Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

# Diplomová práce

# Pokročilé vyhledávání v datech ze zpravodajských portálů

Plzeň 2017 Pavel Přibáň

# Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 14. května 2017

Bc. Pavel Přibáň

# Poděkování

Rád bych poděkoval doc. Ing. Josefu Steinbergerovi, Ph.D. za vstřícnost, trpělivost, odborné rady a cenné připomínky, které mi pomohly tuto diplomovou práci vypracovat.

# Abstract

The purpose of this master thesis is to create simple and advanced searching for MediaGist system in data from news portals. MediaGist is an online system for cross-lingual analysis of aggregated news and commentaries based on summarization and sentiment analysis technologies. In the first part of this work the basic principles of information retrieval are described. The second part deals with the comparison of Elasticsearch and Apache Solr, that allows text searching. Next is described design and implementation of the searching using the Elasticsearch tool. The last part contains testing and evaluation of the created searching.

# Abstrakt

Cílem této diplomové práce je realizovat jednoduché a rozšířené vyhledávání pro systém MediaGist v datech ze zpravodajských portálů. MediaGist je on-line systém pro kroslinguální analýzu agregovaných zpráv a komentářů založený na technologii sumarizace a analýze sentimentu. V první části této práce jsou popsány základní principy vyhledávání informací. Druhá část práce se věnuje porovnání nástrojů Elasticsearch a Apache Solr umožňujících textové vyhledávání. Dále je popsán návrh a implementace vyhledávání za pomoci nástroje Elasticsearch. Poslední část práce zahrnuje testování a vyhodnocení vytvořeného vyhledávání.

# Obsah

**1 Úvod** **1**
1.1 Cíle práce . . . . . 1

**2 Vyhledávání informací** **2**
2.1 Vyhledávač . . . . . 3
2.2 Základní systém vyhledávání informací . . . . . 4
2.2.1 Sekvenční vyhledávání . . . . . 5
2.3 Indexace . . . . . 6
2.3.1 Incidenční matice . . . . . 6
2.4 Invertovaný index . . . . . 8
2.4.1 Vytvoření invertovaného indexu . . . . . 8
2.5 Předzpracování dokumentů . . . . . 10
2.5.1 Tokenizace . . . . . 11
2.5.2 Stop slova . . . . . 12
2.5.3 Stemming a Lematizace . . . . . 12
2.6 Booleovský model vyhledávání informací . . . . . 14
2.7 Vektorový model vyhledávání informací . . . . . 14
2.7.1 Term frekvence . . . . . 16
2.7.2 Inverzní dokument frekvence . . . . . 16
2.7.3 Tf-idf váha . . . . . 16
2.8 Vyhodnocování IR systémů . . . . . 17
2.8.1 Kvalita IR systémů . . . . . 17
2.8.2 Neohodnocené vyhledávání . . . . . 17
2.8.3 Ohodnocené vyhledávání . . . . . 19

**3 Vyhledávací nástroje** **21**
3.1 Apache Lucene . . . . . 21
3.1.1 Dokument . . . . . 21
3.1.2 Předzpracování dokumentu . . . . . 21
3.1.3 Index v Lucene . . . . . 22
3.1.4 Vyhledávání v Lucene . . . . . 23
3.2 Apache Solr . . . . . 23
3.2.1 Dokument v Apache Solr . . . . . 23

3.2.2 Indexace v Apache Solr . . . 24
3.2.3 Vyhledávání v Apache Solr . . . 24
3.2.4 Škálování v Apache Solr . . . 25
3.3 Elasticsearch . . . 26
3.3.1 Dokument v Elasticsearch . . . 26
3.3.2 Cluster . . . 27
3.3.3 Mapování . . . 28
3.3.4 Analyzátor . . . 28
3.3.5 Vyhledávání v Elasticsearch . . . 29
3.4 Porovnání Elasticsearch a Apache Solr . . . 30
3.4.1 Závěr porovnání Elasticsearch a Apache Solr . . . 31

**4 Návrh vyhledávání 32**
4.1 Systém MediaGist . . . 32
4.1.1 Data ze zpravodajských portálů . . . 32
4.2 Požadovaná funkcionalita . . . 33
4.2.1 Správa vyhledávání . . . 34
4.3 Architektura systému . . . 35
4.4 Návrh knihovny . . . 36
4.4.1 Návrh indexu . . . 36
4.4.2 Návrh vyhledávání . . . 39
4.5 Návrh webové aplikace . . . 40
4.5.1 Uživatelské rozhraní . . . 40
4.6 Návrh klienta pro indexaci . . . 40

**5 Realizace vyhledávání 43**
5.1 Načtení dat . . . 43
5.2 Vytvoření indexů . . . 43
5.2.1 Mapování typu indexu . . . 45
5.2.2 Nastavení indexů . . . 46
5.3 Indexace dokumentů . . . 47
5.3.1 Připojení k serveru Elasticsearch . . . 47
5.4 Vyhledávání dokumentů . . . 49
5.4.1 Rozpoznání jazyka . . . 51
5.4.2 Jednoduché vyhledávání . . . 51
5.4.3 Rozšířené vyhledávání . . . 51
5.5 Implementace webové aplikace . . . 52
5.5.1 Našeptávání výsledků . . . 53
5.6 Implementace klienta . . . 54

**6 Testování 55**
6.1 Prostředí pro testování . . . 55
6.2 Automatizované testování . . . 56

6.2.1 Data určená k testování . . . . . 56
6.2.2 Průběh automatizovaného testování . . . . . 57
6.2.3 Výsledky automatizovaného testování . . . . . 58
6.2.4 Zhodnocení získaných výsledků . . . . . 66
6.2.5 Návrh nového nastavení indexů . . . . . 68
6.2.6 Porovnání s CLEF AdHoc úlohami . . . . . 69
6.2.7 Závěr pro automatizované testování . . . . . 71
6.3 Uživatelské testování . . . . . 71
6.3.1 Výsledky uživatelského testování . . . . . 72
6.3.2 Možná rozšíření . . . . . 73

**7 Závěr** **74**

**Seznam použitých zkratek a výrazů** **76**

**Literatura** **78**

**Seznam příloh** **81**

**A Uživatelská příručka** **82**
A.1 Elasticsearch . . . . . 82
A.1.1 Požadavky pro spuštění Elasticsearch . . . . . 82
A.1.2 Spuštění Elasticsearch . . . . . 82
A.1.3 Spuštění Elasticsearch s upravenou konfigurací . . . . . 83
A.2 Klient . . . . . 84
A.2.1 Požadavky pro spuštění klienta . . . . . 84
A.2.2 Spuštění klienta . . . . . 84
A.2.3 Parametry klienta . . . . . 85
A.3 Webová aplikace . . . . . 86
A.3.1 Požadavky pro spuštění webové aplikace . . . . . 86
A.3.2 Spuštění webové aplikace . . . . . 86
A.3.3 Ovládání aplikace . . . . . 87

**B Diagramy, grafy a výsledky testování** **91**

**C Nastavení indexů** **107**

**D Obsah DVD** **117**

# 1 Úvod

Každý den je na internetu v elektronické podobě vygenerováno obrovské množství dat a vyhledávání informací (zpráv, počasí atd.) v těchto datech se v dnešní době stalo pro většinu populace naprosto rutinní a přirozenou záležitostí. Většina uživatelů si už ale neuvědomuje, jak složitým problémem je vyhledávání v tak velkém množství dat, a že za vykonáním jednoho dotazu se mohou skrývat složité algoritmy, tisíce serverů a mnoho let výzkumu. To, že vyhledávání je velmi důležité dokazuje fakt, že např. firma Google se díky svému internetovému vyhledávači stala jednou z největších a nejdůležitějších firem v oblasti IT.

Zjednodušeně lze říci, že systém MediaGist pravidelně seskupuje nové články (zprávy) v několika jazycích ze zpravodajských portálů, do tzv. clusterů (skupin). Cluster zde označuje skupinu článků, které pojednávají o stejné události nebo se jejich obsah vztahuje k velmi podobnému tématu. V článcích clusteru jsou rozpoznány pojmenované entity (osoby, organizace, státy apod.), nad kterými je provedena analýza sentimentu, tzn. jak se o rozpoznaných entitách psalo v článcích a jaký názor na ně uživatelé podle komentářů měli (kladný, neutrální nebo negativní). Z každého clusteru následně systém MediaGist vytvoří souhrn jeho obsahu a komentářů článků. Vytvořené souhrny jsou dostupné na webových stránkách systému MediaGist, kde je možné si je prohlížet, ale není možné v nich vyhledávat.

## 1.1 Cíle práce

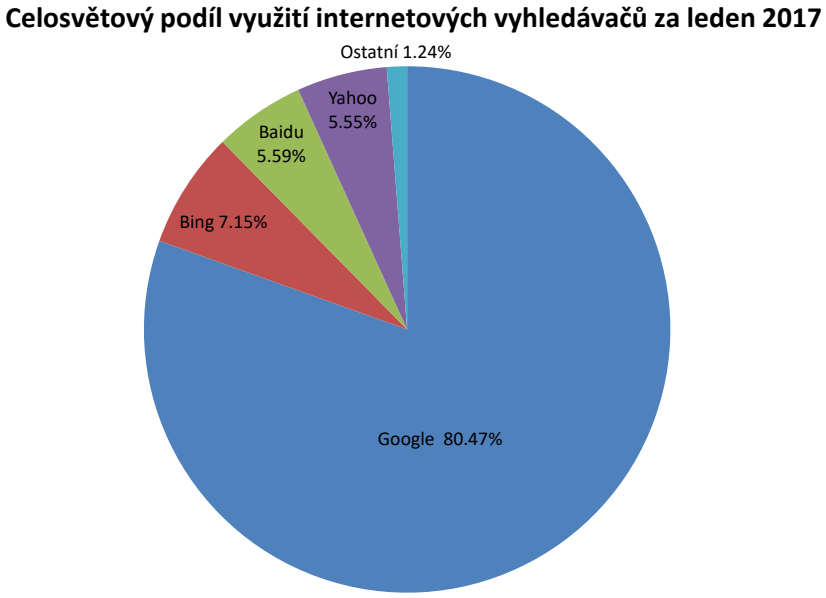
Jedním z cílů této diplomové práce je prozkoumat a analyzovat vybrané nástroje (technologie), které umožňují vyhledávání v textových datech a na základě jejich porovnání vybrat vhodný nástroj pro realizaci vyhledávání v datech systému MediaGist. Důležité je také zmínit, že textová data, pro která bude vyhledávání vytvořeno jsou v pěti jazycích. Návrh a implementace vyhledávání v datech systému MediaGist je hlavním cílem diplomové práce. Protože integrace vyhledávání se systémem MediaGist nepatří mezi cíle této práce, bude vytvořena jednoduchá webová aplikace demonstrující funkcionalitu implementovaného vyhledávání. Posledním cílem práce je otestovat implementované vyhledávání a zhodnotit výsledky testování.

Po přečtení této práce by měl čtenář pochopit jak bylo vytvořené vyhledávání navrženo a jakým způsobem byla realizována jeho implementace a testování. Dále by měl získat základní informace o elementárních principech, na kterých je vyhledávání informací obecně založeno.

# 2 Vyhledávání informací

Tato kapitola se bude zabývat pojmem `Vyhledávání informací`, v anglické literatuře je tento pojem známý jako `Information retrieval (IR)`.

Vyhledávání informací je podle [1] definováno jako hledání materiálů (obvykle dokumentů) v nestrukturované podobě (nejčastěji text), které odpovídají hledané informaci a jsou obsaženy ve velké kolekci dat (zpravidla v elektronické podobě), někdy také nazývaná jako `korpus`. Příkladem `systému vyhledávání informací` (dále v textu také `IR systém`) je asi nejznámější a celosvětově nejpoužívanější (podle [19]) internetový vyhledávač `Google`, viz obr. 2.1, nebo v České republice velmi známý `Seznam.cz`. Dalším příkladem IR systému může být vyhledávání souborů na pevném disku počítače [2].




Obrázek 2.1: Celosvětový podíl využití internetových vyhledávačů za leden 2017 (data převzata z [19])

Nestrukturovanými daty jsou myšlena ta data, která nemají sémanticky jasnou a pro počítač snadno zpracovatelnou strukturu. Opakem nestrukturovaných dat může být například relační databáze, kde je struktura jasně definována. Běžně ale téměř všechna data obsahují alespoň částečnou strukturu. Text obvykle obsahuje nadpisy a odstavce, které jsou v textu explicitně označeny např. podtržením nadpisu a odřádkováním na konci odstavce a v `HTML` stránkách se text uzavírá mezi značky, které vyjadřují sémantiku (význam).

Dále v práci budou za hledané materiály považovány dokumenty v elektronické formě obsahující text. Příkladem dokumentu může být webová stránka, článek na zpravodajském portále nebo textový soubor obsahující úryvek z knihy.

## 2.1 Vyhledávač

Podle [3] je vyhledávač (search engine) praktická aplikace technik vyhledávání informací na velké textové kolekce. Webový vyhledávač je toho ukázkovým příkladem. Vyhledávače existují v mnoha různých aplikacích od jednoduchých vyhledávačů v mobilních aplikacích až po velké podnikové systémy.

Vyhledávače je možné najít v mnoha konfiguracích, které odrážejí aplikace, pro které byly navrženy. Webové vyhledávače jako `Google` nebo `Yahoo!` musí být schopny získávat a zpracovávat mnoho terabytů dat a pak poskytnout odpovědi na dotazy s odezvou v řádech stovek milisekund až jedné sekundy. Další vyhledávače mohou být implementovány jako vlastnost (feature) operačního systému počítače a musí být schopny rychle zahrnout nové dokumenty, webové stránky a emaily, které uživatel počítače vytvořil nebo zobrazil a také poskytnout intuitivní rozhraní pro hledání těchto velmi různorodých informací [3].

Od IR systému se obvykle očekává, že bude splňovat tyto dva hlavní cíle [3]:

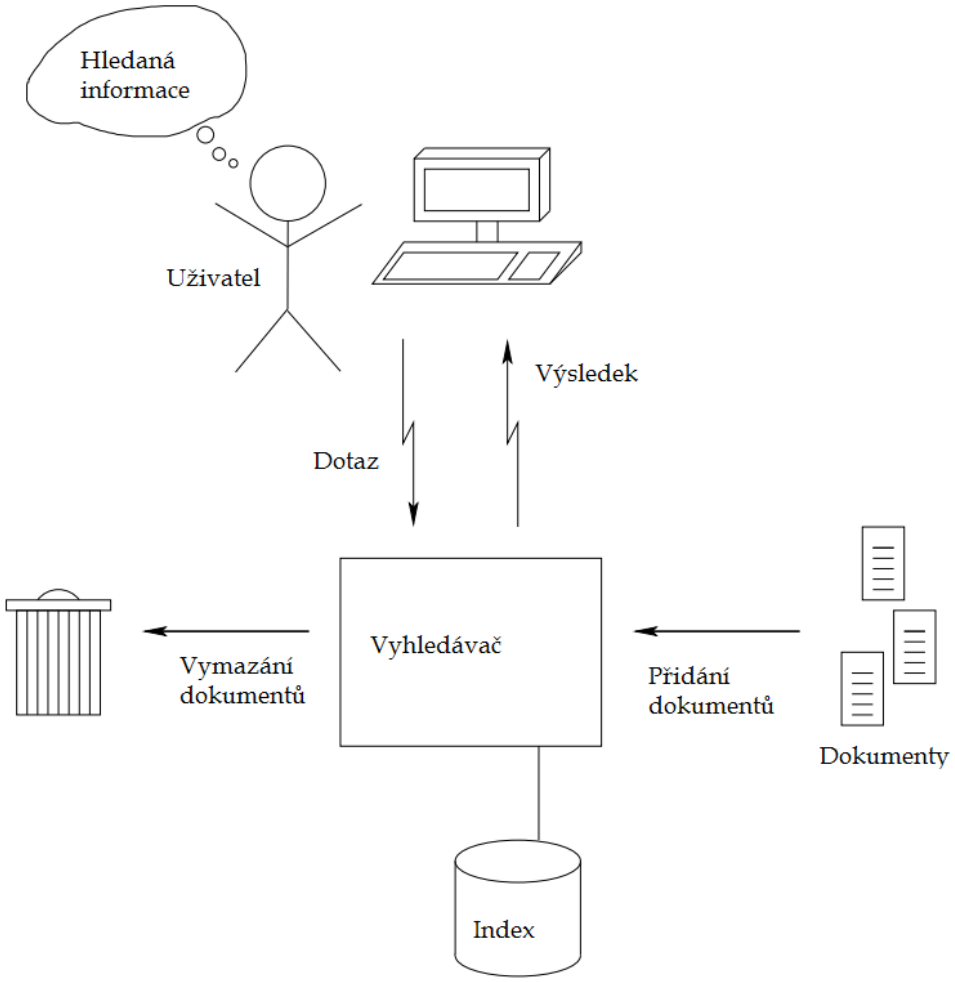
- Kvalita nebo také účinnost (v angl. lit. `effectiveness`) vyhledávání: Je požadováno, aby systém byl schopen vyhledat nejrelevantnější množinu dokumentů pro odpovídající dotaz.
- Výkonnost[1] (v angl. lit. `efficiency`) systému: Je požadováno, aby dotazy od uživatele byly zpracovány a výsledky vráceny tak rychle, jak je to jen možné.

Relevantním dokumentem je myšlen dokument obsahující informace, které uživatele zajímají a snaží se je získat (najít) [1]. Samozřejmě mohou být od systému vyhledávání očekávány další specifické požadavky, ale většinou jde především o kvalitu nebo o rychlost (popř. oboje). Mnoho datových struktur a algoritmů v IR systému je proto uzpůsobeno, aby vyhovovaly těmto dvěma základním požadavkům.

[1] Anglické pojmy `effectiveness` a `efficiency` je možné oba přeložit do češtiny jako účinnost, ale jejich význam je odlišný.

## 2.2 Základní systém vyhledávání informací

Většina IR systémů sdílí stejnou základní architekturu, která je dále upravena pro potřeby specifické aplikace [2]. Na obrázku 2.2 lze vidět hlavní komponenty v IR systému. Tyto komponenty mají především poskytovat dvě hlavní funkce, tj. `indexaci dokumentů` a `vyhodnocení dotazů` [3].




Obrázek 2.2: Základní komponenty IR systému (obrázek převzat a upraven z [2])

Před provedením vyhledávání má uživatel představu o informaci, kterou hledá. Někdy je označovaná jako `téma` (`topic`). Tato informace tvoří základ procesu hledání a řídí jej. Na základě této informace vytvoří uživatel `dotaz` (`query`). Typicky se dotaz

skládá z malého množství tzv. termů[2]. Pojem term se používá místo pojmu slovo, protože v dotazu se obecně nemusí vyskytovat pouze slova. Dotaz může obsahovat čísla, fráze nebo datum. IR systémy většinou podporují bohatší syntaxi dotazu jako jsou booleovské operátory (viz část 2.6) nebo žolíkové znaky.

Dotaz uživatele je následně zpracován vyhledávačem (anglicky search engine), který může být umístěn na lokálním počítači uživatele nebo i na velkých geograficky vzdálených výpočetních clusterech. Hlavní úlohou vyhledávače je spravovat index a manipulovat s ním. Index je vytvořen z kolekce dokumentů a je využíván vyhledávačem pro hledání a ohodnocení výsledků. Obvykle je indexem myšlen tzv. invertovaný index, viz část 2.4, na kterém jsou založeny všechny moderní vyhledávače [3].

Pro podporu řazení získaných výsledků si vyhledávač udržuje kolekci statistik spojenou s indexem. Uchovává se např. počet dokumentů obsahujících daný term a délku každého dokumentu. Kromě toho má vyhledávač obvykle přístup k originálnímu (původnímu) obsahu indexovaných dokumentů, aby mohl vracet smysluplné výsledky zpět uživateli. Za použití indexu, kolekce statistik a dalších dat je dotaz uživatele vyhodnocen a vrácen seřazený seznam výsledků (results). Aby mohly být výsledky seřazeny, je pro každý odpovídající dokument (výsledek) vypočteno skóre, podle kterého je řazení provedeno. Vráceným výsledkem může být celý dokument. Pokud by výsledek byl příliš velký, může být vrácena pouze jeho část, např. odstavec v dokumentu nebo jedna stránka knihy.

Ve většině IR systémů mohou být úpravy relativně jednoduché, tzn. dokumenty mohou být pouze přidány nebo smazány. Jakmile je dokument jednou přidán do IR systému, jeho obsah nemůže být modifikován. Pro většinu IR aplikací je tento přístup dostačující. Pokud např. dojde ke změně nějaké internetové stránky a vyhledávač zjistí, že se stránka změnila, tak aktualizace této stránky v indexu může probíhat jako smazání staré verze a přidání nové.

### 2.2.1 Sekvenční vyhledávání

Jedním z možných řešení vyhledávání informací by mohlo být prohledávání celé kolekce dat při každém dotazu a porovnání hledaného výrazu s daty v kolekci. Příkladem takovéhoto přístupu je Unixový příkaz grep. Pro malé kolekce dat, např. všechny divadelní hry Williama Shakespeara (cca 1 milion slov), je tento přístup při dnešním výkonu počítačů dostačující, viz [1]. Takovéto řešení bude jednoduché

[2]Pro některé anglické pojmy neexistuje nebo se nepoužívá český překlad, protože většina dostupné literatury je v angličtině. Pokud to bude možné budou v textu v prvé řadě používány české překlady anglických pojmů a až v případě, že je nebude možné přeložit nebo nebude překlad vhodný, bude použit originální anglický pojem.

implementovat a pro některé případy bude naprosto dostačující, ale zcela jistě se najdou případy, pro které bude tento přístup v lepším případě neefektivní a v horším případě nebude fungovat vůbec. Dále jsou vyjmenovány některé z těchto případů:

- Zpracování velkých kolekcí dokumentů. Ve velkých[3] kolekcích dokumentů nebude možné rychle vyhledávat, protože bude potřeba projít velké množství dat a doba zpracování dotazu bude tedy příliš dlouhá.
- Flexibilnější vyhledávací operace. Například bude nepraktické provádět dotaz pro vyhledání slova „Automobil“ v blízkosti slova „Vlak“, kde „v blízkosti“ může být definováno jako výskyt slova do vzdálenosti pěti slov nebo jako výskyt ve stejné větě.
- Ohodnocené vyhledávání. Při vyhledání dotazu může být vráceno více výsledků (odpovídajících dokumentů) a je požadováno seřadit výsledky od nejrelevantnějších (tzn. výsledky, které nejlépe odpovídají dotazu).

Řešení, které dovoluje vyhnout se opakovanému procházení celé kolekce dokumentů při vyhledávání, se jmenuje `indexace`.

## 2.3 Indexace

Indexace je proces vytváření datové struktury, která se nazývá `index`. Během indexace jsou dokumenty, ve kterých bude probíhat vyhledávání, nejprve předzpracovány (podrobněji viz část 2.5) a v upravené formě uloženy do indexu. Nad vytvořeným indexem poté může vyhledávač provádět rychlé a efektivní operace hledání. Index se nevytváří před každým dotazem, ale zpravidla pouze jednou[4] při spuštění celého IR systému.

### 2.3.1 Incidenční matice

Datovou strukturou, která bude reprezentovat index může být např. term-dokument incidenční matice, kterou lze vidět na obr. 2.3. Řádky představují jednotlivé termy, sloupce určují dokumenty. Prvek matice `(t,d)` je 1, pokud se v dokumentu ze sloupce

[3] Určit, kdy je kolekce velká a doba zpracování příliš dlouhá není obecně jednoduché a velmi záleží na konkrétním případu, tzn. na typu úlohy, použitém hardwaru apod. Někdy může být doba odezvy 10 sekund akceptovatelná a naopak u jiné aplikace může být doba odezvy 0,5 sekundy příliš dlouhá.

[4] Index může být také uložen na disku a při spuštění IR systému se pouze načte.

`d` vyskytl term z řádku `t`, jinak prvek obsahuje 0. Například z obr. 2.3 lze vidět, že term „Calpurnia“ se vyskytl pouze v dokumentu „Julius Caesar“. Při indexaci jsou pak postupně načítány všechny dokumenty z kolekce. Dokument je zpracováván po jednotlivých termech (slovech) a vždy, když se v dokumentu vyskytne daný term, je na odpovídající pozici v matici nastavena 1. Výsledná matice má rozměr $|V| \times |D|$, kde $|V|$ je velikost slovníku termů (počet řádků) a $|D|$ je počet dokumentů (počet sloupců).

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|---|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 | |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 | |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 | |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 | |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 | |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 | |
| worser | 1 | 0 | 1 | 1 | 1 | 0 | |
| ... | | | | | | | |

Obrázek 2.3: Příklad term-dokument incidenční matice (obrázek převzat z [1])

Nyní můžeme na každý term (řádku) pohlížet jako na vektor, který indikuje v jakých dokumentech se term objevil.

Dále bude uveden příklad z [1], na kterém bude demonstrováno proč je použití incidenční matice nevhodné. Lze předpokládat, že kolekce, nad kterou bude prováděno vyhledávání obsahuje 1 000 000 dokumentů. Každý dokument obsahuje cca 1000 slov a každé slovo zabere v paměti průměrně 6 B včetně mezer a interpunkce. Pak bude mít kolekce dokumentů přibližně 6 GB. Slovník termů vytvořený z kolekce bude obsahovat přibližně 500 000 záznamů. Slovník je zde (v kontextu IR) považován za datovou strukturu, která uchovává seznam všech unikátních termů, tzn. ve slovníku jsou uloženy všechny termy, které se v kolekci vyskytly, ale vždy právě jednou (bez duplicit). V tomto případě není možné vytvořit term-dokument incidenční matici, protože matice jedniček a nul o rozměrech 500K $\times$ 1M (500 miliard prvků) by se nevešla do paměti běžného počítače.

Důležitým poznatkem je, že matice bude extrémně řídká. Pouze malý počet prvků matice bude obsahovat nenulové hodnoty, protože každý dokument má přibližně 1000 slov, v celé matici nebude více jak miliarda jedniček. To znamená, že 99.8 % prvků v matici budou nuly. Mnohem lepším řešením je zaznamenat pouze informaci o tom, že se daný term vyskytl (jedničky v matici) a nikoliv zároveň i to, že se v dokumentu term nevyskytl. Toto řešení využívá `invertovaný index` viz část 2.4.

Smyslem předchozího příkladu bylo ukázat, že výběr datové struktury je u tohoto typu úlohy velmi důležitý.

## 2.4 Invertovaný index

Všechny indexy dnešních moderních vyhledávačů jsou založeny na tzv. `invertovaném indexu`, někdy také označovaný jako `invertovaný soubor`, a jsou považovány za nejefektivnější a nejflexibilnější datové struktury pro vytváření indexů [3]. Základní myšlenka je ukázána na obr. 2.4.

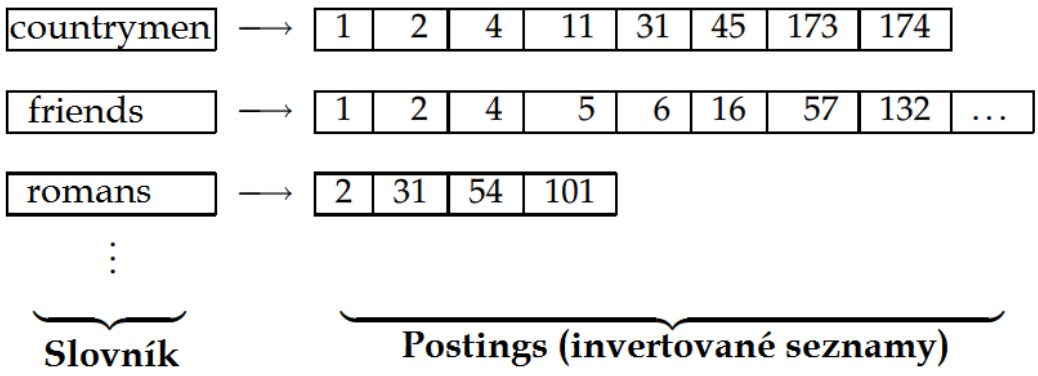
Index si udržuje lexikograficky seřazený slovník termů a pro každý term je dále udržován seznam se záznamy, ve kterých dokumentech se term vyskytl. Prvek seznamu se nazývá `posting` a často obsahuje dodatečné informace, např. pozice výskytů termů v původním dokumentu nebo počet jejich výskytů. Dodatečné informace pak mohou být využity pro výpočet ohodnocení (skóre) dokumentu při vyhodnocování dotazu. Tento seznam se pak označuje v angl. lit. jako `postings list` neboli invertovaný seznam a obvykle jsou záznamy v něm seřazené (např. podle číselného identifikátoru dokumentu, dále jen ID dokumentu). Všechny invertované seznamy dohromady tvoří `postings`. Index se nazývá invertovaným, protože obvykle jsou slova považována za část dokumentů, ale pokud převrátíme tuto myšlenku tak dokumenty jsou asociovány ke slovům (termům).

### 2.4.1 Vytvoření invertovaného indexu

Vytvoření invertovaného indexu lze rozdělit na tyto čtyři hlavní kroky [1]:

1. Získání dokumentů, které budou indexovány:
   | Friends, Romans, countrymen. | So let it be with Caesar. | ...
2. Tokenizace textu, která převede každý dokument na seznam tokenů:
   | Friends | Romans | countrymen | So | let | it | ...
3. Předzpracování (převedení na určitý tvar) jednotlivých tokenů, které jsou pak indexovány jako termy:
   | friends | romans | countrymen | so | let | it | ...
4. Vytvoření datové struktury invertovaného indexu skládajícího se ze slovníku termů a invertovaných seznamů (postings), viz obr. 2.4.

Procesy tokenizace a předzpracování textu jsou detailněji popsány v části 2.5, ale pro tento jednoduchý příklad, který byl převzat z [1], lze považovat tokenizaci za rozdělení vět na jednotlivá slova a předzpracování za převedení písmen ve slově na písmena malá. Ve skutečnosti je předzpracování textu před indexací velmi složitou a obsáhlou úlohou IR.



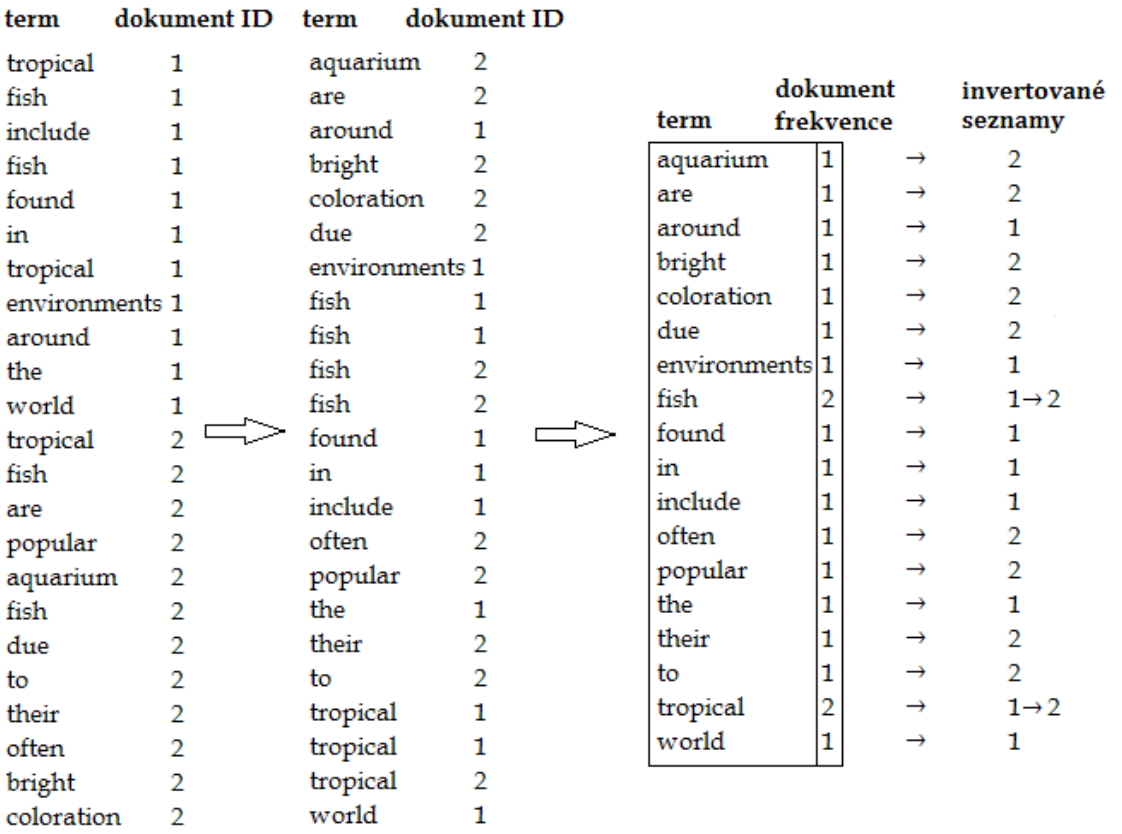

Obrázek 2.4: Dvě části invertovaného indexu. Slovník je obvykle udržován v paměti spolu s odkazy na každý invertovaný seznam, který je uložen na disku (obrázek převzat a upraven z [1])

V kolekci dokumentů se předpokládá, že každý dokument má přiřazený unikátní nejlépe číselný identifikátor (ID), který je přiřazen během vytváření indexu každému novému dokumentu. Vstupem pro indexaci je seznam normalizovaných tokenů z každého dokumentu, což je ekvivalent k seznamu párů term - ID, jak lze vidět na obr. 2.5, který představuje postup při indexaci těchto dvou dokumentů:

> Dokument 1: **Tropical fish include fish found in tropical environments around the world.**
>
> Dokument 2: **Tropical fish are popular aquarium fish, due to their often bright coloration.**

Důležitým krokem je abecední seřazení tohoto seznamu párů (prostřední sloupec obr. 2.5). Výskyty identických termů ve stejném dokumentu jsou sloučeny. Identické termy z různých dokumentech jsou sdruženy a je k nim vytvořen odpovídající invertovaný seznam (pravý sloupec na obr. 2.5). Slovník také uchovává počet dokumentů, ve kterých se daný term vyskytl - `dokument frekvence` (`document frequency`). Tato informace není nutná pro vyhledávač používající booleovský model, ale dovoluje zvýšit kvalitu výsledků u vyhledávačů vracejících ohodnocené výsledky (např. při použití `vektorového modelu`, viz část 2.7). Invertované seznamy jsou seřazeny podle ID dokumentů, čímž poskytují základ pro efektivní zpracování dotazů [1].

Nevýhodou indexace je, že potřebuje dodatečné místo pro uložení indexu. Obvykle je slovník udržován v paměti a invertované seznamy jsou uloženy na disku, protože společně jsou příliš velké a nevešly by se do paměti. Ideální řešením by samozřejmě bylo udržet obě části indexu v paměti, protože čtení z pevného disku je podle [21] přibližně 80krát pomalejší než čtení z paměti, a proto je velikost slov-

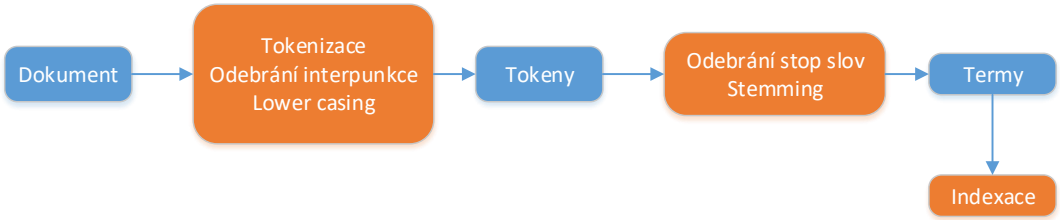



Obrázek 2.5: Znázornění postupu při indexaci dokumentů do invertovaného indexu

níku a invertovaných seznamů velmi důležitá. Místo, které datové struktury (index) zabírají, je možné redukovat a optimalizovat např. kompresí, podrobněji viz [1, 2, 3].

