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

11. Indexování

verze z 26. listopadu 2023

1 Elasticsearch

Elasticsearch[1] je systém na fulltextové vyhledávání. Systém je založen na technologii Apache Lucene [2], který používá datovou strukturu Skip list.

1.1 Instalace serveru

Server stáhněte ze stránek `https://www.elastic.co/downloads/elasticsearch` a rozbalte.

Ve složce serveru zadejte příkaz:

```
elasticsearch-8.11.1 % ./bin/elasticsearch
```

který server spustí. Po spuštění se kromě jiného vytiskne heslo pro uživatele `elastic`:

```
Password for the elastic user (reset with ‘bin/elasticsearch-reset-password
-u elastic‘):
password
```

Heslo si poznamenejte.

Server vypněte například stiskem `Ctrl+C`. V souboru `config/elasticsearch.yml` najděte část:

```
# Enable encryption for HTTP API client connections, such as Kibana, Logstash,
  and Agents
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12
```

a v ní změňte `true` na `false`:

```
  enabled: false
```

[1] https://www.elastic.co
[2] `https://lucene.apache.org`)

Tím dojde k vypnutí šifrování přenosu. Server opět spusťte. Správnost instalace ověříte příkazem, kde za `password` doplníte poznamenané heslo:

```
% curl -u "elastic:password" -GET http://localhost:9200/
{
  "name" : "Jan---Air",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "uC_02YahSWSRABElyGYEMA",
  "version" : {
    "number" : "8.11.1",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "6f9ff581fbcde658e6f69d6ce03050f060d1fd0c",
    "build_date" : "2023-11-11T10:05:59.421038163Z",
    "build_snapshot" : false,
    "lucene_version" : "9.8.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

1.2 Klient

Elasticsearch komunikuje pomocí REST (representational state transfer).

Pro komunikaci se serverem lze použít libovolného REST klienta. Například program pro příkazovou řádku `curl`. My budeme používat rozšíření pro `Visual Studio Code` jménem `REST Client` (`humao.rest-client`). Do souboru `settings.json` je potřeba přidat:

```
    "rest-client.defaultHeaders": {
        "Authorization": "Basic elastic:password",
        "User-Agent": "vscode-restclient",
        "Content-Type": "application/json"
    },
    "rest-client.environmentVariables": {
        "$shared": {
            "host": "http://localhost:9200"
        }
    }
```

kde opět `password` nahradíme poznamenaným heslem. Soubor `settings.json` lze například otevřít přes nastavení rozšíření.

Soubory s příponou `http` popisují dotazy na server. Například:

```
GET {{host}}/ HTTP/1.1
```

Po kliku na `Send request` se zobrazí okno s odpovědí:

```
HTTP/1.1 200 OK
X-elastic-product: Elasticsearch
content-type: application/json
content-encoding: gzip
content-length: 329

{
  "name": "Jan---Air",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "uC_02YahSWSRABElyGYEMA",
  "version": {
    "number": "8.11.1",
    "build_flavor": "default",
    "build_type": "tar",
    "build_hash": "6f9ff581fbcde658e6f69d6ce03050f060d1fd0c",
    "build_date": "2023-11-11T10:05:59.421038163Z",
    "build_snapshot": false,
    "lucene_version": "9.8.0",
    "minimum_wire_compatibility_version": "7.17.0",
    "minimum_index_compatibility_version": "7.0.0"
  },
  "tagline": "You Know, for Search"
}
```

Dotazy se oddělují třemi křížky (`###`). Vše za jedním křížkem do konce řádku je komentář.

Požadavek je dán metodou *`method`*, cestou *`path`* a případně tělem *`body`*. Metoda může být: `GET`, `POST`, `DELETE` nebo `PUT`. Cesta se skládá z kroků oddělených lomítkem (’`/`’). Tělo je zapsáno v JSON formátu.

Pokud požadavek nemá tělo, je zapsán:

```
method {{host}}path HTTP/1.1
```

V případě přítomnosti těla nabyde tvaru:

