Zadání zápočtového projektu:

Rozšířený booleovský model vyhledávání dokumentů

Zadání

Cílem projektu je implementovat webovou aplikaci, která umožní vyhledávání v kolekci dokumentů s využitím rozšířeného booleovského modelu. Dotaz je tvořen booleovským výrazem, který se skládá z hledaných slov (termů), které jsou spojené logickými spojkami AND, OR a NOT. Jednotlivé části dotazu mohou být navíc různě uzávorkované. Můžeme mít tedy např. dotaz ((dog OR cat) AND NOT mouse). Na vstupu bude tedy booleovský dotaz. Na výstupu bude seznam dokumentů z databáze, které splňují daný výraz. U rozšířeného booleovského modelu jsou dokumenty na výstupu navíc setříděné podle relevance. Dokument, který obsahuje hledané termy vícekrát, bude mít tedy lepší skóre než dokument, ve kterém se hledané termy vyskytují méněkrát.

Maximální počet členů v týmu: 2

Podrobnosti

- V první fázi je potřeba implementovat extrakci termů z kolekce dokumentů.
- Ze seznamu termů odstraníme nevýznamová slova (předložky, spojky, členy, …). Seznam tzv. *stop words* lze získat z nějaké knihovny pro zpracování textu (např. NLTK pro Python), lze si ho stáhnout z webu nebo vygenerovat přímo ze zpracovávané kolekce textů (tj. spočítat si frekvence výskytu všech slov a ta nejčastější využít jako seznam stop words).
- U zbývajících slov provedeme tzv. stemming (získání kmenů slov, tj. částí slov, které se nemění při skloňování nebo časování) případně lematizaci (tj. úpravu na základní tvar slova, např. tvary „go, went, gone“ převedeme na společný tvar „go“). Pro stemming/lematizaci lze využít např. Porter Stemmer, který je dostupný pro mnoho programovacích jazyků nebo knihovnu NLTK.
- Po předzpracování textu máme tedy k dispozici slovník termů a seznam dokumentů. Můžeme tedy sestavit tzv. term-by-document matici. U rozšířeného booleovského modelu nejsou v term-by-document matici pouze hodnoty 0/1, ale reálná čísla udávající váhu termu na základě počtu jeho výskytů v daném dokumentu. Váhy termů v jednotlivých dokumentech ($w_{i,j}$) vypočteme pomocí TF-IDF schématu uvedeného v přednášce č. 3, slides 10-12. Vypočtené váhy můžeme dále normalizovat do intervalu <0,1>.
- Nevýhodou term-by-dokument matice je, že je řídká tj. obsahuje mnoho 0 a její ukládání v hlavní paměti není efektivní. Z tohoto důvodu používáme efektivnější uložení dat ve formě tzv. invertovaného seznamu/indexu.
- V základní verzi reprezentujeme invertovaný index jako spojový seznam, kde pro každé slovo ukládáme setříděný seznam celočíselných identifikátorů jednotlivých dokumentů. V moderních programovacích jazycích lze místo spojového seznamu využít i datovou strukturu „mapa“. Alternativně můžeme invertovaný seznam simulovat i v SQL databázi (viz přednáška č. 2, slide 19), kdy nad sloupečky, nad kterými provádíme operaci WHERE v SQL příkazu „SELECT … FROM … WHERE …“, musí být definovaný index příkazem CREATE INDEX. Databázový stroj pak neprochází všechny řádky v tabulce, ale jen některé s využitím indexovací datové struktury (např. B-strom).
- Sekvenční průchod databáze obecně neznamená implementovat jiný vyhledávací algoritmus. Předzpracování dat je stejné, pouze projdeme všechny dokumenty v databázi (tj. vypneme datovou strukturu sloužící pro zrychlení vyhledávání). V našem případě nepoužijeme invertovaný seznam. Sekvenční průchod v případě simulace invertovaného seznamu v SQL databázi odpovídá zrušení indexu příkazem DROP INDEX.

© Ing. Jiří Novák, Ph.D., 2023

- Libovolně uzávorkovaný výraz lze parsovat s využitím bezkontextové gramatiky ETF (expression-term-factor), kde násobení nahradíme logickou spojkou AND, sčítání spojkou OR a můžeme tak respektovat i prioritu operátorů.
- Relevanci daného dokumentu vzhledem k dotazu vypočteme pomocí vzorců pro AND a OR uvedených v přednášce č. 2, slide 27. V případě uzávorkovaného výrazu se vzorce aplikují rekurzivně. Pokud máme hodnoty normalizované do intervalu <0,1>, relevanci negace výrazu můžeme vypočítat jako $1 - x$, kde x je původní relevance.
- Vzorec pro výpočet relevance umožňuje přiřadit skóre všem dokumentům v databázi včetně těch, které nesplňují zadaný booleovský výraz. Výpis nalezených dokumentů je tedy vhodné oříznout, aby byl zadaný výraz splněn a skóre vypisovat pouze v rámci těchto dokumentů. Alternativně však můžeme zobrazit i dokumenty, které danému výrazu přesně neodpovídají, ale jejich skóre vykazuje určitou podobnost k zadanému dotazu. To může být výhodné zejména v případě, že v databázi nebude žádný dokument, který by přesně odpovídal zadanému dotazu.