## 2.5 Předzpracování dokumentů

Cílem předzpracování je převést text dokumentů, který se může lišit např. v kódování nebo ve formátu uložení dat (PDF, DOCX, XML), do podoby, ve které jej bude možné jednotně indexovat. Dalšími důvody pro předzpracování je zefektivnění zpracování a vyhodnocení dotazů, a to jak z důvodu výkonu (rychlost, místo potřebné pro uložení indexu), tak i z důvodu zkvalitnění získaných výsledků (relevance vrácených dokumentů). Protože jednotlivé jazyky se od sebe velmi liší (slovní zásoba, skladba vět a gramatika), musí být některé kroky předzpracování upraveny speciálně pro daný jazyk [1]. V praxi to znamená, že např. při stemmingu (pojem stemming bude vysvětlen dále) je pro každý jazyk potřeba použít jiný algoritmus nebo jiná data (model) algoritmu, nad kterými je založen (např. [24]). To jak může vypadat proces předzpracování lze vidět na obr. 2.6.




Obrázek 2.6: Znázornění možného postupu předzpracování dokumentů před indexací

Důležitou poznámkou je, že při zadání dotazu do IR systému je dotaz předzpracován stejně jako dokumenty (viz obr. 2.6). Pokud by nebyl dotaz takto předzpracován, vyhledávání by téměř nefungovalo.

### 2.5.1 Tokenizace

`Tokenizace` je proces rozdělení sekvence znaků (textu dokumentu) na části, které se nazývají `tokeny`. V průběhu tokenizace nebo po jejím skončení jsou také obvykle odstraněny určité znaky, např. interpunkce. Zde je příklad tokenizace:

Vstup: Fish, sharks and turtles live in sea.

Výstup: | Fish | | sharks | | and | | turtles | | live | | in | | sea |

Tokeny jsou často označovány za termy nebo slova, ale v některých případech je důležité tyto pojmy od sebe odlišit, protože neznamenají totéž. `Token` je seskupená sekvence znaků v určitém dokumentu, která obsahuje sémantickou informaci. `Typ` (angl. `type`) je třída všech tokenů, které mají identickou sekvenci znaků. `Term` je odvozený z `typu` tak, že prošel procesem `normalizace` a dalších úprav (stemming, lematizace) a je zahrnut ve slovníku IR systému (je tedy součástí indexu).

Například pokud by dokument byl reprezentován větou „the first Man on the Moon“, pak z věty vznikne 6 tokenů, ale pouze 5 typů (slovo „the“ se vyskytlo dvakrát). Pokud budou ještě vynechána slova „the“ a „on“, protože jsou obsažena v množině stop slov, do indexu budou uloženy pouze tyto tři termy: „first, man, moon“.

Normalizace je proces odstranění zanedbatelných rozdílů mezi jinak stejnými slovy (tokeny), přičemž může zahrnovat převedení všech písmen na malá (angl. `lowercasing`), odstranění diakritiky a další úpravy [6]. Další úpravou může být převedení německého znaku `ß` (ostré s) na `ss`.

Velmi důležitým a obtížným úkolem je rozhodnout, které sekvence znaků určit jako tokeny a jak je získat. Předchozí příklad byl velmi jednoduchý, stačilo pouze rozdělit text podle mezer a odstranit interpunkci, ale v praxi by tento přístup nemusel vůbec fungovat a nevhodně provedená tokenizace může mít velmi negativní vliv na kvalitu získaných výsledků. Například v čínštině a japonštině mezera nemusí nutně znamenat oddělení jednotlivých slov [4]. Jedním z tokenizérů, který obsahuje podporu některých asijských jazyků je `ICU Tokenizer` [23]. Rozdělení tokenů podle mezer může také rozdělit výrazy, které by bylo dobré indexovat jako jeden term, např. názvy měst „Los Angles“ nebo „Ústí nad Labem“.

### 2.5.2 Stop slova

Přirozený jazyk obsahuje běžná slova, která se vyskytují velmi často a nesou pouze minimální sémantickou informaci, tzn. nemají téměř žádný význam. Taková slova se v IR nazývají `stop slova` (`stop words`). Každý jazyk má svá vlastní stop slova. Příkladem některých stop slov v českém jazyce jsou: „a“, „je“, „z“, „ten“ a v angličtině to mohou být slova: „the“, „a“, „an“ nebo „that“.

Seznam stop slov se vytvoří tak, že pro každé slovo v kolekci dokumentů (nebo v jazyku) je spočtena frekvence jeho výskytů. Dále se vybere $N$ (např. 50) slov s největší frekvencí, která jsou často ještě zkontrolována manuálně člověkem, zda se v seznamu neobjevila slova, která by byla významově důležitá vzhledem k množině dokumentů, které budou indexovány [1]. Během předzpracování jsou po tokenizaci tokeny obsažené v seznamu stop slov odebrány a nejsou dále zpracovávány.

Prvním důvodem pro odstranění stop slov je místo, které je potřebné pro jejich uložení v indexu. Odstraněním stop slov se redukuje počet položek v invertovaných seznamech, které musí být uloženy. Tento důvod je spíše historickým, protože dříve byly velikosti disků opravdu malé ve srovnání s dnešními a dnes již není nezbytně nutné zbavovat se stop slov pro ušetření místa. Druhým důvodem je to, že stop slova jsou velmi běžná v jakémkoliv dokumentu (textu) a jen zřídka nesou relevantní informaci o obsahu dokumentu a pokud budeme uvažovat pouze hledání jednotlivých slov (nikoliv frází), nejsou stop slova v indexu téměř užitečná a můžeme je odstranit. Odstraněním snížíme velikost indexu a získáme lepší kvalitu získaných výsledků při vyhledávání [1].

### 2.5.3 Stemming a Lematizace

V přirozeném jazyce je možné vyjádřit konkrétní věc (myšlenku) mnoha různými způsoby. To může způsobovat problémy vyhledávačům, kteří při získávání relevant-

ních dokumentů spoléhají na nalezení shodných slov dotazu se slovy v prohledávaných dokumentech. Aby vyhledávače při vyhodnocování dotazů nemusely spoléhat pouze na přesnou shodu slov (termů), využívá se několika technik, které dovolují nalézt slova sémanticky podobná. Takovými technikami jsou `Stemming` a `Lematizace` [3].

Cílem stemmingu a lematizace je zredukovat různé tvary slova, ve kterých se dané slovo může vyskytnout (at' už z důvodu skloňování nebo jiného odvozování) na společný základní tvar. Například:

jsem, je, jsi ⇒ být
car, cars, car's, cars' ⇒ car

Lematizace obvykle získává základní tvar slova použitím slovníku a morfologické analýzy slov. Výstup (základní tvar) lematizace se nazývá `lemma`. Stemming je heuristický postup, který se snaží odebírat přípony z konců slov a získat tak základní tvar slova (přiblížit se k lematu zpracovávaného slova) [1]. Rozdíl mezi lematizací a stemmingem lze ukázat na příkladu se slovem „jíst“ ve tvaru „jíme“. Jednoduchý stemmer může fungovat tak, že bude u slov končících příponou „-me“ tuto příponu odebírat. Výsledkem stemmingu slova „jíme“ bude „jí“, zatímco při použití lematizace bude vráceno „jíst“.

Asi nejznámějším algoritmem pro stemming angličtiny je `Porterův algoritmus` [22], který se zároveň ukázal (empiricky) jako velmi efektivní. Algoritmus se skládá z pěti kroků a každý krok obsahuje množinu pravidel pro odstranění přípon. Jednotlivé kroky jsou vykonávány sekvenčně, na slova (tokeny) je vybráno a aplikováno pravidlo na nejdelší možnou příponu. V tabulce 2.1 lze vidět ukázku pravidel prvního kroku.

| **Pravidlo** | **Příklad** |
|---|---|
| SSES → SS | stresses ⟶ stress |
| S → | gaps ⟶ gap |
| IES → I | ponies ⟶ poni |
| SS → SS | caress ⟶ caress |

Tabulka 2.1: Ukázka pravidel z první fáze `Porterova algoritmu` [22] spolu s příklady aplikace

Použití stemmeru pro vyhledávání nad anglickým textem přináší patrné zlepšení v kvalitě výsledků. V některých jazycích, ve kterých se slova často skloňují (např. ruština, čeština nebo němčina) je stemming kritickou součástí potřebnou pro získání kvalitních výsledků [3].

## 2.6 Booleovský model vyhledávání informací

`Booleovský model vyhledávání informací` (`Boolean retrieval model`) je model pro vyhledávání informací, kterému je možné zadat dotaz ve formě booleovského výrazu. Dotaz může být složen z termů a logických operátorů $AND$, $OR$ *a* $NOT$ a dokumenty jsou vráceny pouze pokud odpovídají přesně zadanému dotazu. Model pohlíží na každý dokument pouze jako na množinu slov. Dokumenty vrácené tímto modelem nejsou nijak seřazeny podle jejich relevance, protože model předpokládá, že všechny vrácené výsledky jsou stejně relevantní. Booleovský model byl používán prvními vyhledávači a v některých případech je používán dodnes [3, 1].

Například pro matici na obr. 2.3 a dotaz `Brutus` $AND$ `Caesar` je třeba vybrat vektory termů `Brutus` a `Caesar` a provést logickou operaci bitového součinu (AND), jak je ukázáno ve výrazu 2.1, výsledkem je vektor. Pokud je na pozici `d` ve výsledném vektoru nastavena jednička, tak dokument odpovídající sloupci `d` z incidenční matice vyhovuje zadanému dotazu. Ve výrazu 2.1 tedy dotazu odpovídají dokumenty `Antony and Cleopatra`, `Julius Caesar` a `Hamlet`. Tyto dokumenty by byly vráceny uživateli jako výsledek na dotaz `Brutus` $AND$ `Caesar`. Velkou výhodou bitových operací nad vektory složených z nul a jedniček (zde vzniklé z incidenční matice 2.3) je jejich rychlost. Zcela jistě bude mnohonásobně rychlejší provést bitový součin ($AND$) nad dvěma vektory čísel než místo toho porovnávat odpovídající řetězce slov.

$$110100 \text{ AND } 110111 = 110100 \tag{2.1}$$

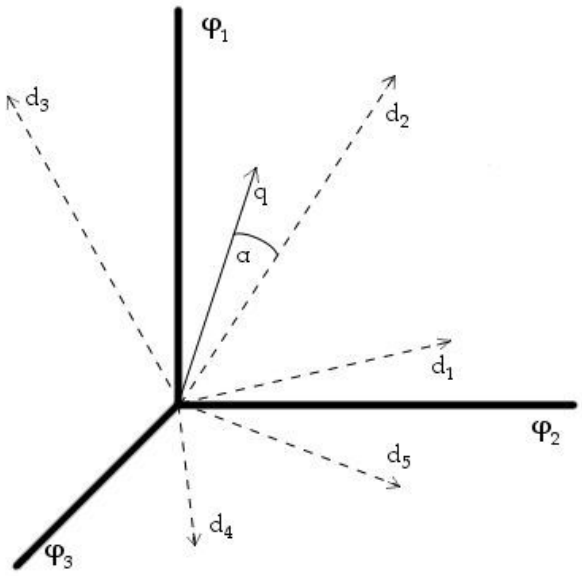
## 2.7 Vektorový model vyhledávání informací

Nevýhodou booleovského modelu při vyhledávání informací je, že vrácené výsledky není možné řadit dle relevance, protože dokument je k zadanému dotazu buď relevantní nebo nerelevantní. Pro velké kolekce dokumentů to v praxi může znamenat, že vrácených výsledků bude buď příliš mnoho nebo naopak nebudou vráceny žádné.

`Ohodnocené vyhledávání` (`ranked retrieval`) je možné implementovat `vektorovým modelem` (`vector space model`), který každému výsledku (dokumentu) dokáže přiřadit skóre relevance vzhledem k zadanému dotazu.

`Vektorový model` je založen na představě $m$-dimenzionálního vektorového prostoru (viz obr. 2.7). Každá dimenze tohoto prostoru představuje unikátní `příznak` $\varphi_i$ (`feature`) v kolekci dokumentů. Každý dokument a dotaz je v tomto modelu reprezentován vektorem o $m$ složkách. Dimenze prostoru je rovna počtu indexovaných

termů (velikosti slovníku indexu). Každý příznak odpovídá jednomu termu v indexu. V jednotlivých dokumentech je každému termu přiřazena váha (ta je také hodnotou příznaku) právě když se daný term v dokumentu vyskytl, jinak je váha příznaku nula.




Obrázek 2.7: Reprezentace dotazu $q$ a kolekce pěti dokumentů $d_i$ v prostoru se třemi dimenzemi (označené jako $\varphi_i$)

Model také předpokládá, že všechny příznaky (a tím pádem i termy) jsou na sobě vzájemně nezávislé. Dokument $D_i$ je vyjádřen jako vektor ve výrazu 2.2, kde $d_{ij}$ reprezentuje váhu $j$-tého termu v tomto dokumentu [4].

$$\vec{d} = (d_{i1}, d_{i2}, \ldots, d_{im}) \tag{2.2}$$

Výsledné skóre dokumentu (jak moc relevantní je dokument k dotazu) se poté určí jako vzdálenost (podobnost) vektoru dotazu a vektoru dokumentu. Místo euklidovské vzdálenosti pro určení podobnosti se nejčastěji používá kosinová podobnost (viz vzorec 2.3), jejímž výsledkem je kosinus úhlu mezi dvěma vektory. Nejrelevantnější dokumenty jsou takové, u kterých je úhel mezi vektorem dokumentu a vektorem dotazu nejmenší (na obr. 2.7 to je úhel $\alpha$) [3].

$$\cos(\vec{q}, \vec{d}) = \frac{\vec{q} \cdot \vec{d}}{||\vec{q}||\ ||\vec{d}||} = \frac{\sum_{i=1}^{m} q_i d_i}{\sqrt{\sum_{i=1}^{m} q_i^2} \sqrt{\sum_{i=1}^{m} d_i^2}} = \cos(\alpha) \tag{2.3}$$

### 2.7.1 Term frekvence

Nejjednodušší možností určení váhy je přiřadit termu $t$ váhu rovnu počtu výskytů v dokumentu $d$, tento typ se nazývá `term frekvence` (`term frequency`) a značí se $tf_{t,d}$. Dotaz obsahuje term $t$, získaný dokument v němž se daný term $t$ vyskytne desetkrát ($tf_{t,d} = 10$) je jistě relevantnější než dokument, ve kterém se term $t$ vyskytne pouze jednou ($tf_{t,d} = 1$), ale určitě není desetinásobně relevantnější. Z tohoto důvodu se používá logaritmovaná frekvence $wf_{t,d}$, která se vypočte podle vztahu 2.4 [4].

$$wf_{t,d} = \begin{cases} 1 + \log tf_{t,d} & \text{pro } tf_{t,d} > 0 \\ 0 & \text{jinak} \end{cases} \tag{2.4}$$

### 2.7.2 Inverzní dokument frekvence

Inverzní dokument frekvence $idf_t$ termu $t$ vyjadřuje důležitost termu v celé kolekci dokumentů. V čím více dokumentech se term vyskytl, tím méně bude jednotlivé dokumenty (ve kterých se vyskytl) od sebe odlišovat a následně pak bude méně užitečný při vyhledávání. Proto, pokud se term vyskytuje v kolekci dokumentů často, je potřeba snížit jeho váhu (je méně důležitý) a opačně méně častým (jsou více důležité) termům váhu zvýšit. Obvykle se pro tento účel používá vzorec 2.5, kde $N$ je počet dokumentů v kolekci, a $df_t$ je počet dokumentů v kolekci, které obsahují term $t$ [4].

$$idf_t = \log \frac{N}{df_t} \tag{2.5}$$

### 2.7.3 Tf-idf váha

Vynásobením předchozích vah, tj. `term frekvence` a `inverzní dokument frekvence`, vznikne `tf-idf` váha (viz vztah 2.6, ve kterém je značena jako $w_{t,d}$), která je nejznámějším typem váhy v IR [3].

$$w_{t,d} = wf_{t,d} \times idf_t \tag{2.6}$$

## 2.8 Vyhodnocování IR systémů

`Vyhodnocování` (`evaluation`) IR systémů se zaměřuje na dvě vlastnosti, které jsou od IR systémů primárně vyžadovány, tj. kvalita vyhledávání (effectiveness) a výkonnost systému (efficiency). Základními ukazateli pro vyhodnocení kvality vyhledávání v IR systémech je `přesnost` (`precision`) a `úplnost` (`recall`). Tato kapitola se zaměří na vyhodnocení kvality (relevance) vyhledávání. Je snahou IR systémy neustále zdokonalovat a vyvíjet. Jednotlivá vyhodnocení (testy) musí být opakovatelná, aby bylo možné zjistit, zda po úpravě IR systému došlo ke zlepšení nebo zhoršení.

### 2.8.1 Kvalita IR systémů

Při posuzování kvality v IR systémech se vychází ze dvou základních ukazatelů, tj. `přesnost` (viz vzorec 2.7) a `úplnost` (viz vzorec 2.8). Přesnost je poměr počtu vrácených relevantních dokumentů ku celkovému počtu vrácených dokumentů a úplnost je poměr počtu vrácených relevantních dokumentů ku celkovému počtu relevantních dokumentů [1].

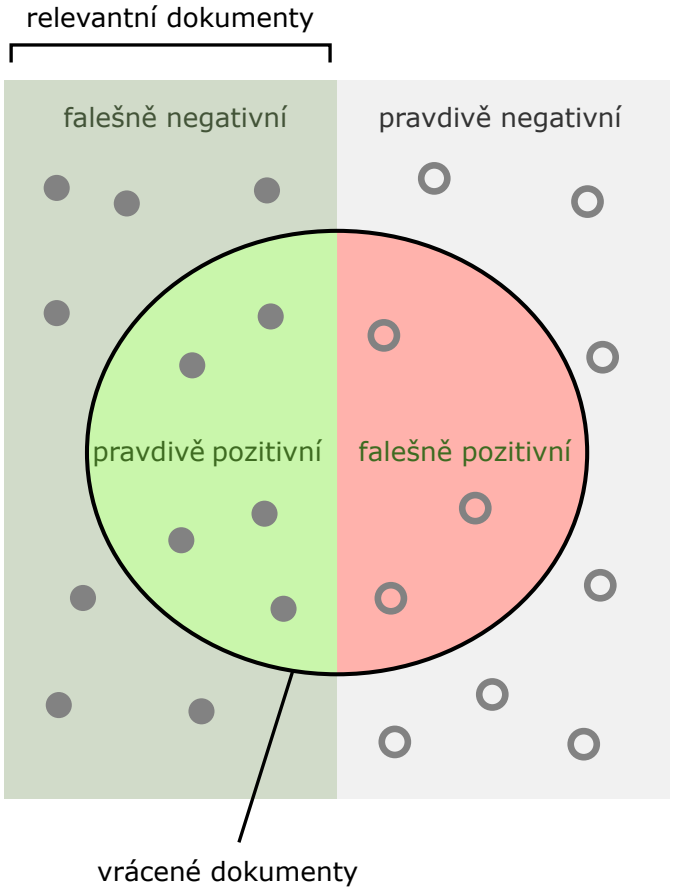
$$P = \frac{tp}{tp + fp} \tag{2.7}$$

$$R = \frac{tp}{tp + fn} \tag{2.8}$$

Na obr. 2.8 jsou vyobrazeny jednotlivé množiny pro označení dokumentů. Falešně negativní (false negative – fn) dokumenty jsou takové, které nebyly vráceny, ale jsou relevantní. Pravdivě negativní (true negative – tn) jsou nevrácené nerelevantní dokumenty. Pravdivě pozitivní (true positive – tp) jsou relevantní vrácené a falešně pozitivní (false positive – fp) jsou vrácené, ale nerelevantní. Z přesnosti a úplnosti vychází složitější ukazatele a metriky, které jsou popsány dále.

### 2.8.2 Neohodnocené vyhledávání

Vyhodnocení obvykle probíhá tak, že je vytvořena kolekce testů skládající se z dokumentů a dotazů s označenými relevantními dokumenty. Dokumenty jsou zaindexovány a poté jsou pro každý případ z kolekce testů provedeny dotazy nad vyhledávačem. Následně je z vrácených výsledků (relevantní dokumenty podle vyhledávače) vypočtena přesnost a úplnost.




Obrázek 2.8: Znázornění množin pro označení dokumentů při vyhodnocování vyhledávání (obrázek převzat z [27])

Ideální případ by nastal pokud by obě míry dosahovaly hodnoty jedna (100 %), ale problémem zůstává, že pokud se podaří zlepšit jednu z nich, druhá se zpravidla zhorší. Alternativou je `F-míra` $F_\beta$ (nejčastěji s parametrem $\beta = 1$), která je založena na přesnosti a úplnosti a je definována jako jejich `harmonický` průměr, viz vzorec 2.9, kde $P$ je přesnost a $R$ úplnost. Vzorec 2.10 je pouze vyjádřením vzorce 2.9 s $\beta = 1$.

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \tag{2.9}$$

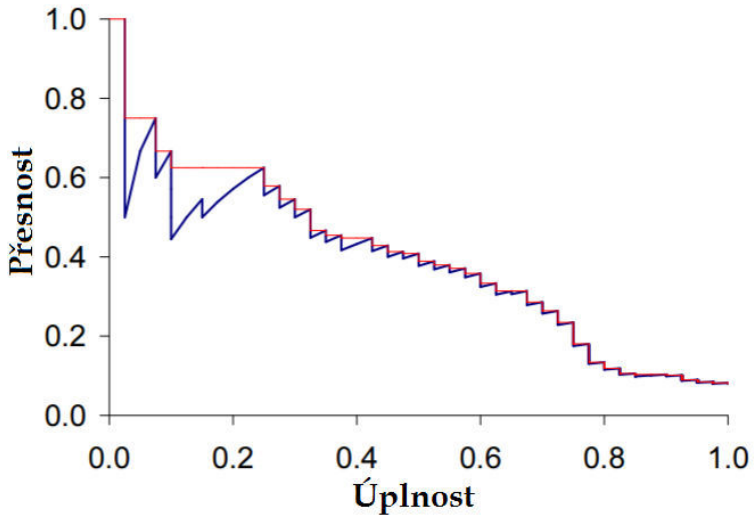
$$F_1 = \frac{2PR}{(P + R)} \tag{2.10}$$

Nevýhodou F-míry je, že nebere v potaz ohodnocení (relevanci) získaných výsledků a všechny výsledky mají stejnou váhu. Tento typ hodnocení je možné použít

například pro booleovský model, ale většina dnešních vyhledávačů vrací výsledky seřazené a hodnotit výsledky podle F-míry není vhodné.

### 2.8.3 Ohodnocené vyhledávání

Aby bylo možné použít přesnost a úplnost je potřeba modifikovat jejich použití a to tak, že obě míry jsou postupně spočteny pro prvních $k$ výsledků ($k = 1, 2, 3, 4, ...$). Nejdříve se tedy vypočte přesnost a úplnost pro první výsledek, dále pro první dva výsledky a tak dále. Takovou množinu dat je možné vykreslit jako graf (viz modrá křivka na obr. 2.9), kde osa y představuje přesnost a osa x úplnost. Tento graf se typicky nazývá `přesnost/úplnost graf` (`precision/recall graph`) [1]. Každý bod je určen dvojicí přesnost/úplnost pro určitý počet $k$ nejrelevantnějších výsledků.




Obrázek 2.9: Modrá barva reprezentuje přesnost/úplnost graf. Červená barva představuje interpolovanou přesnost $p_{int}$ (obrázek převzat a upraven z [1])

Často je užitečné odstranit z grafu „skoky“, které vznikají, když pro některé body (určené dvojicí přesnost/úplnost spočtenou z $k$ nejrelevantnějších výsledků) vyjde stejná úplnost. Odstranění se typicky provádí interpolací přesnosti. Interpolovaná přesnost $p_{int}$ pro určitou hodnotu úplnosti $r$ je definována vzorcem 2.11 jako nejvyšší přesnost z jakékoliv hodnoty úplnosti $r' \geq r$. Díky této definici je možné určit interpolovanou přesnost i pro úplnost s hodnotou nula. Na obr. 2.9 je graf interpolované přesnosti vykreslen červenou barvou.

$$p_{int}(r) = \max_{r' \geq r} p(r') \tag{2.11}$$

`MAP` (`Mean Average Precision`) je definován podle vzorce 2.13 jako průměrná hodnota průměrných přesností $AP$ pro každý dotaz $q_i$ v množině dotazů $Q$. Průměrná přesnost $AP_i$ dotazu $q_i$ je vypočtena podle vzorce 2.12 jako průměr přesností na pozicích (ohodnocení), na kterých byl získán relevantní dokument (tzn. když se zvětšila hodnota úplnosti viz obr. 2.9), podrobněji viz [3]. Ve výrazu 2.12 označuje $m_i$ velikost množiny relevantních dokumentů $\{d_1, ...d_{m_i}\}$ pro dotaz $q_i$ a $Prec(R_{ij})$ je přesnost, která byla vypočtena, právě když se dokument $d_j$ dostal mezi $k$ nejrelevantnějších výsledků.

$$AP_i = \frac{1}{m_i} \sum_{j=1}^{m_i} Prec(R_{ij}) \tag{2.12}$$

$$MAP(Q) = \frac{1}{|Q|} \sum_{i=1}^{|Q|} AP_i \tag{2.13}$$

MAP míra se stala jednou z nejpoužívanějších v odborných článcích [3]. Alternativou k MAP je `GMAP` míra, která místo aritmetického průměru používá geometrický průměr, viz vztah 2.14.

$$GMAP(Q) = \exp\left(\frac{1}{|Q|} \sum_{i=1}^{|Q|} \log AP_i\right) \tag{2.14}$$

# 3 Vyhledávací nástroje

V této kapitole budou krátce popsány a na závěr porovnány dva vyhledávací nástroje (technologie) `Elasticsearch` a `Apache Solr`. Oba nástroje jsou založeny na knihovně pro textové vyhledávání `Apache Lucene`. Elasticsearch i Apache Solr jsou vyvíjeny jako `open-source` (otevřený zdrojový kód) software, tzn. zdrojový kód je volně přístupný např. na internetových stránkách `www.github.com` [1]. Na základě porovnání nástrojů Elasticsearch a Apache Solr bude jeden z nich vybrán pro realizaci vyhledávání v praktické části této diplomové práce.

## 3.1 Apache Lucene

Apache Lucene (dále jen Lucene) je dnes pravděpodobně nejpokročilejší, velmi výkonná, škálovatelná a plně vybavená open-source knihovna pro vyhledávání, která je implementována v programovacím jazyce `JAVA` [6]. Lucene poskytuje nástroje (API) pro vyhledávání a je pouze knihovnou, nikoliv hotovým vyhledávačem. Následuje popis základních postupů a částí, které Lucene využívá.

### 3.1.1 Dokument

Základní datovou jednotkou (datovou strukturou), kterou Lucene používá pro reprezentaci dat je `dokument`. Dokument se typicky skládá z několika oddělených pojmenovaných `polí` (`fields`) obsahujích reálná data. Lucene poskytuje API pro vytváření takových dokumentů, ale nedefinuje žádnou logiku (např. z jakých polí má být dokument vytvořen) pro vytváření dokumentů [5].

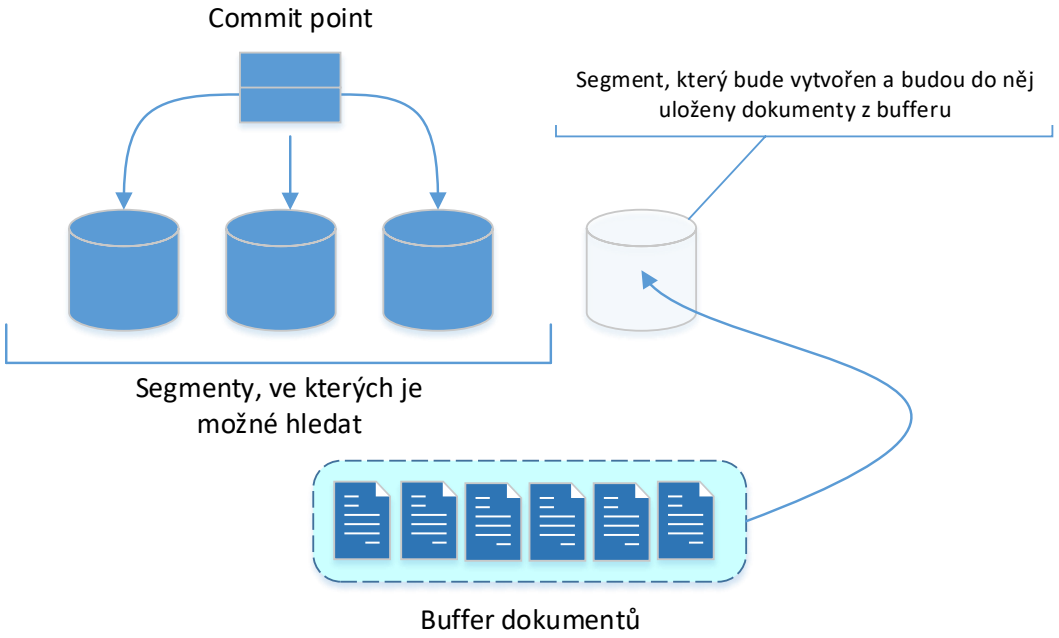
### 3.1.2 Předzpracování dokumentu

Po vytvoření dokumentu následuje jeho předzpracování (tokenizace, normalizace nebo stemming textu dokumentu, viz část 2.5). Lucene poskytuje množství připravených analyzátorů, které zvládají celý proces analýzy dokumentu. Lucene také umožňuje vytvořit vlastní analyzátor nebo upravit již hotový [5].

[1] Internetová adresa pro Elasticsearch: `https://github.com/elastic/elasticsearch` a pro Apache Solr: `https://github.com/apache/lucene-solr`

### 3.1.3 Index v Lucene

Po analýze je dokument přidán do indexu. Lucene využívá datovou strukturu invertovaného indexu (viz část 2.4) a zavádí pojem `segment`. Segment je invertovaným indexem a slovo index v Lucene znamená kolekci segmentů a soubor, který se jmenuje `commit point`. V tomto souboru je uložen seznam všech segmentů (viz obr. 3.1). Nové dokumenty jsou před zapsáním do jednotlivých segmentů uložených na disku přidány do bufferu, který je v paměti. Dokud jsou dokumenty v bufferu, není možné v nich vyhledávat. Z bufferu jsou cyklicky vybírány přidané dokumenty, které jsou pak zapsány do nového segmentu a jméno nového segmentu je také přidáno do commit point souboru [6].




Obrázek 3.1: Znázornění struktury indexu, segmentů a bufferu v Apache Lucene

Segmenty jsou neměnitelné (immutable), takže dokumenty v segmentech nemohou být přímo smazány ani upraveny. Když obdrží Lucene příkaz smazat dokument, tak je tento dokument pouze označen jako smazaný, ale není okamžitě fyzicky smazán ze segmentu. Commit point (soubor) obsahuje seznam, jaký dokument byl v jakém segmentu označen jako smazaný [6]. Dokumenty označené jako smazané jsou fyzicky z disku odstraněny až při procesu slučování segmentů (podrobněji viz [5, 6]). Při aktualizaci je stará verze dokumentu označena jako smazaná a nová verze dokumentu je zapsána do nového segmentu.

### 3.1.4 Vyhledávání v Lucene

Vyhledávání je prováděno nad všemi segmenty indexu. Při zadání dotazu uživatelem musí být dotaz transformován do objektu pojmenovaného `Query`. Lucene poskytuje nástroj nazvaný `QueryParser`, který z dotazu uživatele vytvoří objekt Query. Dotaz může obsahovat booleovské operátory, frázový dotaz nebo žolíkové znaky. Následně proběhne vyhodnocení dotazu a vrácení výsledků. K vyhodnocení dotazů používá Lucene kombinaci booleovského a vektorového modelu (viz části 2.6 a 2.7).

## 3.2 Apache Solr

Vyhledávací nástroj Apache Solr (dále také jen Solr) využívá knihovnu Lucene jako jádro pro indexaci a vyhledávání. Solr je vysoce škálovatelný, poskytuje distribuované vyhledávání a replikaci indexů. Solr je implementován v jazyce `JAVA` jako samostatný server běžící v servletovém kontejneru (např. `Jetty`[2]) a je vyvíjen v rámci projektu Lucene [9].

Pro komunikaci se Solr serverem je možné použít HTTP protokol a REST (Representational state transfer) rozhraní, které podporuje tyto datové formáty: XML (Extensible Markup Language) a JSON (JavaScript Object Notation). Další možností jak komunikovat se serverem Solr je použití nativních klientů, které jsou implementovány pro většinu nejpoužívanějších jazyků: Python, PHP, Java .NET a Ruby [7].

### 3.2.1 Dokument v Apache Solr

Základní datovou jednotkou pro indexaci a dotazování je jako u Lucene `dokument`, který obsahuje jednotlivá `pole` (`field`). U každého pole je možné specifikovat, jaký druh dat obsahuje (tzv. `field type`), a jak s těmito daty pracovat či způsob indexace (tj. jaký použít tokenizér, stemmer atd.). Tyto popisné informace jednotlivých polí se nazývají `schéma`. Schéma se deklaruje ve formátu XML souboru, viz ukázka kódu 3.1.

Definovat schéma pro každý typ dokumentu není povinné, protože může nastat případ, kdy bude potřeba indexovat dokumenty, jejichž struktura ještě není známa nebo tyto dokumenty ještě ani neexistují. V takovém případě se Solr pokusí vytvořit schéma automaticky, včetně rozpoznání datových typů jednotlivých polí [7].

[2]Jetty: `http://www.eclipse.org/jetty/`

```
<schema name="example" version="1.5">
 <fields>
   <field name="id" type="string" indexed="true" stored="true" .../>
   <field name="name" type="text_general" indexed="true" ... />
   <field name="article" type="string" indexed="true" .../>
 </fields>
 <uniqueKey>id</uniqueKey>
 <types>
   <fieldType name="string" class="solr.StrField" .../>
   <fieldType name="text_general" class="solr.TextField">
      <analyzer type="index">
        <tokenizer class="solr.StandardTokenizerFactory"/>
        <filter class="solr.StopFilterFactory" .../>
        <filter class="solr.LowerCaseFilterFactory"/>
      </analyzer>
   </fieldType>
 </types>
</schema>
```

Ukázka kódu 3.1: Ukázka schématu v Apache Solr

### 3.2.2 Indexace v Apache Solr

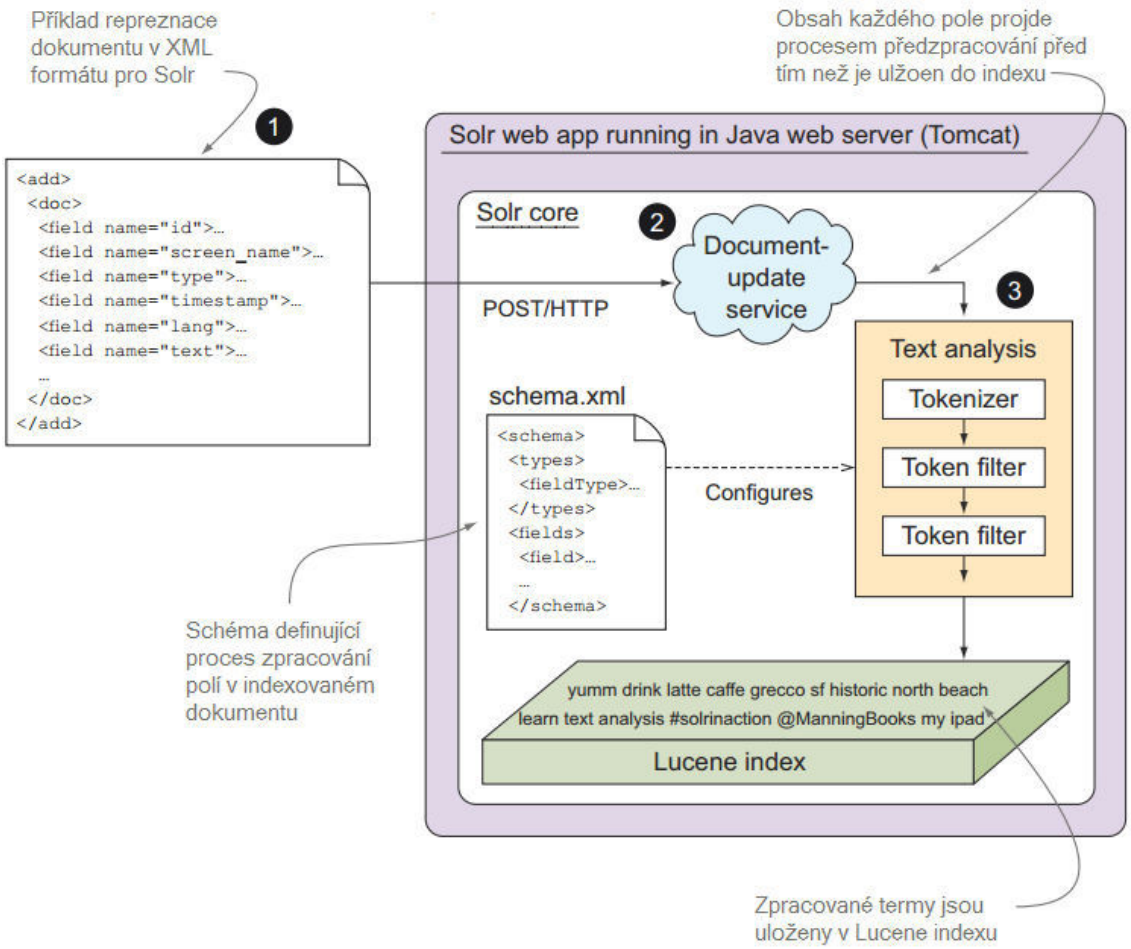
Indexaci v Solr je možné rozdělit na tři části:

1. Převedení dokumentu z jeho nativní reprezentace jako je PDF (Portable Document Format) do formátu, který podporuje Solr (např. XML, JSON).
2. Odeslat dokument na Solr server, typicky protokolem HTTP a jeho metodou `POST`.
3. Provedení předzpracování dokumentu a jeho uložení do indexu.

