April 26, 2019

Vektorové vyhledávání

Martin Koucký, Jakub Schinko

Popis projektu

Cílem projektu je implementace vektorového systému ukládání dat (tj. preprocessing a indexování) a vytvoření základního webového GUI kvůli zobrazování výsledků.

Vstupem je buď uživatelem zadaný dotaz ve formě n-krát dvojice [slovo] [váha], po kterém se ve výstupu zobrazí dokumenty nejvíce odpovídající danému vstupu a nebo po rozkliknutí daných dokumentů přímo dokument samotný. To znamená, že celý tento vybraný dokument je převeden na dotaz. Poté jsou v dolní části stránky s dokumentem zobrazeny dokumenty z databáze, které jsou nejvíce podobné danému vybranému dokumentu (převedenému na dotaz)

Způsob řešení

Algoritmus vektorového vyhledávání začíná předzpracováním databáze dokumentů. To proběhne následovně:

- Načtení slov z dokumentů do pole
- Odstranění tzv. stopwords a lematizace, zde je využita knihovna Porter Stemming
- Načtení do reprezentace strukturou Counter, ten reprezentuje dokument jako set dvojic: [slovo], [počet výskytů slova v dokumentu]

Poté se vytvoří index, který jednotlivým ID z counteru přiřazuje jméno skutečného souboru, tento index se uloží v textové formě na disk. Dále se vypočte IDF a s jeho využitím váhy jednotlivých slov v dokumentech (tyto váhy přepisují počty slov ve struktuře Counter popsané výše)

Nyní se vytvoří invertovaný index a sekvenční matice. Invertovaný index je reporezentován jako dictionary, kde klíč je slovo a hodnota je pole párů. Každý tento pár se skládá z ID dokumentu a z váhy. Záznam pro dokument se vyplňuje pouze pokud daný dokument dané slovo obsahuje (neboli má nenulovou váhu)

Sekvenční matice vypadá podobně, ale má záznam pro každé slovo ve slovníku a pro každý dokument, tedy celkem [počet slov] * [počet dokumentů] záznamů. Tam kde dokument neobsahuje dané slovo je za váhu vyplněna 0.

Invertovaný index i sekvenční matice se nyní uloží do textového dokumentu, aby se nemuseli pokaždé vytvářet znovu.

Implementace

Projekt je z větší části napsán v Pythonu s využitím webového frameworku Flask. Zobrazení do webového rozhraní funuguje na základě komunikace app.py se soubory .html ve složce templates. Většina algoritmické logiky je uložena v souboru logic.py a většina práce se soubory je uložena v souboru fileManagement.py. Využity jsou zde knihovny collections, Flask, PorterStemmer. Aplikace nemusí fungovat na jiných systémech než UNIX, kvůli práci se soubory.

Příklad výstupu

Vector Search

science 0.4 computer 0.5 school 0.2

dotaz

Vyhledat

Název		
"COMPUTE!"	7.050665424015399	Detail
"Hackmeeting"	5.693653900679995	Detail
"Deus Ex SDK"	4.954110948807097	Detail
"Liquid State Machine"	4.662219717410007	Detail
"Cusp neighborhood"	3.6954919816535465	Detail
"Trey Anastasio"	3.677128064394924	Detail
"Global Information Grid"	3.6747414616464957	Detail
"Peripheral Component Interconnect"	3.617926678488958	Detail
"Tri-Institutional MD-PhD Program"	3.0157033232481716	Detail

výsledek dotazu

[název] [váha] [reference]

"COMPUTE!"

COMPUTE! () was a computer magazine that was published from 1979 to 1994. In its 1980s heyday it covered all major platforms, and several single-platform spinoffs of the magazine were launched. One of these was COMPUTE!'s Gazette, catering to Commodore computer users. Its original goal was to write about and publish programs for all of the computers that used some version of the MOS Technology 6502 CPU. It started out with the Commodore PET, Commodore Vic-20, the Atari 8-bit series, the Apple II plus, and some 6502-based computers one could build from kits, such as the Rockwell AIM 65, the KIM-1 by MOS Technology, and others from companies such as Ohio Scientific. Support for the kit computers and the Commodore PET were eventually dropped. The platforms that became mainstays at the magazine were the Commodore Vic-20, Commodore 64, Atari 8-bit series, TI-99/4A, and the Apple II series. Later on the IBM PC, Atari ST series, and the Commodore Amiga series computers were added to its line-up. In addition, COMPUTE! published a large number of computer books. Most personal computers of the time came with some version of the BASIC programming language. The magazine often featured type-in programs written in these versions of BASIC for their respective computers. Machine code programs were also published, usually for simple video games listed as hexadecimal numbers that could be POKEd into the memory of a home computer such as a VIC-20 or Atari 400. Machine language listings could also be entered in decimal with a program provided in each issue called MLX (available for Apple II and Commodore hardware, and written in Basic). It was noted particularly for software such as the multiplatform word processor SpeedScript, and the spreadsheet SpeedCalc. Editors of the magazine included Founder Robert Lock, Richard Mansfield, Charles Brannon, and Tom R. Halfhill. buys Compute assets -- Newsbytes News Network, August 10, 1994; External links. COMPUTE! at The Classic Computer Magazine Archive website; Southern Arts Journal; DevX.com; Richard Mansfield's article OOP is Much Better in Theory Than in Practice; Microprocessor Report; Maximum PC; ENDOFARTICLE.

