Komplexní IR systém
KIV/IR

David Markov

Fakulta aplikovaných věd
Západočeská univerzita v Plzni
8. května 2023

Obsah

1 Zadání

Cílem semestrální práce je naučit se implementovat komplexní IR systém s využitím hotových knihoven pro preprocessing. Vedlejším produktem bude hlubší porozumění indexerům, vyhledávacím systémům a přednáškám.

Systém po předchozím předzpracování zaindexuje zadané dokumenty a poté umožní vyhledávání nad vytvořeným indexem. Vyhledávání je možné zadáním dotazu s logickými operátory AND, OR, NOT a s použitím závorek. Výsledek dotazu by měl vrátit top x (např. 10) relevantních dokumentů seřazených dle relevance.

Minimální nutná funkčnost semestrání práce pro získání 15 bodů (a tedy potenciálně zápočtu):

Tokenizace, preprocessing (stopwords remover, stemmer/lemmatizer), vytvoření in-memory invertovaného indexu, tf-idf model, cosine similarity, vyhledávání dotazem vrací top x výsledků seřazených dle relevance (tj. vektorový model - vector space model), vyhledávání logickými operátory AND, OR, NOT (booleovský model), podrobná dokumentace (programátorská i uživatelská), podpora závorek pro vynucení priority operátorů.

Semestrální práce musí umožňovat zaindexování dat stažených na cvičení (1. bodované cvičení Crawler) a libovolných dalších dat ve stejném formátu. Obě sady dat je možné zaindexovat nezávisle na sobě.

Semestrální práce musí umožňovat zadávat dotazy z GUI nebo CLI (command line interface) a při zadávání dotazů je možno vybrat index a model vyhledávání (vector space model vs. boolean model). Výsledky vyhledávání obsahují i celkový počet dokumentů, které odpovídají zadanému dotazu.

Nadstandardní funkčnost (lze získat až dalších 15 bodů), např.:

- File-based index
- Pozdější doindexování dat - přidání nových dat do existujícího indexu
- Ošetření HTML tagů
- Detekce jazyka dotazu a indexovaných dokumentů
- Vylepšení vyhledávání
- Vyhledávání frází (i stop slova)
- Vyhledávání v okolí slova
- Více scoring modelů
- Indexování webového obsahu - zadám web, program stáhne data a rovnou je zaindexuje do existujícího indexu
- Další předzpracování normalizace
- GUI/webové rozhraní

- Napovídání keywords
- Podpora více polí pro dokument (např. datum, od do)
- Zvýraznění hledaného textu v náhledu výsledků
- Vlastní implementace parsování dotazů bez použití externí knihovny
- Implementace dalšího modelu (použití sémantických prostorů, doc2vec, Transformers - BERT) atd.

2 Popis implementovaného systému

V rámci semestrální práce byl vyvinut systém pro vyhledávání a indexaci dokumentů. Aplikace uživateli umožňuje vyhledávat v zaindexovaných dokumentech pomocí řetězcových dotazů z uživatelského rozhraní. Výsledky vyhledávání jsou uživateli následně kompaktně zobrazovány.

Systém navíc umožňuje spuštění bez přidaných vlastních dat, kdy si sám vyhledá data z odpovídajícího webu a automaticky je zaindexuje ihned po startu.

2.1 Uživatelská příručka

Pro spuštění aplikace je třeba mít nastaven pracovní adresář jako kořenový adresář projektu a z terminálu spustit následující:

```
java -jar search-engine.jar [-OPTION]
```

Při spuštění systému z příkazové řádky je možné předávat následující parametry:

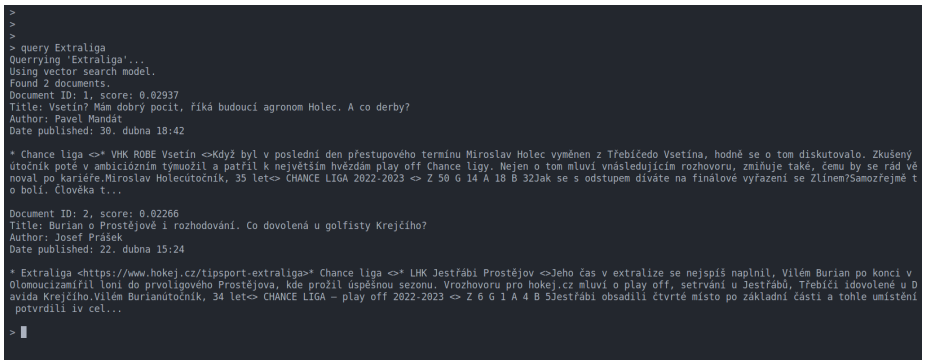
- `-h (--help)` - vypíše nápovědu
- `-s (--storage) <memory/file>` - specifikuje implementaci úložiště dokumentů k pozdějšímu zaindexování - v paměti nebo na disku (výchozí variantou je implementace diskového úložiště)
  - POZNÁMKA: paměťová implementace úložiště dokumentů je určena pouze pro účely vývoje a neměla by se využívat při reálném běhu aplikace; aktuálně není toto úložiště podporováno a jeho volba ukončí systém výjimkou jelikož některé operace rozhraní úložiště nejsou momentálně podporovány

Požadavkem pro spuštění předem sestaveného spustitelného souboru je nainstalovaný *Java Runtime Environment (JRE) 17* nebo novější.

Po spuštění může uživatel se systémem interagovat pomocí uživatelského rozhraní a ovládat tak vyhledávací engine. Podporované příkazy jsou:

- `clear` - vyčistí obrazovku terminálu
- `exit` - korektně ukončí aplikaci
- `query <query string> [--model <boolean/vector]` - vyhledá v indexu dokumenty, které odpovídají zadanému dotazu
  - volitelným parametrem `--model` lze specifikovat jaký vyhledávací model se má při vyhledávání použít; aktuálně podporovanými variantami jsou *Booleovský model* a *Vektorový model*, přičemž vektorový model je výchozí variantou
- `url <url>` - vyhledá zadanou URL (musí být absolutní z `<https://hokej.cz>` nebo relativní), ze které zpracuje článek; zpracovaný článek je uložen do úložiště dokumentů a je zaindexován při dalším spuštění aplikace

Po zadání dotazu jsou vyhledány odpovídající dokumenty v indexu a jsou kompaktně prezentovány na uživatelském rozhraní. Uveden je použitý vyhledávací model, počet nalezených dokumentů a u každého nalezeného dokumentu je uvedeno jeho celočíselné ID, skóre, titulek, autor článku, datum publikace a krátký úryvek ze začátku obsahu. Seznam vyhledaných výsledků je navíc seřazen podle skóre sestupně (viz obr. 1).



Obrázek 1: Uživatelské rozhraní systému, prezentující výsledky vyhledávání.

2.2 Sestavení systému z přiložených zdrojových souborů

Spustitelný soubor systému je samozřejmě možné snadno sestavit z přiložených zdrojových souborů. Stačí jednoduše nastavit pracovní adresář jako kořenový adresář projektu a z terminálu spustit příkaz

```
mvn clean package
```

Tento příkaz sestaví všechny *Maven* moduly systému a vytvoří spustitelný `.jar` soubor v adresáři `core/target` s názvem `search-engine-<verze>-jar-with-dependencies.jar`. Ten je následně možné spustit stejným způsobem, jak již bylo zmíněno dříve.

Požadavky pro sestavení ze zdrojových souborů jsou nainstalované programy *Maven* a *Java Development Kit (JDK) 17* nebo novější.

2.3 Indexace vlastních dokumentů

Systém umožňuje snadno indexovat své vlastní dokumenty, které odpovídají očekávanému formátu. Jednoduše stačí dokumenty přidat do adresáře úložistě dokumentů `storage` a při dalším spuštění budou zaindexovány. Pokud soubor nebude odpovídat předpokládanému formátu, bude zalogována chyba a přejde se na načtení dalšího souboru.

