
    }
  }
}
```

Reindex:

```
POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard",
    "pipeline": "my_pipeline"
  }
}
```

D.2

API requests to create **`cacm_termvector`**:

Mappings:

```
PUT /cacm_termvector
{
  "mappings": {
    "properties": {
      "author": {
        "type": "keyword"
      },
      "date": {
        "type": "date"
      },
      "id": {
        "type": "unsigned_long"
      },
      "summary": {
        "type": "text",
        "term_vector": "with_positions"
      },
      "title": {
        "type": "text"
      }
    }
  }
}
```

Reindex:

```
POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_termvector",
    "pipeline": "my_pipeline"
  }
}
```

D.3

API request to query a term vector:

`GET /cacm_termvector/_termvectors/gNa1ZYAB7VfE5TWZZFs7`

`gNa1ZYAB7VfE5TWZZFs7` being the ID of a document that has a `summary` field

D.4

The official documentation of Elasticsearch describes a term vector as the following:

> *Term vectors contain information about the terms produced by the analysis process, including:*
>
> - *A list of terms.*
> - *The position (or order) of each term.*
> - *The start and end character offsets mapping the term to its origin in the original string.*
> - *Payloads (if they are available) — user-defined binary data associated with each term position.*

D.5

Sizes of the indexes:

- `cacm_raw` : **1.34MB**
- `cacm_standard` : **1.48MB**
- `cacm_termvector` : **2.07MB**

2.3 Reading Index

D.6

Using the following request, we observe that **Thacher Jr., H. C.** is the author with the highest number of publications. He has **38** publications.

Request:

```
GET /cacm_standard/_search
{
  "aggs": {
    "genres": {
      "terms": {
        "field": "author",
        "size": 1
      }
    }
  }
}
```

D.7

Using the following request, we observe that the top 10 terms are:

1. *of*
2. *algorithm*
3. *a*
4. *for*
5. *the*
6. *and*
7. *in*
8. *on*
9. *an*
10. *computer*

Request:

```
GET /cacm_standard/_search
{
  "aggs": {
    "genres": {
      "terms": {
        "field": "title",
        "size": 10
      }
    }
  }
}
```

2.4 Using different Analyzers

D.8

The following requests create indexes with the required analyzers.

whitespace analyzer

```
PUT /cacm_standard_whitespace
{
  "settings": {
    "analysis": {
      "analyzer": "whitespace"
    }
  },
  "mappings": {
    "properties": {
      "id":{"type": "unsigned_long"},
      "author": {"type": "keyword"},
      "title":{"type": "text", "fielddata": true},
      "date":{"type": "date"},
      "summary":{"analyzer" : "whitespace", "type": "text", "fielddata" : true}
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_whitespace",
    "pipeline": "my_pipeline"
  }
}
```

english analyzer

```
PUT /cacm_standard_english
{
  "settings": {
    "analysis": {
      "analyzer": "english"
    }
  },
  "mappings": {
    "properties": {
      "id":{"type": "unsigned_long"},
      "author": {"type": "keyword"},
      "title":{"type": "text", "fielddata": true},
      "date":{"type": "date"},
      "summary":{"analyzer": "english", "type": "text", "fielddata" : true}
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_english",
    "pipeline": "my_pipeline"
  }
}
```

standard analyzer with shingles of size 1 and 2

```
PUT /cacm_standard_myanalyzer1
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer1": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "custom_shingle"
          ]
        }
      },
      "filter": {
        "custom_shingle": {
          "type": "shingle",
          "max_shingle_size": 2
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "id": { "type": "unsigned_long" },
      "author": { "type": "keyword" },
      "title": { "type": "text",
        "fielddata": true },
      "date": { "type": "date" },
      "summary": { "analyzer": "my_analyzer1", "type": "text", "fielddata": true }
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_myanalyzer1",
    "pipeline": "my_pipeline"
  }
}
```

standard analyzer with shingles of size 3

```
PUT /cacm_standard_myanalyzer2
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_analyzer2": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "custom_shingle"
          ]
        }
      },
      "filter": {
        "custom_shingle": {
          "type": "shingle",
          "min_shingle_size": 3,
          "max_shingle_size": 3
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "id": { "type": "unsigned_long" },
      "author": { "type": "keyword" },
      "title": { "type": "text", "fielddata": true },
      "date": { "type": "date" },
      "summary": {"analyzer": "my_analyzer2", "type": "text", "fielddata": true }
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_myanalyzer2",
    "pipeline": "my_pipeline"
  }
}
```

stop analyzer