Na obr. 3.2 je tento postup znázorněn podrobněji. Ve schématu dokumentu je možné definovat tzv. `token filtery`. V každém filtru je nad tokenem provedena nějaká akce, např. transformace (lowercasing, stemming).

### 3.2.3 Vyhledávání v Apache Solr

Solr poskytuje množství nástrojů, které mohou být použity v dotazu a to včetně logických operátorů, žolíkových znaků, regulárních výrazů, frázových dotazů nebo dotazů obsahujících rozsah pro data a čísla [7].




Obrázek 3.2: Znázornění postupu indexace v Apache Solr (obrázek převzat a upraven z [7])

### 3.2.4 Škálování v Apache Solr

Solr umožňuje škálování, které se typicky provádí ze dvou důvodů. Prvním je propustnost dotazů, což je počet dotazů za sekundu, které mohou být najednou zpracovány. Propustnost je možné zvýšit replikací indexu na jiné servery, a tím také rozložit zátěž. Replikace také přináší bezpečnostní výhodu při splnění určitých podmínek. Pokud jedna replika selže, systém je stále schopný pracovat [7].

Druhým důvodem je počet indexovaných dokumentů. Pokud bude index obsahovat velké množství dokumentů, bude výkonnost (rychlost) vyhledávání klesat. Řešením je rozdělit index na menší části, které se nazývají `shards` a vyhledávání se poté provádí distribuovaně nad jednotlivými částmi. Pokud je index rozdělen na jednotlivé části (shards), pak jsou replikovány tyto části [7].

Solr umožňuje škálování automatizovat pomocí nástroje `SolrCloud`. SolrCloud využívá `Apache ZooKeeper`[3] pro distribuci konfigurací, správu replik a jednotlivých částí indexů (shards).

## 3.3 Elasticsearch

Elasticsearch (dále také jen ES) je vysoce škálovatelný nástroj pro vyhledávání a analýzu dat založený na Apache Lucene. Dovoluje uložit, hledat a analyzovat velké objemy dat téměř v reálném čase. ES je používán velkými webovými stránkami a institucemi jako je například `Wikipedia, Netflix, The New York Times, IBM` nebo `GitHub`, nejen pro vyhledávání, ale i pro analýzu dat a sběr logů. ES je stejně jako Apache Solr a Lucene implementován v jazyce `JAVA` [8].

ES je distribuován jako samostatný server, se kterým je možné komunikovat přes REST API pomocí protokolu HTTP nebo klienty implementovaných v různých programovacích jazycích (Groovy, JavaScript, .NET, PHP, Perl, Python a Ruby) [6].

### 3.3.1 Dokument v Elasticsearch

Stejně jako v Lucene a Solr je v ES základní datovou jednotkou, která je indexována a nad níž probíhá vyhledávání, dokument. Dokument je v ES reprezentován ve formátu JSON, příklad JSON dokumentu lze vidět v ukázce kódu 3.2. Nástroje pro práci s JSON formátem jsou dostupné v naprosté většině programovacích jazyků. JSON je jednoduchý, stručný a snadno čitelný. V ES je pro reprezentaci většiny dat používán téměř výhradně JSON (dotazy, výsledky dotazů, konfigurace) [6].

V ES každý dokument patří k nějakému `typu` (`type`), který je umístěn v indexu. Typ lze přirovnat k tabulce v tradičních relačních databázích. Pro lepší představu je dále znázorněna paralela mezi ES a tradiční relační databází:

| | | | | |
|---|---|---|---|---|
| Relační databáze | ⇒ Instance databáze | ⇒ Tabulky | ⇒ Řádky | ⇒ Sloupce |
| Elasticsearch | ⇒ Indexy | ⇒ Typy | ⇒ Dokumenty | ⇒ Pole |

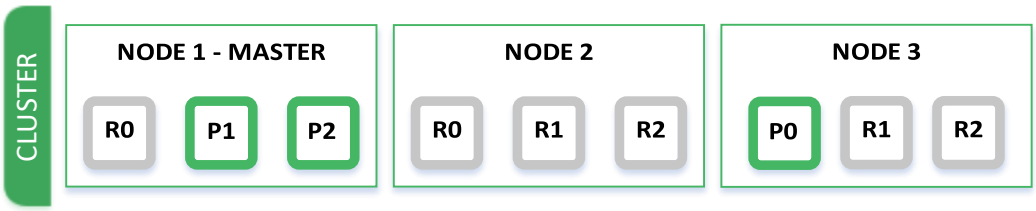
[3] Apache ZooKeeper `http://zookeeper.apache.org/`

```
{
  ”email”: ”pribanp@students.zcu.cz”,
  ”first_name”: ”Pavel”,
  ”last_name”: ”Priban”,
  ”info”: {
        ”gender”: ”male”,
        ”age”: 24,
        ”interests”: [”information retrieval”, ”sport” ]
  },
  ”join_date”: ”2014/05/01”
}
```

Ukázka kódu 3.2: Příklad použití reprezentace dokumentu jako JSON

### 3.3.2 Cluster

Běžící instance ES serveru se nazývá `uzel` (`node`) a skupina spolupracujících uzlů je pak označována jako `Cluster`. Cluster také může být tvořen jediným uzlem. V clusteru je vždy zvolen jeden hlavní uzel, tzv. `master`, kterému jsou podřízeny ostatní uzly. Tento uzel řídí vytváření nebo mazání indexů a přidávání dalších uzlů do clusteru. U ES platí stejné důvody a principy pro škálování a replikaci jako u Apache Solr (viz část 3.2.4).




Obrázek 3.3: Znázornění clusteru se třemi uzly, každý primární shard disponuje dvěma replikami

Stejně jako u Apache Solr i ES rozděluje index na menší části, které se jmenují `shards` a to z důvodu škálování. V ES je `shard` jednou instancí indexu knihovny Lucene. Celý index v ES je tak složen z několika indexů knihovny Lucene. Shard může být buď `primární` nebo `replikovaný`. Každý dokument v indexu patří k právě jednomu primárnímu shard. Replikovaný shard je pouze kopií primárního shard a slouží k redundanci dat jako ochrana před hardwarovým selháním. Počet primárních shards je zvolen při vytvoření indexu a poté už je počet neměnný [6]. Na obr. 3.3 je znázorněn cluster se třemi uzly, třemi primárními shards (P0 - P2), přičemž každý primární shard disponuje dvěma kopiemi (replikami) (R0-R2) [6].

### 3.3.3 Mapování

Jak bylo uvedeno v části 3.3.1, každý dokument v indexu je přiřazen k typu. Každý typ má svoje vlastní `mapování`, které definuje strukturu pro dokument, tzn. datové typy polí a postup jejich zpracování při indexaci (obdoba schématu u Apache Solr). Pokud není mapování definováno, ES podle dat v jednotlivých polích dokumentu vytvoří mapování sám (tzv. dynamické mapování). Příklad mapování lze vidět v ukázce kódu 3.3.

```
{"my_type": {
    "properties": {
      "title": {
        "type": "string",
        "analyzer": "standard"
      },
      "text": {
        "type": "text",
        "analyzer": "custom_text_analyzer"
      }
}}}
```

Ukázka kódu 3.3: Ukázka mapování pro dokument s poli `text` a `title` a nastavením analyzátorů

### 3.3.4 Analyzátor

Před indexací (uložením) dokumentu je provedeno předzpracování pomocí `analyzátoru`, což je jen název pro balíček tří funkcí. Analyzátory se definují v mapování typu (viz část 3.3.3). Nejprve projde indexovaný dokument `filtry znaků` (`character filters`), které mají za úkol pročistit text dokumentu před tokenizací (např. odstranit HTML značky nebo převést znak & na slovo `and`). Druhou funkcí je `tokenizér` (viz část 2.5.1). Poslední část představují `filtry tokenů`, které mohou zahrnovat stemming, lowercasing, odebrání stop slov apod. [6].

Filtrů znaků a filtrů tokenů může být v jednom analyzátoru obecně použito nula až $N$, tokenizér je vždy právě jeden. Zpracovávané tokeny (popř. text u znakových filtrů) postupně projdou všemi definovanými filtry. ES obsahuje množství již hotových analyzátorů, filtrů a tokenizerů, které je možné použít a případně upravit. Vlastní analyzátory se definují pro každý index zvlášť. V ukázce kódu 3.4 lze vidět příklad definice nového analyzátoru.

```
{"settings": {
    "analysis": {
      "analyzer": {
          "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": ["icu_normalizer" ,"czech_stop" ,"czech_stemmer" ,"icu_folding" ]
 }}}}}
```

Ukázka kódu 3.4: Definice vlastního analyzátoru se čtyřmi token filtry

Indexace v ES probíhá téměř stejným postupem jako u Apache Solr. Z původního formátu indexovaného dokumentu je vytvořen JSON, který je přes REST API odeslán na běžící instanci ES serveru, kde je předzpracován a následně uložen do indexu.

### 3.3.5 Vyhledávání v Elasticsearch

API v ES dovoluje vyhledávat dvěma způsoby. První, tzv. `query-string` verze, předává všechny parametry vyhledávacího dotazu jako jeden řetězec (string). Query-string vyhledávání je vhodné pro testování a ladění z příkazové řádky, např. pomocí programu `curl` [6], viz ukázka kódu 3.5, kde je v indexu „cities“ a všech polích pojmenovaných „title“ hledáno slovo „plzen“.

```
$ curl -XGET "http://localhost:9200/cities/_search?q=title:plzen"
```

Ukázka kódu 3.5: Použití programu curl pro vyhledání dotazu „plzen“ v polích pojmenovaných „title“ v indexu „cities“ na ES serveru

Druhou možností je vytvoření dotazu s tělem obsahujícím JSON. Tento JSON je zapsaný v dotazovacím jazyku `Query domain-specific language (DSL)`, který dovoluje vytvářet velmi komplexní a robustní vyhledávací dotazy. Předchozí příklad přepsaný do query DSL je znázorněn v ukázce kódu 3.6.

```
$ curl -XGET "http://localhost:9200/dp-index-cs/_search" -d "
{
        "query" :{
            "match" :{
              "title":"praha"
}}}
"
```

Ukázka kódu 3.6: Ukázka dotazu zapsaného v query DSL

## 3.4 Porovnání Elasticsearch a Apache Solr

Oba systémy v jejich aktuálních verzích (Solr 6.5.0 a Elasticsearch 5.3.0, duben 2017) lze považovat za podobné nástroje v tom smyslu, že nabízí téměř totožnou funkcionalitu a mají srovnatelný výkon, společně jsou ve své oblasti (IR) nejpoužívanějšími technologiemi. I přesto, že se v některých ohledech liší, oba nástroje jsou vhodným řešením pro naprostou většinu běžných úloh, ve kterých je potřeba vyhledávání informací. ES i Solr jsou vyspělými a stabilními nástroji a stojí za nimi silná a aktivní komunita uživatelů [13, 14, 15]. Dále budou uvedeny některé rozdíly, výhody a nevýhody těchto nástrojů.

Solr je spíše orientovaný na textové vyhledávání, zatímco ES navíc nabízí sadu nástrojů pro analýzu dat, známou jako `ELK stack` (často se používá pro ukládání logů a vyhledávání v nich) [14, 16]. Podle [17] (viz tab. 3.1) je ES v současné době (duben 2017) nejpopulárnějším vyhledávacím nástrojem. Instalace ES je jednoduchá v porovnání s Apache Solr. Velikost instalačního balíčku ES ve verzi 5.3.0 je přibližně 36 MB, velikost instalačního balíčku Solr ve verzi 6.5.0 je cca 160 MB [15, 16, 13].

| **Pořadí** | **Vyhledávací nástroj** | **Skóre** |
|---|---|---|
| 1. | Elasticsearch | 105, 67 |
| 2. | Apache Solr | 64, 37 |
| 3. | Splunk | 55, 50 |
| 4. | MarkLogic | 11, 20 |
| 5. | Sphinx | 6, 66 |

Tabulka 3.1: Pět nejpopulárnějších vyhledávacích nástrojů podle [17] k dubnu 2017, postup výpočtu skóre a odkazy na jednotlivé stránky vyhledávacích nástrojů lze nalézt v [17]

Solr podporuje rozdělení a přidání nových shards (viz části 3.3.2 a 3.2.4), v ES primární shard není možné rozdělit ani přidat. V ES je možné přidávat repliky primárních shards, a pokud je přidán nový uzel (tj. běžící instance ES, zpravidla na jiném fyzickém serveru), ES bude automaticky řídit load-balancing (zatížení jednotlivých serverů), což může zahrnovat přesouvání primárních shards na jiné uzly (node). Dokumentace u Apache Solr je na velmi dobré úrovni a obsahuje řadu užitečných příkladů použití API a konfigurací. Dokumentace ES je organizovaná, ale na rozdíl od Solr jí často chybí dobré a samovysvětlující příklady [15].

### 3.4.1 Závěr porovnání Elasticsearch a Apache Solr

Hlavní výhodou obou nástrojů je, že zapouzdří funkcionalitu knihovny Apache Lucene do vyhledávače. Ten je možné téměř ihned nasadit a používat, není tak potřeba detailně znát knihovnu Apache Lucene nebo mít hluboké znalosti v oblasti vyhledávání informací. Navíc oba systémy také řeší škálování a distribuované nasazení.

Jedním z cílů diplomové práce (viz část 1.1) je implementovat vyhledávač nad textovými daty ze zpravodajských portálů. Pokud jde o funkcionalitu vyhledávání v textových datech, mezi oběma systémy nejsou žádné výrazné rozdíly, a jak bylo již zmíněno v předchozí části 3.4, pro většinu běžných úloh, kde je potřeba vyhledávání, je Elasticsearch i Apache Solr vhodným řešením. Na základě prostudování [6, 7, 12, 13, 14, 15, 16] by nemělo upřednostněním jednoho nástroje před druhým (pro tento případ užití) dojít k získání nějakých výrazných výhod. Protože autor diplomové práce má již s nástrojem Elasticsearch dřivější zkušenosti a vedoucí diplomové práce preferuje použití nástroje Elasticsearch, bude pro implementaci vyhledávání použit Elasticsearch.

# 4 Návrh vyhledávání

Tato kapitola se bude věnovat návrhu vyhledávání pro systém `MediaGist`. Nejprve bude krátce představen systém MediaGist a data ze zpravodajských portálů, nad kterými bude vyhledávání implementováno. Následně bude popsán návrh softwarového řešení vyhledávání.

## 4.1 Systém MediaGist

MediaGist[1] je on-line systém pro kroslinguální analýzu agregovaných zpráv a komentářů založený na technologii sumarizace a analýze sentimentu. Články ze zpravodajských portálů vydané za poslední týden jsou samostatně shlukovány v pěti jazycích a shluky jsou poté propojeny napříč jazyky. Systém Mediagist je navržen k detekci a prozkoumávání zpravodajských témat, která jsou kontroverzně reportována nebo komentována napříč různými zeměmi [10].

### 4.1.1 Data ze zpravodajských portálů

Data, která budou indexována, a ve kterých bude následně prováděno vyhledávání, jsou uložena ve formátu XML. Zjednodušenou strukturu XML souboru (celý dokument by byl příliš velký) lze vidět v ukázce kódu 4.1. Kořenovým elementem XML souboru je element `rss`, ve kterém je umístěn element `channel`. Element `channel` obsahuje informace o clusteru jako jeho popis, sumarizace textu či komentářů, id clusteru a jazyk, pro který je daný cluster vytvořen.

Dále jsou v elementu clusteru položky `item`, které obsahují informace o zpravodajských článcích, z nichž byl cluster vytvořen. Jejich počet se pro různé clustery liší. V jednotlivých elementech článků je uveden jazyk, titulek, id a odkaz na původní článek. V každém elementu článku jsou ještě obsaženy informace o rozpoznaných pojmenovaných entitách a sentimentu.

[1] Systém MediaGist je dostupný na `http://mediagist.kiv.zcu.cz`

```
<rss xmlns:emm="http://emm.jrc.it" version="2.0">
  <channel>
    <title>Tuhý je podezřelý z úniku informací</title>
    <description>Policejní prezident Tomáš Tuhý</description>
    <language>cs</language>
    <guid>b0f5d9350f6cc03449c4015b7aa3233d</guid>
    <pubDate>2016-06-21 05:32:23 CEST</pubDate>
    <emm:summary>[#] Podle ní se...</emm:summary>
    <commentSummary>Hele, přestante spek...</commentSummary>

    <item emm:id="acec3af282a52ce1881ca613bb769030">
       <title>Ostravský ÚOOZ podezřívá....</title>
       <link>http://domaci.ihned.cz/c1-65339...</link>
       <description>Vedoucí ostravského ÚOOZ...</description>
       <emm:contentType>text/html</emm:contentType>
       <pubDate>2016-06-20 18:34:00 CEST</pubDate>
       <iso:language>cs</iso:language>
       <guid>acec3af282a52ce1881ca613bb769030</guid>
       <emm:entity name="Milan Chovanec">Milan Chovanec</emm:entity>
       <emm:tonality>-13</emm:tonality>
       <commentTonality pos="0" neut="0" neg="0">0</commentTonality>
       <commentSummary>Pane Tomáši Tuhý,... </commentSummary>
        ...
       <emm:text>Vedoucí ostravského ÚOOZ...</emm:text>
    </item>
    <item emm:id="336b2a4bf74c7f01ba8a73ec18cea014">...</item>
    <item emm:id="9f052b1729c7b1c6cabfadab48f1dd0a">...</item>
    <item emm:id="02405fc8d78ce7b17274caf0b0cd9426">...</item>
    <item emm:id="8054eca78e90c590785083320c3c5018">...</item>
    <item emm:id="b0f5d9350f6cc03449c4015b7aa3233d">...</item>
      ...
  </channel>
</rss>
```

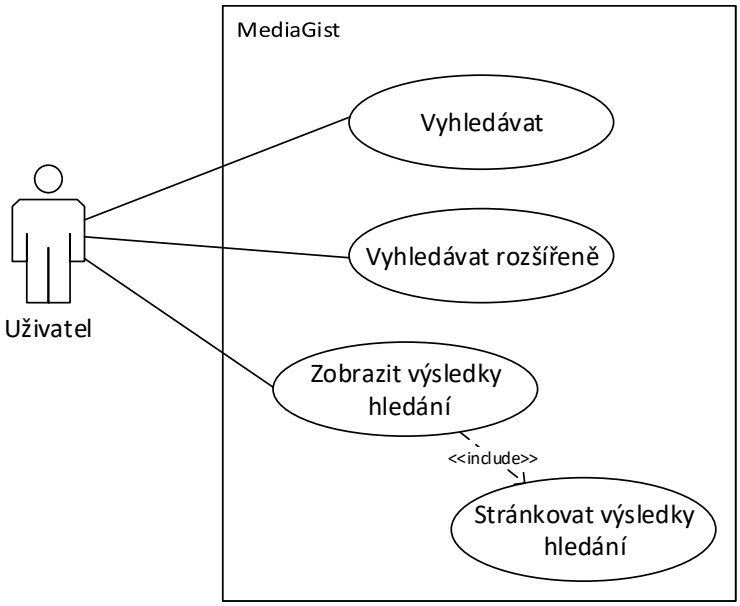
Ukázka kódu 4.1: Zjednodušená ukázka struktury dat ze zpravodajských portálů ve formátu XML

## 4.2 Požadovaná funkcionalita

Jedním z cílů této diplomové práce je implementovat vyhledávání v datech ze zpravodajských portálů (zpravodajské články) v systému MediaGist. Protože systém je dostupný jako webová služba (webové stránky, viz část 4.1), bude vyhledávání navrženo a implementováno tak, aby jej bylo možné integrovat na webovém serveru systému MediaGist. Systém by měl být rozšířen o funkcionalitu, která by uživatelům měla umožňovat následující:

- jednoduché vyhledávání,
- rozšířené vyhledávání,
- zobrazení a stránkování výsledků,
- zobrazení detailu výsledku.

Na obrázku 4.1 je zobrazen `UML` diagram případů užití pro předchozí požadavky. UML (Unified Modeling Language) je „univerzální jazyk pro vizuální modelování systémů“ [28], který se používá především v softwarovém inženýrství. UML diagramy použité v této práci byly vytvořeny především podle [28] a [29].

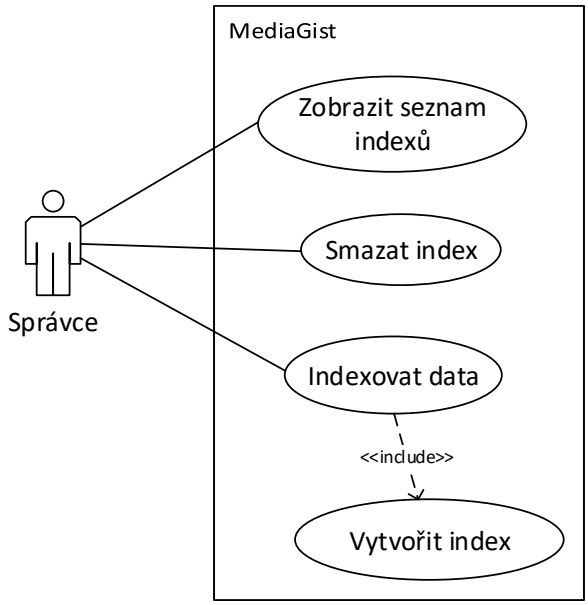



Obrázek 4.1: UML diagram případů užití znázorňující funkcionalitu vyhledávání

### 4.2.1 Správa vyhledávání

Z pohledu správce systému je potřeba, aby správce mohl nová data indexovat, případně mazat původní nebo o nich získávat další informace. Tyto funkce, jež by měl systém umožňovat, jsou zobrazeny na obr. 4.2, který je diagramem případů užití. Indexovat nová data nemusí jen správce (člověk), ale může to být např. jiný program nebo jiná část systému, která tato data cyklicky získává z internetu.

Všechny tyto akce je možné vykonat přímo posláním požadavků na běžící server Elasticsearch z příkazové řádky programem `curl` nebo pomocí nástroje `Kibana` (viz [8]). Nevýhodou tohoto přístupu je, že uživatel (správce) musí znát syntaxi dotazů, které zpracovává Elasticsearch. Pokud MediaGist tuto funkcionalitu zapouzdří a implementuje do sebe, správce pak nebude muset znát konkrétní syntaxi příkazů Elasticseach pro vybranou funkcionalitu.




Obrázek 4.2: UML diagram případů užití znázorňující správu indexu vyhledávání

## 4.3 Architektura systému

Na základě požadované funkcionality (viz část 4.2), kterou by měl systém obsahovat, bude vytvořen návrh architektury. Požadovaná funkcionalita bude rozdělena do `dvou aplikací` (komponent), tj. webového klienta a klienta pro indexaci. Webová část by měla být přidána ke stávající funkcionalitě současného webu a bude obsahovat vyhledávání a zobrazení výsledků. Integrace nové funkcionality k současnému webovému serveru ale není předmětem této diplomové práce, a proto bude funkcionalita vyhledávání demonstrována na samostatné webové aplikaci. Při budoucí integraci požadované funkcionality bude možné použít nově vytvořené části (stránkování, zobrazení výsledků, vyhledávací formuláře apod.) v současné webové aplikaci.

Druhá aplikace bude samostatný program umožňující správu Elasticsearch serveru (viz část 4.2.1). Obě aplikace budou využívat `knihovnu`, jež bude obsahovat rozhraní deklarující metody pro indexaci, vyhledávání a práci (správu) se serverem Elasticsearch.

Protože současný webový server systému MediaGist je naprogramován za pomoci programovacího jazyka JAVA, technologií servletů a JSP (JavaServer Pages), pro všechnu nově požadovanou funkcionalitu bude pro implementaci použit také jazyk JAVA.

Obrázek 4.3 obsahuje diagram komponent, který znázorňuje návrh architektury vyhledávání. Komponenta `Request Controller` zpracovává všechny příchozí webové požadavky a volá funkce pro vyhledávání komponenty `Knihovna`, která komunikuje přímo se serverem ES. V klientovi (programu) pro indexaci jsou příkazy zpracovány v komponentě `Ovládání klienta`, z níž jsou taktéž volány funkce komponenty `Knihovna`.

Vytvořením samostatné aplikace pro správu serveru Elasticsearch je možné provádět indexaci nezávisle na webovém serveru. Výhodou zapouzdření nástrojů (funkcí) pro práci se serverem Elasticsearch do knihovny je, že knihovna může být vyvíjena a testována nezávisle na ostatních částech systému. Dále ji bude možné použít i v případných dalších projektech. Na diagramu nasazení (viz obr. 4.4) je ukázán možný příklad reálného nasazení.

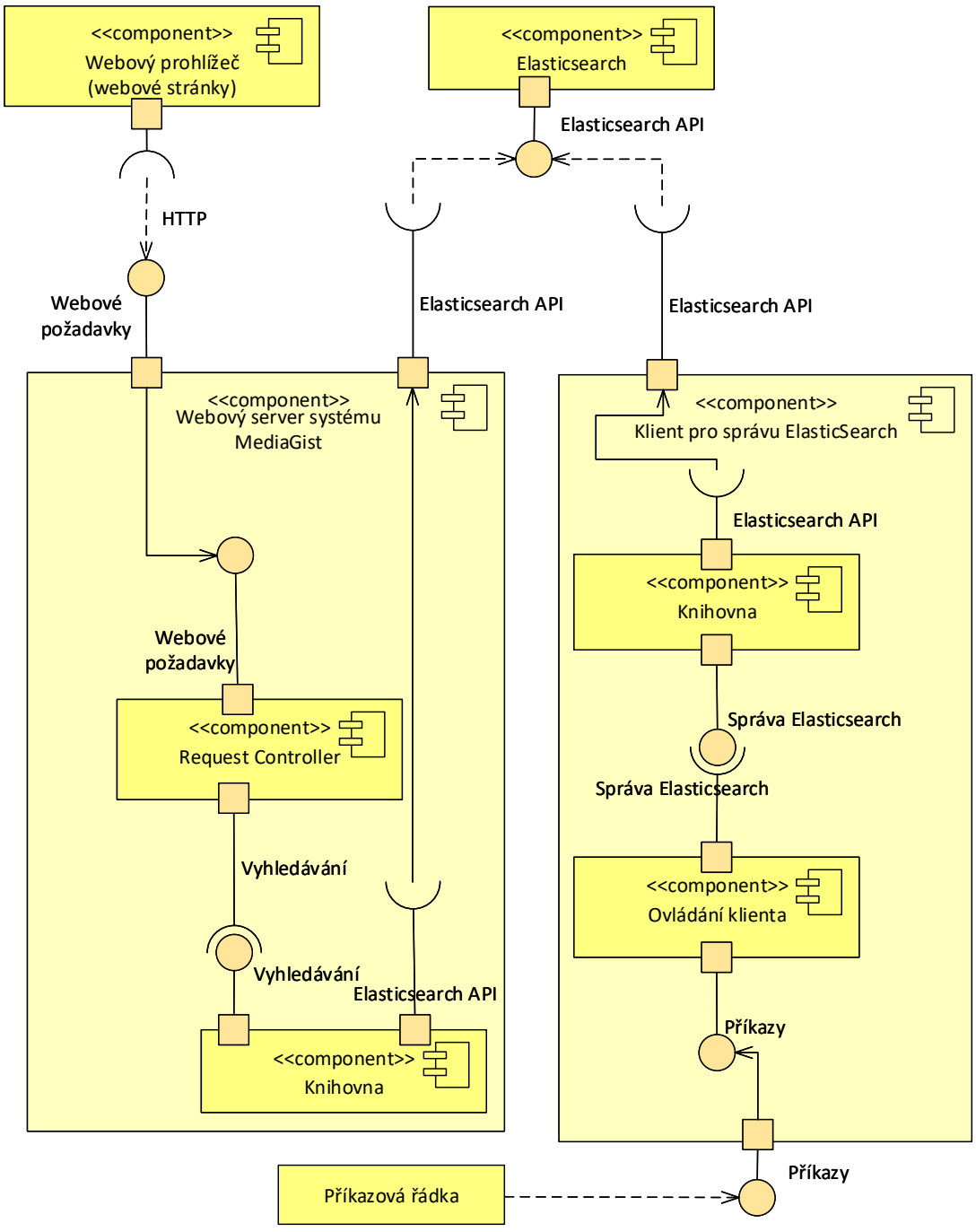
## 4.4 Návrh knihovny

Nejdůležitější částí implementace bude knihovna, která bude poskytovat především funkcionalitu v podobě vyhledávání (jednoduché a pokročilé) a indexace dat (XML souborů) v Elasticsearch. Dále bude zahrnovat funkce pro vytváření, mazání a získávání seznamu vytvořených indexů.

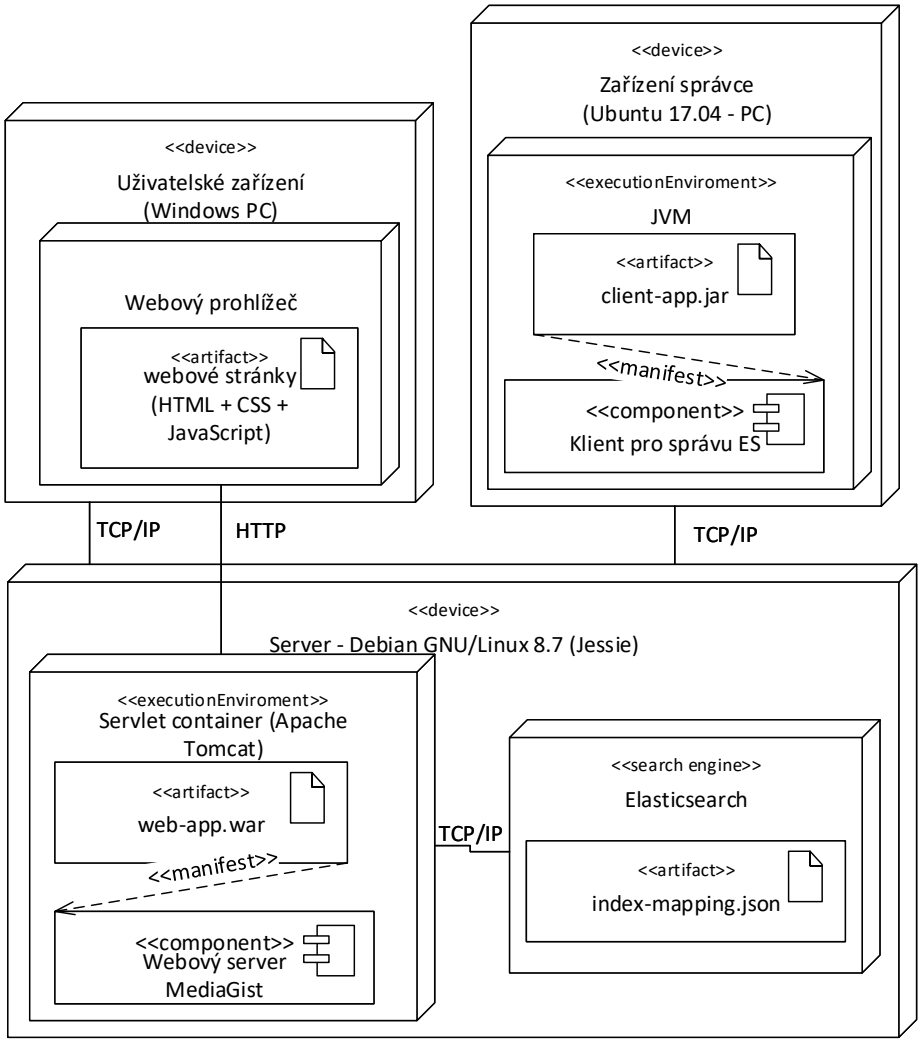
### 4.4.1 Návrh indexu

Jedním z nejdůležitějších rozhodnutí při tvorbě knihovny je návrh indexu a postup zpracování dokumentů při indexaci. Protože dokumenty určené k indexaci se vyskytují ve více jazycích (konkrétně v pěti), je důležité zvolit řešení, které tento fakt zohledňuje. Některé procesy při předzpracování dokumentů (odebrání stop slov, stemming apod.) musí být pro každý jazyk definovány samostatně, ale zároveň musí být možné vyhledávat ve všech jazycích najednou. Není možné tedy mít jeden index pro všechny jazyky, který bude obsahovat pouze pole odpovídající elementům z XML souboru (viz část 4.1.1). Mezi možná řešení patří tato tři:

1. Všechny dokumenty budou v jediném indexu, ale mapování typu bude obsahovat zvlášť jednotlivá pole pro každý jazyk.
2. Všechny dokumenty budou v jediném indexu, ale každý jazyk bude mít svůj vlastní typ (mapování).
3. Pro každý jazyk bude vytvořen jeden index.




Obrázek 4.3: Diagram komponent znázorňující návrh architektury vyhledávání a správy serveru Elasticsearch systému MediaGist




Obrázek 4.4: Diagram možného způsobu nasazení

Příklad, jak by mohlo vypadat mapování pro první případ, lze vidět v ukázce kódu 4.2. Nevýhodou tohoto přístupu indexace je, že v případě dokumentů ze systému MediaGist, by každý dokument při indexaci využil pouze ta pole, která se vztahují k danému jazyku dokumentu. Toto řešení je vhodné v případě, že indexované dokumenty jsou obsahově identické a jsou pouze přeloženy do jiných jazyků. Tyto totožné dokumenty by pak byly sloučeny do jednoho dokumentu uloženého v indexu, který by obsahoval všechny jazykové verze pro daný dokument.

```
{"mappings":{
      "movie":{
         "properties":{
            "title_cz":{
               "type":"string",
               "analyzer":"czech" },
            "title_en":{
               "type":"string",
               "analyzer":"english"}
}}}
```

Ukázka kódu 4.2: Způsob mapování, kdy všechny dokumenty jsou uložené

Druhý a třetí způsob se může zdát téměř stejný. Pro všechny jazyky bude v obou případech definováno vlastní mapování (typ), ale je potřeba si uvědomit, jaké dopady budou tato řešení mít. Pokud bude použito řešení s pouze jedním indexem, (více typů v jednom indexu) budou všechny dokumenty uloženy v jednom indexu, který je rozdělen na pevně daný počet částí (tzv. shards tvořené indexy knihovny Apache Lucene, viz část 3.2.4). Při použití třetího řešení s jedním indexem pro každý jazyk se počet primárních shards znásobí počtem indexů.