3 Detaily implementace

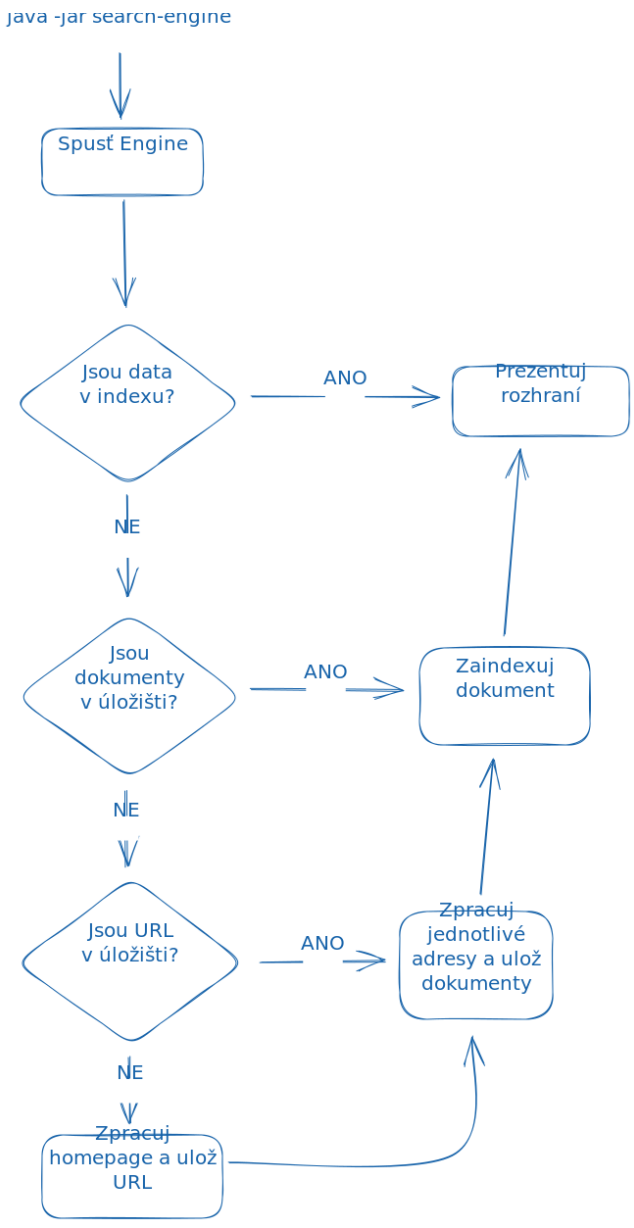
Spuštění systému je provedeno v několika po sobě jdoucích krocích.

Nejprve se zkontroluje přítomnost zaindexovaných dat v indexu. Pokud jsou nalezena zaindexovaná data, je uživateli ihned prezentováno uživatelské rozhraní. Je třeba zmínit, že tento krok byl přidám primárně pro účely implementace indexu v souboru, který uchovává data napříč jednotlivými běhy systému. Jelikož je podporován pouze paměťový, data nebudou při startu nikdy nalezena a spouštění postoupí do druhé fáze.

Ve druhém kroku je zkontrolována přítomnost dat v úložišti dokumentů k indexaci v adresáři `storage`. Pokud jsou uloženy záznamy k indexaci, jsou objektem `Storage` načteny do paměti a zaindexovány. Pokud v úložišti nejsou žádné záznamy, spouštění postoupí do třetího kroku.

V tomto kroku následuje kontrola úložiště URL adres článků ke zpracování Crawlerem, uchování článků do úložiště a jejich následná indexace. Procházení potřebného webu a zpracování jednotlivých článků je časově náročné a proto je implementován mezikrok, ve kterém jsou tyto články založeny do úložiště. To umožňuje výrazně urychlit budoucí běhy vyhledávacího enginu, jelikož nemusí články neustále zpracovávat.

Pokud nejsou přítomny ani uložené URL adresy jednotlivých článků, je nejprve zpracována domovská stránka webu `<https://hokej.cz>`, ze které jsou načteny všechny potřebné URL a jsou přidány do úložiště, aby je nebylo nutné znovu načítat v příštím běhu. Celý proces je znázorněn v sekvenčním diagramu na obrázku 2.



Obrázek 2: Sekvenční diagram, popisující kroky spuštění systému.

3.1 Moduly systému

Systém je rozdělen do dvou samostatných modulů, na kterých je dále závislý modul jádra systému.

