KATEDRA INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI
www.inf.upol.cz

Jan Laštovička
https://www.inf.upol.cz/lide/jan-lastovicka
jan.lastovicka@upol.cz
17. listopadu 12, 771 46 Olomouc

Databáze ◇ poznámky k přednášce

12. Nepřesnost

verze z 3. prosince 2023

1 Levenštejnova vzdálenost

Vezměme libovolné řetězce x, y. Definujeme vzdálenost řetězců x, y následovně.

1. Pokud je řetězce y prázdný, pak je vzdálenost rovna délce řetězce x,
2. pokud je řetězec x prázdný, pak je vzdálenost rovna délce řetězce y,
3. začínají-li oba řetězce x i y stejným znakem, pak je vzdálenost rovna vzdálenosti řetězců x,y bez prvních znaků,
4. pokud oba řetězce nezačínají stejným znakem, pak je vzdálenost rovna jedna plus minimum z hodnot
   (a) vzdálenost řetězce x bez prvního znaku a řetězce y,
   (b) vzdálenost řetězce x a řetězce y bez prvního znaku
   (c) a vzdálenost řetězců x, y bez prvních znaků.

Například: Vzdálenost `""` a `""` je nula. Vzdálenost `"ab"` a `"ab"` je nula. Vzdálenost `""` a `"a"` je jedna. Vzdálenost `"ab"` a `""` je dva. Vzdálenost `"a"` a `"ab"` je jedna. Vzdálenost `"ba"` a `"aa"` je jedna.

Podmínka na hodnotu v podmínce na dokument `match` může obsahovat položku `fuzziness` s hodnotou ***`distance`***, která může být 0, 1 nebo 2. Pokud je položka `fuzziness` přítomna, pak se dva termy rovnají, pokud je jejich vzdálenost menší nebo rovna než ***`distance`***.Vzdálenost také ovlivňuje *boost* složku skóre. Čím menší vzdálenost, tím větší *boost*.

Příklad dotazu:

```
GET /book/_search
{
  "query": {
    "match": {
      "title": {
        "query": "king",
        "fuzziness": 1
      }
    }
  }
}
```

2 Prefix

Dokument splňuje podmínku

```
{
  "prefix": {
    field: {
      "value": prefix
    }
  }
}
```

pokud v položce *field* má term začínající *prefix*. Vyhovující dokumenty mají skóre 1.

Dokument splňuje podmínku

```
{
  "match_phrase_prefix": {
    field: {
      "query": string
    }
  }
}
```

pokud položka *field* obsahuje za sebou termy *string*. Poslední term v *string* se považuje za prefix.

Položka `slop` určuje maximální počet vynechaných termů mezi uvedenými termy.

Například:

```
GET /book/_search
{
  "query": {
    "match_phrase_prefix" : {
      "title" : {
        "query" : "Fellowship Rin",
        "slop" : 2
      }
   }
  }
}
```

3 Vyhledávání termů

Dokument splňuje podmínku

```
{
  "term": {
    field: {
      "value": term
    }
  }
}
```

pokud má v položce *field* term *term*.

Například:

```
GET /book/_search
{
  "query": {
    "term": {
      "title": {
        "value": "fellowship"
      }
    }
  }
}
```

Podmínky `term` a `match` mohou mít položku `boost` udávající koeficient, kterým se násobí skóre vyhovujících dokumentů. Například:

```
GET /book/_search
{
  "query": {
    "match": {
      "title": {
        "query": "king",
        "boost": 2
      }
    }
  }
}
```

4 Typy položek

Každá položka dokumentu uložená v indexu má deklarovaný typ. Deklarace typů se také nazývá *mapování*. Deklaraci typů položek indexu získáme požadavkem:

```
GET /index/_mapping
```

Například požadavek:

```
GET /book/_mapping
```

může mít odpověď:

```
{
  "book": {
    "mappings": {
      "properties": {
        "title": {
          "type": "text",
          "fields": {
            "keyword": {
              "type": "keyword",
              "ignore_above": 256
            }
          }
        }
      }
    }
  }
}
```

Na cestě *index*`.mappings.properties` v odpovědi je objekt, kde položky popisují typy hodnot položek dokumentů v indexu.

