Booleovský model

Daniel Honys

Popis projektu

Cílem projektu je implementace Booleovského modelu pro získávání informací z kolekce textových dokumentů. Model umožňuje uživateli využití booleovského dotazu pro získání seznamu dokumentů, které daný dotaz splňují.

Způsob řešení

Předpokládáme, že máme nějakou kolekci textových dokumentů. Před schopností dotazování nad kolekcí je nutné jednotlivé dokumenty kolekce předzpracovat. Z jednotlivých dokumentů jsou vyextrahovány dostatečně výrazné termy, které se v textu dokumentu vyskytují. Takové termy v podstatě tvoří jakýsi souhrn důležitých informací v dokumentu. Termy ze všech dokumentů pak tvoří slovník. Každý term má vlastní list dokumentů, kde se vyskytuje, což je pak využito při vyhodnocení dotazu.

Implementace

Implementace projektu byla rozdělena do podprojektů a pro automatizaci sestavování využívala **Gradle**. Vše bylo vyvíjeno v jazyce **Java**.

bm-scraper

Podprojekt, který má za úkol vytvoření kolekce různých článků z Wikipedie. Využívá odkaz, který uživatele přesměruje na random článek na Wikipedii (Wikipedia Special Random). Ze stránky článku se pak vezme několik odstavců. Pro extrakci odstavců je využita knihovna **jsoup**. Odstavce společně s názvem a odkazem na článek jsou uloženy do souboru. Pro každý článek je jeden soubor.

bm-preprocess

Tento podprojekt se zabývá předzpracováním kolekce. Předzpracování zahrnuje načtení souboru obsahující článek, zpracování textu a extrakce jeho termů, zde jsem využil **CoreNLP** knihovnu od Stanfordovi univerzity. Text článku, jeho termy a relace jsou zapsány do databáze.

Postup:

1. Načtení textu ze souboru
2. Tokenizace
3. Lemmanizace

4. Odstranění nedůležitých znaků a slov (číslovky, speciální znaky, zkratky, ...)
5. Odstranění stopslov (často vyskytujících se slov)
6. Odstranění slov pod určitou délku
7. Filtrace unikátních slov
8. Uložení článku, termů a relací do databáze

Databázové schéma

Jako relační databázi jsem zvolil **MySQL**.

bm-web-app

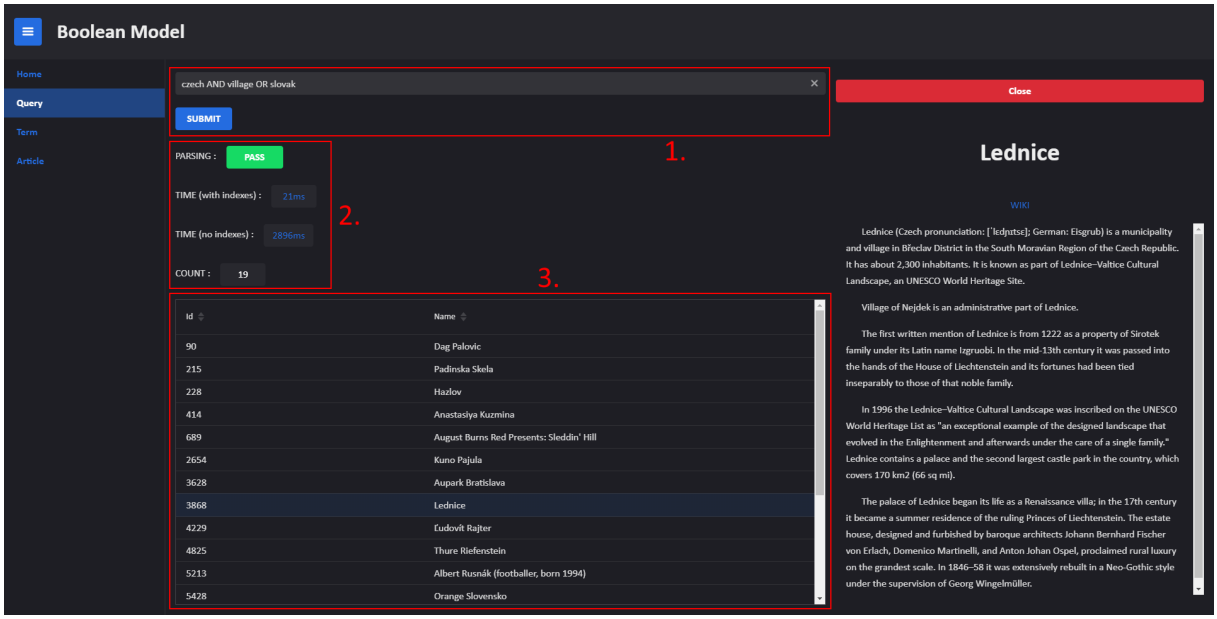
V tomto sub projektu je realizovaná webová aplikace. Aplikace využívá **Spring boot** jako backend framework. A frontend aplikace je realizován pomocí **Vaadin** frameworku.

