

09 - Textová analýza

Při ukládání dat do Elasticsearch probíhá na polích typu `text` textová analýza. Během **analýzy** Elasticsearch rozdělí text na slova (**tokeny**), převede je do základní formy (například převede množné číslo na jednotné) a uloží je do speciální datové struktury zvané **invertovaný index**.

Standard analyzer

Výchozí analyzátor se jmenuje `standard`. S jeho pomocí je možné implementovat full textové vyhledávání nezávislé na velikosti písmen. Pro kontrolu, jak bude vypadat výsledek analýzy pro daný text lze využít endpoint `_analyze`:

```
POST _analyze
{
  "analyzer": "standard",
  "text": "Order with ID d03fd6k-2da consists of four products."
}
```

```
{
  "tokens" : [
    {
      "token" : "order",
      "start_offset" : 0,
      "end_offset" : 5,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "with",
      "start_offset" : 6,
      "end_offset" : 10,
      "type" : "<ALPHANUM>",
      "position" : 1
    },
    {
      "token" : "id",
      "start_offset" : 11,
      "end_offset" : 13,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "d03fd6k",
      "start_offset" : 14,
```

```

    "end_offset" : 21,
    "type" : "<ALPHANUM>",
    "position" : 3
  },
  {
    "token" : "2da",
    "start_offset" : 22,
    "end_offset" : 25,
    "type" : "<ALPHANUM>",
    "position" : 4
  },
  {
    "token" : "consists",
    "start_offset" : 26,
    "end_offset" : 34,
    "type" : "<ALPHANUM>",
    "position" : 5
  },
  {
    "token" : "of",
    "start_offset" : 35,
    "end_offset" : 37,
    "type" : "<ALPHANUM>",
    "position" : 6
  },
  {
    "token" : "four",
    "start_offset" : 38,
    "end_offset" : 42,
    "type" : "<ALPHANUM>",
    "position" : 7
  },
  {
    "token" : "products",
    "start_offset" : 43,
    "end_offset" : 51,
    "type" : "<ALPHANUM>",
    "position" : 8
  }
]
}

```

Při definici mapování lze zvolit, jaký analyzátor má být pro dané pole použit. Stejný analyzátor pak bude použit i pro vyhledávání.

```
PUT data_analyzed
{
  "mappings": {
    "properties": {
      "description": {
        "type": "text",
        "analyzer": "standard"
      }
    }
  }
}
```

Jazykové analyzéry

Přestože bude `standard` analyzer pro mnoho případů dostačující, při práci s přirozeným jazykem dosáhneme lepších výsledků s specifickými analyzéry pro dané jazyky, tzv. [language analyzers](#). Pro porovnání můžeme vyzkoušet anglický analyzátor (`english`) pro shodný vstupní text:

```
POST _analyze
{
  "analyzer": "english",
  "text": "Order with ID d03fd6k-2da consists of four products."
}
```

```
{
  "tokens" : [
    {
      "token" : "order",
      "start_offset" : 0,
      "end_offset" : 5,
      "type" : "<ALPHANUM>",
      "position" : 0
    },
    {
      "token" : "id",
      "start_offset" : 11,
      "end_offset" : 13,
      "type" : "<ALPHANUM>",
      "position" : 2
    },
    {
      "token" : "d03fd6k",
      "start_offset" : 14,
      "end_offset" : 21,
      "type" : "<ALPHANUM>",
      "position" : 3
    }
  ]
}
```

```

},
{
  "token" : "2da",
  "start_offset" : 22,
  "end_offset" : 25,
  "type" : "<ALPHANUM>",
  "position" : 4
},
{
  "token" : "consist",
  "start_offset" : 26,
  "end_offset" : 34,
  "type" : "<ALPHANUM>",
  "position" : 5
},
{
  "token" : "four",
  "start_offset" : 38,
  "end_offset" : 42,
  "type" : "<ALPHANUM>",
  "position" : 7
},
{
  "token" : "product",
  "start_offset" : 43,
  "end_offset" : 51,
  "type" : "<ALPHANUM>",
  "position" : 8
}
]
}

```

Jazyková analýza navíc převede:

- products => product
- consists => consist

Díky tomu je možné vyhledávat v dokumentech nezávisle na tvarosloví.

Custom analyzer

Pokud však ani jazyková analýza nedostačuje, je nutné vytvořit vlastní (`custom`) analyzér. Každý analyzér sestává z tří částí:

1. Char filter: Filtruje vstupní text na úrovni znaků, lze využít například k odstranění HTML značek
2. Tokenizer: Dělí vstupní text na tokeny
3. Token filter: Modifikuje token (například je převádí na malá písmena, nebo do základního tvaru), odstraňuje nepotřebné tokeny (stopslova) nebo přidává další tokeny (synonyma)

Custom analyzer lze definovat v rámci nastavení indexu:

```
PUT custom_html_analyzer
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_html_analyzer": {
          "type": "custom",
          "tokenizer": "standard",
          "char_filter": ["html_strip"],
          "filter": ["lowercase", "asciifolding"]
        }
      }
    }
  }
}
```

jakmile je index vytvořen, lze otestovat textovou analýzu:

```
POST custom_html_analyzer/_analyze
{
  "analyzer": "my_html_analyzer",
  "text": "<b>Hello</b> world!"
}
```

Nejpoužívanější filtry jsou:

Character filter

- `html_strip`: odstraní HTML značky
- `mapping`: nahradí znaky
- `pattern_replace`: nahradí text dle zadaného regulárního výrazu