Search Engine Utilities je modul, obsahující pomocné metody a funkce, které je možné využít napříč širokou škálou systémů. Jsou to například pomocné funkce pro práci se soubory, řetězci či URL odkazy. Dále jsou zde implementovány třídy, specifické pro tento systém, které mohou být dále využity v dalších modulech systému. Těmi jsou například třída *Storage*, zodpovědná za správu úložiště dokumentů pro indexaci nebo *FileLoader*, který poskytuje rozhraní pro načítání objektů ze souborů.

Search Engine Crawler obsahuje veškerou logiku, potřebnou k procházení webu, zpracování webových stránek a získávání potřebných dat z jednotlivých článků. Základem celého modulu je třída *Crawler*, která pro všechny tyto operace poskytuje rozhraní. Modul je závislý na již zmíněném modulu `utils`, jelikož využívá například třídu *Storage* k ukládání obsahu zpracovaných článků z webu do úložiště.

Search Engine Core je nosným prvkem celého systému. Obsahuje logiku spouštění aplikace, uživatelské rozhraní, které zpracovává požadavky uživatele, implementaci indexu pro vyhledávání dokumentů i parser jednotlivých dotazů do indexu. Spojuje celý systém do jednoho celistvého programu a je závislý na všech ostatních samostatných modulech projektu. Pro indexaci dokumentů poskytuje rozhraní *Index* a jeho implementaci *TfIdfIndex*, která pro indexování a vážení jednotlivých dokumentů a slov používá výpočet *TF-IDF*.

Logování v celém systému je zajištěno knihovnou *SLF4J*. Všechny logy jsou zaznamenávány na různých úrovních detailu do souboru v adresáři `logs`. Zde jsou logy uchovávany po jeden den, poté jsou přeměněny do komprimované podoby pro pozdější využití.

4 Implementovaná nadstandardní funkčnost

Zde je uveden seznam nadstandardních implementovaných funkčností oproti základnímu zadání:

- Česká dokumentace v TeXu a anglické README v Markdownu
- Systém logování s následnou komprimací záznamů po uplynutí časového intervalu.
- Modularizace projektu do dílčích *Maven* modulů
- Snadno rozšiřitelná parametrizace spouštění systému pomocí přepínačů

- Cache jednotlivých kroků při startu systému
- Integrace web-crawleru do systému
- Crawlování zadané URL adresy a získání článku, který je později zaindexován

5 Výsledky TREC evaluace

Při evaluaci třídou *TestTrecEval.java* bylo dosaženo následujících výsledků:

- *MAP* při vyhledání pomocí `Topic::getTitle` - 0.1074
- *MAP* při vyhledání pomocí `Topic::getNarrative` - 0.1004
- *MAP* při vyhledání pomocí `Topic::getDescription` - 0.1085
- *MAP* pri vyhledání spojením `Topic::getTitle` a `Topic::getNarative` - 0.1348
- *MAP* pri vyhledání spojením `Topic::getTitle` a `Topic::getDescription` - 0.1173

Nejlepší výsledek tedy dosáhla evaluace vyhledáváním `Topic::getTitle` a `Topic::getNarrative`. Celý výstup lze vidět na obrázku 3.

```
./trec_eval.8.1/./trec_eval ./trec_eval.8.1/czech ./TREC/results-2023-05-08_17_48_115.txt
TREC EVAL output:
num_q           all     50
num_ret         all     500
num_rel         all     762
num_rel_ret     all     120
map             all     0.1348
gm_ap           all     0.0065
R-prec          all     0.1989
bpref           all     0.1611
recip_rank      all     0.4012
ircl_prn.0.00   all     0.4314
ircl_prn.0.10   all     0.3493
ircl_prn.0.20   all     0.2640
ircl_prn.0.30   all     0.1941
ircl_prn.0.40   all     0.1609
ircl_prn.0.50   all     0.1503
ircl_prn.0.60   all     0.0718
ircl_prn.0.70   all     0.0453
ircl_prn.0.80   all     0.0222
ircl_prn.0.90   all     0.0125
ircl_prn.1.00   all     0.0125
P5              all     0.2400
P10             all     0.2400
P15             all     0.1600
P20             all     0.1200
P30             all     0.0800
P100            all     0.0240
P200            all     0.0120
P500            all     0.0048
P1000           all     0.0024
```

Obrázek 3: Výsledek evaluace nad sadou dokumentů TREC s použitím `Topic::getTitle` a `Topic::getNarrative` jako dotazu.