Uživatel zadá dotaz, který chce provést nad databází. Jako první dojde ke kontrole, zda dotaz splňuje pravidla definované gramatiky. Pro tuto kontrolu jsem využil knihovny **ANTLR v4** (v ní si člověk definuje pravidla gramatiky a knihovna pak kontroluje správnost dotazu). Pokud je dotaz v pořádku, tak je proveden nad databází a výsledek je pak vyobrazen uživateli. Pokud je dotazem něco v nepořádku, je uživateli zobrazena error zpráva.

Aplikace dále umožnuje uživateli si prohlédnou jednotlivé termy a články, které se v databázi nacházejí.

Webová aplikace společně s databází jsou pak hostované za pomoci **Dockeru** ve VPS.

Příklad výstupu



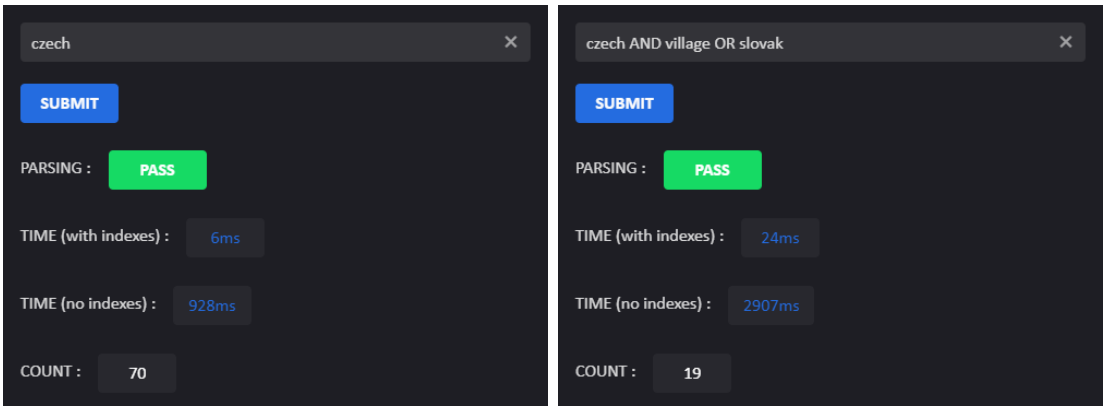
Na obrázku je vidět vyhodnocení dotazu „czech AND village OR slovak“.

1. Místo pro zadání dotazu
2. Informace o vyhodnocení dotazu
3. Seznam dokumentů, které splňují dotaz

Při selekci nějakého článku je uživateli otevřen náhled článku, kde se nachází i odkaz na původní stránku Wikipedie, kde se článek nachází.

Experimentální sekce

V této sekci bych se chtěl podívat na rychlost vyhodnocení dotazu.



Zde můžeme vidět dva různě komplexní dotazy. Vidíme, že komplexnější dotaz trval na vyhodnocení déle a tím pádem můžeme říct, že rychlost vyhodnocení je přímo závislá na komplexitě dotazu.

Při porovnání času s indexem a bez, je vidět že bez použití indexu je vyhodnocení značně pomalejší. Index jako takový je datová struktura, která pomáhá databázovému stroji urychlit procházení tabulky nad kterou je zrovna prováděný dotaz. Bez jeho využití je databázový stroj nucen projít veškeré záznamy v tabulce sekvenčně, proto je čas bez využití indexu o tolik větší.

Diskuse

V projektu je prostor pro přidání pár funkcionalit a optimalizací (např.: u dokumentu ukazovat seznam termů, které se v něm vyskytují; optimalice booleovského dotazu přezávorkováním podle počtu dokumentů ve kterých se term vyskytuje, ...) ale myslím si, že moje řešení je celkem obstojným „proof of concept“.

Závěr

Boolean model se ukázal jako dobrý způsob vyhledávání za předpokladu, že přesně víme, co hledáme. Jeho nevýhodou však je že nebere v potaz žádný ranking či jiné relace mezi dokumenty.