Tokenizer

- `standard`: výchozí tokenizer
- `keyword`: zachová text nerozdělený
- `letter`: vytváří tokeny podle všeho, co není písmeno
- `whitespace`: vytváří tokeny pouze podle bílých znaků
- `pattern`: token může být reprezentován regulárním výrazem
- `ngram` a `edge_ngram`: vytváří tokeny pro našeptávání a fuzzy vyhledávání
- `uax_url_email`: rozdělí e-mailovou adresu na tokeny
- `path_hierarchy`: cesty v filesystému

Token filter

- `asciifolding`, `icu_folding`: odstranění diakritiky
- `lowercase`, `uppercase`: převedení na malá nebo velká písmena
- `stop`: odstraní stopslova
- `stemmer`: algoritmické převedení slova na základní tvar
- `hunspell`: slovníkové převedení slova na základní tvar
- `synonym`: přidá synonyma jako nové tokeny
- `shingle`: vytváří sousloví pro frázové vyhledávání
- `length`: odstraní tokeny na základě jejich délky
- `unique`: ponechá pouze unikátní tokeny

Custom analyzer: příklad českého analyzáru

Při vytváření českého analyzáru je třeba vzít v úvahu:

- Tvarosloví: slova se mění dle časování nebo skloňování
- Stopslova
- Diakritiky: vyhledávání často probíhá bez použití diakritiky
- Velikost písmen: vyhledávání je často zapsáno jen malými písmeny
- Někdy dává smysl zařadit do vyhledávání práci s synonymy

```
PUT product-czech
{
  "settings": {
    "analysis": {
      "analyzer": {
        "my_czech": {
          "type": "custom",
          "tokenizer": "standard",
          "filter": [
            "stemmer_cs",
            "lowercase",
            "stop_cs",
            "asciifolding",
            "synonym_cz",
            "unique"
          ]
        }
      },
      "filter": {
        "stemmer_cs": {
          "type": "stemmer",
          "language": "czech"
        },
        "stop_cs": {
          "type": "stop",
```

```

        "stopwords": [
            "_czech_"
        ]
    },
    "synonym_cz": {
        "type": "synonym",
        "synonyms" : [
            "lednicka,lednice=>chladnicka"
        ]
    }
}
},
"mappings": {
    "properties": {
        "name": {
            "type": "text",
            "analyzer": "my_czech"
        }
    }
}
}

POST product-czech/_analyze
{
    "analyzer": "my_czech",
    "text": "Kombinovaná lednička s mrazákem BOSCH KGN39VL35"
}

POST product-czech/_doc
{
    "name": "Chladnička BOSCH KWG22VL30"
}

GET product-czech/_search
{
    "_source": "name",
    "query": {
        "match": {
            "name": "lednicka"
        }
    }
}
}

```

Alternativou k algoritmickeému stemmeru může být slovníkový, který je v případě Elasticsearch reprezentován `hunspell11` token filtrem. Jakmile přidáte slovníky do konfigurační složky Elasticsearch, můžete nadefinovat v rámci analýzy vlastní filtr:

```
...
"filter": {
  "hunspell_cs": {
    "type": "hunspell",
    "locale": "cs_CZ"
  },
  ...
}
```

Částečná shoda, překlepy

Uživatelský vstup často obsahuje chyby ve formě překlepů. Vypořádat se s nimi je možné:

1. V době dotazu
 1. `match` query s argumentem `fuzziness` (pro pole typu `text`)
 2. `fuzzy` query (pro pole typu `keyword`)
2. Při indexování dat
 1. `ngram` tokenizer

Při hledání s překlepy dávejte pozor na výkonnost. Fuzzy query může vyhledávání zdatelně zpomalit. Na druhé straně při použití n-gramů dochází k nárůstu velikosti indexů.

Alternativou k n-gramům může být suggester ([docs](#)). Nabízí rychlé vyhledávání za cenu vyššího využití paměti. Dále není tak konfigurovatelný.

Našeptávač

Situace je obdobná jako v předchozím případě s jediným rozdílem: Přesně víme, že hledáme od začátku slova. Opět máme k dispozici obdobné přístupy:

1. V době dotazu
 1. `match_phrase_prefix` query pro pole typu `text`
 2. `prefix` query pro pole typu `keyword`
2. Při indexování dat
 1. `edge_ngram` tokenizer
 2. `search_as_you_type` datový typ:

```
PUT brands
{
  "mappings": {
    "properties": {
      "name": {
        "type": "search_as_you_type"
      }
    }
  }
}
```



```
}  
}  
  
PUT brands/_doc/1?refresh  
{  
  "name": "Western Digital"  
}  
  
GET brands/_search  
{  
  "query": {  
    "multi_match": {  
      "query": "western di",  
      "type": "bool_prefix",  
      "fields": [  
        "name",  
        "name._2gram",  
        "name._3gram"  
      ]  
    }  
  }  
}
```

Úkol: custom analyzer

1. Připravte nastavení indexu pro nový index nazvaný `my_em_analyzer` — zkopírujte nastavení anglického analyzáru z <https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-lang-analyzer.html#english-analyzer>
2. Rozšiřte analyzáru:
 1. Přidejte character filter, který odstraní HTML značky z vstupního textu
 2. Přidejte custom synonym filter, který převede slovo `failure` na `error`
3. Vytvořte následující mapping:
 1. Název pole: `message`
 2. Typ pole: `text`
 3. Analyzer: pole `message` by mělo využívat analyzer definovaný v předchozím kroku
4. Uložte následující dokumenty do ES:

```
POST my_em_analyzer/_doc
{
  "message": "An error occured"
}

POST my_em_analyzer/_doc
{
  "message": "There was a failure during processing request"
}
```

5. Vyhledejte `error` v poli `message`. Oba dokumenty by měly být nalezeny.