```
method {{host}}path HTTP/1.1

body
```

Například:

```
GET {{host}}/ HTTP/1.1
```

Zjednodušeně budeme dotazy zapisovat jako:

```
method path
```

nebo:

```
method path
body
```

Například:

```
GET /
```

Odpověď na požadavek má tvar:

```
HTTP/1.1 status
headers

body
```

Část ***headers*** tvoří *hlavičky* odpovědi určující dodatečné informace (například délku odpovědi). Tělo ***body*** je ve formátu JSON.

Běžné statusy odpovědi:

- `200 OK`
- `201 Created`
- `400 Bad Request`
- `401 Unauthorized`
- `404 Not Found`

Například:

```
HTTP/1.1 200 OK
X-elastic-product: Elasticsearch
content-type: application/json
content-encoding: gzip
content-length: 329

{
  "name": "Jan---Air",
  "cluster_name": "elasticsearch",
  "cluster_uuid": "uC_02YahSWSRABElyGYEMA",
  "version": {
    "number": "8.11.1",
    "build_flavor": "default",
```

```
    "build_type": "tar",
    "build_hash": "6f9ff581fbcde658e6f69d6ce03050f060d1fd0c",
    "build_date": "2023-11-11T10:05:59.421038163Z",
    "build_snapshot": false,
    "lucene_version": "9.8.0",
    "minimum_wire_compatibility_version": "7.17.0",
    "minimum_index_compatibility_version": "7.0.0"
  },
  "tagline": "You Know, for Search"
}
```

Odpovědi často zkrátíme jen na jejich tělo.

2 Model

Dokument je libovolný objekt v JSON formátu. Pojmenovaná množina dokumentů se nazývá *index*. Každý dokument v indexu je určen identifikátorem.

Práci s indexem ***index*** zajišťují následující požadavky. Vytvoření indexu:

```
PUT /index
```

informace o indexu:

```
GET /index
```

smazání indexu:

```
DELETE /index
```

V rámci indexu musí mít každý dokument jedinečný identifikátor. Přidání dokumentu:

```
PUT /index/_doc/id
document
```

Získání dokumentu:

```
GET /index/_doc/id
```

Smazání dokumentu:

```
DELETE /index/_doc/id
```

Přidání dokumentu s automaticky vytvořeným identifikátorem:

```
POST /index/_doc
document
```

Identifikátor přidaného dokumentu se nalézá v odpovědi na dotaz v položce `_id`.

3 Rozdělení textu na termy

Před uložením dokumentu do indexu je každá textová hodnota převedena na *termy*.

Požadavek:

```
POST /_analyze
{
  "analyzer": "standard",
  "text": string
}
```

vrátí v odpověď termy vytvořené z řetězce ***string***.

Například řetězec:

```
"The Hobbit, or There and Back Again"
```

se rozloží na termy:

```
"the"
"hobbit"
"or"
"there"
"and"
"back"
"again"
```

Řetězec se rozdělí podle bílých a pomocných znaků (jako čárka) a získané řetězce se převedou na malá písmena.

4 Dotazovací jazyk

Dotaz je vyjádřen pomocí JSON.

Požadavek na dotaz indexu má tvar:

```
POST /index/_search
{
  "query": query
}
```

kde ***query*** je objekt nazývaný *dotaz*.

Tělo odpovědi obsahuje:

```
{
  "hits": {
```

```
    "total": {
      "value": hits_count,
    },
    "hits": hits
  }
}
```

kde *hits_count* je počet zásahů a *hits* je pole zásahů. Zásah obsahuje:

```
{
  "_id": document_id,
  "_source": document
}
```

kde *document* je zasažený dokument a *document_id* jeho identifikátor.

Dotaz

```
{
   "match_all": { }
}
```

zasáhne každý dokument.

Dotaz

```
{
  "match": {
    name: value_query
  }
}
```

zasáhne dokumenty, které mají v položce *name* hodnotu vyhovující predikátu *value_query*.

Predikát může mít tvar:

```
{
  "query": query_string
}
```

kde *query_string* je řetězec. Řetězec *query_string* je také rozložen na termy. Například v dotazu:

```
{
  "query": {
    "match": {
      "title": {
        "query": "THE, HoBBit"
```

```
      }
    }
  }
}
```

je řetězec `"THE, HoBBit"` rozložen na termy:

```
"the"
"hobbit"
```

Dokument je zasažen pokud se aspoň jeden term dotazu *`query_string`* shoduje s termem položky.

Pokud chceme, aby položka obsahovala všechny uvedené termy, použijeme predikát:

```
{
  "query": query_string ,
  "operator": "AND"
}
```

5 Skóre zásahu

Zásahy mají v položce `_score` desetinné číslo nazývané *skóre* zásahu. Čím větší je skóre zásahu, tím lépe dokument vyhovuje dotazu. Zásahy jsou uspořádané podle skóre sestupně.

Výpočet skóre probíhá podle algoritmu BM25.[3] Písmena BM jsou zkratkou za *best matching*.

Skóre dotazu Q obsahující tokeny $q_1, \ldots, q_n$ na hodnotu zadané položky dokumentu D je dáno výpočtem:

$$score(D, Q) = \sum_{i=1}^{n} boost \cdot IDF(q_i) \cdot tf(q_i, D)$$

Základní hodnota *boost* je $k_1 + 1$. Výchozí hodnota parametru k_1 (*term saturation parameter*) je 1.2. Základní hodnota *boost* je tedy 2.2.

Hodnota $IDF(q_i)$ je dána vzorcem

$$ln\left(1 + \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5}\right)$$

Zkratka IDF znamená *inverse document frequency*. Hodnota $n(q_i)$ udává počet dokumentů obsahující v zadané položce term q_i a hodnota N celkový počet dokumentů s položkou.

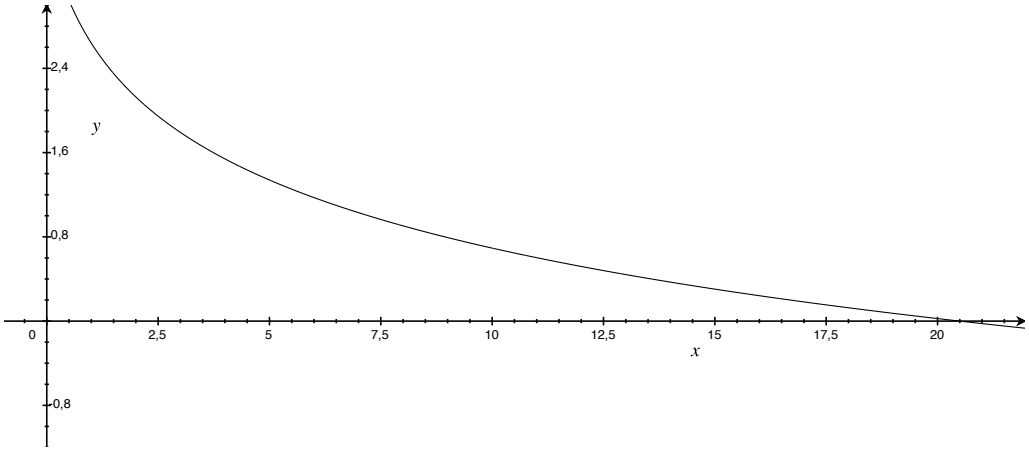
[3] `https://en.wikipedia.org/wiki/Okapi_BM25`

Hodnota $tf(q_i, D)$ je dána vzorcem

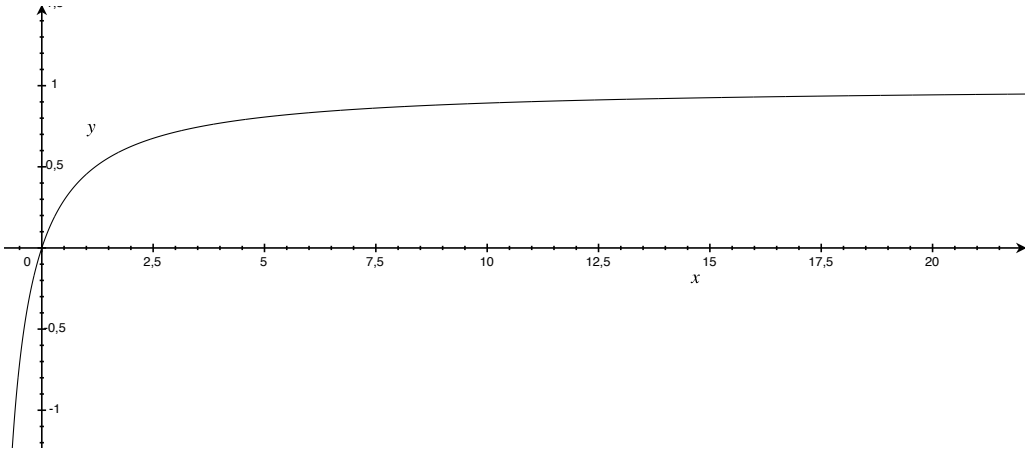
$$\frac{f(q_i, D)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{dl}{avgdl})}$$

Zkratka tf znamená *term frequency*. Hodnota $f(q_i, D)$ udává počet výskytů termu q_i v zadané položce dokumentu D, $k_1 = 1.2$ (*term saturation parameter*), $b = 0.75$ (*length normalization parameter*), hodnota dl (*document length*) udává počet termů v zadané položce dokumentu D a $avgdl$ (*average document length*) udává průměrný počet termů v zadané položce u všech dokumentů.