Popis typu obsahuje:

```
{
  "type": type
}
```

kde *type* je typ hodnoty.

Textové typy, u kterých se provádí analýza (rozdělování na termy), mají typ `"text"`. Při vložení dokumentu se automaticky deklarují typy nových položek. Výchozí typ pro řetězec je právě typ `"text"`. Typ `"keyword"` (klíčové slovo) je textový typ, který neprovádí analýzu textu.

Deklaraci typu hodnoty položky lze provést požadavkem:

```
PUT /<index>/_mapping
{
  "properties": {
    field: { "type": type }
  }
}
```

Popis typu hodnoty může obsahovat:

```
{
  "fields": {
    field1: type1,
    field2: type2,
    ⋮
  }
}
```

kde *field1*, *field2*, ... jsou názvy položek a *type1*, *type2*, ... popisy jejich typů. Takový popis hodnoty pak definuje další způsoby indexování.

Například popis typu:

```
{
  "type": "text",
  "fields": {
    "keyword": {
      "type": "keyword",
      "ignore_above": 256
  }
}
```

znamená, že hodnota se indexuje jako `"text"` a pod položkou `"keyword"` jako klíčové slovo. (Položka `ignore_above` rovna 256 znamená, že řetězce delší jak 256 znaků nebudou indexovány jako klíčová slova.)

K dalším indexovaným hodnotám lze přistupovat přes tečkovou notaci. Například:

```
GET /book/_search
{
  "query": {
    "term": {
      "author.keyword": {
        "value": "John Ronald Reuel Tolkien"
      }
    }
  }
}
```

Deklaraci typů položek můžeme provést při vytvoření indexu požadavkem:

```
PUT index/book
{
  "mappings": {
    "properties": types
  }
}
```

Například:

```
PUT /book
{
  "mappings":{
    "properties": {
      "tags": { "type": "keyword" },
      "title": { "type": "text" }
    }
  }
}
```

5 Pole

Pokud je hodnota položky dokumentu pole, pak se na položku nahlíží tak, jako by každý prvek pole byl její hodnotou.

Například pokud vytvoříme index:

```
PUT /book
{
  "mappings":{
    "properties": {
      "tags": { "type": "keyword" },
      "title": { "type": "text" }
    }
  }
}
```

a přidáme dokument:

```
PUT /book/_doc/1
{
  "title": "The Fellowship of the Ring",
  "tags": ["fantasy", "english"]
}
```

pak odpověď na dotaz:

```
GET /book/_search
{
  "query": {
    "term": {
      "tags": {
        "value": "fantasy"
      }
```

```
    }
  }
}
```

bude obsahovat:

```
{
  "hits": {
    "hits": [
      {
        "_source": {
          "title": "The Fellowship of the Ring",
          "tags": [
            "fantasy",
            "english"
          ]
        }
      }
    ]
  }
}
```

6 Složené podmínky

Podmínka na dokument může mít tvar:

```
{
  "bool": clauses
}
```

kde *clauses* je objekt, který může obsahovat položky: `must`, `filter`, `must_not` a `should`. Hodnoty těchto položek se nazývají *klauzule*. Každá klauzule je podmínka na dokument, nebo pole podmínek na dokument.

Podmínky zadané v klauzulích `must` a `filter` musí být splněny. Podmínky zadané v klauzuli `must_not` nesmí být zasaženy. Podmínky v klauzuli `should` mohou být splněny. Je vyžadováno splnění aspoň jedné podmínky.

Skóre zásahu se rovná součtu skóre všech splněných podmínek v klauzuli `must` a `should`. Pokud použijeme klauzuli `should` s `filter` nebo `must`, pak nemusí být podmínky v `should` splněny.

Například:

```
{
  "bool": {
    "must_not": {
```

```
      "term" : { "tags" : { "value" : "scifi" } }
    },
    "should": [
      {
        "term" : { "tags" : { "value" : "english" } }
      }, {
        "term" : { "tags" : { "value" : "american" } }
      }
    ]
  }
}
```

Minimální počet splněných podmínek v klauzuli `should` lze nastavit položkou `minimum_should_match` objektu *`clauses`*.