Vyhledávání je vždy prováděno nad všemi částmi indexů (shards) a výsledky z každé části je poté potřeba sloučit. Pokud je částí (shards) příliš mnoho, může slučování výsledků spotřebovávat velké množství zdrojů (výpočetní čas CPU a paměť). Ovšem pokud bude dokumentů velké množství, pak prohledávání jednoho velkého indexu (druhé řešení) bude trvat déle než prohledání více menších indexů [25]. Protože počet indexovaných dokumentů (velikost indexu) v systému MediaGist v budoucnu stále poroste (jsou vytvářeny stále nové clustery), tak na základě prostudování [25, 26] bude vhodnějším řešením použít poslední možnost z nabízených, tedy každý jazyk bude mít vlastní index. Navíc počet částí (shards) jednotlivých indexů (výchozí počet je 5) je možné změnit (zmenšit) při vytváření indexu.

### 4.4.2 Návrh vyhledávání

Jednoduché vyhledávání umožní uživatelům použít závorky pro vynucení precedence a booleovských operátorů AND, OR a NOT. V pokročilém vyhledávání bude možné zadat, v jakých jazycích (nad jakými indexy) bude vyhledávání probíhat, vybrat části (nadpis, text, komentáře) dokumentů, které budou do vyhledávání zahrnuty a časové rozmezí, kdy byly clustery (dokumenty) vytvořeny. Dále bude možné v pokročilém vyhledávání upřesnit dotaz za pomoci fráze, výčtu slov, jež se musí současně všechna v hledaném dokumentu vyskytnout, výčtu slov, z nichž alespoň jedno se musí v hledaném dokumentu vyskytnout a výčtu slov, která se v hledaném dokumentu nesmí vyskytnout.

## 4.5 Návrh webové aplikace

Protože integrace vyhledávání (které bude implementováno) se současným systémem není cílem této práce, bude vytvořena jednoduchá webová aplikace demonstrující vytvořenou funkcionalitu. Webová aplikace bude vytvořena tak, aby při integraci bylo možné použít její důležité části bez velkých úprav, nejlépe je jen zkopírovat a vložit do současného systému. Současný systém je napsán kombinací technologie JSP, servletů a programovacího jazyka JAVA z tohoto důvodu bude webová aplikace implementována také technologií JSP, jazyka JAVA a frameworku `Spring`[2].

### 4.5.1 Uživatelské rozhraní

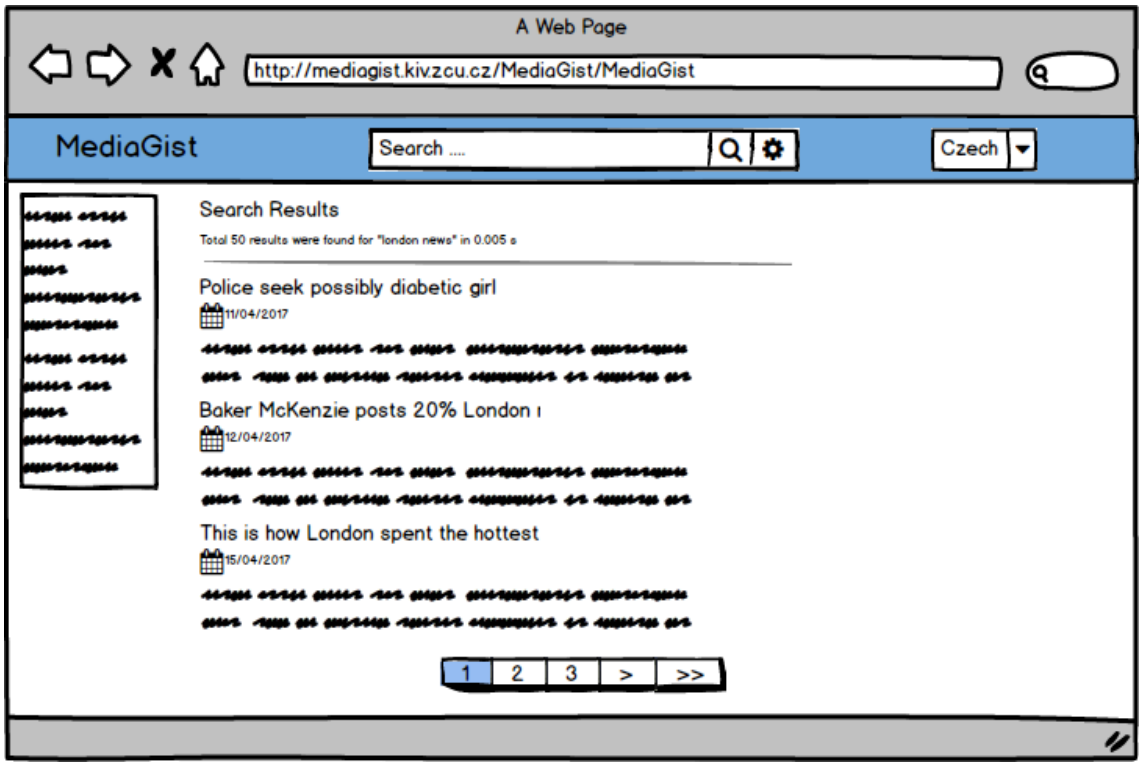
Uživatelské rozhraní (UI) bude realizováno za pomoci webových stránek, jejichž vzhled a základní struktura bude převzata z aktuálního webu a bylo tak možné použít nově vzniklé části (zobrazení výsledků, stránkování apod.). Stránky nebudou obsahovat žádnou současnou funkcionalitu systému (pouze zaslepené odkazy). Protože systém MediaGist analyzuje články v různých jazycích, budou stránky lokalizovány do českého a anglického jazyka. Pro implementaci stránek bude použit framework `Bootstrap` [3].

Webové stránky budou vytvořeny na základě obr. 4.5 a 4.6, které představují návrh pro ovládání vyhledávání. Na obr. 4.5 je možné vidět návrh vyhledávacího formuláře pro jednoduché vyhledávání spolu s rozvržením vyhledaných výsledků a stránkováním. Jednoduché vyhledávání bude také podporovat našeptávání výsledků, tzn. pokud uživatel začne zadávat dotaz, budou mu ihned nabízeny možné relevantní výsledky. Obrázek 4.6 obsahuje návrh pro pokročilé vyhledávání, jenž uživateli umožňuje upřesnit dotaz, např. frází, která se musí v dokumentu vyskytnout apod. Dále má uživatel možnost omezit vyhledávání jen na některé části dokumentů (nadpisy, komentáře nebo text článků) a jazyky, nad kterými bude vyhledávání probíhat.

## 4.6 Návrh klienta pro indexaci

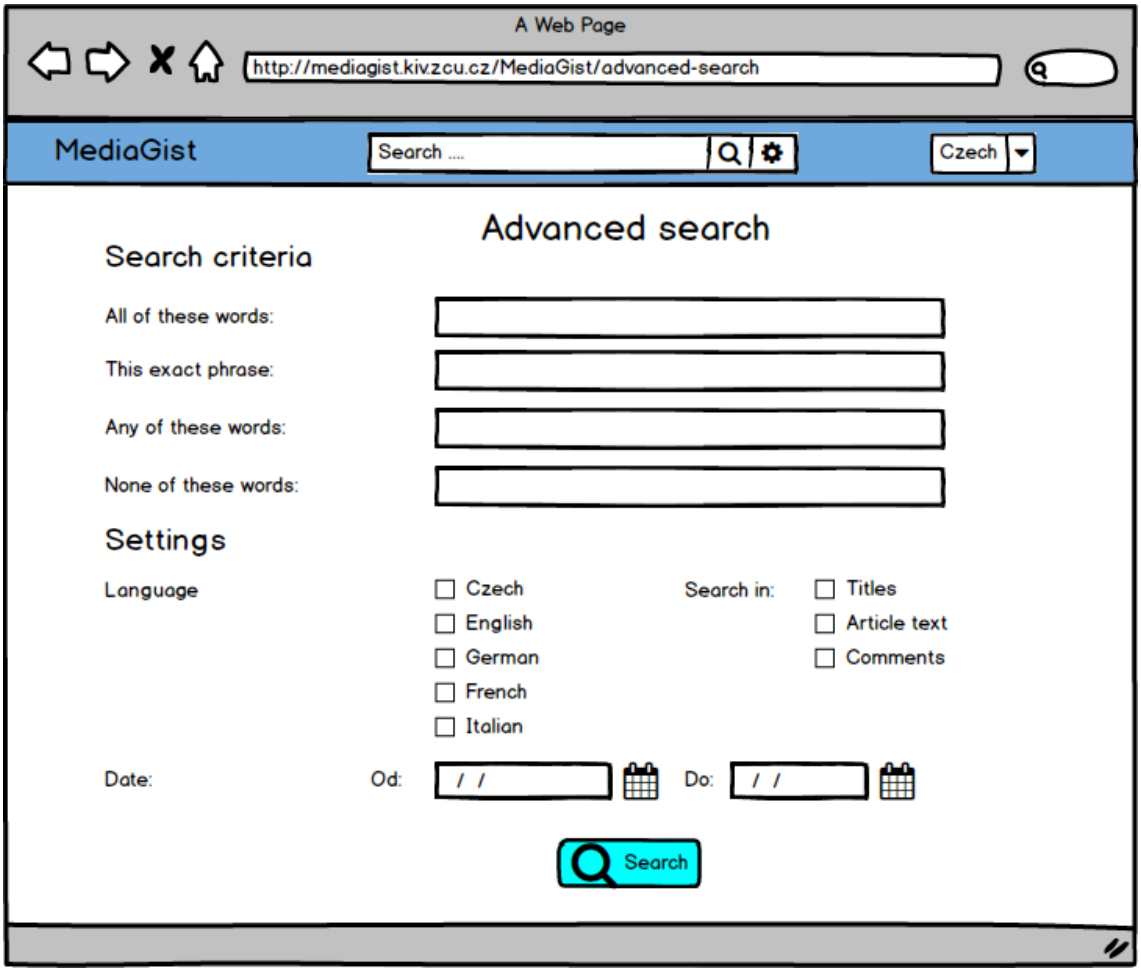
Klient pro indexaci (správu) bude samostatným programem umožňujícím indexaci dat (dokumentů) ze systému MediaGist a případné odstraňování indexů. Protože clusterů (dokumentů) je zpravidla vytvořeno více najednou, je žádoucí, aby mohly

[2] https://www.spring.io/
[3] http://www.getbootstrap.com/




Obrázek 4.5: Návrh formuláře pro jednoduché vyhledávání a zobrazení výsledků spolu se stránkováním

být indexovány hromadně (v dávkách). Program bude implementován v programovacím jazyce JAVA.




Obrázek 4.6: Návrh formuláře pro pokročilé vyhledávání

# 5 Realizace vyhledávání

V této kapitole bude popsána realizace požadované funkcionality, která byla navržena v kapitole 4. Nejprve bude popsána vytvořená knihovna a poté oba programy, jež tuto knihovnu využívají - webová aplikace a klient pro indexaci. K oběma aplikacím jsou dostupné uživatelské příručky, které se nacházejí v příloze A. Zdrojové soubory aplikací jsou umístěny na přiloženém DVD (viz příloha D). Část tříd knihovny je pokryta `JUnit` testy, jenž byly vytvořeny na základě některých doporučení z [20]. Při implementaci byly využívány tyto zdroje: [6, 8]. Implementace byla provedena ve vývojovém prostředí `Intellij IDEA` [1] a byl používán verzovací systém Git.

## 5.1 Načtení dat

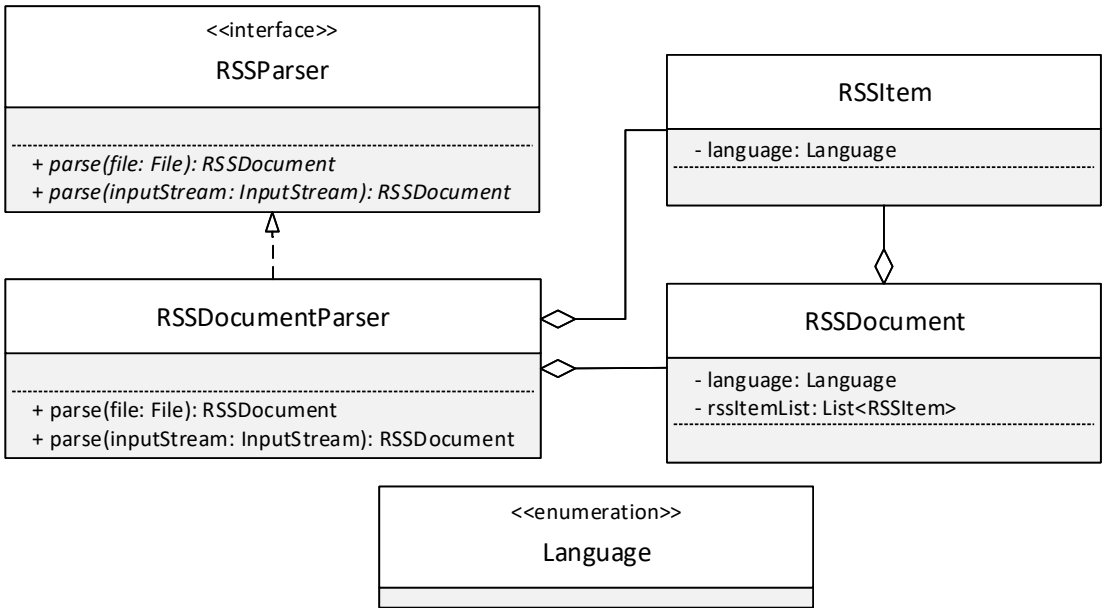
Před samotnou indexací je potřeba nejdříve dokumenty uložené na disku načíst do paměti. Pro načtení dokumentů obsahuje knihovna rozhraní `RSSParser` a třídu `RSSDocumentParser` implementující toto rozhraní. Třída `RSSDocumentParser` umožňuje načtení dokumentů, které jsou ve formátu popsaném v části 4.1.1. Pokud by bylo případně v budoucnu potřeba načítat data, která budou v jiném formátu (např. JSON), stačí pouze vytvořit novou třídu implementující rozhraní `RSSParser`. Na obr. 5.1 je vidět diagram tříd, které jsou používány při načítání.

## 5.2 Vytvoření indexů

Po načtení dokumentů jsou při indexaci, pokud neexistují, vytvořeny indexy pro jednotlivé jazyky dokumentů. Indexy jsou pojmenovány jako `dp-index-`*jazyk*, např. index pro češtinu se bude jmenovat `dp-index-cs`. V případě potřeby je možné názvy indexů upravit ve složce projektu `\resources\index-config`, která obsahuje konfigurace pro jednotlivé indexy. Vytvoření vlastních analyzátorů a nastavení jednotlivých indexů, tzn. jejich mapování, použité tokenizéry a analyzátory bylo nejnáročnějším a také nejdůležitějším úkolem při implementaci.

Pro každý index (jazyk) je potřeba sestavit mapování jeho typu a nastavení, které definuje vlastní filtry a analyzátory. Protože struktura dokumentů je u všech jazyků stejná, je možné použít stejné mapování (totožné názvy polí a jejich datových typů a názvy použitých analyzátorů) pro všechny indexy za podmínky, že nastavené analyzátory budou ve všech indexech pojmenovány identicky. Výhodou je, že není

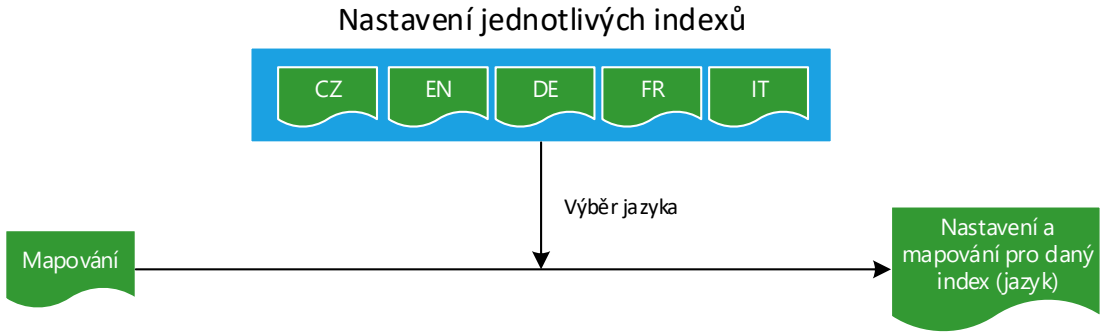
[1] https://www.jetbrains.com/idea/




Obrázek 5.1: Diagram tříd používaných pro načítání dokumentů systému MediaGist

potřeba udržovat mapování v pěti různých souborech, ale jen v jednom a případné úpravy je možné provést jen na jediném místě. Dále, díky stejnému mapování, je možné vytvořit dotaz, který je stejný pro všechny indexy.

Mapování je tedy uloženo v jediném souboru (`index-mapping.json`) a nastavení je pro každý index v jiném souboru (*jazyk*`-index.json`). Nastavení zahrnuje definici vlastních analyzátorů, token filtrů a počet replik primárních shards (samozřejmě mohou být přidány další nastavení, která Elasticsearch podporuje). Při vytvoření indexu je poté ze dvou souborů (viz obr. 5.2) vytvořen jeden textový řetězec ve formátu JSON, který je použit pro nastavení a mapování daného indexu.




Obrázek 5.2: Znázornění sestavení JSON řetězce pro mapování a nastavení indexu

### 5.2.1 Mapování typu indexu

Mapování typu indexu[2] (soubor `index-mapping.json`) se skládá z polí vytvořených podle struktury XML souboru popsaného v části 4.1.1. Každé pole je zpracováváno a ukládáno více způsoby. V ukázce kódu 5.1 je mapování titulku clusteru. V tomto případě je titulek indexován (zpracováván) třikrát, jednou bez jakéhokoliv zpracování, dále je zpracován nově definovaným analyzátorem `custom_analyzer` (viz část 5.2.2) a navíc jsou z titulku vytvořeny tzv. `n-gramy` za pomoci analyzátoru pojmenovaného `shingle_analyzer`.

N-gram je sekvence $n$ po sobě jdoucích slov nebo znaků (n-gramy mohou být slovní nebo znakové) [3]. Například z věty „Jakub snědl rybu“, pro $n$ rovno dvěma, vzniknou dva slovní n-gramy „Jakub snědl“ a „snědl rybu“. Pro $n$ rovné dvěma se n-gramy nazývají bigramy.

```
"properties": {
   "title":{
     "type": "text",
     "fields": {
         "analyzed":{
           "type": "text",
           "analyzer": "custom_analyzer"
          },
         "shingle":{
           "type": "text",
           "analyzer": "shingle_analyzer"}
   }},
   ...
}
```

Ukázka kódu 5.1: Část mapování titulku clusteru

Titulek clusteru je ukládán bez předzpracování, aby bylo možné vyhledat cluster s titulkem obsahující stop slova. Pokud by titulek obsahoval pouze stop slova, např. „Být, či nebýt ?“, nebylo by ho možné nikdy vyhledat, protože stop slova by byla při předzpracování odstraněna a vůbec by se nedostala do indexu.

N-gramy jsou indexovány z toho důvodu, aby zachytily kontext (větší množství informací) mezi slovy. Pokud jsou slova použita společně ve spojení, tak zpravidla vyjadřují myšlenku nebo význam mnohem přesněji nebo smysluplněji než samotná izolovaná slova.

Protože každý cluster obsahuje články, ze kterých byl vytvořen, jsou tyto články mapovány jako tzv. `nested objects` (vnořené objekty). Každý vnořený objekt je mapován jako samostatný dokument, který je svázán se svým nadřazeným dokumentem. Vnořený objekt se mapuje téměř stejně jako ostatní pole. Pole vnořeného

[2] V tomto případě je ekvivalentní označení mapování indexu, protože každý index obsahuje pouze jeden typ.

objektu je pojmenováno a označeno jako `nested`, což lze vidět v ukázce kódu 5.2. Podrobně jsou vnořené objekty a jejich vlastnosti popsané v [6] a [8].

```
"items":{
    "type": "nested",
    "properties": {
      "title":{
       "type": "text",
        "fields": {...}
      },
      "description" :{
        "type": "text",
        "analyzer": "custom_analyzer",
         "fields": {...}
      }}
},...
```

Ukázka kódu 5.2: Zjednodušená ukázka mapování vnořených objektů

### 5.2.2 Nastavení indexů

Jak již bylo zmíněno, index pro každý jazyk obsahuje vlastní nastavení (soubor JSON). V nastavení jsou definovány filtry tokenů a analyzátory, jež využívají tyto filtry. Mezi filtry tokenů jsou filtry pro odstraňování stop slov, filtr pro vytváření n-gramů a stemmer. N-gramy jsou vytvářeny ze dvou až čtyř slov (bigramy, trigramy a čtyřgramy). Analyzátory jsou definovány z jednoho tokenizéru a několika filtrů, přes které zpracovávaný text (dokument) postupně při indexaci prochází. Všechny analyzátory napříč jazyky mají stejný název, aby mohlo být použito jednotné mapování.

Nejdůležitějším analyzátorem je `custom_analyzer`, jehož definici pro francouzský index je možné vidět v ukázce kódu 5.3. Pro tokenizaci je použit `ICU Tokenizer` [23], tokeny jsou poté normalizovány za pomoci `ICU Normalizátoru` (proces normalizace je podrobněji popsán v [23]). Následně je použit filtr `elision`, který odstraňuje pro francouzštinu typickou elizi, např. z tokenu „l'avion“ (letadlo) vznikne použitím filtru `elision` pouze „avion“. Dále jsou vyjmuta francouzská stop slova, aplikován francouzský stemmer a odstraněna diakritika (filtr `icu_folding`).

U ostatních jazyků je struktura `custom_analyzer` analyzátoru podobná, tj. jsou aplikovány stejné typy filtrů (stemmer, stop slova, normalizace) určené pro daný jazyk, případně jsou použity doplňující filtry stejně jako filtr `elision` pro francouzštinu.

```
"custom_analyzer":{
  "type": "custom",
  "tokenizer": "icu_tokenizer",
  "filter": [
    "icu_normalizer",
    "elision",
    "french_stop",
    "french_stemmer",
    "icu_folding"]
}
```

Ukázka kódu 5.3: Definice `custom_analyzer` analyzátoru pro francouzštinu

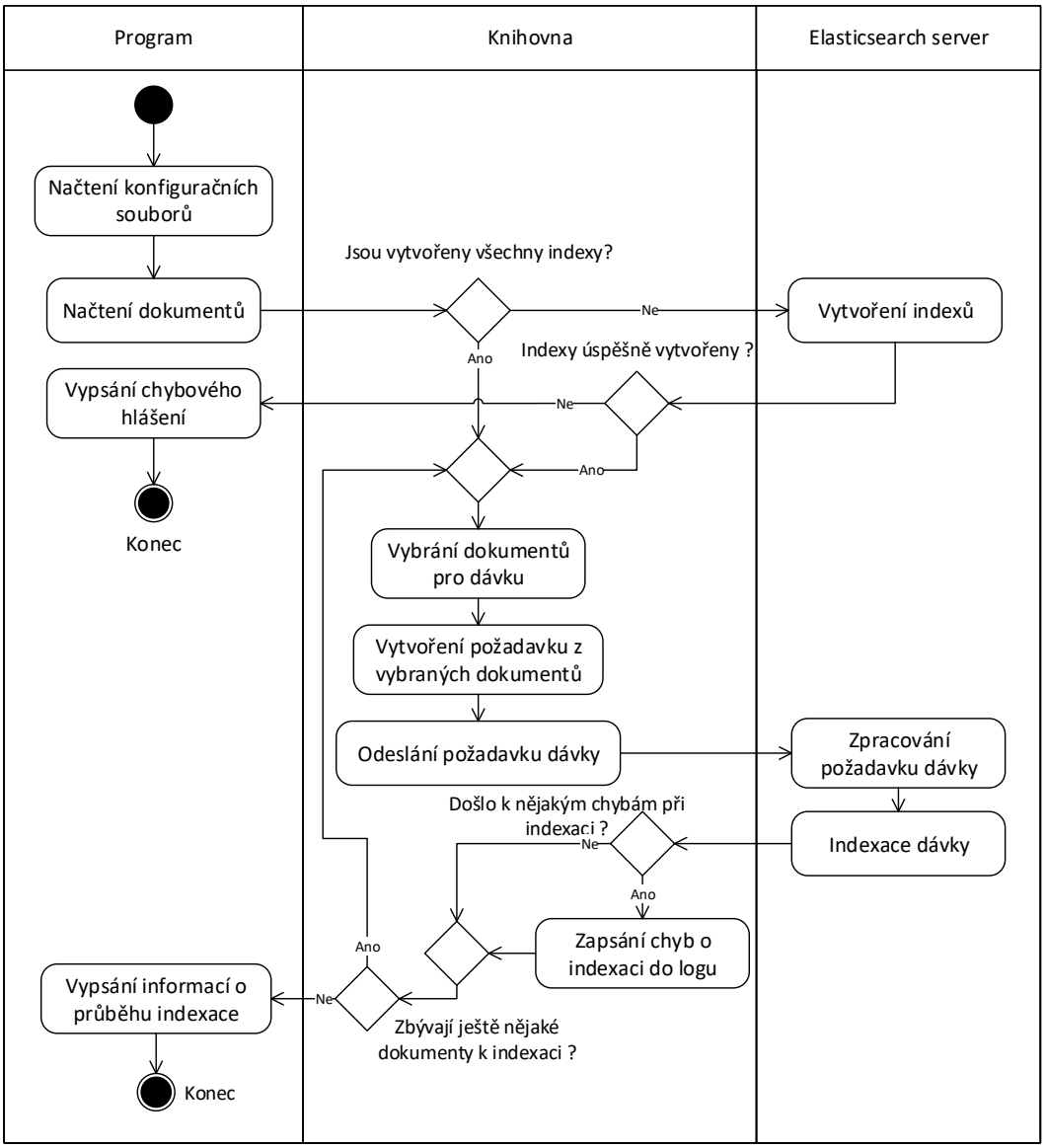
## 5.3 Indexace dokumentů

Pokud jsou dokumenty načtené a jednotlivé indexy vytvořeny, je možné za pomoci instance třídy `ElasticIndexer` implementující rozhraní `Indexer` provést samotnou indexaci. Rozhraní deklaruje dvě metody pro indexaci, pomocí nichž je možné indexovat dokumenty po jednom, a nebo použít dávkovou indexaci. V případě indexace více dokumentů by měla být použita dávka, protože velmi výrazně snižuje dobu indexace [6]. Pro správu indexů (vytváření a mazání) obsahuje knihovna rozhraní `IndicesAdmin`, které deklaruje potřebné metody. Tyto metody jsou implementovány ve třídě `ElasticIndicesAdmin`.

Velikost dávky je možné nastavit v konstruktoru třídy `ElasticIndexer`, standardně je velikost dávky (počet dokumentů, které budou najednou odeslány na server Elasticsearch) nastavena na 500. Na obr. 5.3 je zobrazen diagram aktivit, který popisuje postup při indexaci a obr. 5.4 obsahuje diagram tříd používaných při vytváření indexů a indexaci.

### 5.3.1 Připojení k serveru Elasticsearch

Elasticsearch poskytuje v jazyce JAVA oficiální API. Důležitou částí z tohoto API je rozhraní `Client` z balíku `org.elasticsearch.client`, které deklaruje všechny potřebné metody pro komunikaci s Elasticsearch serverem. Aby bylo možné využívat Elasticsearch API, je potřeba získat instanci objektu, jehož třída toto rozhraní implementuje. Pro vytvoření takovéto instance je potřeba znát adresu serveru, jméno clusteru a port, na kterém je instance Elasticsearch dostupná. Tyto údaje jsou standardně uloženy v konfiguračním souboru `config.properties`, jehož příklad je možné vidět v ukázce kódu 5.4.

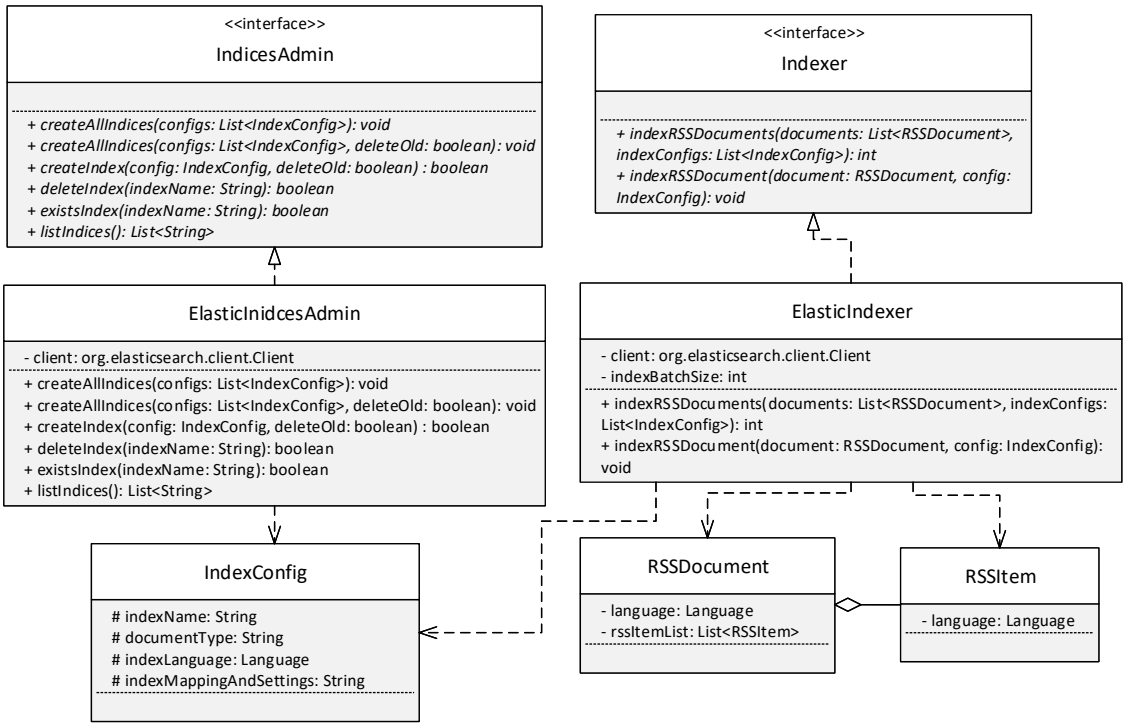



Obrázek 5.3: Diagram aktivit znázorňující aktivity při indexaci dokumentů

```
cluster.name=dp-cluster
host=localhost
port=9300
```

Ukázka kódu 5.4: Standardní konfigurační soubor `config.properties` pro připojení k Elasticsearch serveru

Knihovna obsahuje třídu `IndexPropertiesLoader`, která dokáže tento nebo jiný konfigurační soubor se stejnou strukturou načíst do objektu `Properties`. Objekt je poté předán statické metodě třídy `ClientFactory` a ta zajistí vytvoření instance




Obrázek 5.4: Diagram tříd používaných pro indexaci a správu

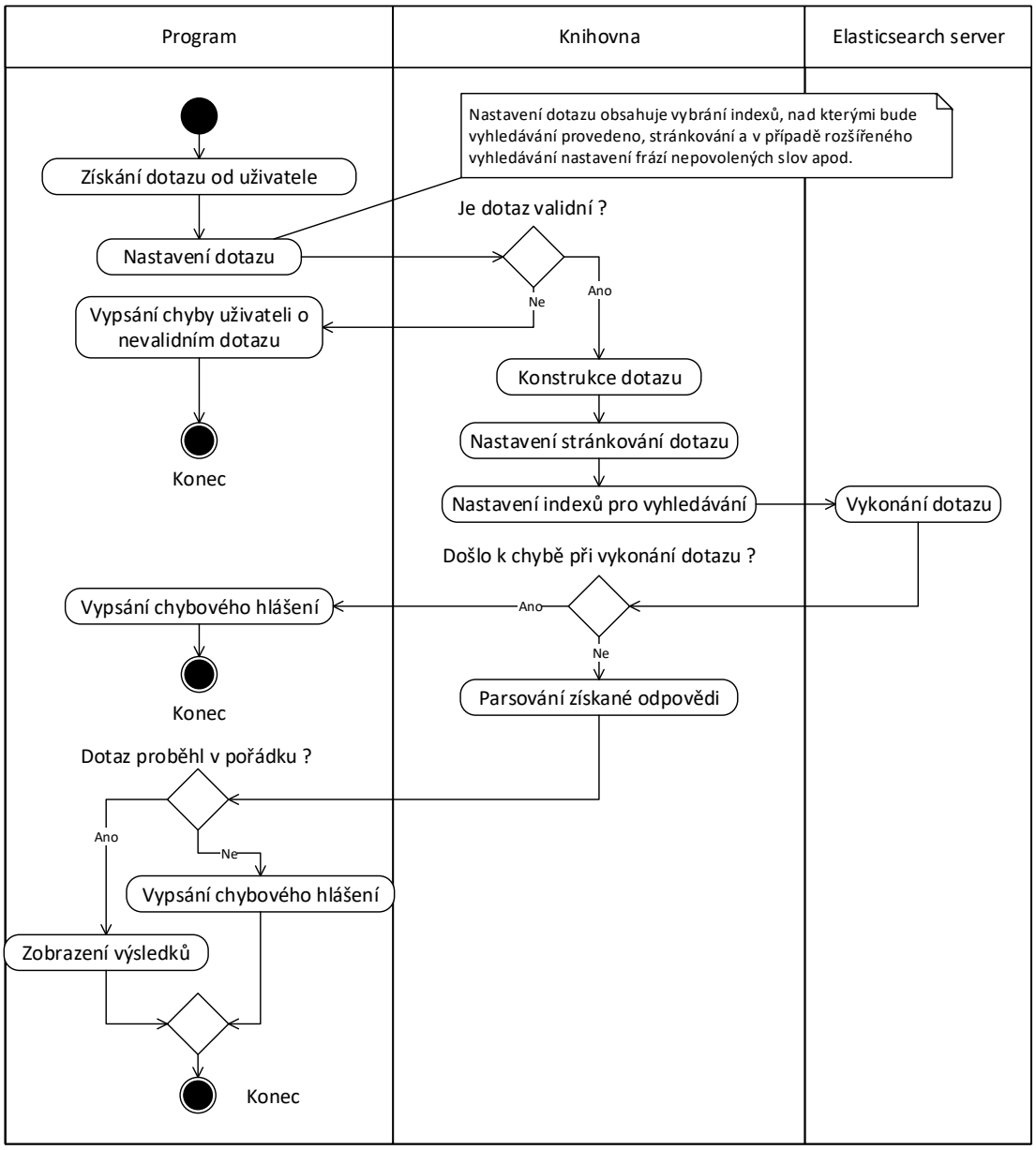
objektu implementujícího rozhraní `Client`. `Client` je zpravidla předáván v konstruktoru při vytváření objektů knihovny. Výhodou umístění údajů pro připojení do konfiguračního souboru je, že při jejich změně není potřeba měnit zdrojové soubory.

## 5.4 Vyhledávání dokumentů

Implementovaná funkcionalita vyhledávání poskytuje čtyři funkce (metody pro vyhledávání) deklarované v rozhraní `SearchService`. Na obr. B.16 v příloze B je zobrazen diagram tříd, které jsou používány při vyhledávání. Obrázek 5.5 obsahuje diagram aktivit, který zobrazuje obecný (zahrnuje jednoduché i rozšířené vyhledávání) postup při vyhledávání. Deklarované metody jsou určeny pro zvládnutí těchto úkonů:

- Jednoduché textové vyhledávání dokumentů.
- Pokročilé textové vyhledávání dokumentů (zahrnuje upřesnění dotazu v podobě frází, nepovolených slov, jazyka apod.).

- Našeptávání (angl. „autocomplete“ nebo „search as you type“) možných relevantních výsledků při zadávání dotazu pro jednoduché textové vyhledávání.
- Získání dokumentu na základě jeho ID.