```
PUT /cacm_standard_stopwords
{
  "settings": {
    "analysis": {
      "analyzer": {
        "stopwords": {
          "tokenizer": "lowercase",
          "filter": [ "custom_stopwords" ]
        }
      },
      "filter" : {
        "custom_stopwords" : {
          "type" : "stop",
          "stopwords_path" : "data/common_words.txt"
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "id": { "type": "unsigned_long" },
      "author": { "type": "keyword" },
      "title": { "type": "text", "fielddata": true },
      "date": { "type": "date" },
      "summary": { "analyzer" : "stopwords", "type": "text", "fielddata": true }
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_stopwords",
    "pipeline": "my_pipeline"
  }
}
```

D.9

Explanation of the analyzers, according to the Elasticsearch documentation:

- `whitespace` : Breaks text into terms whenever a whitespace is encountered.
- `english` : Targeted for english text. It features relevant stop words, plural to singular conversion and other similar language-specific filters.
- `standard` with shingles of size 1 and 2 : Produce shingles (or word n-gram) up to a size of two,

  The text `"I Love MAC"` would produce `["I", "I Love", "Love", "Love MAC", "MAC"]`.
- `standard` with shingles of size 3 only : Produce shingles (or word n-gram) of size 3,

  The text `"I Love MAC"` would produce `["I", "I Love MAC", "Love", "MAC"]`.
- `stop` : Uses a list of words as stop words that will be removed from the the requested text.

D.10

Using the Index stats and search APIs with the following types of requests:

```
GET /${INDEX_NAME}/_stats
```

```
GET /${INDEX_NAME}/_search
```

The results are:

Analyzer type:	`whitespace`	`english`	`standard` shingles 1-2	`standard` shingles 3	`stop`
a)	3'202 docs	3'202 docs	3'202 docs	3'202 docs	3'202 docs
b)	103'275 terms	72'298 terms	237'189 terms	242'248 terms	59'988 terms
c)	of	which	the	the	computer
	the	us	of	of	system
	is	comput	a	a	paper
	and	program	is	is	presented
	a	system	and	and	time
	to	present	to	to	program
	in	describ	in	in	data
	for	paper	for	for	method
	The	can	are	are	algorithm
	are	gener	of the	this	discussed
d)	13'542'719 B	19'859'606 B	2'597'942 B	3'615'184 B	2'833'980 B
e)	350 ms	250 ms	340 ms	300 ms	340 ms

D.11

Several statements can be made regarding the previous results, here are our 3 concluding ones:

1. All the indexes have the same number of document, the presentation of the documents is not altered.
2. The shingle-based indexes have the most terms. This make sense because they are the only indexes that add new terms in summary (the shingles).
3. The custom stop words provided in `stop` are more restrictive than the default ones of `english`. This is confirmed by the lower number of terms in the `stop` index.

2.5 Searching

D.12

Here are the API requests of the relevant queries:

1.

```
GET /cacm_standard_english/_search
{
  "query" : {
    "query_string" : {
      "query" : "Information Retrieval",
      "default_field": "summary"
    }
  },
  "_source": "id"
}
```

2.

```
GET /cacm_standard_english/_search
{
  "query" : {
    "query_string" : {
      "query" : "Information AND Retrieval",
      "default_field": "summary"
    }
  },
  "_source": "id"
}
```

3.

```
GET /cacm_standard_english/_search
{
  "query" : {
    "query_string" : {
      "query" : "(Retrieval OR (Retrieval AND Information)) AND NOT Database",
      "default_field": "summary"
    }
  },
  "_source": "id"
}
```

4.

```
GET /cacm_standard_english/_search
{
  "query" : {
    "query_string" : {
      "query" : "Info*",
      "default_field": "summary"
    }
  },
  "_source": "id"
}
```

5.

```
GET /cacm_standard_english/_search
{
  "query" : {
    "query_string" : {
      "query" : "\"Information Retrieval\"~5",
      "default_field": "summary"
    }
  },
  "_source": "id"
}
```

D.13

Here are the results of the previous API requests:

1. **240** hits
2. **36** hits
3. **69** hits
4. **205** hits
5. **30** hits

2.6 Custom similarity

D.14

The new index with custom scoring method is created with the following API request:

```
PUT /cacm_standard_score
{
  "settings": {
    "number_of_shards": 1,
    "similarity": {
      "scripted_tfidf": {
        "type": "scripted",
        "script": {
          "source": "double tf = 1 + Math.log(doc.freq); double idf = Math.log((field.docC
        }
      }
    }
  },
  "mappings": {
    "properties": {
      "author": { "type": "keyword" },
      "date": { "type": "date" },
      "id": { "type": "unsigned_long" },
      "summary": { "similarity": "scripted_tfidf", "type": "text", "fielddata": true },
      "title": { "type": "text", "fielddata": true }
    }
  }
}

POST _reindex
{
  "source": {
    "index": "cacm_raw"
  },
  "dest": {
    "index": "cacm_standard_score",
    "pipeline": "my_pipeline"
  }
}
```

D.15

Here are the top 10 results of the `compiler program` query with and without the custom scoring system:

Default scoring:


```
"hits" : [
      {
        "_index" : "cacm_standard",
        "_id" : "-njFZYAB9pJXcpxGir2n",
        "_score" : 4.59885,
        "_source" : {
          "id" : "3130"
        }
      },
      {
        "_index" : "cacm_standard",
        "_id" : "n3jFZYAB9pJXcpxGibfn",
        "_score" : 4.5247345,
        "_source" : {
          "id" : "1503"
        }
      },
      {
        "_index" : "cacm_standard",
        "_id" : "B3jFZYAB9pJXcpxGibLh",
        "_score" : 4.391566,
        "_source" : {
          "id" : "71"
        }
      },
      {
        "_index" : "cacm_standard",
        "_id" : "2XjFZYAB9pJXcpxGibbm",
        "_score" : 4.308632,
        "_source" : {
          "id" : "1305"
        }
      },
      {
        "_index" : "cacm_standard",
        "_id" : "MnjFZYAB9pJXcpxGir2m",
        "_score" : 4.308632,
        "_source" : {
          "id" : "2930"
        }
      },
```

```
  {
    "_index" : "cacm_standard",
    "_id" : "GnjFZYAB9pJXcpxGibbm",
    "_score" : 4.2708445,
    "_source" : {
      "id" : "1114"
    }
  },
  {
    "_index" : "cacm_standard",
    "_id" : "rXjFZYAB9pJXcpxGibrp",
    "_score" : 4.139876,
    "_source" : {
      "id" : "2285"
    }
  },
  {
    "_index" : "cacm_standard",
    "_id" : "-XjFZYAB9pJXcpxGibTl",
    "_score" : 4.1280527,
    "_source" : {
      "id" : "825"
    }
  },
  {
    "_index" : "cacm_standard",
    "_id" : "1njFZYAB9pJXcpxGibLj",
    "_score" : 4.0737095,
    "_source" : {
      "id" : "278"
    }
  },
  {
    "_index" : "cacm_standard",
    "_id" : "mHjFZYAB9pJXcpxGibTl",
    "_score" : 4.0378237,
    "_source" : {
      "id" : "728"
    }
  }
]
```

Custom scoring:


```
"hits" : [
      {
        "_index" : "cacm_standard_score",
        "_id" : "B3jFZYAB9pJXcpxGibLh",
        "_score" : 11.760702,
        "_source" : {
          "id" : "71"
        }
      },
      {
        "_index" : "cacm_standard_score",
        "_id" : "q3jFZYAB9pJXcpxGibXl",
        "_score" : 11.38064,
        "_source" : {
          "id" : "1003"
        }
      },
      {
        "_index" : "cacm_standard_score",
        "_id" : "n3jFZYAB9pJXcpxGibfn",
        "_score" : 11.299849,
        "_source" : {
          "id" : "1503"
        }
      },
      {
        "_index" : "cacm_standard_score",
        "_id" : "-njFZYAB9pJXcpxGir2n",
        "_score" : 11.153636,
        "_source" : {
          "id" : "3130"
        }
      },
      {
        "_index" : "cacm_standard_score",
        "_id" : "1njFZYAB9pJXcpxGibLj",
        "_score" : 10.735809,
        "_source" : {
          "id" : "278"
        }
      },
      {
        "_index" : "cacm_standard_score",
        "_id" : "5XjFZYAB9pJXcpxGibjo",
        "_score" : 10.39101,
```

```
    "_source" : {
      "id" : "1829"
    }
  },
  {
    "_index" : "cacm_standard_score",
    "_id" : "03jFZYAB9pJXcpxGir2n",
    "_score" : 10.323089,
    "_source" : {
      "id" : "3091"
    }
  },
  {
    "_index" : "cacm_standard_score",
    "_id" : "vHjFZYAB9pJXcpxGirym",
    "_score" : 9.628572,
    "_source" : {
      "id" : "2812"
    }
  },
  {
    "_index" : "cacm_standard_score",
    "_id" : "jHjFZYAB9pJXcpxGibXl",
    "_score" : 9.099797,
    "_source" : {
      "id" : "972"
    }
  },
  {
    "_index" : "cacm_standard_score",
    "_id" : "23jFZYAB9pJXcpxGibbm",
    "_score" : 9.099797,
    "_source" : {
      "id" : "1307"
    }
  }
]
```

D.16

Here is the query with the custom function score:

```
GET /cacm_standard_stopwords/_search
{
  "query": {
    "function_score": {
      "query": {
        "query_string": {
          "query": "computer program",
          "default_field": "summary"
        }
      },
      "linear": {
        "date": {
          "origin": "1970-01-01",
          "scale": "90d",
          "offset": "0d",
          "decay": 0.5
        }
      }
    }
  }
}
```