dotaz – celý text

similar files

"COMPUTE!"	1254.7792561456336	Detail
"Trey Anastasio"	107.88223754600602	Detail
"Booker T. Washington"	76.74329712237055	Detail
"Optus Television"	71.88932247073575	Detail
"Books of Kings"	71.12590315621806	Detail
"Howard Scott Warshaw"	70.84815813869203	Detail
"Hampden-Sydney College"	70.83664441183714	Detail
"Waterfall model"	69.59014060069886	Detail
"Absolutely Fabulous"	66.28404218733742	Detail
"C. F. Martin & Company"	62.08830398830153	Detail

Zpět

výsledek dotazu

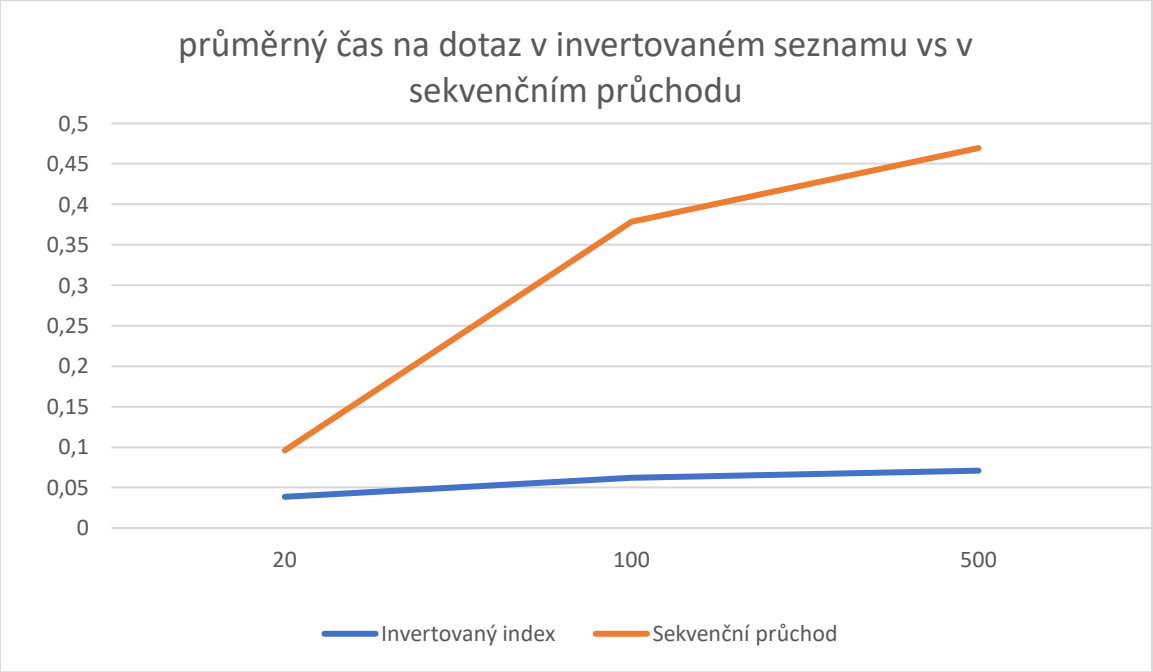
[název] [váha] [reference]

Experimentální sekce

Porovnávali jsme jak rychlý bude sekvenční průchod v dotazování oproti invertovanému seznamu na různých velikostech databáze. Bohužel, jelikož je naše databáze reprezentována přímo v paměti programu a sekvenční průchod má velkou velikost ([počet slov ve slovníku]*[počet dokumentů v databázi]) nebyly jsme schopni porovnávat na větších velikostech než 500 dokumentů, ale i na tomto menším vzorku byla jasně vidět výhoda v rychlosti vyhledávání při reprezentaci invertovaným seznamem.

Tabulka a graf průměrného času na dotaz

Dokumentů v databázi	Invertovaný index	Sekvenční průchod
20	0,03862381	0,09593964
100	0,06222725	0,37841797
500	0,07095337	0,46954155



Diskuze

Databáze dokumentů a její zpracování by bylo potřeba vylepšit, v současné podobě se invertovaný seznam musí buď vytvořit znovu nebo načíst ze souboru, ale v každém případě musí být uložen v paměti programu. Což by u větší databáze dokumentů nebylo dále udržitelné, a pro sekvenční matici není prakticky možné mít databázi větší než 1000 dokumentů. Proto by bylo potřeba ukládat vše sofistikovanějším způsobem, například do nějáké SQL databáze.

Závěr

Aplikace v nynější podobě implementuje funkční základní vektorový vyhledávací model. Umí si zpracovat dokumenty stažené z internetu (v určitém formátu, například wikipedia raw) a uložit je do databáze. Tu umí efektivně prohledávat a na výstupu zobrazovat uspokojující výsledky. Dále umí databázi prohledávat neefektivně, tedy sekvenčně, jak jsme zjistili v experimentální části.