Obrázek 5.5: Diagram aktivit znázorňující obecný postup při vyhledávání

### 5.4.1 Rozpoznání jazyka

Knihovna dovoluje také automatické rozpoznání jazyka zadaného textu, což je implementováno využitím knihovny `Apache Tika`, jenž automatické rozpoznání jazyka obsahuje [18]. Zamýšlené použití je takové, že program používající knihovnu může nejprve rozpoznat jazyk dotazu a na základě rozpoznaného jazyka vyhledávat jen v určitém indexu, uživatel tím získá výsledky pouze v rozpoznaném jazyce. Metoda pro rozpoznání jazyka je deklarována rozhraním `LangDetectionService` a implementována ve třídě `LangDetectionServiceImpl`.

### 5.4.2 Jednoduché vyhledávání

Jednoduché vyhledávání je implementováno pomocí dotazu typu `Query String Query` z API Elasticsearch a je prováděno nad všemi indexovanými poli dokumentů. Tento dotaz podporuje rozšířenou syntaxi a při vytváření dotazu je možné použít booleovské operátory `AND, OR` a `NOT`, závorky pro vynucení precedence a frázové vyhledávání pomocí uvozovek. Elasticsearch API samo provede analýzu dotazu a zkontroluje, zda je zadaný dotaz validní.

Validním dotazem tak může být např. `("plzen vyhrála") AND fotbal`. Pro tento dotaz by měly být nalezeny dokumenty obsahující frázi „plzen vyhrála“, a zároveň by se v těchto dokumentech mělo vyskytnout slovo „fotbal“. Samotný dotaz typu `Query String Query` podporuje i další funkcionalitu (žolíkové znaky, regulární výrazy aj., podrobněji viz [8]), ale většina je velmi náročných na výkon a při reálném nasazení by mohly velmi výrazně zpomalit dobu odezvy vyhledávání, a proto je v aplikaci možné použít pouze výše jmenované.

Třída `SearchServiceImpl` implementuje rozhraní `SearchService` (tedy veškerou funkcionalitu vyhledávání). Při volání jejích metod pro vyhledávání je možné vybrat indexy, nad kterými bude probíhat vyhledávání nebo nastavení stránkování (počet vrácených výsledků a pozice podle skóre, od které bude daný počet vrácen).

### 5.4.3 Rozšířené vyhledávání

Implementované rozšířené vyhledávání umožňuje vyhledávat podle kritérií popsaných v části 4.4.2. Při volání metody `advancedSearch()` (třídy `SearchServiceImpl`) obsahující funkcionalitu rozšířeného vyhledávání je jako parametr předán objekt se všemi požadovanými kritérii a indexy, ve kterých má být vyhledáváno. Tato omezení jsou poté spolu s dotazem nastavena v metodě `prepareSearch`, viz ukázka kódu 5.5.

```
//kriteria omezeni a nastaveni dotazu
AdvancedSearchOptions opt = ...
//vytvoreni dotazu
QueryBuilder query = createQuery(opt);
//nastaveni dotazu
SearchRequestBuilder bld = client.prepareSearch(opt.getIndices())
    .setFrom(opt.getFrom())
    .setSize(opt.getSize())
    .setTypes(opt.getTypes())
    .setQuery(query);
//provedeni dotazu a ziskani odpovedi
SearchResponse response = builder.get();
```

Ukázka kódu 5.5: Postup při vytvoření, nastavení dotazu a získání odpovědi

Dotaz u rozšířeného vyhledávání se skládá z dotazů typu `Bool Query`, `Multi Match Query` a `Nested Query`. `Nested Query` je použit pro vyhledávání nad vnořenými objekty dokumentu (články clusteru), viz část 5.2.1.

Knihovna také obsahuje metodu pro získání konkrétního dokumentu podle jeho ID. Ve webové aplikacie (část 5.5) je použita pro získání dokumentu po kliknutí na výsledek vyhledávání. Poslední poskytovanou funkcionalitou je podpora našeptávání, která hledá frázový prefix zadaného dotazu. Vyhledávání v tomto případě probíhá pouze nad titulky clusterů a vrácené výsledky obsahují zvýrazněné části, jenž odpovídají zadanému dotazu.

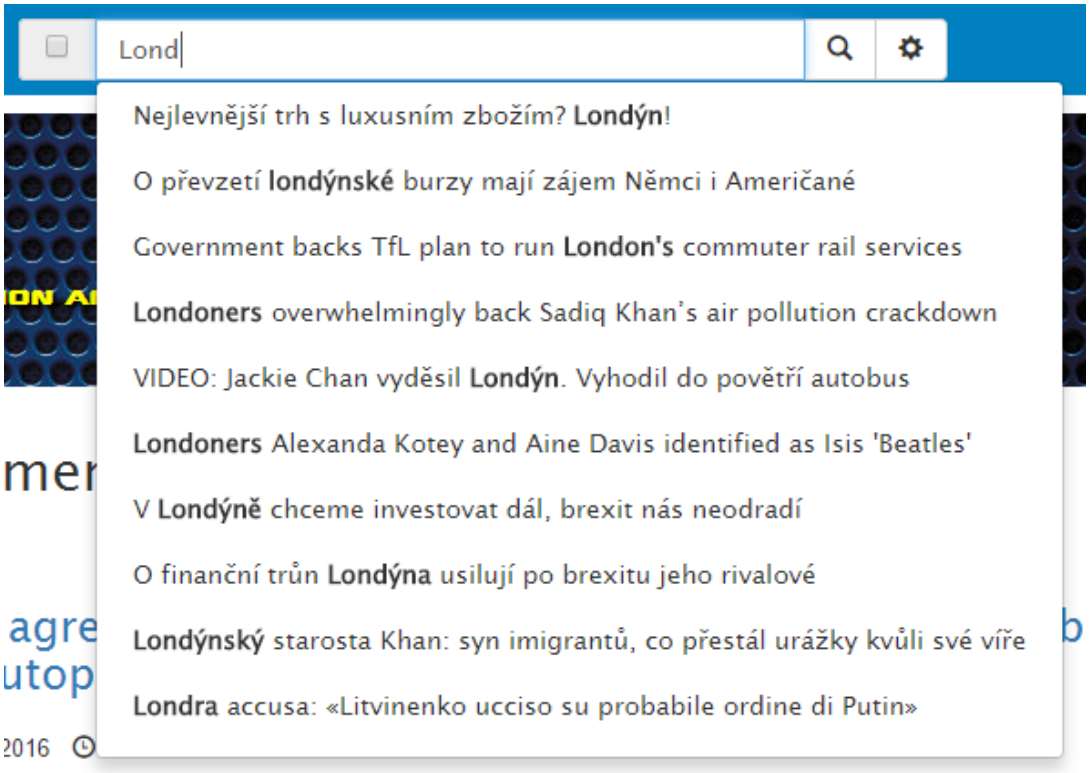
## 5.5 Implementace webové aplikace

Webová aplikace byla implementována za pomoci frameworku `spring`. Při generování jednotlivých webových stránek je používána technologie `JSP`. Požadavky a návod pro spuštění webové aplikace jsou popsány v příloze A. Cílem webové aplikace je ukázat funkcionalitu vyhledávání, kterou vytvořená knihovna poskytuje.

Všechny webové požadavky jsou zpracovávány ve třídě `SearchController`. Grafický vzhled stránek a rozvržení pokročilého vyhledávání je vytvořen podle návrhu z části 4.5.1. Funkcionalita veškerého vyhledávání je zapouzdřena do třídy implementující rozhraní `SearchManager`, které vytváří nastavení jednotlivých dotazů uživatele (stránkování, indexy, nad kterými bude probíhat vyhledávání apod.) a volá metody knihovny.

Rozšířené vyhledávání obsahuje veškerou funkcionalitu popsanou v části 4.2. Jednoduché vyhledávání navíc umožňuje rozpoznat jazyk dotazu. Pokud uživatel použije tuto funkcionalitu (vybere zaškrtávací pole u formuláře jednoduchého vyhledávání, viz obr. 5.6), pokusí se program rozpoznat jazyk dotazu a v případě, že se podaří rozpoznat jeden z pěti podporovaných jazyků (čeština, angličtina, francouz-

ština, němčina nebo italština) pak je vyhledávání provedeno pouze nad indexem rozpoznaného jazyka.




Obrázek 5.6: Ukázka formuláře pro jednoduché vyhledávání a zobrazení našeptávání

Konfigurace a vytvoření tzv. `bean` objektů pro framework spring je umístěna ve třídě `SearchConfiguration`.

### 5.5.1 Našeptávání výsledků

Při zadávání dotazu pro jednoduché vyhledávání jsou uživateli nabízeny možné relevantní výsledky, viz obr 5.6. Tato funkcionalita se nazývá našeptávání. Dotaz pro získání možných relevantních výsledků probíhá vždy jen nad titulky dokumentů (clusterů), proto pro některé dotazy nemusí být zobrazeno našeptávání, ale mohou být nalezeny výsledky. Zobrazeno je vždy nula až deset výsledků.

Našeptávání je implementováno pomocí skriptu v jazyce `JavaScript` (JS). Vždy když uživatel upraví dotaz ve formuláři, je odeslán asynchronní požadavek na webový server, který jej zpracuje. Webový server zavolá funkci knihovny a z vrácených výsledků vygeneruje JSON. Tento JSON je vrácen jako odpověď, pomocí JS zpracován a zobrazen uživateli.

## 5.6 Implementace klienta

Klient pro indexaci je vytvořen jako konzolová aplikace v jazyce JAVA. Požadavky a návod na ovládání a spuštění klienta jsou dostupné v příloze A. Klient umožňuje získat seznam všech indexů v běžícím clusteru. Při používání klienta je možné specifikovat cestu k souboru s konfigurací (jméno clusteru, adresa a port serveru, viz část 5.3.1) pro připojení k běžícímu clusteru. Pokud konfigurační soubor není specifikován je použit standardní konfigurační soubor s konfigurací, kterou je možné vidět v ukázce kódu 5.4.

Dále klient umožňuje mazat index dle zadaného jména a indexovat soubory XML (popsané v části 4.1.1) systému MediaGist. Indexovat je možné samotný soubor nebo složku obsahující více souborů. V případě indexace celé složky jsou nejprve získány rekurzivně (včetně podsložek) všechny soubory s příponou XML a následně jsou tyto soubory (s podporovanou strukturou) postupně indexovány. Do paměti je vždy načtena jen část souborů a tato část je indexována, po jejím dokončení jsou načteny další soubory. Protože v případě, kdy by celková velikost souborů pro indexaci byla příliš velká (v řádu gigabytů), nemusely by se všechny soubory najednou vejít do paměti a program by mohl skončit chybou.

# 6 Testování

Posledním cílem této diplomové práce je otestování úspěšnosti vyhledávání. Nejprve bude popsáno automatizované testování za pomoci testovacích dat určených k ověřování kvality IR systémů. Pro porovnání a vyhodnocení získaných výsledků bude použita `MAP` míra, která je popsána v části 2.8.3. Balíček dat (`CLEF AdHoc - News 2004-2008`) pro testování pochází z `Cross-Language Evaluation Forum` (`CLEF`) konferencí z let 2004 až 2008, kde hlavním úkolem byla evaluace IR systémů [11].

Dále bude vyhledávání otestováno uživateli za pomoci webové aplikace. Uživatelé budou mít zadané úkoly, tj. dokumenty nebo informace, které budou muset nalézt a bude sledováno, zda se jim tyto úkoly daří plnit. Nakonec budou získané výsledky zhodnoceny a porovnány. Na základě získaných výsledků z obou typů testování bude případně možné navrhnout úpravy mapování a nastavení indexu pro zlepšení kvality vyhledávání.

## 6.1 Prostředí pro testování

Testování bude provedeno na notebooku `Dell Inspiron 15 (7000)` v této konfiguraci:

- Procesor: `Intel Core i7 6700HQ Skylake 2,6 GHz` (4 fyzická jádra, 8 virtuálních s techonologií HyperThreading)
- RAM: `16 GB DDR4`
- Grafická karta: `NVIDIA GeForce GTX 960M`
- HDD: `TOSHIBA MQ01ABD100` - 1 TB
- SSD: `SanDisk Z400s M.2 2280` - 128 GB
- OS: `Windows 10 Home 64-bit`

Elasticsearch server i webový server budou společně spuštěny na uvedeném zařízení na SSD disku (pokud není řečeno jinak). První část testování (nad balíčkem CLEF dat) bude také zároveň spouštěna na uvedeném zařízení. Při uživatelském testování budou uživatelé přistupovat na webové stránky ze svých zařízení a prohlížečů.

## 6.2 Automatizované testování

Pro automatizované testování jsou potřebná data, ke kterým jsou dostupné dotazy s označenými relevantními dokumenty. Vedoucím práce byla poskytnuta reálná data (clustery) systému MediaGist, jenž vznikla v letech 2015 a 2016 a obsahují 23000 clusterů a jejich velikost na disku činí 810 MB. Testovány budou tři jazyky (resp. indexy), které mají v poskytnutých datech největší zastoupení, tj. čeština, angličtina a francouzština.

Protože vytvoření dotazů a k nim relevantních dokumentů určených k testování z poskytnutých dat by bylo časově příliš náročné, budou použita testovací data balíčku `CLEF AdHoc - News 2004-2008`, která obsahují dokumenty k indexaci, dotazy a relevantní dokumenty k dotazům.

### 6.2.1 Data určená k testování

Jak bylo zmíněno výše, data určená k testování pochází z balíčku `CLEF AdHoc - News 2004-2008`. Pro český jazyk se jedná o 81 735 novinových článků z roku 2002 o celkové velikosti 178 MB, v CLEF balíčku jsou pojmenované jako `Mladna frontaDnes`[1] a `Lidove Noviny`. Anglických článků bylo použito celkem 135 153, jejich velikost byla 434 MB a v CLEF balíčku jsou pojmenované jako `Los Angeles Times 2002`. Testovací data pro francouzštinu se skládala ze čtyř částí v CLEF balíčku pojmenovaných `Le Monde 1994`, `Le Monde 1995`, `SDA French 1994` a `SDA French 1995` obsahujících 177 452 dokumentů o celkové velikosti 487 MB.

Vyhledávané informace (topics) pro český a anglický jazyk pochází z CLEF úlohy `CLEF AdHoc 2007` a pro francouzštinu z CLEF úlohy `CLEF AdHoc 2006`. Každá hledaná informace je popsána třemi částmi, tj. `title`, `description` a `narrative`, viz ukázka kódu 6.1. Z těchto tří částí je možné sestavit dotaz pro vyhledávač.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<topics>
 <topic lang="cs">
  <identifier>10.2452/401-AH</identifier>
  <title>Inflace Eura</title>
  <description>Najděte dokumenty o růstech cen po zavedení Eura.</description>
  <narrative>Relevantní jsou jakékoliv dokumenty, které poskytují informace o růstu cen v jakékoliv zemi.</
      narrative>
 </topic>
```

Ukázka kódu 6.1: Ukázka popisu hledané informace (topic) pro český jazyk

[1] V názvu dat ve slově `Mladna` je pravděpodobně překlep, ale v CLEF balíčku jsou tato data opravdu takto pojmenována.

Nevýhodou těchto dat je, že jejich struktura neodpovídá struktuře dat systému MediaGist. Proto bude každý článek indexován jako samostatný cluster, který neobsahuje žádné vnořené dokumenty.

### 6.2.2 Průběh automatizovaného testování

Vyhodnocení vyhledávání, tzn. výpočet MAP míry a interpolovaných přesností (popsaných v části 2.8.3) byl proveden programem `trec_eval.8.1`[2], který jako vstup přijímá dva soubory. První soubor obsahuje seznam relevantních dokumentů k jednotlivým dotazům (hledaným informacím) a druhý soubor obsahuje získané výsledky pro každý dotaz (hledanou informaci). Výstup programu lze vidět na obr. 6.1. Výstup obsahuje, kromě jiných ukazatelů, také interpolovanou přesnost pro hodnoty 0 až 1 s krokem 0.1, ze kterých je vytvářen přesnost/úplnost graf.