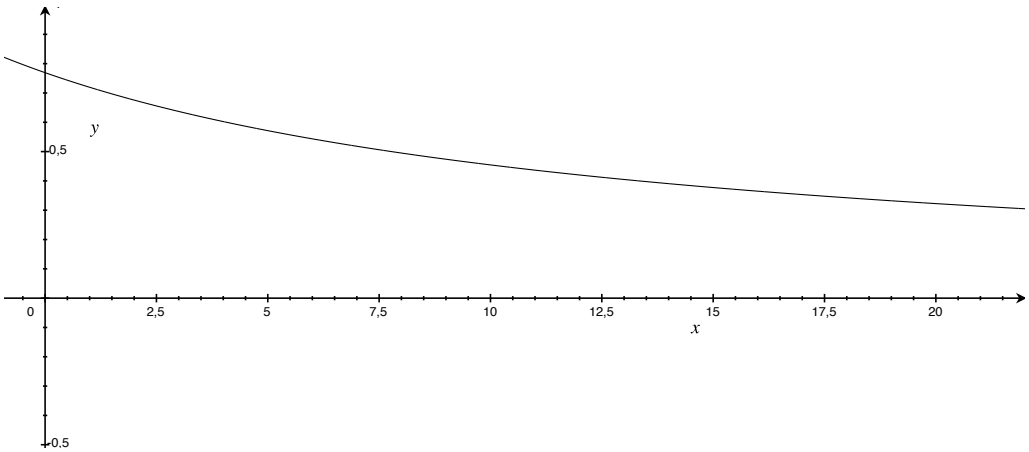
Graf funkce $f(x) = ln\left(1 + \frac{20-x+0.5}{x+0.5}\right)$, která vyjadřuje závislost IDF na počtu dokumentů s hledaným termem pro dvacet dokumentů:



Graf funkce $f(x) = \frac{x}{x+1.2}$, která vyjadřuje závislost tf na počtu nalezených termů, kde položka má průměrný počet termů:



Graf funkce $f(x) = \frac{1}{1+1.2\left(0.25+0.75\frac{x}{10}\right)}$, která vyjadřuje závislost tf na počtu termů, kde se vyskytuje hledaný term pouze jednou:



Uvažujme dokumenty, kde zadaná položka má termy:

1. `the fellowship of the ring`
2. `the two towers`
3. `the return of the king`
4. `the hobbit or there and back again`
5. `titus groan`
6. `gormenghast`
7. `titus alone`

Dotaz Q bude tvořen jediným termem $r_1 =$„`the`“. Spočítáme skóre:

$$\begin{aligned} score(D_1, Q) &= \sum_{i=1}^{n} boost \cdot IDF(q_i) \cdot tf(q_i, D) \\ &= boost \cdot IDF(q_1) \cdot tf(q_1, D) \\ &= 2.2 \cdot IDF(\texttt{the}) \cdot tf(\texttt{the}, D) \end{aligned}$$

Kde:

$$IDF(\texttt{the}) = ln\left(1 + \frac{N - n(\texttt{the}) + 0.5}{n(\texttt{the}) + 0.5}\right)$$

Dosadíme $n(\texttt{the}) = 4$ a $N = 7$ získáme:

$$\begin{aligned} IDF(\texttt{the}) &= ln\left(1 + \frac{7 - 4 + 0.5}{4 + 0.5}\right) \\ &= ln\left(\frac{8}{4.5}\right) \\ &= 0.575364144903562\ldots \end{aligned}$$

Pokračujeme:

$$tf(\texttt{the}, D) = \frac{f(\texttt{the}, D)}{f(\texttt{the}, D) + k_1 \cdot (1 - b + b \cdot \frac{dl}{avgdl})}$$

Dosazením $f(\texttt{the}, D) = 2$, $k_1 = 1.2$, $b = 0.75$, $dl = 5$, $avdl = 3.5714285\ldots$ obdržíme:

$$\begin{aligned} tf(\texttt{the}, D) &\approx \frac{2}{2 + 1.2 \cdot (1 - 0.75 + 0.75 \cdot \frac{5}{3.5714285})} \\ &\approx \frac{2}{2 + 1.2 \cdot (1 - 0.75 + 0.75 \cdot 1.4)} \\ &= \frac{2}{2 + 1.2 \cdot 1.3} \\ &= \frac{2}{3.56} \\ &= 0.561797752808989 \end{aligned}$$

Celkem:

$$\begin{aligned} score(D_1, Q) &= 2.2 \cdot IDF(\texttt{the}) \cdot tf(\texttt{the}, D) \\ &\approx 2.2 \cdot 0.5753 \cdot 0.5617 \\ &\approx 0.711 \end{aligned}$$

Podrobnosti o výpočtu skóre lze získat přidáním položky `explain` s hodnotou `true` k dotazu:

```
{
  "query": query,
  "explain": true
}
```

Například:

```
GET /book/_search
{
  "query": {
    "match": {
      "title": {
        "query": "the"
      }
    }
  },
  "explain": true
}
```

Odpověď obsahuje (popisy jsou zkráceny):

```
{
    "hits": [
      {
        "_source": {
          "title": "The Fellowship of the Ring",
        },
        "_explanation": {
          "value": 0.71112424,
          "details": [
            {
              "value": 0.71112424,
              "description": "score",
              "details": [
                {
                  "value": 2.2,
                  "description": "boost"
                },
                {
                  "value": 0.5753642,
                  "description": "idf",
                  "details": [
                    {
                      "value": 4,
                      "description": "n"
                    },
                    {
                      "value": 7,
                      "description": "N"
                    }
                  ]
                },
                {
                  "value": 0.56179774,
                  "description": "tf",
                  "details": [
                    {
                      "value": 2.0,
                      "description": "freq"
                    },
                    {
                      "value": 1.2,
                      "description": "k1"
                    },
                    {
                      "value": 0.75,
                      "description": "b"
```

```
                    },
                    {
                      "value": 5.0,
                      "description": "dl"
                    },
                    {
                      "value": 3.5714285,
                      "description": "avgdl"
                    }
                  ]
                }
              ]
            }
          ]
        }
      }
    ]
  }
}
```