Implementace

Aplikace by měla obsahovat následující součásti:

- extrakci termů, identifikaci nevýznamových slov, stemming/lematizaci
- implementaci sekvenčního průchodu databáze a invertovaného seznamu
- výpočet vah termů v jednotlivých dokumentech pomocí TF-IDF a jejich uložení v invertovaném seznamu
- parsování a vyhodnocení libovolně uzávorkovaného výrazu
- výpočet relevance/skóre jednotlivých dokumentů pro zadaný dotaz
- webové rozhraní pro zadání vstupního dotazu (případně i dalších vstupních parametrů) a přehledné zobrazení vyhledaných dokumentů setříděných podle skóre

Data

- Vstupní kolekce dokumentů může být reprezentována adresářem s textovými soubory.
- Zdroj dat může být libovolný. Vhodná datová sada by měla obsahovat stovky až jednotky tisíc dokumentů, aby bylo možné pozorovat zrychlení vyhledávání při použití invertovaného seznamu oproti sekvenčnímu prohledání celé databáze. Textové soubory nemusí být příliš dlouhé. Stačí např. jeden až dva odstavce textu.
- Pro vyhledání vhodné kolekce dokumentů můžete využít klíčové slovo *corpus*, což bývají různé balíky předpřipravených textů pro testování kompresních algoritmů, apod. Alternativně je možné využít offline verzi Wikipedie, stáhnout si nějaké webové stránky očištěné od HTML značek, apod.
- V tomto projektu se předpokládá využití statické kolekce textů. Přidání nového záznamu totiž znamená přepočítání vah všech termů v jednotlivých dokumentech pomocí TF-IDF. Pro reálné použití je proto obvykle praktičtější znovu zkonstruovat index a přepočítat všechny váhy (po uplynutí určitého časového intervalu pomocí cronu, apod.).
- Vhodné je volit spíše anglické texty než české, kvůli širší dostupnosti knihoven pro práci s textem.

Experimenty

- V rámci projektu lze provést porovnání rychlosti vyhledávání při použití invertovaného seznamu vůči sekvenčnímu prohledání celé kolekce dokumentů pro různé typy dotazů. Lze vytvořit tabulku/graf v závislosti zvětšující se velikosti databáze dokumentů.
- Dále lze měřit rychlost předzpracování dat, analyzovat paměťovou náročnost tvorby term-by-document matice resp. invertovaného indexu, apod.

© Ing. Jiří Novák, Ph.D., 2023

- Rovněž lze experimentovat s optimalizacemi při vyhodnocování uzávorkovaných dotazů např. dotazy „Caesar AND (Brutus AND Calpurnia)“ a „(Caesar AND Brutus) AND Calpurnia“ lze vyhodnotit s různou rychlostí v závislosti na počtu dokumentů, ve kterých se hledané termy vyskytují (viz přednáška č. 2, slide 17).
- V rámci testování přesnosti hledání je možné experimentovat i s různým nastavením mocniny při výpočtu relevance (viz přednáška č. 2, slide 29). Místo původního parametru p = 2, můžeme zvolit libovolnou hodnotu p ≥ 1 (např. p = 5) a pozorovat, jak je ovlivněno skóre jednotlivých dokumentů.

Reference

[1] Tomáš Skopal, BI-VWM.21 - Vyhledávání na webu a v multimediálních databázích, Přednáška č. 2: Boolean model of information retrieval.
[2] Tomáš Skopal, BI-VWM.21 - Vyhledávání na webu a v multimediálních databázích, Přednáška č. 3: Vector model of information retrieval.
[3] Salton G. et al., Extended Boolean Information Retrieval, *Communications of the ACM*, vol. 26, no. 11, pp. 1022–1036, 1983. https://dl.acm.org/doi/10.1145/182.358466
[4] Porter Stemmer, https://tartarus.org/martin/PorterStemmer/
[5] NLTK pro Python, https://www.nltk.org/
[6] List of text corpora, https://en.wikipedia.org/wiki/List_of_text_corpora
[7] Wikipedia, Database download, https://en.wikipedia.org/wiki/Wikipedia:Database_download
[8] MySQL, CREATE INDEX, https://dev.mysql.com/doc/refman/8.0/en/create-index.html
[9] B-tree, https://en.wikipedia.org/wiki/B-tree
[10] Context-free Grammars, ETF, https://lambda.uta.edu/cse5317/notes/node12.html

© Ing. Jiří Novák, Ph.D., 2023