```
num_q             all     50
num_ret           all     50000
num_rel           all     762
num_rel_ret       all     657
map               all     0.3204
gm_ap             all     0.1935
R-prec            all     0.3337
bpref             all     0.2993
recip_rank        all     0.5674
ircl_prn.0.00     all     0.6116
ircl_prn.0.10     all     0.5409
ircl_prn.0.20     all     0.4903
ircl_prn.0.30     all     0.4422
ircl_prn.0.40     all     0.4090
ircl_prn.0.50     all     0.3565
ircl_prn.0.60     all     0.2845
ircl_prn.0.70     all     0.2462
ircl_prn.0.80     all     0.1865
ircl_prn.0.90     all     0.0962
ircl_prn.1.00     all     0.0697
```

Obrázek 6.1: Ukázka výstupu programu trec_eval.8.1

Testování probíhalo tak, že nejprve bylo vyhodnoceno standardní nastavení indexů (MAP míra a interpolované přesnosti) pro dokumenty a hledané informace (dotazy) popsané v předchozí části 6.2.1. Toto nastavení pro jednotlivé indexy lze vidět v příloze C a ukázkách kódu C.1, C.2 a C.3.

Dále byla postupně upravována jednotlivá nastavení indexů, tj. konfigurace jednotlivých filtrů tokenů a analyzátorů. Po každé úpravě proběhlo stejné vyhodnocení (nad stejnými daty i dotazy).

Každá hledaná informace byla reprezentována textovým řetězcem. Řetězec byl vytvořen spojením elementů `title` a `description` (viz ukázka kódu 6.1) z popisu

[2]Program trec_eval.8.1 je volně dostupný z webové adresy `http://trec.nist.gov/trec_eval/`.

hledané informace. Následně byla zavolána metoda knihovny pro jednoduché vyhledávání, které byl jako dotaz zadán vytvořený řetězec. Programem trec_eval.8.1 bylo vždy vyhodnocováno prvních tisíc získaných výsledků. Provedené úpravy a získané výsledky jsou popsány v následující části 6.2.3.

### 6.2.3 Výsledky automatizovaného testování

Každé vyhodnocení úpravy nastavení indexu bude porovnáno s vyhodnocením standardního nastavení indexu. Sledovanými hodnotami budou velikosti jednotlivých indexů, MAP míra a interpolované přesnosti pro hodnoty úplnosti 0 až 1 s krokem 0.1.

U jednotlivých případů (úprav nastavení) budou uvedeny tabulky se získanými hodnotami a standardními hodnotami (původní nastavení indexu). Interpolované přesnosti nebudou uvedeny v tabulce, ale vykresleny do přesnost/úplnost grafu, kde `plné` křivky reprezentují výsledky získané ze standardního nastavení indexů a `čárkované` křivky představují hodnoty získané úpravou nastavení indexu.

Všechna naměřená data jsou uložena v souboru **namerena-data.xlsx**, který je dostupný na přiloženém DVD. Umístění tohoto souboru na DVD je popsané v příloze D. Na základě porovnání výsledků získaných úpravou nastavení indexu bude možné navrhnout úpravy nastavení indexů pro zlepšení kvality vyhledávání.

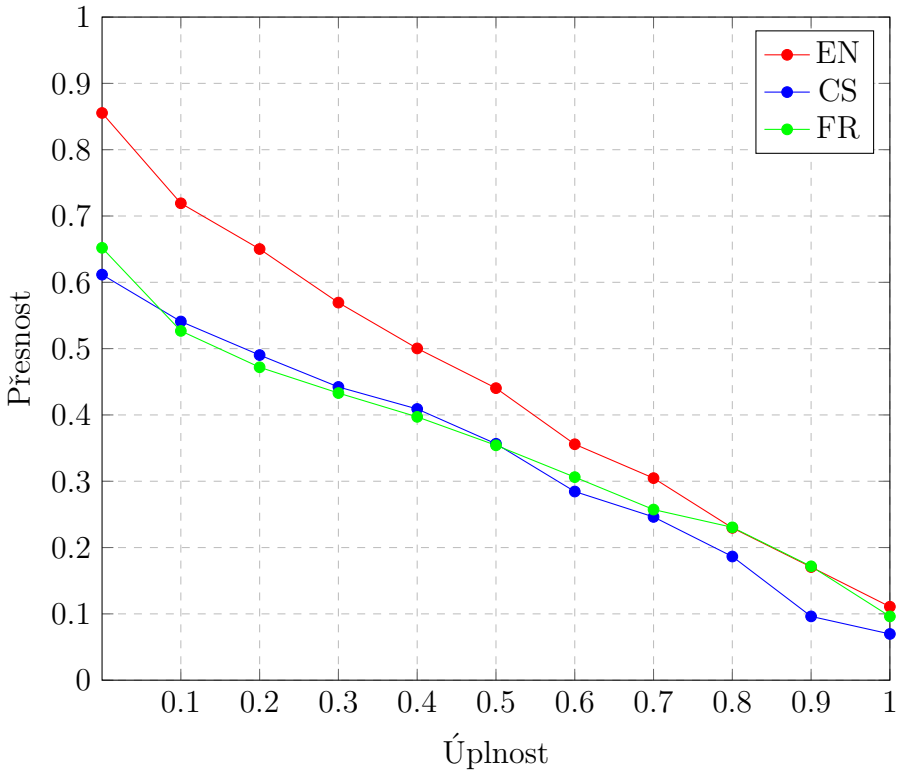
V tabulkách 6.1 a 6.2 jsou uvedeny naměřené hodnoty standardního nastavení indexu, se kterými budou porovnávány ostatní naměřené hodnoty. V tabulce 6.1 ve sloupci jazyk jsou uvedeny zkratky pro jazyky jednotlivých indexů, EN - angličtina, CS - čeština, FR - francouzština. Na obr. 6.2 je zobrazen přesnost/úplnost graf pro standardní nastavení indexů. Zkratky použité v obr. 6.2 odpovídají zkratkám jazyků z tabulky 6.1 a 6.2 a jsou použity i v ostatních tabulkách a obrázcích.

| **Jazyk** | **Velikost indexu** |
|---|---|
| EN | 2.64 GB |
| CS | 1.37 GB |
| FR | 3.21 GB |

Tabulka 6.1: Velikosti indexů pro standardní nastavení indexů

| **Jazyk** | **MAP** |
|---|---|
| EN | 0.4264 |
| CS | 0.3204 |
| FR | 0.3384 |

Tabulka 6.2: Získané hodnoty MAP míry pro standardní nastavení indexů




Obrázek 6.2: Přesnost/úplnost graf testovacích dat pro standardní nastavení indexů

V následujícím číselném výčtu jsou v tabulkách uvedeny získané hodnoty pro jednotlivé případy testování. Každá položka výčtu představuje jeden testovací případ, pro který byl upraven index. Příslušná úprava standardního nastavení indexu je v každé položce popsána. Význam upravených parametrů filtrů a analyzátorů (např. název stemmeru apod.) je podrobně popsán v dokumentaci Elasticsearch [8]. Protože přesnost/úplnost grafy pro všechny případy by v textu zabraly příliš mnoho místa, jsou tyto grafy umístěny do přílohy B.

U položek výčtu jsou uvedeny dvě tabulky, tj. tabulka s velikostmi indexů a tabulka se spočtenými hodnotami MAP míry. V obou tabulkách jsou uvedeny hodnoty pro standardní nastavení indexu (sloupečky `Velikost indexu` a `MAP`) a hodnota získaná po příslušné úpravě indexu a zopakování testu (sloupečky `Velikost indexu po úpravě` a `MAP po úpravě`). Dále rozdíl a poměr mezi hodnotou pro standardní nastavení a hodnotou pro nastavení po úpravě (sloupce `Rozdíl` a `Poměr`). Do přílohy B jsou umístěny tabulky pro případy, u kterých po provedené úpravě nastavení indexu nedošlo k výrazným změnám velikostí indexů.

Pro sloupeček `Rozdíl` u velikostí indexů platí, že čím větší kladné číslo, tím lepší výsledek (je snaha index zmenšit) a naopak záporné číslo znamená zhoršení výsledků (index se zvětšil). Sloupec `Poměr` udává kolikrát je index pro standardní nastavení

větší než index po provedené úpravě, tzn. ke zlepšení dojde, pokud je poměr větší než jedna (naopak ke zhoršení, pokud je hodnota menší než jedna).

U MAP míry vyšší hodnota po úpravě znamená zlepšení. Pokud tedy v tabulce s hodnotami MAP míry ve sloupci `Rozdíl` bude záporné číslo, došlo úpravou nastavení ke zlepšení. Sloupec `Poměr` udává kolikrát úprava indexu zhoršila MAP míru, tj. pokud je číslo větší než jedna, došlo ke zhoršení (pokud je menší než jedna, došlo ke zlepšení).

1. Úprava nastavení indexu č. 1: ve filtru `shingle_filter` generujícího n-gramy byl přenastaven parametr `output_unigrams` z hodnoty `false` na hodnotu `true`. Tato úprava způsobí, že spolu s n-gramy budou generovány i unigramy. Získané hodnoty jsou uvedeny v tabulkách 6.3 a 6.4 a na obr. B.1 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.78 GB | −0.14 | 0.95 |
| CS | 1.37 GB | 1.45 GB | −0.08 | 0.94 |
| FR | 3.21 GB | 3.53 GB | −0.32 | 0.91 |

Tabulka 6.3: Velikosti indexů po úpravě č. 1

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4120 | 0.0144 | 1.0349 |
| CS | 0.3204 | 0.2934 | 0.027 | 1.0920 |
| FR | 0.3384 | 0.3145 | 0.0239 | 1.0759 |

Tabulka 6.4: Získané hodnoty MAP míry po úpravě č. 1

2. Úprava nastavení indexu č. 2: filtr `shingle_filter` byl upraven tak, aby generoval pouze bigramy a trigramy (standardně generuje ještě čtyřgramy). Získané hodnoty jsou uvedeny v tabulkách 6.5 a 6.6 a na obr. B.2 je zobrazen přesnost/úplnost graf.

| Jazyk | Velikost indexu | Velikost indexu po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 2.64 GB | 1.55 GB | 1.09 | 1.70 |
| CS | 1.37 GB | 0.86 GB | 0.50 | 1.58 |
| FR | 3.21 GB | 1.94 GB | 1.27 | 1.65 |

Tabulka 6.5: Velikosti indexů po úpravě č. 2

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4249 | 0.0015 | 1.0035 |
| CS | 0.3204 | 0.3224 | −0.002 | 0.9937 |
| FR | 0.3384 | 0.3413 | −0.0029 | 0.9915 |

Tabulka 6.6: Získané hodnoty MAP míry po úpravě č. 2

3. Úprava nastavení indexu č. 3: filtr `shingle_filter` generuje pouze bigramy. Získané hodnoty jsou uvedeny v tabulkách 6.7 a 6.8 a na obr. B.3 je zobrazen přesnost/úplnost graf.

| Jazyk | Velikost indexu | Velikost indexu po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 2.64 GB | 0.79 GB | 1.84 | 3.31 |
| CS | 1.37 GB | 0.46 GB | 0.91 | 2.95 |
| FR | 3.21 GB | 0.98 GB | 2.23 | 3.28 |

Tabulka 6.7: Velikosti indexů po úpravě č. 3

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4231 | 0.00033 | 1.0077 |
| CS | 0.3204 | 0.3228 | −0.0024 | 0.9925 |
| FR | 0.3384 | 0.3461 | −0.0077 | 0.9777 |

Tabulka 6.8: Získané hodnoty MAP míry po úpravě č. 3

4. Úprava nastavení indexu č. 4: filtr `shingle_filter` generuje pouze trigramy. Získané hodnoty jsou uvedeny v tabulkách 6.9 a 6.10 a na obr. B.4 je zobrazen přesnost/úplnost graf.

| Jazyk | Velikost indexu | Velikost indexu po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 2.64 GB | 1.16 GB | 1.48 | 2.28 |
| CS | 1.37 GB | 0.63 GB | 0.74 | 2.16 |
| FR | 3.21 GB | 1.45 GB | 1.76 | 2.21 |

Tabulka 6.9: Velikosti indexů po úpravě č. 4

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4318 | −0.0054 | 0.9874 |
| CS | 0.3204 | 0.3222 | −0.0018 | 0.9944 |
| FR | 0.3384 | 0.3372 | 0.0012 | 1.0035 |

Tabulka 6.10: Získané hodnoty MAP míry po úpravě č. 4

5. Úprava nastavení indexu č. 5: filtr `shingle_filter` generuje pouze čtyřgramy. Získané hodnoty jsou uvedeny v tabulkách 6.11 a 6.12 a na obr. B.5 je zobrazen přesnost/úplnost graf.

| Jazyk | Velikost indexu | Velikost indexu po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 2.64 GB | 1.48 GB | 1.16 | 1.78 |
| CS | 1.37 GB | 0.76 GB | 0.60 | 1.78 |
| FR | 3.21 GB | 1.89 GB | 1.32 | 1.70 |

Tabulka 6.11: Velikosti indexů po úpravě č. 5

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4305 | −0.0041 | 0.9904 |
| CS | 0.3204 | 0.3234 | −0.003 | 0.9907 |
| FR | 0.3384 | 0.3486 | −0.0102 | 0.9707 |

Tabulka 6.12: Získané hodnoty MAP míry po úpravě č. 5

6. Úprava nastavení indexu č. 6: filtr `shingle_filter` generuje pouze trigramy a čtyřgramy. Získané hodnoty jsou uvedeny v tabulkách 6.13 a 6.14 a na obr. B.6 je zobrazen přesnost/úplnost graf.

| Jazyk | Velikost indexu | Velikost indexu po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 2.64 GB | 2.23 GB | 0.41 | 1.18 |
| CS | 1.37 GB | 1.13 GB | 0.24 | 1.21 |
| FR | 3.21 GB | 2.87 GB | 0.34 | 1.12 |

Tabulka 6.13: Velikosti indexů po úpravě č. 6

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4324 | −0.006 | 0.9861 |
| CS | 0.3204 | 0.3194 | 0.001 | 1.0031 |
| FR | 0.3384 | 0.3498 | −0.0114 | 0.9674 |

Tabulka 6.14: Získané hodnoty MAP míry po úpravě č. 6

7. Úprava nastavení indexu č. 7: v analyzátoru `custom_analyzer` se oproti standardnímu nastavení nepoužívá filtr `icu_folding`, který odstraňuje diakritiku. Získané hodnoty jsou uvedeny v tabulkách B.1 a 6.15 a na obr. B.7 je zobrazen přesnost/úplnost graf.

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4237 | 0.0027 | 1.0063 |
| CS | 0.3204 | 0.3185 | 0.0019 | 1.0059 |
| FR | 0.3384 | 0.3413 | −0.0029 | 0.9915 |

Tabulka 6.15: Získané hodnoty MAP míry po úpravě č. 7

8. Úprava nastavení indexu č. 8: v analyzátoru `custom_analyzer` se oproti standardnímu nastavení nepoužívají filtry odstraňující stop slova. Získané hodnoty jsou uvedeny v tabulkách B.2 a 6.16 a na obr. B.8 je zobrazen přesnost/úplnost graf.

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4232 | 0.0032 | 1.0075 |
| CS | 0.3204 | 0.3237 | −0.0033 | 0.9898 |
| FR | 0.3384 | 0.3413 | −0.0029 | 0.9915 |

Tabulka 6.16: Získané hodnoty MAP míry po úpravě č. 8

9. Úprava nastavení indexu č. 9: v analyzátoru `custom_analyzer` se oproti standardnímu nastavení nepoužívají filtry pro stemming (není použit žádný stemmer). Získané hodnoty jsou uvedeny v tabulkách B.3 a 6.17 a na obr. B.9 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.3941 | 0.0323 | 1.0819 |
| CS | 0.3204 | 0.2214 | 0.099 | 1.4471 |
| FR | 0.3384 | 0.2997 | 0.0387 | 1.1291 |

Tabulka 6.17: Získané hodnoty MAP míry po úpravě č. 9

10. Úprava nastavení indexu č. 10: v analyzátoru `custom_analyzer` se oproti standardnímu nastavení nepoužívá filtr `icu_normalizer` pro normalizaci. Získané hodnoty jsou uvedeny v tabulkách B.4 a 6.18 a na obr. B.10 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4262 | 0.0002 | 1.0004 |
| CS | 0.3204 | 0.3292 | −0.0088 | 0.9732 |
| FR | 0.3384 | 0.3431 | −0.0047 | 0.9863 |

Tabulka 6.18: Získané hodnoty MAP míry po úpravě č. 10

11. Úprava nastavení indexu č. 11: v analyzátoru `custom_analyzer` se oproti standardnímu nastavení pro index francouzštiny nepoužívá filtr `elision` a pro index angličtiny se nepoužívá filtr `english_possessive_stemmer`. V indexu pro češtinu nebylo nic upraveno (výsledky nebudou uvedeny). Získané hodnoty jsou uvedeny v tabulkách B.5 a 6.19 a na obr. B.11 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4312 | −0.0048 | 0.9888 |
| FR | 0.3384 | 0.3262 | 0.0122 | 1.0374 |

Tabulka 6.19: Získané hodnoty MAP míry po úpravě č. 11

12. Úprava nastavení indexu č. 12: v indexu pro francouzštinu je použit stemmer `french`, v indexu pro angličtinu je použit stemmer `light_english`. V indexu pro češtinu nebylo nic upraveno a ve výsledcích tak nebude uveden. Získané hodnoty jsou uvedeny v tabulkách B.6 a 6.20 a na obr. B.12 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4314 | −0.0050 | 0.9884 |
| FR | 0.3384 | 0.3373 | 0.0011 | 1.0032 |

Tabulka 6.20: Získané hodnoty MAP míry po úpravě č. 12

13. Úprava nastavení indexu č. 13: v indexu pro francouzštinu je použit stemmer `minimal_french`, v indexu pro angličtinu je použit stemmer `minimal_english`. V indexu pro češtinu nebylo nic upraveno a ve výsledcích tak nebude uveden. Získané hodnoty jsou uvedeny v tabulkách 6.21 a B.7 a na obr. B.13 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4149 | 0.0115 | 1.0277 |
| FR | 0.3384 | 0.3343 | 0.0041 | 1.0122 |

Tabulka 6.21: Získané hodnoty MAP míry po úpravě č. 13

14. Úprava nastavení indexu č. 14: v indexu pro angličtinu je použit stemmer `porter2`. V indexech pro češtinu a francouzštinu nebylo nic upraveno a ve výsledcích tak nebudou uvedeny. Získané hodnoty jsou uvedeny v tabulkách B.8 a 6.22 a na obr. B.14 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0.4261 | 0.0003 | 1.0007 |

Tabulka 6.22: Získané hodnoty MAP míry po úpravě č. 14

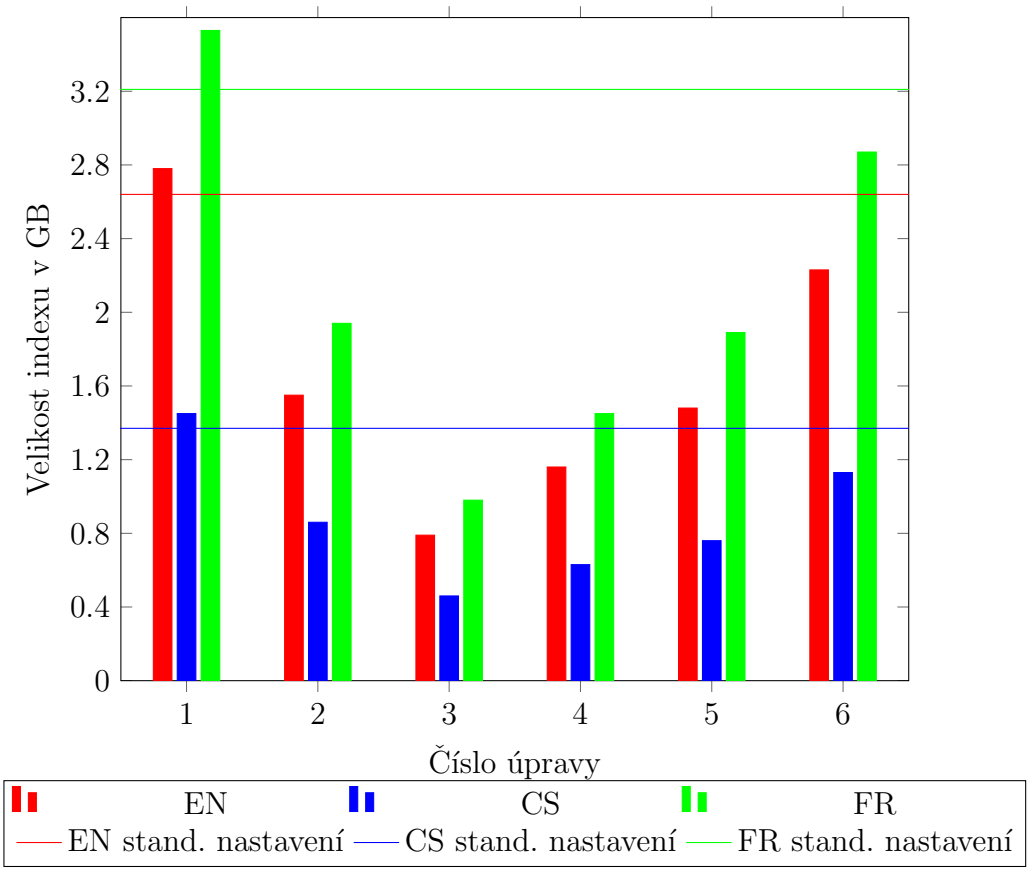
15. Úprava nastavení indexu č. 15: v indexu pro angličtinu je použit stemmer `lovins`. V indexech pro češtinu a francouzštinu nebylo nic upraveno a ve výsledcích tak nebudou uvedeny. Získané hodnoty jsou uvedeny v tabulkách B.9 a 6.23 a na obr. B.15 je zobrazen přesnost/úplnost graf.

| Jazyk | MAP | MAP po úpravě | Rozdíl | Poměr |
|---|---|---|---|---|
| EN | 0.4264 | 0.4128 | 0.0136 | 1.0329 |

Tabulka 6.23: Získané hodnoty MAP míry po úpravě č. 15

### 6.2.4 Zhodnocení získaných výsledků

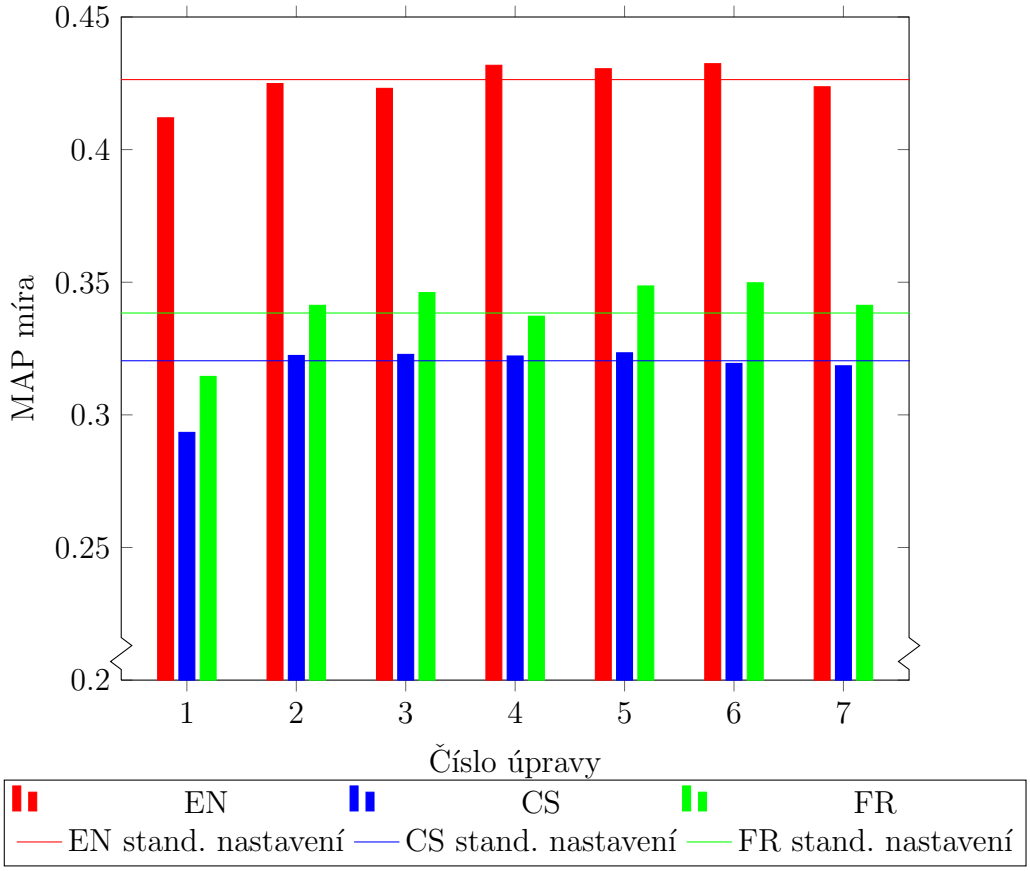
Při úpravách generování n-gramů došlo (logicky) k poměrně velkému zmenšení velikosti indexů (kromě úpravy č. 1, viz obr. 6.3), protože oproti standardnímu nastavení, kde byly generovány bigramy, trigramy a čtyřgramy, v úpravách (č. 2 až č. 6) byl generován menší počet n-gramů.




Obrázek 6.3: Porovnání velikostí indexů pro jednotlivé úpravy (úpravy č. 1 až č. 6) s velikostmi indexů při standardním nastavení (vodorovné linky)

Na obr. 6.3 je zobrazen graf s velikostmi indexů pro úpravy nastavení č. 1 až č. 6. Pro každý jazyk indexu je na obr. 6.3 také vykreslena hranice (čára), určující velikost indexu při jeho standardním nastavení. U dalších úprav nastavení indexů nedošlo k výrazné změně velikosti indexu a proto nejsou v grafu uvedeny.

I přesto, že byl generován menší počet n-gramů nedošlo k výraznému snížení MAP míry (viz obr. 6.4) a v některých případech dokonce nastalo zlepšení. Při úpravě č. 1 (generování unigramů spolu s n-gramy) došlo k výraznému zhoršení MAP míry.

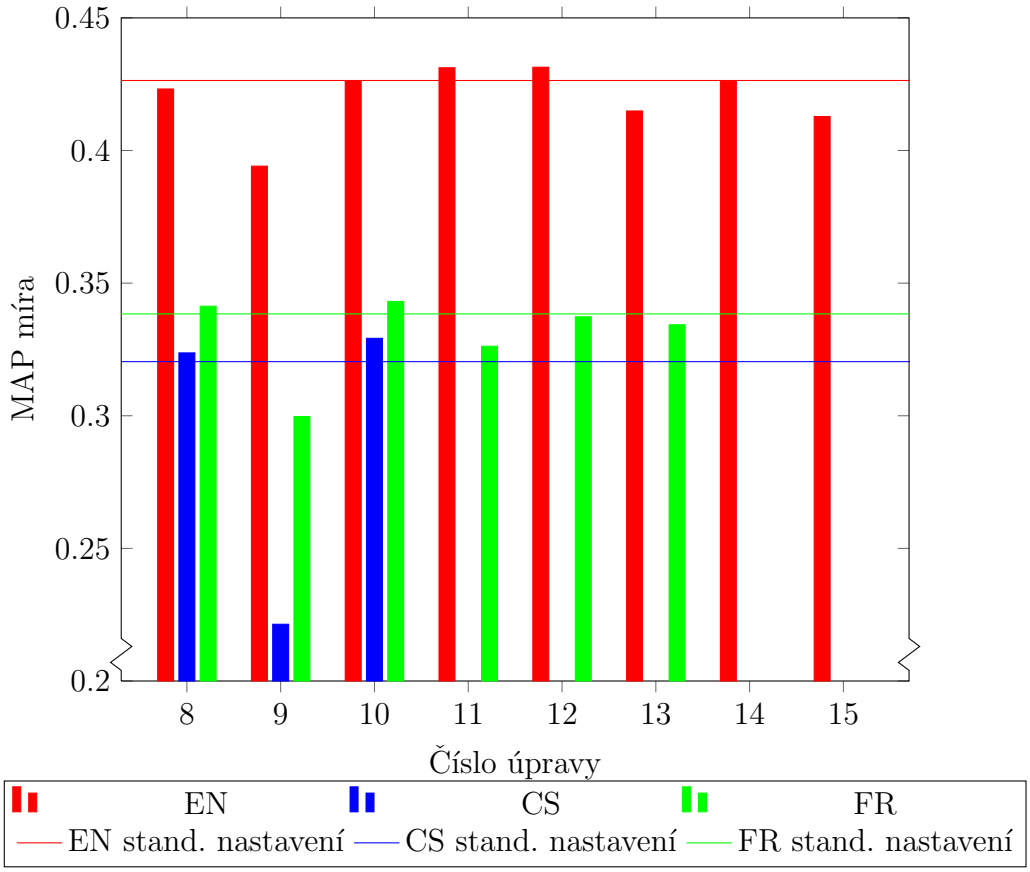



Obrázek 6.4: Porovnání získaných MAP mír pro jednotlivé úpravy (úpravy č. 1 až č. 7) s MAP mírami pro standardní nastavení (vodorovné linky)

Nepoužití filtru `icu_folding` odstraňujícího diakritiku (úprava č. 7) vedlo jen k částečnému zlepšení MAP míry pro francouzštinu a zhoršení u ostatních jazyků, viz obr. 6.4. Odebrání filtrů odstraňujících stop slova (úprava č. 8) téměř neovlivnilo velikost indexů, ale u češtiny a francouzštiny došlo k mírnému zlepšení hodnot MAP míry.

Jak bylo uvedeno v části 2.5.3, stemmer je klíčový pro jazyky, ve kterých se slova často skloňují což potvrdila úprava č. 9, po níž došlo k velmi výraznému zhoršení MAP míry, a to především u češtiny (viz obr. 6.5). Zlepšení pro češtinu a francouzštinu nastalo také po úpravě č. 10, při které nebyl použit filtr `icu_normalizer`. Odebráním filtru `english_possessive_stemmer` (úprava č. 11) u angličtiny se mírně podařilo zvýšit MAP míru.

Při úpravách č. 12 až č. 15 byly testovány všechny stemmery, které poskytuje Elasticsearch pro dané jazyky. Protože pro češtinu je v ES standardně dostupný pouze jeden stemmer, který je použit ve standardním nastavení, nebyla čeština dále testována. Podle výsledků MAP míry je nejlepší pro angličtinu stemmer `light_english` a pro francouzštinu `light_french`, který je již ve standardním nastavení.




Obrázek 6.5: Porovnání získaných MAP mír pro jednotlivé úpravy (úpravy č. 8 až č. 15) s MAP mírami pro standardní nastavení (vodorovné linky)

### 6.2.5 Návrh nového nastavení indexů

Na základě získaných výsledků testování byly navrženy změny pro jednotlivé indexy. Tyto změny by měly zlepšit kvalitu vyhledávání. Filtr `shingle_filter` generující n-gramy v indexu pro češtinu bude generovat pouze bigramy. Z analyzátoru `custom_analyzer` byl odebrán filtr `icu_normalizer` a při indexaci budou ponechána stop slova (nebudou odstraňována). Celé nastavení indexu pro češtinu po navržených změnách lze vidět v ukázce kódu C.8 v příloze C.

V nastavení pro anglický index je použit stemmer `light_english`, filtr `shingle_filter` generuje pouze trigramy a v analyzátoru `custom_analyzer` již není používaný filtr `english_possessive_stemmer`. Úprava nastavení pro francouzský index zahrnuje odstranění filtru `icu_normalizer` z analyzátoru `custom_analyzer` a úpravu filtru `shingle_filter`, tak aby generoval trigramy a čtyřgramy. Upravené nastavení indexu pro angličtinu a francouzštinu je možné vidět v ukázkách kódu C.9 a C.10 v příloze C.

Pro upravená nastavení indexů bylo provedeno vyhodnocení s testovacími daty (stejně jako u úprav v části 6.2.3). Získané hodnoty jsou uvedeny v tabulkách 6.24 a 6.25 a na obr. 6.6 je zobrazen přesnost/úplnost graf.

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 1.17 GB | 1.47 | 2.25 |
| CS | 1.37 GB | 0.474 GB | 0.9 | 2.89 |
| FR | 3.21 GB | 2.85 GB | 0.36 | 1.13 |

Tabulka 6.24: Velikosti indexů po úpravě nastavení indexů. Nastavení indexů bylo upraveno na základě výsledků testování

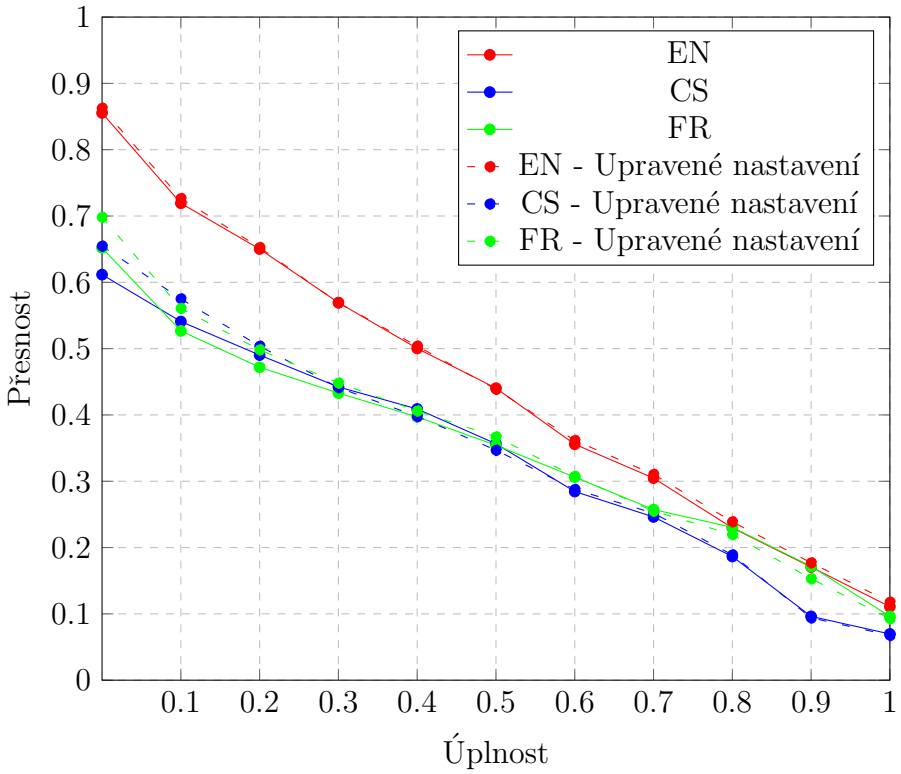
| **Jazyk** | **MAP** | **MAP po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 0.4264 | 0,4317 | $-0,0053$ | 0,9877 |
| CS | 0.3204 | 0,3267 | $-0,0063$ | 0,9807 |
| FR | 0.3384 | 0,349 | $-0,0106$ | 0,9696 |

Tabulka 6.25: Získané hodnoty MAP míry po úpravě nastavení indexů. Nastavení bylo upraveno na základě výsledků testování

### 6.2.6 Porovnání s CLEF AdHoc úlohami

Získané výsledky je možné porovnat s `CLEF AdHoc` úlohami, protože u těchto úloh byla používána stejná testovací data jako při automatizovaném testování. CLEF AdHoc úlohy z let 2007 a 2006 byly především zaměřeny na evaluaci, tzv. `monolingual` a `bilingual` vyhledávání informací.

Termínem `monolingual` je myšleno klasické vyhledávání, kdy dotaz a vrácené výsledky jsou ve stejném jazyce. `Bilingual` IR označuje vyhledávání informací takové, že vrácené výsledky (dokumenty) jsou v jiném jazyce než je zadaný dotaz. Získané výsledky automatizovaného testování budou porovnány s výsledky CLEF AdHoc úloh pro `monolingual` IR z let 2007 a 2006. Z CLEF AdHoc úlohy roku




Obrázek 6.6: Přesnost/úplnost graf testovacích dat pro nastavení indexů. Nastavení indexů bylo upraveno na základě výsledků testování

2006 bude pro porovnání použita francouzština a z roku 2007 budou použity angličtina a čeština. Monolingual úlohu pro anglický jazyk řešilo deset účastníků, pro český a francouzský jazyk řešilo úlohu celkem 8 účastníků. V tabulce 6.26 je uvedeno pět nejlepších výsledků MAP míry pro jednotlivé jazyky v CLEF AdHoc úlohách z let 2007 (angličtina a čeština) a 2006 (francouzština).

| **Pořadí** | **MAP** | | |
|---|---|---|---|
| | **Angličtina** | **Čeština** | **Francouzština** |
| 1. | 0.4402 | 0.4242 | 0.4468 |
| 2. | 0.4342 | 0.3586 | 0.4096 |
| 3. | 0.4274 | 0.3484 | 0.4077 |
| 4. | 0.4057 | 0.3419 | 0.3828 |
| 5. | 0.4016 | 0.3203 | 0.3794 |

Tabulka 6.26: Pět nejlepších výsledků MAP míry z CLEF AdHoc úloh z let 2007 (čeština a angličtina) a 2006 (francouzština), data převzata z [30] a [31]

Porovnáním výsledků z CLEF AdHoc úloh (tab. 6.26) a výsledků získaných navrženou úpravou indexů (tab. 6.25), lze zjistit, že implementované vyhledávání za pomoci Elasticsearch u angličtiny a češtiny dosahuje podobných výsledků, jakých bylo dosaženo při řešení monolingual CLEF AdHoc úlohy. Pokud by byly získané výsledky přidány do tabulky 6.26, umístilo by se vytvořené řešení na třetím místě pro anglický jazyk a na pátém místě pro český jazyk. Pouze u francouzštiny byl rozdíl MAP míry 0.0304 mezi pátým výsledkem z CLEF AdHoc úlohy a MAP mírou pro upravené nastavení indexu a v tabulce 6.26 by se tak řešení pro francouzský jazyk umístilo na 6 až 9 místě (výsledky byly zveřejněny jen pro prvních pět řešení).

Porovnání s CLEF AdHoc úlohami ukazuje, že vytvořené řešení je podle MAP míry téměř na stejné úrovni a nijak výrazně nezaostává. Je třeba také zmínit, že implementované vyhledávání je navrženo pro data systému MediaGist, která jsou jinak strukturovaná než použitá testovací data, což může mít také vliv na výslednou kvalitu vyhledávání, tzn. i na MAP míru. Zlepšení výsledků u češtiny by pravděpodobně mohlo být dosaženo použitím jiného stemmeru (Elasticsearch standardně obsahuje zabudovaný pouze jeden), např. [24].

### 6.2.7 Závěr pro automatizované testování

Cílem testování bylo určit úspěšnost vyhledávání. Při automatizovaném testování byly testovány různé konfigurace nastavení indexů pro angličtinu, češtinu a francouzštinu. Testování probíhalo nad daty z balíčku `CLEF AdHoc - News 2004-2008` a sledovanými veličinami byly velikosti indexů a MAP míry.

Získané výsledky byly porovnávány s hodnotami získaných z původních nastavení indexů a na jejich základě byly navrženy úpravy pro testované indexy (angličtina, čeština a francouzština). Navrženými úpravami se u indexu pro český a anglický jazyk podařilo pro testovací data výrazně snížit velikost indexu a u všech tří jazyků mírně zvýšit MAP míru, která reprezentuje kvalitu získaných výsledků. Na závěr byly (v části 6.2.6) porovnány dosažené výsledky upraveného nastavení indexů s výsledky z CLEF AdHoc úloh.

## 6.3 Uživatelské testování

Cílem uživatelského testování je zjistit, zda jsou uživatelé schopni za pomoci implementované funkcionality používat vyhledávání a získávat tak relevantní výsledky pro hledané informace. Dále je cílem tohoto testování objevit případné chyby a nedostatky webové aplikace a knihovny a na jejich základě vytvořit doporučení úprav při

integraci vyhledávání se systémem MediaGist. Při uživatelském testování měli vybraní uživatelé (testeři) prozkoumat webovou aplikaci, vyzkoušet všechny její funkce a splnit následující úkoly:

1. V Ženevě byl vydražen vzácný diamant.
   a) Zjistěte jeho jméno v angličtině.
   b) Na základě nalezeného jména zjistěte jeho cenu v dolarech.
2. Co se dělo s cenou ropy podle dokumentů publikovaných v polovině listopadu roku 2015 ?
3. K jakému tématu se vztahuje fráze „Co chceš od slepice“, která se vyskytla v komentářích daného dokumentu.
4. Jak skončilo první fotbalové utkání mezi Německem a Mexikem na letních olympijských hrách 2016 (dotaz je potřeba zadat v německém jazyce) ?
   a) Z jakého serveru byla data získána (z jakých článků byl souhrn vytvořen) ?
5. Vyhledejte informace o útocích, které se udály v Londýně a zprávy byly publikovány mezi 1.11.2015 a 1.5.2016. Hledejte pouze v anglických článcích a pouze v titulcích (dotaz je potřeba zadat v anglickém jazyce).
   a) V jaké části Londýna došlo k útoku ?
   b) Na koho bylo útočeno ?

Při vytváření výše uvedených úkolů byla snaha, aby pro jejich splnění museli uživatelé využívat nástrojů pokročilého vyhledávání (hledání frází, změnit rozsah data apod.). Testování probíhalo nad poskytnutými daty ze systému MediaGist a na základě těchto dat byly úkoly vytvořeny.

Webová aplikace a server Elasticsearch byly společně spuštěny na notebooku s popsanou konfigurací (viz 6.1) a jednotliví testeři přistupovali na stránky webové aplikace ze svých zařízení (webových prohlížečů).

### 6.3.1 Výsledky uživatelského testování

Webovou aplikaci otestovalo celkem osm uživatelů. Při testování uživatelé neobjevili žádné závažné chyby a to jak ve webové aplikaci tak ani v samotném vyhledávání. Všem testerům se zadané úkoly podařilo splnit (nalezli požadované informace nebo dokumenty). Při plnění úkolu č. 2 většina testerů nejprve napsala do dotazu měsíc

listopad slovy a teprve, až když se nepodařilo najít hledanou informaci použili filtr pro omezení data v pokročilém vyhledávání, jenž bylo nutné použít k nalezení dané informace. Aby i pro takovýto dotaz uživatelé získali relevantní výsledky mohlo by být číslo měsíce převáděno na textový řetězec a ten by pak mohl být indexován společně s daným dokumentem. Problémem tohoto řešení je ale fakt, že ne vždy bude datum vytvoření dokumentu odpovídat i datu o hledané informaci. Např. dokument může pojednávat o událostech, které se udály v prosinci, ale dokument bude vytvořen v lednu.

Dále bylo zjištěno, že např. při hledání dotazu obsahujícího slovo „Mexico“ v německých článcích nemohly být nalezeny některé dokumenty, protože Mexiko se v německém jazyce píše s písmenem `k` nikoliv s `c` jako v anglickém jazyce. Poslední připomínkou testerů bylo, že postrádali tlačítko pro vymazání textu ve všech polích pokročilého vyhledávání. Tento nedostatek se nevztahuje k samotnému vyhledávání, ale spíše k uživatelskému rozhraní a snadno může být vyřešen přidáním tlačítka na stránku pokročilého vyhledávání při integraci s webovými stránkami systému MediaGist.

### 6.3.2 Možná rozšíření

Knihovnu by samozřejmě bylo možné rozšířit o funkcionalitu, která by uživatelům poskytovala další možnosti. Mezi možná rozšíření, která by mohla být do knihovny (popř. do webové aplikace a následně na webové stránky systému MediaGist) přidána, a to nejen na základě uživatelského testování, patří:

- Indexace a vyhledávání v rozpoznaných pojmenovaných entitách.
- Zvýraznění hledaných slov dotazu v získaných výsledcích.
- Oprava překlepů v dotazu.
- Řazení výsledků (podle data, skóre, jazyku apod.).

# 7 Závěr

Cílem této diplomové práce bylo prozkoumat a analyzovat vybrané nástroje umožňující textové vyhledávání a na základě jejich porovnání vybrat vhodný nástroj pro realizaci vyhledávání v datech systému MediaGist. Dále bylo cílem navrhnout a za pomoci vybraného nástroje realizovat jednoduché a rozšířené vyhledávání a následně jej otestovat.

V první části práce jsou nejprve popsány základní principy a postupy používané při vyhledávání informací, včetně vyhodnocování kvality IR systémů. Následně jsou porovnány dva nástroje umožňující textové vyhledávání – Elasticsearch a Apache Solr. Pro řešenou úlohu nejsou mezi těmito nástroji žádné výrazné rozdíly a upřednostněním jednoho nástroje před druhým by nedošlo k získání žádných výrazných výhod. Oba nástroje byly vhodné pro realizaci požadovaného vyhledávání, ale vybrán byl Elasticsearch, protože autor diplomové práce měl již částečné zkušenosti s tímto nástrojem a vedoucí práce také preferoval jeho použití.

Dále je popsán návrh a realizace požadovaného vyhledávání. Funkcionalita vyhledávání v datech systému MediaGist je implementována formou knihovny, která poskytuje funkce (vyhledávání, indexace atd.) pro komunikaci se serverem Elasticsearch. Knihovna byla naprogramována v jazyce JAVA, protože systém MediaGist je také implementován pomocí jazyka JAVA. Stěžejní částí práce při vytváření knihovny byl návrh a vytvoření nastavení indexů pro jednotlivé jazyky.

Spolu s knihovnou byly vytvořeny další dvě aplikace, které demonstrují použitelnou funkcionalitu knihovny, tj. webová aplikace a klient pro indexaci. Klient je konzolová aplikace, která umožňuje indexovat data (dokumenty) systému MediaGist nezávisle na webové aplikaci. Webová aplikace využívá funkcí vytvořené knihovny a umožňuje vyhledávat v těchto datech pomocí jednoduchého a pokročilého vyhledávání.

V poslední části práce je popsáno uživatelské a automatizované testování. Automatizované testování bylo provedeno na datech z balíčku CLEF AdHoc - News 2004-2008 a získané výsledky byly porovnány s výstupy CLEF AdHoc úloh z let 2007 a 2006, které používaly stejná data. K porovnání byla použita MAP míra. Získané výsledky pro angličtinu a češtinu dosahovaly v průměru stejných a částečně i lepších výsledků než řešení monolingual CLEF AdHoc úloh (řešení pro anglický jazyk by se umístilo na 3. místě, a řešení pro český jazyk na 5. místě). Řešení pro francouzštinu jen mírně zaostávalo za řešeními CLEF AdHoc úloh (umístilo by se na 6. až 9. místě).

Elasticsearch se ukázal jako vyhovující nástroj pro podobný typ úloh a vytvořená knihovna představuje vhodné řešení pro vyhledávání v datech systému MediaGist.

# Seznam použitých zkratek a výrazů

| | |
|---|---|
| **API** | Application Programming Interface |
| **CLEF** | Cross-Language Evaluation Forum |
| **CPU** | Central Processing Unit |
| **CS** | Český jazyk |
| **CSS** | Cascading Style Sheets |
| **DOCX** | Formát souboru aplikace Microsoft Office Word |
| **DSL** | Domain Specific Language |
| **DVD** | Digital Versatile Disc |
| **EN** | Anglický jazyk |
| **ES** | Elasticsearch |
| **FR** | Francouzský jazyk |
| **Framework** | Nástroj (obecně) pro usnadnění a urychlení vývoje aplikace |
| **HDD** | Hard Disk Drive |
| **HTML** | HyperText Markup Language |
| **HTTP** | HyperText Transfer Protocol |
| **IR** | Information Retrieval |
| **IT** | Informační Technologie |
| **JDK** | Java Development Kit |
| **JRE** | Java Runtime Environment |
| **JS** | JavaScript |
| **JSON** | JavaScript Object Notation |
| **JSP** | JavaServer Pages |
| **JVM** | Java Virtual Machine |

| | |
|---|---|
| **MAP** | Mean Average Precision |
| **Open Source** | Software s otevřeným zdrojovým kódem |
| **OS** | Operační Systém |
| **PDF** | Portable Document Format |
| **RAM** | Random Access Memory |
| **REST** | Representational state transfer |
| **SSD** | Solid State Drive |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |
| **UI** | User Interface |
| **UML** | Unified Modeling Language |
| **XML** | Extensible Markup Language |

# Literatura


[1] MANNING, Christopher D., Prabhakar. RAGHAVAN a Hinrich. SCHÜTZE. *Introduction to information retrieval*. New York: Cambridge University Press, 2008. ISBN 05-218-6571-9.

[2] BÜTTCHER, Stefan., Charles L. A. CLARKE a Gordon V. CORMACK. *Information retrieval: implementing and evaluating search engines*. Cambridge, Mass.: MIT Press, c2010. ISBN 02-620-2651-1.

[3] CROFT, Bruce, Donald METZLER a Trevor STROHMAN. *Information retrieval in practice*. International ed. Upper Saddle River, N.J: Pearson Education, 2009. ISBN 978-013-1364-899.

[4] PETERS, C., Martin. BRASCHLER a Paul CLOUGH. *Multilingual information retrieval: From Research To Practice*. New York: Springer, c2012. ISBN 978-3-642-23007-3.

[5] MCCANDLESS, Michael., Erik. HATCHER a Otis. GOSPODNETIC. *Lucene in action*. 2nd ed. Greenwich: Manning, c2010. ISBN 978-1-933988-17-7.

[6] GORMLEY, Clinton a Zachary TONG. *Elasticsearch: the definitive guide*. Gravenstein Highway North, Sebastopol,: O'Reilly Media, 2015. ISBN 978-1-449-35854-9.

[7] GRAINGER, Trey. a Timothy POTTER. *Solr in action*. Shelter Island: Manning, 2014. ISBN 978-161-7291-029.

[8] *Elastic Stack and Product Documentation: Elasticsearch Reference* [online]. Elastic, 2017 [cit. 2017-04-03]. Dostupné z: `https://www.elastic.co/guide/index.html`

[9] *Apache Solr* [online]. The Apache Software Foundation, 2017 [cit. 2017-04-04]. Dostupné z: `http://lucene.apache.org/solr/`

[10] STEINBERGER, J. MediaGist: A cross-lingual analyser of aggregated news and commentaries. In *The 54th Annual Meeting of the Association for Computational Linguistics*. Stroudsburg: Association for Computational Linguistics (ACL), 2016. s. 145-150. ISBN: 978-1-945626-03-6

[11] *Cross-Language Evaluation Forum* [online]. PISA: ISTI-CNR, 2009 [cit. 2017-04-20]. Dostupné z: `http://clef.isti.cnr.it/`

[12] *Apache Solr vs Elasticsearch: The Feature Smackdown* [online]. Kelvin Tan, 2017 [cit. 2017-04-04]. Dostupné z: `http://solr-vs-elasticsearch.com/`

[13] LUBURIĆ, Nikola a Dragan IVANOVIĆ. *Comparing Apache Solr and Elasticsearch search servers*. Novi Sad, 2016.

[14] Solr vs. Elasticsearch: 5 Factors to Consider. *CMS Wire* [online]. Kamran Khan, 2016 [cit. 2017-04-04]. Dostupné z: `http://www.cmswire.com/information-management/solr-vs-elasticsearch-5-factors-to-consider/`

[15] Solr vs. Elasticsearch: Who's The Leading Open Source Search Engine? *Logz.io* [online]. New York: Yigal, 2016 [cit. 2017-04-11]. Dostupné z: `https://logz.io/blog/solr-vs-elasticsearch/`

[16] Solr or Elasticsearch–That Is the Question. *Datanami* [online]. San Diego: Gospodnetić, 2015 [cit. 2017-04-11]. Dostupné z: `https://www.datanami.com/2015/01/22/solr-elasticsearch-question/`

[17] DB-Engines Ranking of Search Engines. *DB-Engines* [online]. Wien: solid IT, 2017 [cit. 2017-04-11]. Dostupné z: `https://db-engines.com/en/ranking/search+engine`

[18] *Apache Tika* [online]. The Apache Software Foundation, 2017 [cit. 2017-04-20]. Dostupné z: `https://tika.apache.org/`

[19] Desktop Search Engine Market Share. *Netmarketshare* [online]. [cit. 2017-02-21]. Dostupné z: `https://www.netmarketshare.com/search-engine-market-share.aspx?qprid=4&qpcustomd=0`

[20] HEROUT, Pavel. *Testování pro programátory*. České Budějovice: Kopp, 2016. ISBN 978-807-2324-811.

[21] Latency Numbers Every Programmer Should Know. In: *GitHubGist* [online]. San Francisco: Bonér, 2017 [cit. 2017-04-18]. Dostupné z: `https://gist.github.com/jboner/2841832`

[22] The Porter Stemming Algorithm. *Martin Porter's Home Page* [online]. Cambridge: Porter, 2006 [cit. 2017-04-18]. Dostupné z: `https://tartarus.org/martin/PorterStemmer/`

[23] *ICU - International Components for Unicode: ICU-TC Home Page* [online]. 2017 [cit. 2017-04-18]. Dostupné z:`http://site.icu-project.org/home`

[24] BRYCHCÍN, Tomáš a Miloslav KONOPÍK. HPS: High precision stemmer. *Information Processing* [online]. 2015, **51**(1), 68-91 [cit. 2017-04-18]. DOI: 10.1016/j.ipm.2014.08.006. ISSN 03064573. Dostupné z: `http://linkinghub.elsevier.com/retrieve/pii/S0306457314000843`

[25] Index vs. Type. *Elastic blog* [online]. Elastic, 2017 [cit. 2017-04-18]. Dostupné z: `https://www.elastic.co/blog/index-vs-type`

[26] Elastic search, multiple indexes vs one index and types for different data sets? *Stackoverflow* [online]. 2013 [cit. 2017-04-18]. Dostupné z: `http://stackoverflow.com/a/14554767/2211656`

[27] Precision and Recall. In: *Wikimedia Commons* [online]. 2014 [cit. 2017-04-04]. Dostupné z: `https://upload.wikimedia.org/wikipedia/commons/2/26/Precisionrecall.svg`

[28] HANA, Kanisová a Müller MIROSLAV. *UML srozumitelně*. Vyd. 1. Brno: Computer Press, 2004, 157 s. ISBN 80-251-0231-9.

[29] ARLOW, Jim a Ila NEUSTADT. *UML 2 a unifikovaný proces vývoje aplikací: objektově orientovaná analýza a návrh prakticky*. 2. akt. a dopl. vyd. Překlad Bogdan Kiszka. Brno: Computer Press, 2007, 567 s. ISBN 978-80-251-1503-9.

[30] DI NUNZIO, Giorgio, et al. Clef 2007: Ad hoc track overview. Advances in Multilingual and Multimodal Information Retrieval, 2008, 13-32.

[31] DI NUNZIO, Giorgio M., et al. CLEF 2006: Ad hoc track overview. In: Workshop of the Cross-Language Evaluation Forum for European Languages. Springer Berlin Heidelberg, 2006. p. 21-34.

# Seznam příloh

Uživatelská příručka - Příloha A

Diagramy, grafy a výsledky testování - Příloha B

Nastavení indexů C

Obsah CD - Příloha D

# A Uživatelská příručka

Uživatelská příručka obsahuje základní informace o webové aplikaci a klientovi pro indexaci, tj. co aplikace umožňují (jejich funkce), popis jejich instalace a ovládání.

## A.1 Elasticsearch

Aby bylo možné používat webovou aplikaci a klienta pro indexaci je potřeba spustit instanci serveru Elasticsearch. Elasticsearch je distribuován na DVD v adresáři **\dist\elasticsearch** ve verzi 5.0.2 s již zaindexovanými daty systému MediaGist.

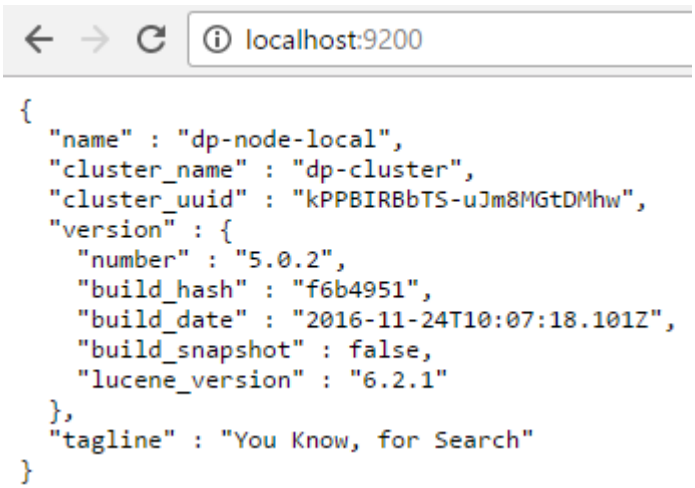
### A.1.1 Požadavky pro spuštění Elasticsearch

Přiloženou verzi Elasticsearch je možné spustit na Windows i GNU/Linux (otestována na Windows 10 Home a Linuxové distribuci Debian 8.7 (jessie)). Požadavky pro spuštění:

- Nainstalované běhové prostředí `JRE` jazyka JAVA ve verzi alespoň `1.8`.
- Alespoň 4GB volné paměti RAM (potřebné množství paměti je možné snížit změnou konfigurace, jenž je popsána v části A.1.3).
- Pokud je to možné je vhodné spustit Elasticsearch na SSD disku.
- Při spuštění je potřeba, aby byly volné porty 9300 a 9200, na kterých Elasticsearch naslouchá.

### A.1.2 Spuštění Elasticsearch

Elasticsearch je možné spustit za pomoci připravených skriptů **run-diploma.sh** pro GNU/Linux a **run-diploma.bat** pro Windows. V případě úspěchu by po zadání adresy `http://localhost:9200/` do prohlížeče měla být zobrazena stránka podobná té na obr. A.1.




Obrázek A.1: Ukázka výstupu stránky `http://localhost:9200/` při úspěšném spuštění Elasticsearch

### A.1.3 Spuštění Elasticsearch s upravenou konfigurací

Elasticsearch je možné spustit s upravenou konfigurací tzn. na jiném portu či s jinou velikostí přidělené paměti. Přidělenou paměť je možné upravit v adresáři **config** distribuce Elasticsearch v souboru **jvm.options**, konkrétně parametry `Xms` a `Xmx`.

Např. pro nastavení přidělené paměti pro Elasticsearch na 2GB je třeba tyto parametry nastavit na hodnotu `2g` jak je ukázáno na obr. A.2.

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms2g
-Xmx2g
```

Obrázek A.2: Ukázka nastavení parametrů pro přidělení 2GB paměti pro Elasaticsearch

Změna portů, na kterých bude Elasticsearch je možná v souboru **elasticsearch.yml**, konkrétně změnou parametrů `http.port:` a `transport.tcp.port:` jak je ukázáno na obr. A.3. Nicméně webová aplikace očekává, že Elasticsearch bude spuštěn na výchozích portech.

```
# Set a custom port for HTTP:
#
 http.port: 9500
 transport.tcp.port: 9600
```

Obrázek A.3: Ukázka nastavení parametrů pro změnu portů, na kterých je dostupný Elasticsearch

## A.2 Klient

Klient je konzolová aplikace sloužící pro indexaci dokumentů systému MediaGist. Umožňuje také zobrazit seznam indexů na serveru ES či je mazat.

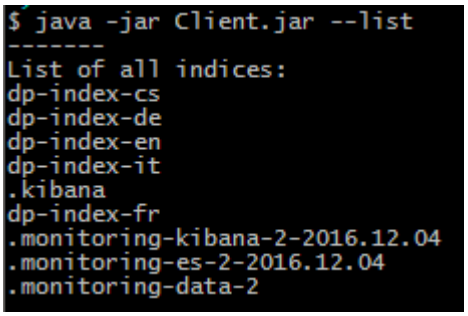
### A.2.1 Požadavky pro spuštění klienta

Pro spuštění aplikace je potřeba mít na zařízení, na kterém bude aplikace spuštěna, nainstalované běhové prostředí `JRE` jazyka JAVA ve verzi alespoň `1.8`. Aplikaci je možné spustit na operačním systému Windows i GNU/Linux. Aplikace byla otestována na Windows 10 Home a Linuxové distribuci Debian 8.6 (jessie).

### A.2.2 Spuštění klienta

Před spuštěním klienta je potřeba aby běžela instance serveru Elasticsearch. Postup spuštění instance serveru Elasticsearch je popsán v části A.1. Klient je distribuován jako spustitelný `JAR` soubor pojmenovaný `Client.jar`, který je umístěný na přiloženém DVD konkrétně v adresáři **\dist\client**.

Na DVD jsou také umístěna vzorová data (dokumenty), které je možné pomocí klienta indexovat. Tato data jsou umístěna na přiloženém DVD v adresáři **\data**. Data s jinou strukturou klient nedokáže zpracovat. Program je možné z příkazové řádky spustit příkazem `java -jar Client.jar [parametry]`. Například pro vypsání seznamu indexů je možné použít příkaz `java -jar Client.jar --list`, na obr. A.4 je zobrazen možný výstup pro tento parametr. Všechny použitelné parametry klienta jsou popsány v následující části A.2.3.




Obrázek A.4: Výstup při spuštění klienta s parametry pro vypsání seznamu indexů

### A.2.3 Parametry klienta

V následujícím výčtu jsou popsány jednotlivé parametry, které akceptuje klient pro indexaci. Parametry je možné kombinovat a zadávat v krátké (např. `-l`) i dlouhé podobě (např. `--list`).

- `-c`, `--config` – Soubor s konfigurací pro připojení s serveru Elasticsearch. Vzor a struktura souboru je uvedena v ukázce kódu A.1.
- `-d`, `--delete` – Smaže zadaný index.
- `-f`, `--file-index` – Soubor XML, který bude zaindexován.
- `-h`, `--help` – Vytiskne nápovědu, pokud je tento parametr nastaven všechny ostatní parametry jsou ignorovány.
- `-i`,`--index` – Složka určená pro indexaci, všechny XML soubory s odpovídající strukturou dokumentů systému MediaGist budou zaindexovány. Složka může obsahovat další podsložky, které budou také indexovány (rekurzivně zpracuje všechny podsložky).
- `-l`, `--list` – Vytiskne seznam všech indexů
- `-r`, `--reindex` – Při indexaci smaže všechny staré indexy. Může být použit pouze s parametrem `-i`.
- `-s`, `--size` – Velikost dávky při indexaci (kolik dokumentů bude odesláno v jednom požadavku). Může být použit pouze s parametrem `-i`. Výchozí hodnota je 500.
- `-v`, `--verbose` – Verbose mod logování. Loguje všechny zprávy programu.

```
cluster.name=dp-cluster
host=192.168.2.103
port=9300
```

Ukázka kódu A.1: Ukázka souboru pro připojení k Elasticsearch serveru

## A.3 Webová aplikace

Webová aplikace slouží pro demonstraci funkcí, které poskytuje knihovna.

### A.3.1 Požadavky pro spuštění webové aplikace

Pro spuštění webové aplikace je potřeba mít na zařízení, na kterém bude aplikace spuštěna, nainstalované běhové prostředí `JRE` jazyka JAVA ve verzi alespoň `1.8`. Webovou aplikaci je možné spustit na operačním systému Windows i GNU/Linux. Aplikace byla otestována na Windows 10 Home a Linuxové distribuci Debian 8.6 (jessie).

### A.3.2 Spuštění webové aplikace

Webová aplikace je distribuována jako soubor s příponou `war`. Tento soubor je pojmenována `web-app.war` a je umístěn na přiloženém DVD v adresáři **\bin\web**.

Webovou aplikaci je možné spustit v servletovém kontejneru, např. v `Jetty`. Před spuštěním webové aplikace je potřeba spustit server Elasticsearch podle popisu v části A.1. Webová aplikace očekává, že server Elasticsearch poběží na stejném zařízení jako webová aplikace, bude dostupný na adrese `127.0.0.1` (resp. `localhost`) a pro připojení JAVA klienta bude poslouchat na portu `9300` (výchozí hodnoty nastavení při spuštění serveru Elasticsearch podle části A.1).

Pro spuštění webové aplikace je již připraven nakonfigurovaný servletový kontejner `Jetty` na přiloženém DVD v adresáři **\dist\jetty-web-app**, kde se také nacházejí skripty **run-diploma-web.sh** pro GNU/Linux a **run-diploma-web.bat** pro Windows. Vykonáním jednoho z nich (např. z příkazové řádky) dojde ke spuštění servletového kontejneru `Jetty`, který na portu `8080` spustí webovou aplikaci. Pokud spuštění proběhne v pořádku, tak po zadání adresy `http://localhost:8080/` do internetového prohlížeče by měla být zobrazena úvodní stránka webové aplikace.

Verze přiloženého servletového kontejneru `Jetty` je `9.4.3` a byla stažena z oficiálních webových stránek `http://www.eclipse.org/jetty/`. Webovou aplikaci je možné spustit na jiném portu než výchozím portu `8080`, ale je potřeba přejít do adresáře **\dist\jetty-web-app\news-searching** a v příkazové řádce zadat příkaz `java -jar ../start.jar jetty.http.port=[číslo portu]` tedy např. pro spuštění na portu `8085` zadat příkaz `java -jar ../start.jar jetty.http.port=8085`.

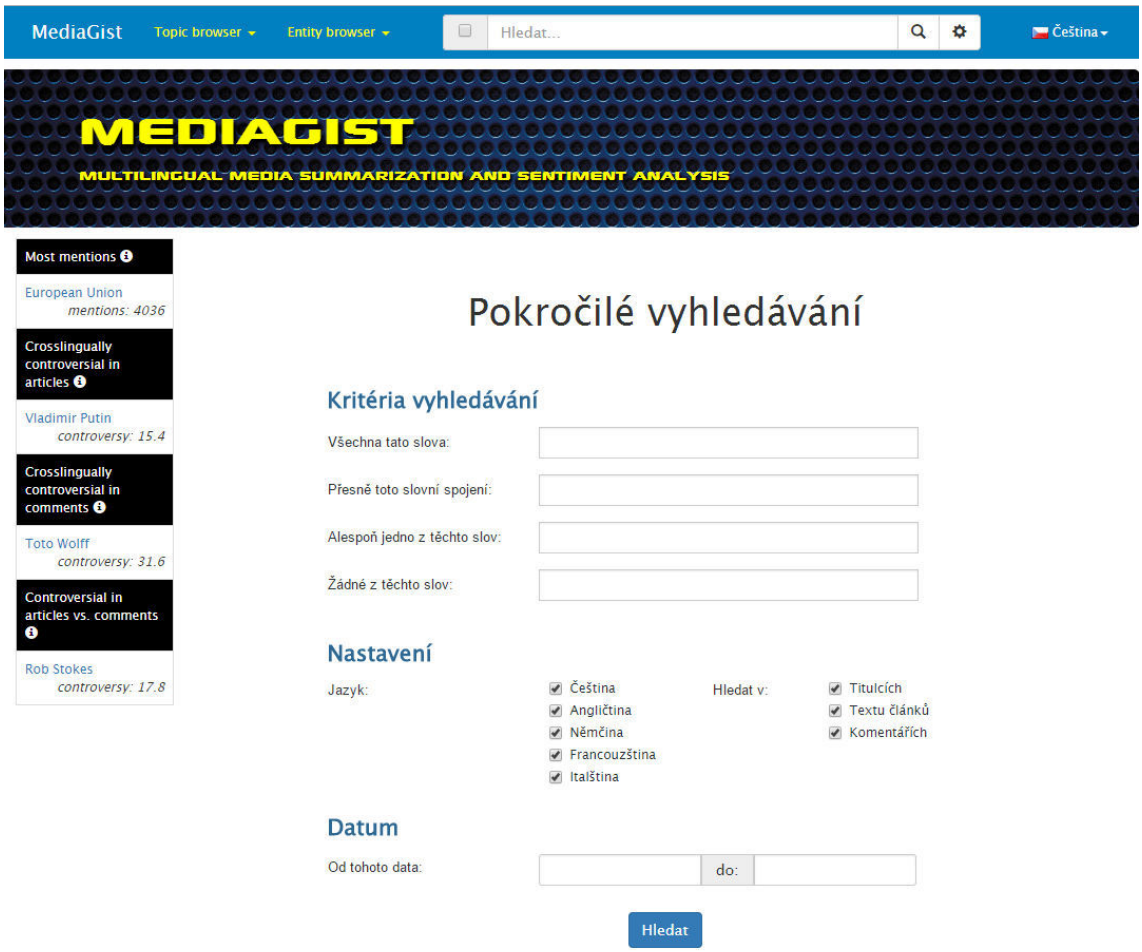
### A.3.3 Ovládání aplikace

Webová aplikace poskytuje dvě možnosti vyhledávání – `Jednoduché` a `Pokročilé`. Jednoduché vyhledávání je dostupné jako formulář v horním panelu stránky viz obr. A.5. V levé části formuláře je umístěno `zaškrtávací pole`, které při použití jednoduchého vyhledávání aktivuje `detekci jazyka`. Při vyhodnocování dotazu se aplikace pokusí rozpoznat jazyk dotazu a pokud se podaří rozpoznat jeden z jazyků (angličtina, čeština, němčina, francouzština, italština), ve kterých se vyskytují dokumenty systému MediaGist, vyhledávání bude provedeno pouze v rozpoznaném jazyku.

V dotazu jednoduchého vyhledávání je možné použít booleovské operátory `AND`, `OR` a `NOT`, dále také závorky pro vynucení precedence. V pravé části formuláře jednoduchého vyhledávání je umístěno tlačítko pro odeslání dotazu (ikona lupy) a tlačítko pro přechod na stránku pokročilého vyhledávání (ikona ozubeného kola). Při zadávání dotazu pro jednoduché vyhledávání jsou nabízené (našeptávané) možné relevantní výsledky. Možné výsledky při našeptávání jsou získávány pouze z titulků dokumentů (clusterů), při ostatních vyhledáváních jsou samozřejmě do hledání zahrnuta i ostatní pole dokumentů.

Na stránce pokročilého vyhledávání (viz obr. A.5) je možné vyhledávání omezit těmito kritérii:

- V hledaném dokumentu se musí objevit všechna zadaná slova (pole s popiskem „Všechna tato slova“).
- V hledaném dokumentu se musí objevit zadané slovní spojení (pole s popiskem „Přesně toto slovní spojení“).
- V hledaném dokumentu se musí objevit alespoň jedno ze zadaných slov (pole s popiskem „Alespoň jedno z těchto slov“).
- V hledaném dokumentu se nesmí objevit žádné ze zadaných slov (pole s popiskem „Žádné z těchto slov“).
- Jazyk hledaného dokumentu.

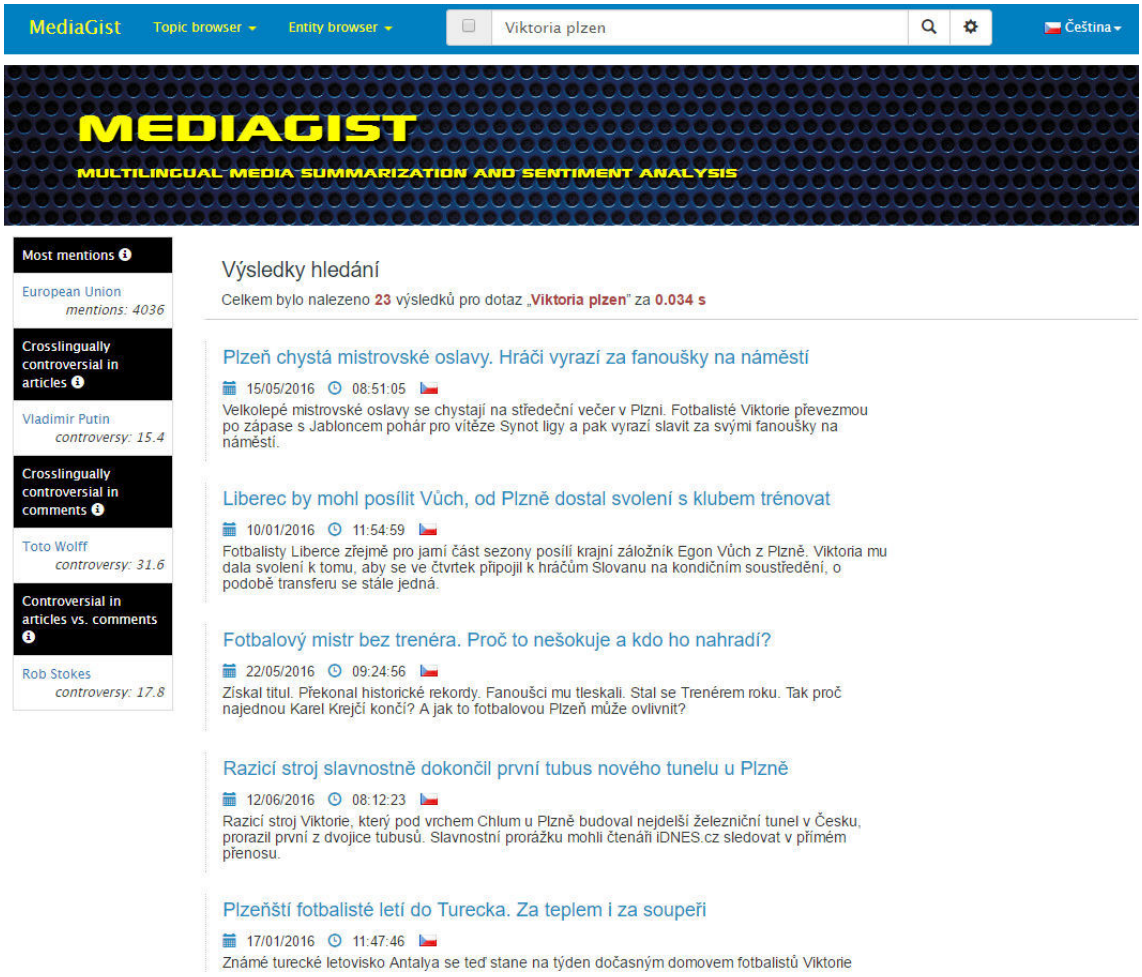



Obrázek A.5: Ukázka stránky s formulářem pro pokročilé vyhledávání

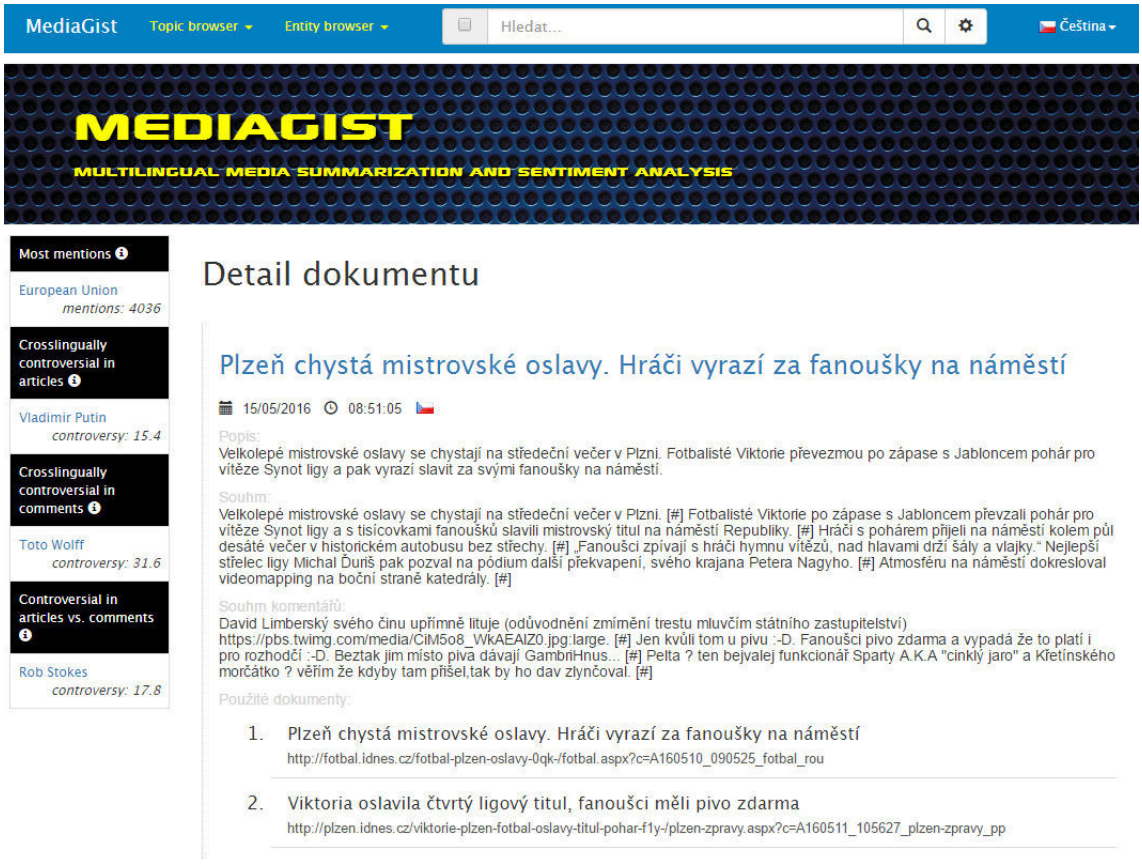
- Omezení na hledání ve vybraných polích dokumentů (titulky, text článků, komentáře).
- Rozmezí data, ve kterém byl článek publikován.

Po odeslání dotazů (jednoduchého i pokročilého vyhledávání) jsou zobrazeny výsledky (viz obr. A.6). Výsledky jsou stránkovány po deseti od nejrelevantnějších. Po kliknutí na určitý výsledek je zobrazen detail dokumentu (clusteru), viz obr. A.7. V detailu výsledku jsou uvedeny souhrny pro daný cluster a odkazy na články, ze kterých byl vytvořen.

Všechny ostatní odkazy na stránkách webové aplikace jsou zaslepené (nikam neodkazují). Webová aplikace je také lokalizována do češtiny a angličtiny.

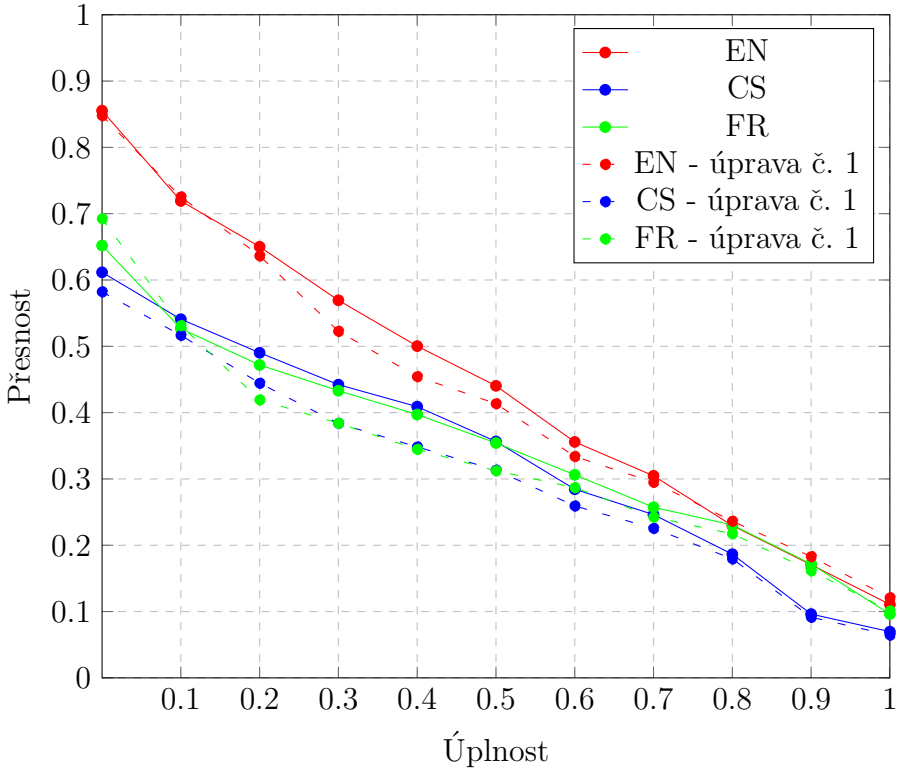



Obrázek A.6: Ukázka stránky s výsledky vyhledávání

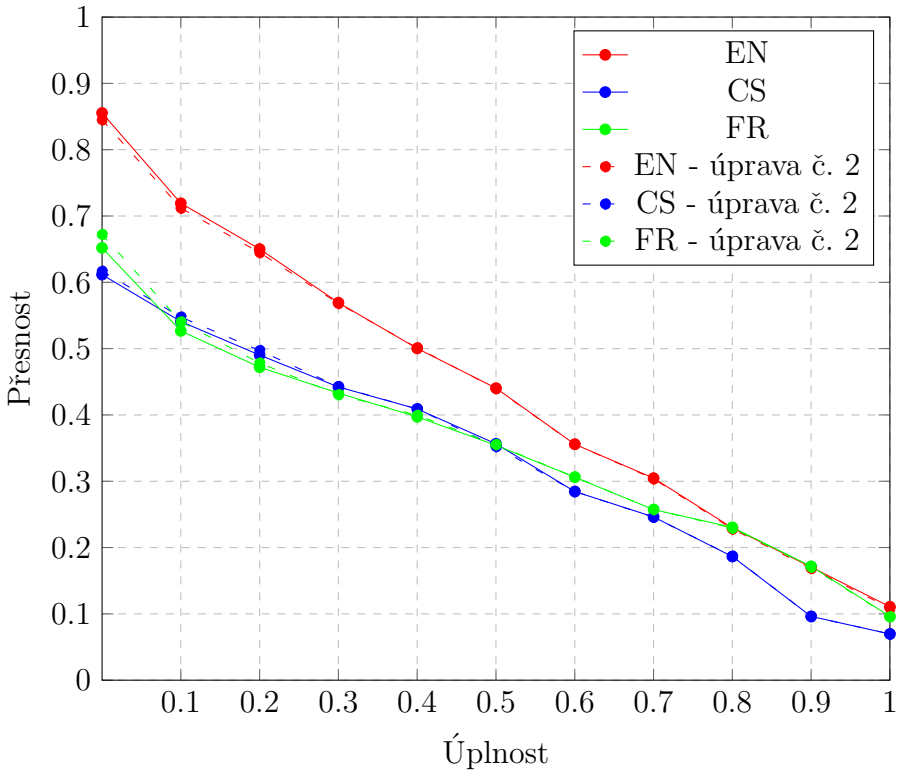



Obrázek A.7: Ukázka stránky s detailem výsledku (dokumentu)

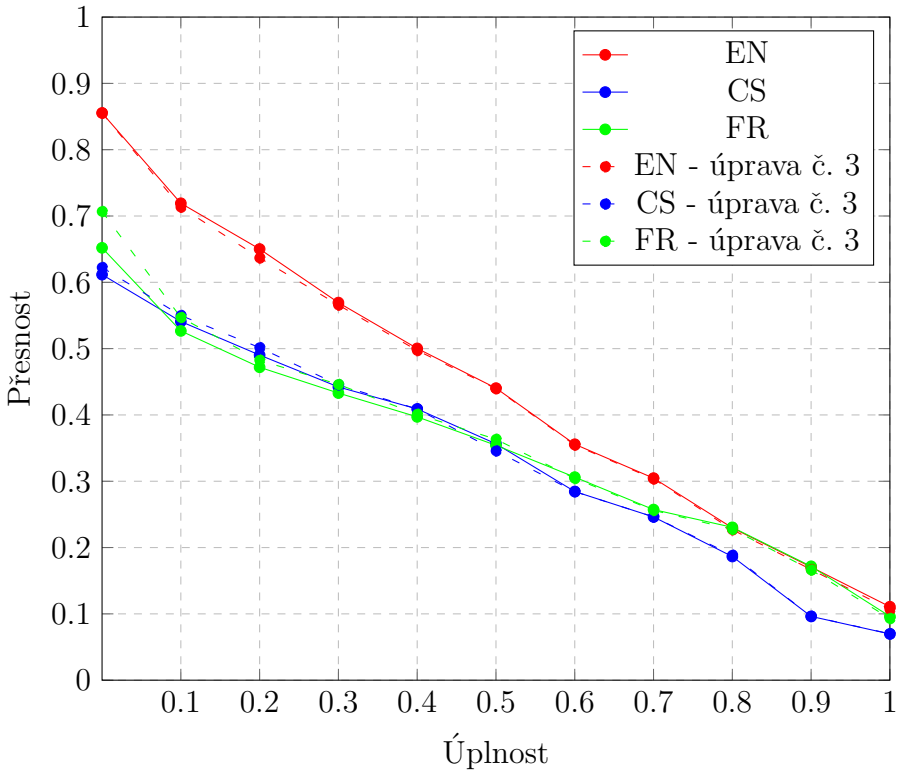
# B Diagramy, grafy a výsledky testování



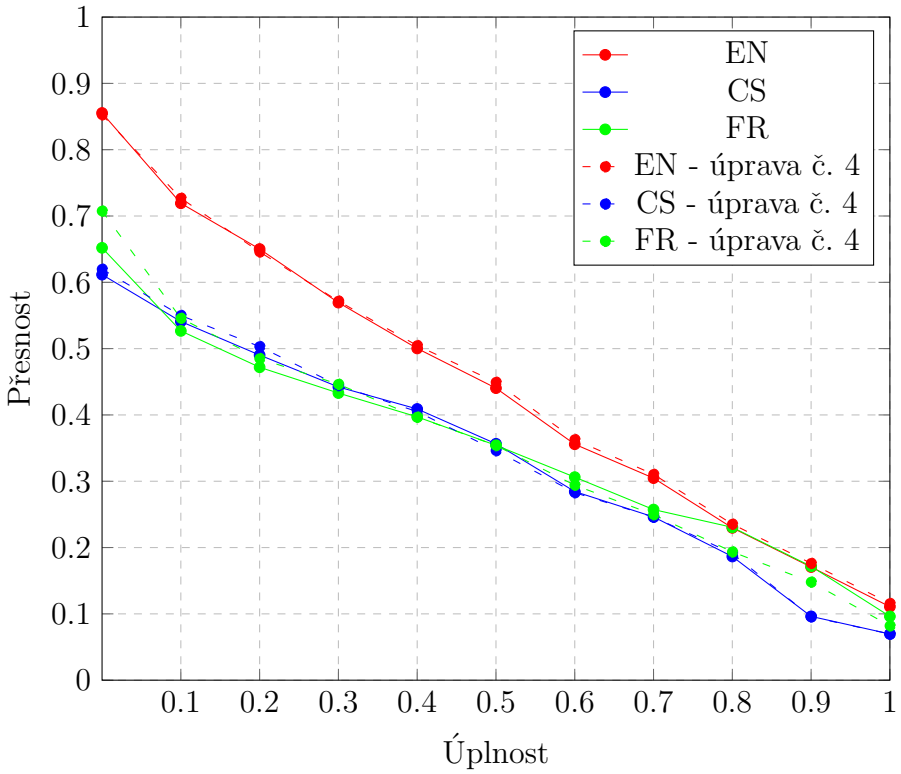

Obrázek B.1: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 1 (ve filtru `shingle_filter` přenastaven parametr `output_unigrams` z hodnoty `false` na hodnotu `true`)



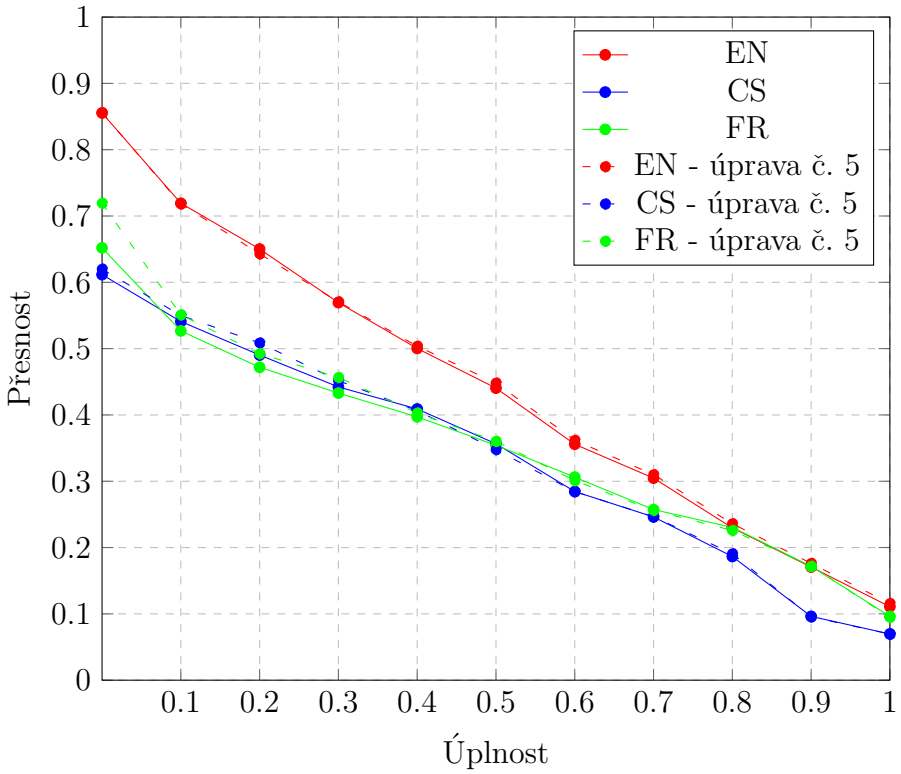

Obrázek B.2: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 2 (filtr `shingle_filter` upraven tak, aby generoval pouze bigramy a trigramy)



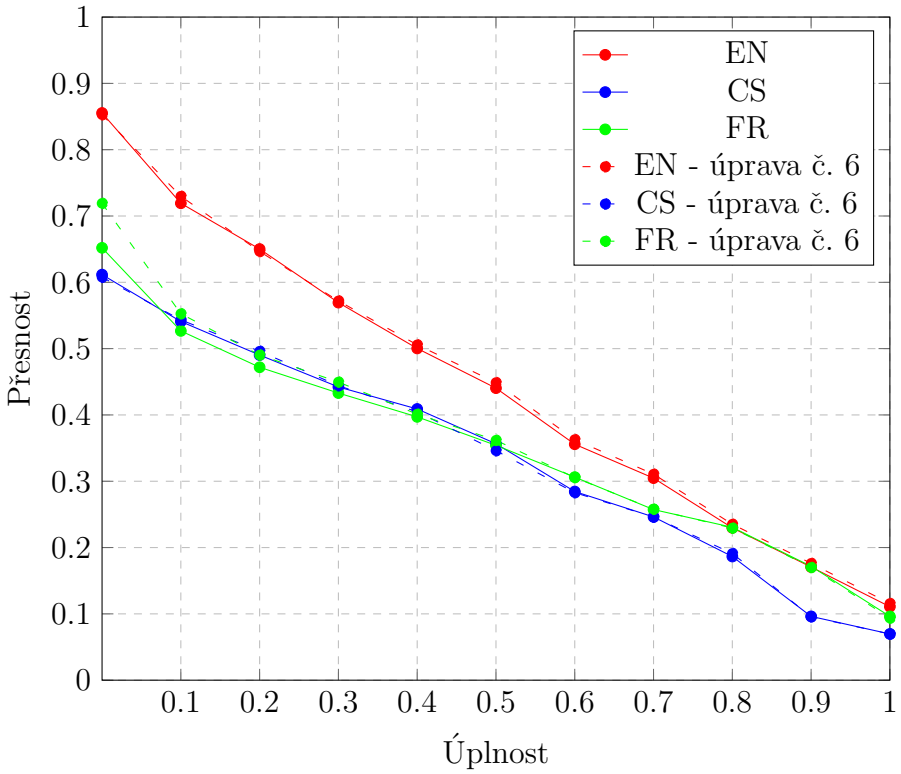

Obrázek B.3: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 3 (filtr `shingle_filter` generuje pouze bigramy)



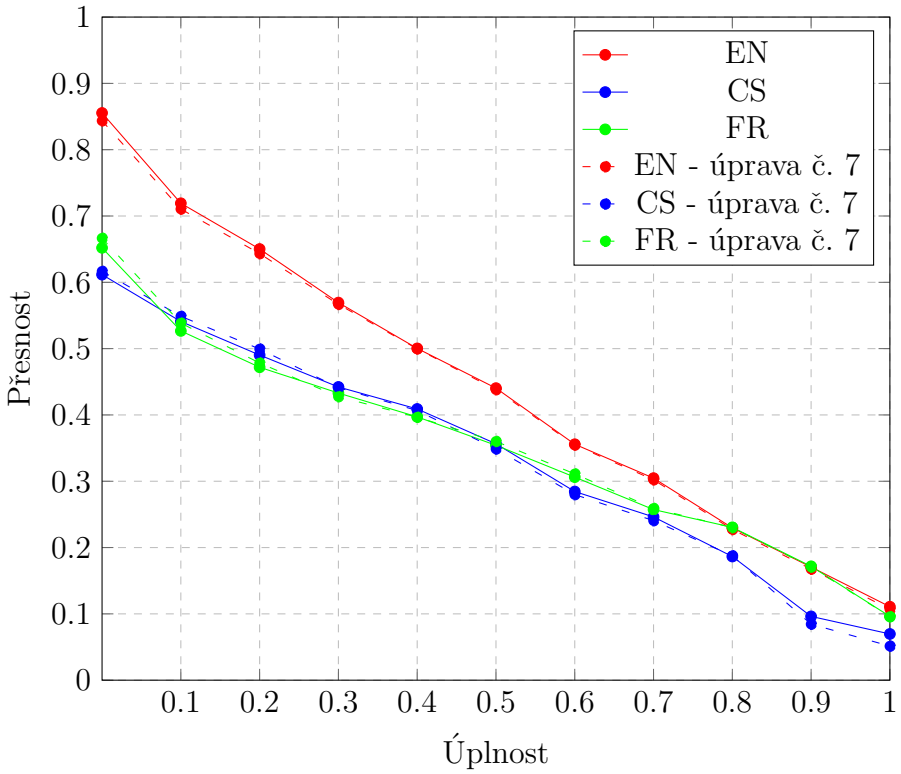

Obrázek B.4: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 4 (filtr `shingle_filter` generuje pouze trigramy)




Obrázek B.5: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 5 (filtr `shingle_filter` generuje pouze čtyřgramy)



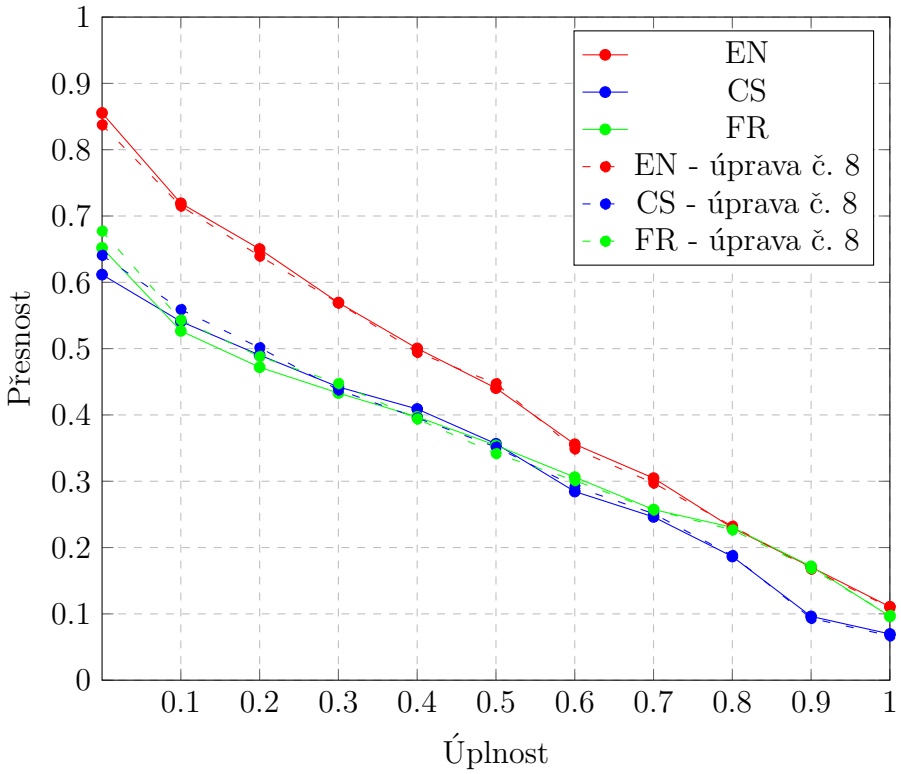

Obrázek B.6: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 6 (filtr `shingle_filter` generuje pouze trigramy a čtyřgramy)




Obrázek B.7: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 7 (v analyzátoru `custom_analyzer` se nepoužívá filtr `icu_folding`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.61 GB | 0.03 | 1.01 |
| CS | 1.37 GB | 1.35 GB | 0.02 | 1.01 |
| FR | 3.21 GB | 3.28 GB | −0.07 | 0.98 |

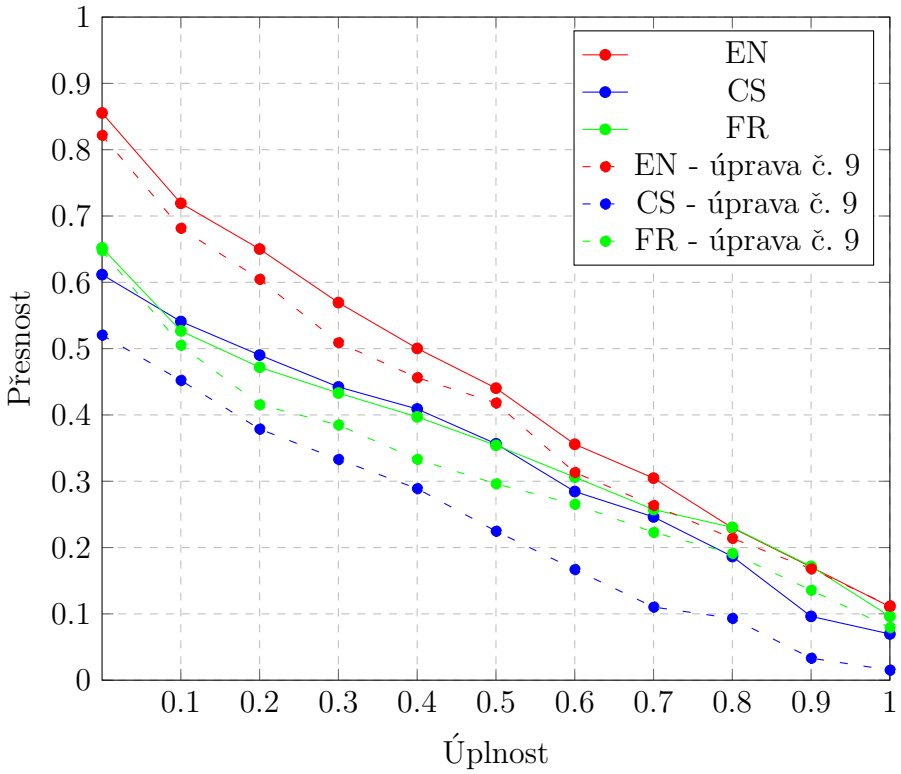
Tabulka B.1: Velikosti indexů po úpravě č. 7




Obrázek B.8: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 8 (v analyzátoru `custom_analyzer` nejsou použity filtry odstraňující stop slova)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.66 GB | −0.02 | 0.99 |
| CS | 1.37 GB | 1.38 GB | −0.01 | 0.99 |
| FR | 3.21 GB | 3.31GB | −0.10 | 0.97 |

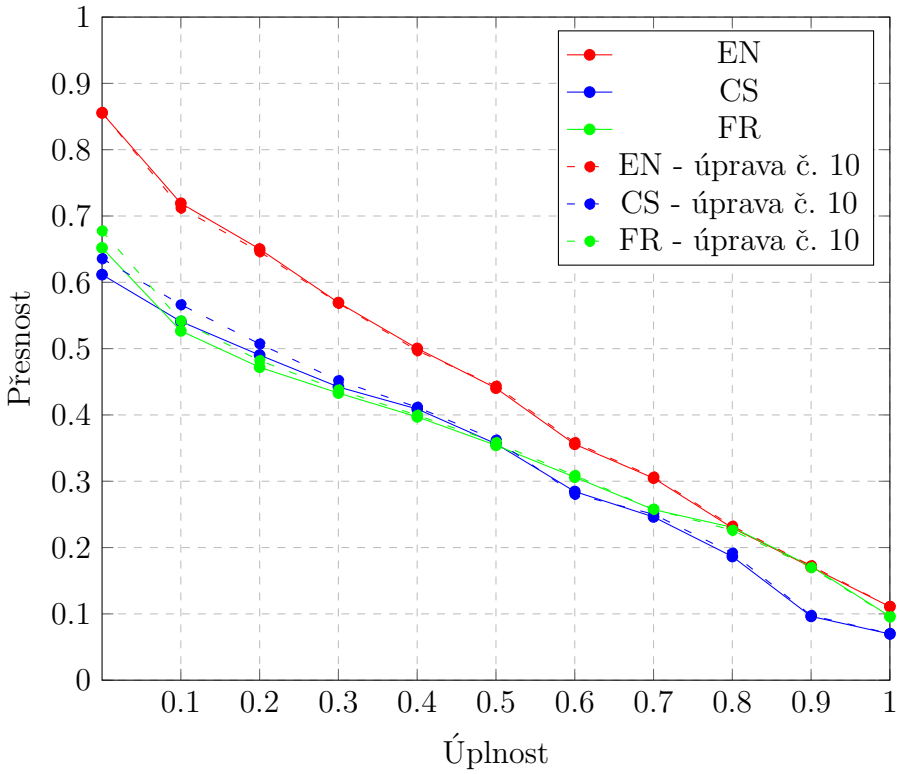
Tabulka B.2: Velikosti indexů po úpravě č. 8




Obrázek B.9: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 9 (v analyzátoru `custom_analyzer` není používán stemming)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.64 GB | 0.00 | 1.00 |
| CS | 1.37 GB | 1.40 GB | −0.03 | 0.98 |
| FR | 3.21 GB | 3.29 GB | −0.08 | 0.98 |

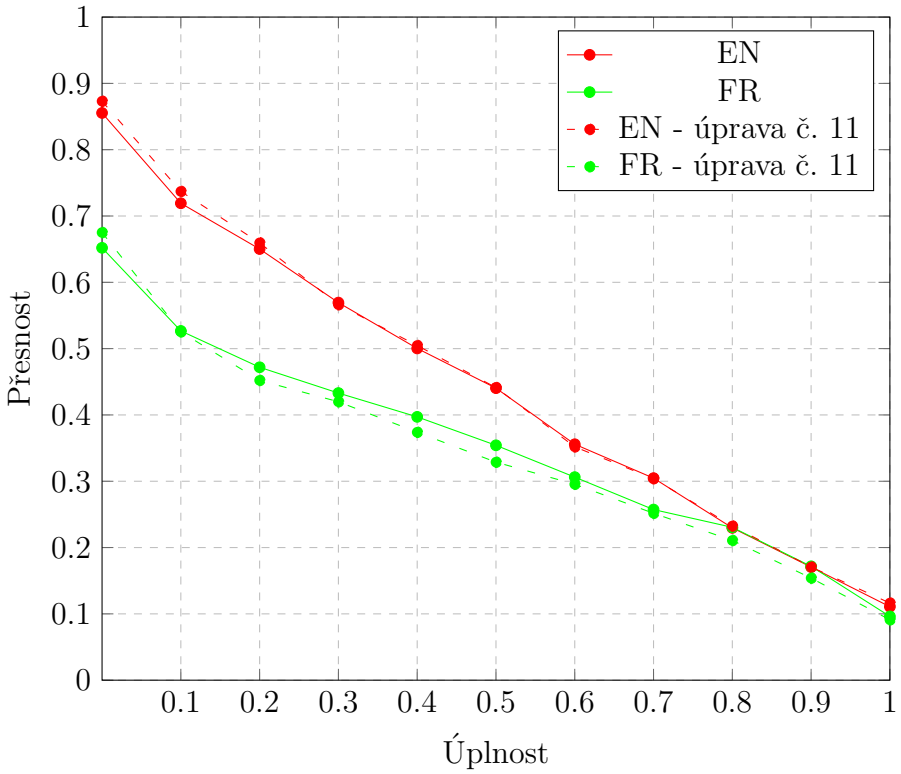
Tabulka B.3: Velikosti indexů po úpravě č. 9




Obrázek B.10: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 10 (v analyzátoru `custom_analyzer` není používán filtr `icu_normalizer` pro normalizaci)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.65 GB | −0.01 | 1.00 |
| CS | 1.37 GB | 1.42 GB | −0.05 | 0.96 |
| FR | 3.21 GB | 3.27 GB | −0.06 | 0.98 |

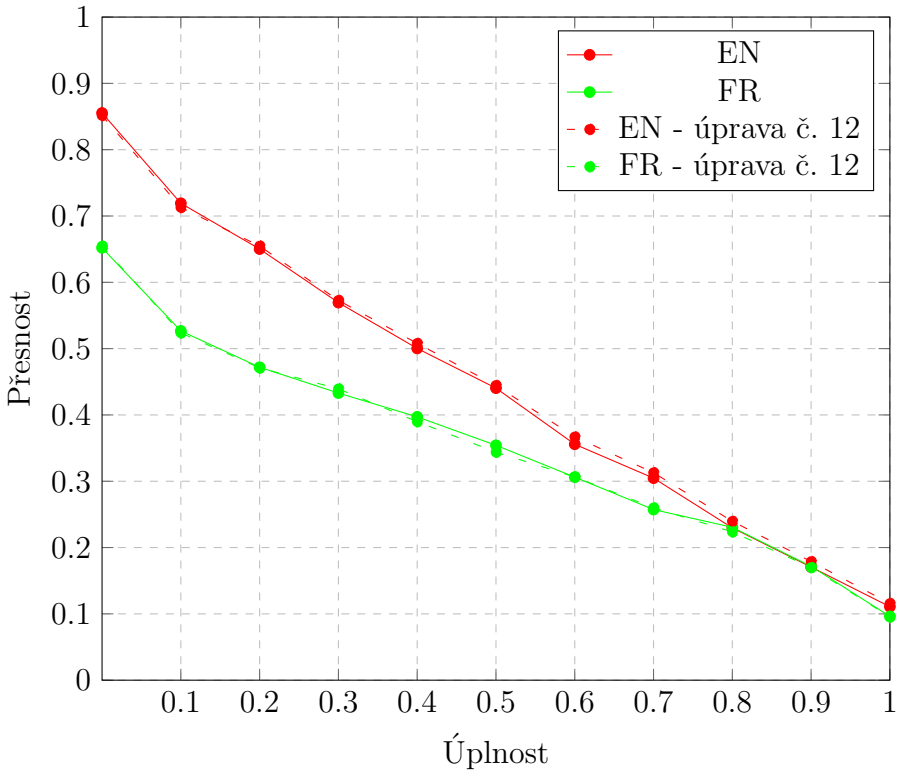
Tabulka B.4: Velikosti indexů po úpravě č. 10




Obrázek B.11: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 11 (v analyzátoru `custom_analyzer` pro index francouzštiny se nepoužívá filtr `elision` a pro index angličtiny není používán filtr `english_possessive_stemmer`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.67 GB | −0.03 | 0.99 |
| FR | 3.21 GB | 3.29 GB | −0.08 | 0.98 |

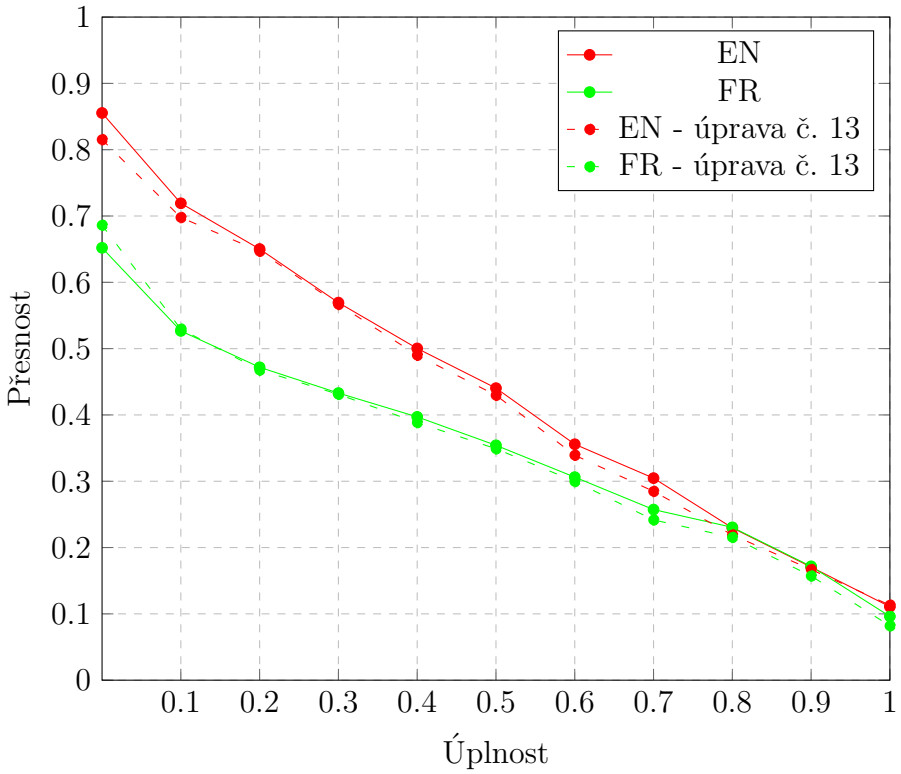
Tabulka B.5: Velikosti indexů po úpravě č. 11




Obrázek B.12: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 12 (v indexu pro francouzštinu je použit stemmer `french`, v indexu pro angličtinu je použit stemmer `light_english`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.64 GB | 0.00 | 1.00 |
| FR | 3.21 GB | 3.36 GB | −0.15 | 0.96 |

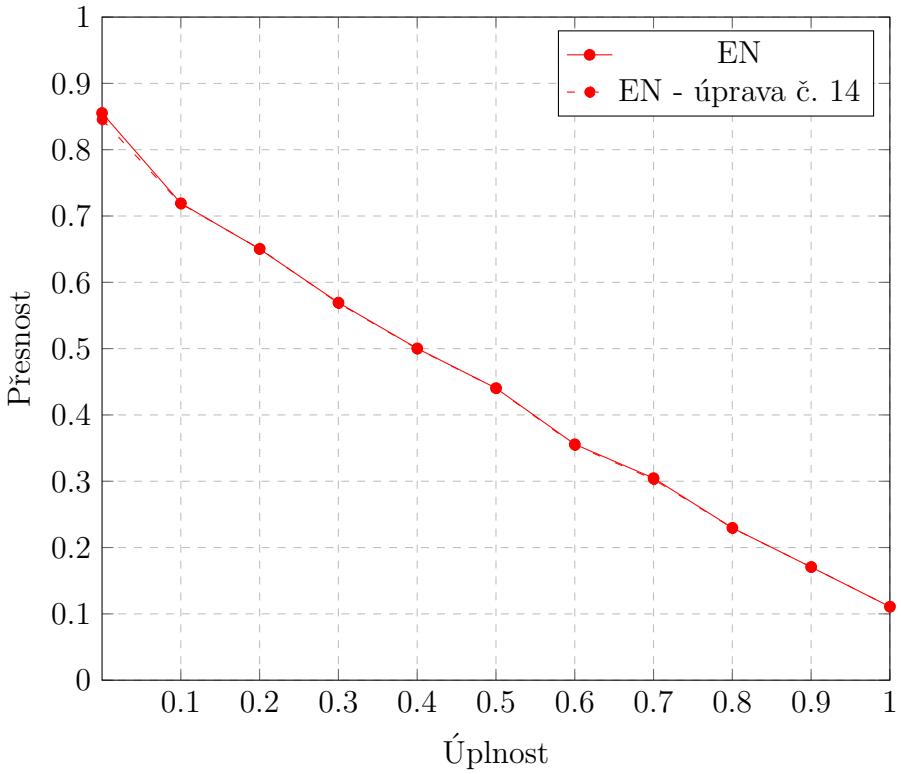
Tabulka B.6: Velikosti indexů po úpravě č. 12




Obrázek B.13: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 13 (v indexu pro francouzštinu je použit stemmer `minimal_french`, v indexu pro angličtinu je použit stemmer `minimal_english`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.65 GB | −0.01 | 1.00 |
| FR | 3.21 GB | 3.37 GB | −0.16 | 0.95 |

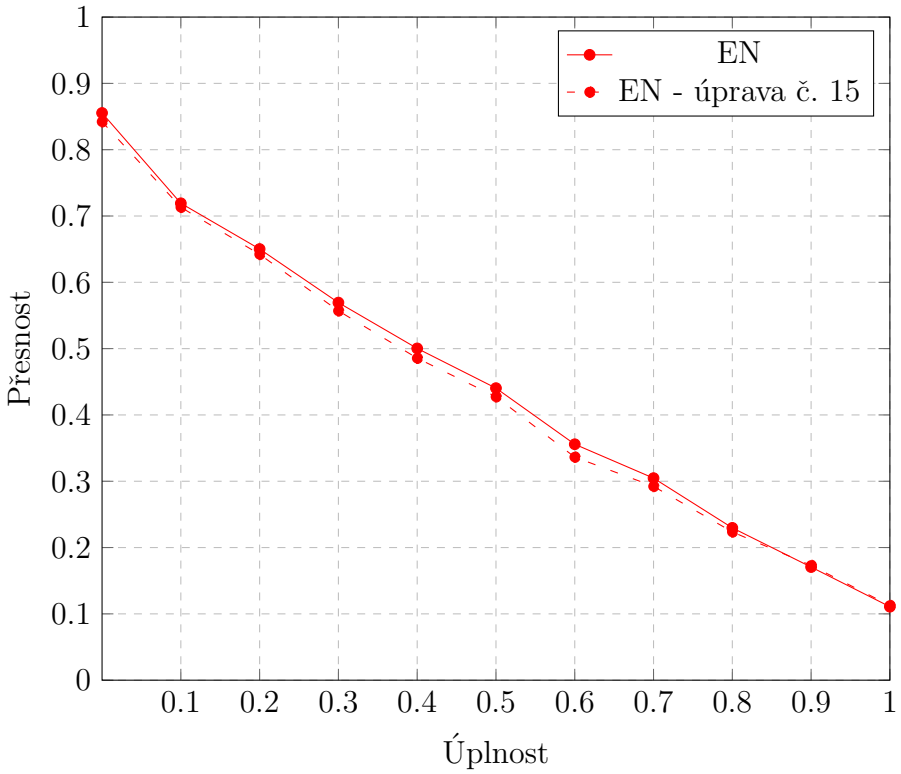
Tabulka B.7: Velikosti indexů po úpravě č. 13




Obrázek B.14: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 14 (v indexu pro angličtinu je použit stemmer `porter2`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.64 GB | 0.00 | 1.00 |

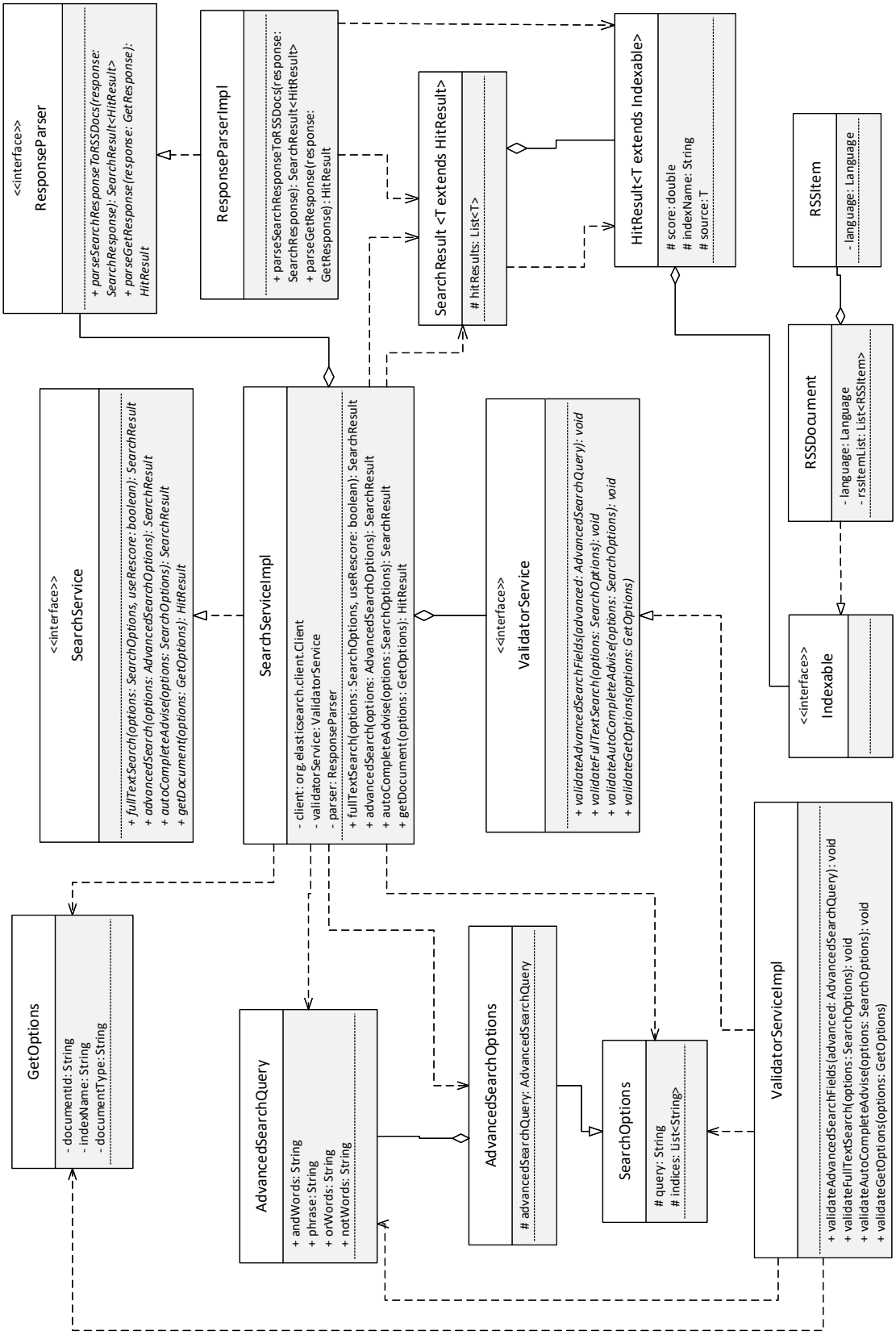
Tabulka B.8: Velikost anglického indexů po úpravě č. 14




Obrázek B.15: Přesnost/úplnost graf testovacích dat pro úpravu indexu č. 15 (v indexu pro angličtinu je použit stemmer `lovins`)

| **Jazyk** | **Velikost indexu** | **Velikost indexu po úpravě** | **Rozdíl** | **Poměr** |
|---|---|---|---|---|
| EN | 2.64 GB | 2.64 GB | 0.00 | 1.00 |

Tabulka B.9: Velikosti indexů po úpravě č. 15




Obrázek B.16: Diagram tříd používaných při vyhledávání

# C Nastavení indexů

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 4,
          "output_unigrams": false },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"] },
        "czech_stop":{
          "type":"stop",
          "stopwords": "_czech_" },
        "czech_stemmer": {
          "type": "stemmer",
          "language": "czech" }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "czech_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "czech_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "icu_normalizer",
            "czech_stop",
            "czech_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.1: Standardní nastavení indexu pro češtinu

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 4,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "english_stop":{
          "type":"stop",
          "stopwords": "_english_"
        },
        "english_stemmer": {
          "type": "stemmer",
          "language": "english"
        },
        "english_possessive_stemmer": {
          "type": "stemmer",
          "language": "possessive_english"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "english_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "english_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "english_possessive_stemmer",
            "icu_normalizer",
            "english_stop",
            "english_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.2: Standardní nastavení indexu pro angličtinu

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 4,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "french_stop":{
          "type":"stop",
          "stopwords": "_french_"
        },
        "french_stemmer": {
          "type": "stemmer",
          "language": "light_french"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "french_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "french_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "icu_normalizer",
            "elision",
            "french_stop",
            "french_stemmer",
            "icu_folding" ]
 }}}},
```

Ukázka kódu C.3: Standardní nastavení indexu pro francouzštinu

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 4,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "german_stop":{
          "type":"stop",
          "stopwords": "_german_"
        },
        "german_stemmer": {
          "type": "stemmer",
          "language": "light_german"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "german_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "german_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "icu_normalizer",
            "german_stop",
            "german_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.4: Standardní nastavení indexu pro němčinu

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 4,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "italian_stop":{
          "type":"stop",
          "stopwords": "_italian_"
        },
        "italian_stemmer": {
          "type": "stemmer",
          "language": "light_italian"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "italian_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "italian_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "icu_normalizer",
            "italian_stop",
            "italian_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.5: Standardní nastavení indexu pro italštinu

```
"mappings": {
    "rss-document": {
      "_all": { "enabled": false},
      "properties": {
        "title":{
          "type": "text",
          "fields": {
            "analyzed":{
              "type": "text",
              "analyzer": "custom_analyzer"
            },
            "shingle":{
              "type": "text",
              "analyzer": "shingle_analyzer"
            }
          }
        },
        "language":{
          "type": "keyword",
          "index": "false"
        },
        "pubDate":{
          "type": "date",
          "format": "yyyy-MM-dd HH:mm:ss z||epoch_millis||dd.MM.yyyy"
        },
        "description" : {
          "type": "text",
          "analyzer": "custom_analyzer",
          "fields": {
            "not_analyzed":{
                "type": "text"
            },
            "shingle":{
              "type": "text",
              "analyzer": "shingle_analyzer"
            }
          }
        },
        "commentSummary":{
          "type": "text",
          "analyzer": "custom_analyzer"
        },
        "summary":{
          "type": "text",
          "analyzer": "custom_analyzer",
          "fields": {
            "shingle":{
              "type": "text",
              "analyzer": "shingle_analyzer_stop"
            }
          }
        },
```

Ukázka kódu C.6: Standardní mapování indexů, část 1

```
        "items":{
          "type": "nested",
          "properties": {
            "title":{
              "type": "text",
              "fields": {
                "analyzed":{
                  "type": "text",
                  "analyzer": "custom_analyzer"
                },
                "shingle":{
                  "type": "text",
                  "analyzer": "shingle_analyzer"
                }
              }
            },
            "guid":{
              "type" : "keyword",
              "index": "no"
            },
            "link": {
              "type" : "text",
              "analyzer": "url_analyzer"
            },
            "language":{
              "type": "keyword",
              "index": "false"
            },
            "pubDate":{
              "type": "date",
              "format": "yyyy-MM-dd HH:mm:ss z||epoch_millis||dd.MM.yyyy"
            },
            "description" : {
              "type": "text",
              "analyzer": "custom_analyzer",
              "fields": {
                "shingle":{
                  "type": "text",
                  "analyzer": "shingle_analyzer_stop"
                }
              }
            },
            "commentSummary":{
              "type": "text",
              "analyzer": "custom_analyzer"
            },
            "text":{
              "type": "text",
              "analyzer": "custom_analyzer",
              "fields": {
                "shingle":{
                  "type": "text",
                  "analyzer": "shingle_analyzer_stop"
}}}}}}}}
```

Ukázka kódu C.7: Standardní mapování indexů, část 2

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 2,
          "max_shingle_size": 2,
          "output_unigrams": false },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"] },
        "czech_stop":{
          "type":"stop",
          "stopwords": "_czech_" },
        "czech_stemmer": {
          "type": "stemmer",
          "language": "czech" }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "czech_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "czech_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "czech_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.8: Upravené nastavení indexu pro češtinu, které je vytvořené na základě výsledků testování

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 3,
          "max_shingle_size": 3,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "english_stop":{
          "type":"stop",
          "stopwords": "_english_"
        },
        "english_stemmer": {
          "type": "stemmer",
          "language": "light_english"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "english_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "english_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "icu_normalizer",
            "english_stop",
            "english_stemmer",
            "icu_folding" ]
}}}},
```

Ukázka kódu C.9: Upravené nastavení indexu pro angličtinu, které je vytvořené na základě výsledků testování

```
"settings" : {
    "number_of_replicas" : 0,
    "analysis": {
      "filter": {
        "shingle_filter": {
          "type" : "shingle",
          "min_shingle_size": 3,
          "max_shingle_size": 4,
          "output_unigrams": false
        },
        "url_stop" : {
          "type" : "stop",
          "stopwords" : ["http", "https", "ftp", "www"]
        },
        "french_stop":{
          "type":"stop",
          "stopwords": "_french_"
        },
        "french_stemmer": {
          "type": "stemmer",
          "language": "light_french"
        }},
      "analyzer": {
        "url_analyzer":{
          "type": "custom",
          "tokenizer": "lowercase",
          "char_filter": [ "html_strip" ],
          "filter" : [
            "icu_normalizer",
            "url_stop",
            "french_stop" ]
        },
        "shingle_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "shingle_filter" ]
        },
        "shingle_analyzer_stop":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter" : [
            "icu_normalizer",
            "french_stop",
            "shingle_filter" ]
        },
        "custom_analyzer":{
          "type": "custom",
          "tokenizer": "icu_tokenizer",
          "filter": [
            "elision",
            "french_stop",
            "french_stemmer",
            "icu_folding" ]
 }}}},
```

Ukázka kódu C.10: Upravené nastavení indexu pro francouzštinu, které je vytvořené na základě výsledků testování

# D Obsah DVD

Přiložené DVD obsahuje text práce ve formátu `pdf` spolu s jeho zdrojovými LaTeX soubory (adresář **text**). Dále obsahuje složku **dist** obsahující Elasticserch, který je možné spustit. V přiloženém Elasticsearch jsou již zaindexována data systému MediaGist. Pro spuštění webové aplikace je ve složce **jetty-web-app** umístěna distribuce servletového kontejneru, který po spuštění také automaticky spustí webovou aplikaci. Zdrojové soubory jsou umístěny ve složce **src**. V této složce jsou obsaženy projekty pro knihovnu, klienta i webovou aplikaci. Ve složce **data** jsou umístěna vzorová data, která je možné indexovat a také tabulka s naměřenými hodnotami při testování. V adresáři **poster** jsou umístěny soubory s posterem ve formátu `pdf` a `pub`. Adresářová struktura DVD je vidět na obr. D.1.

- `/`
  - `bin` ..... Adresář obsahující spustitelné soubory aplikací.
    - `Client.jar` ..... Spustitelný `jar` soubor klienta pro indexaci.
    - `web-app.war` ..... Soubor obsahující webovou aplikaci.
  - `data` ..... Adresář obsahující vzorové dokumenty, které je možné indexovat pomocí klienta pro indexaci a dále naměřená data při testování.
  - `dist` ..... Adresář se spustitelnými programy pro prezentaci diplomové práce.
    - `client` ..... Program klienta pro indexaci.
    - `elasticsearch` ..... Elasticsearch s již zaindexovanými daty systému MediaGist.
    - `jetty-web-app` ..... Servletový kontejner `Jetty` s vytvořenou webovou aplikací.
  - `poster` ..... Adresář obsahující poster ve formátu .pdf a .pub.
  - `src` ..... Adresář obsahující zdrojové soubory.
    - `Project` ..... Projekt knihovny pro vyhledávání.
    - `Project-Client` ..... Projekt klienta pro indexaci.
    - `Project-Web` ..... Projekt webové aplikace.
  - `text` ..... Adresář obsahující text práce.
    - `latex` ..... Zdrojové soubory textu práce v LaTeX.
    - `DP_pribanp.pdf` ..... Text práce ve formátu `pdf`.
  - `Obsah_DVD.txt` ..... Textový soubor se strukturou DVD.

Obrázek D.1: Adresářová struktura přiloženého DVD