INFORMATION RETRIEVAL:
VEKTOROVÝ MODEL

POPIS PROJEKTU:

Cílem projektu je implementace vektorového systému ukládání dat (tj. preprocessing těchto dat a jejich indexování) spolu s možností dotazování z webové aplikace. V našem projektu jsme využili kolekci sta knih z projektu Gutenberg.

Dotaz se obecně může skládat z kolekce termů a jejich vah. Pro komfortní dotazování ze strany uživatele jsme se rozhodli implementovat webové rozhraní takovým způsobem, že uživatel samotné termy a jejich váhy nezadává, ale pouze si vybere jednu z nabízených knih a její obsah je následně použit jako dotaz (tj. po vyhledání jsou ve výsledné kolekci sestupně seřazené knihy dle jejich podobnosti k uživatelem vybrané knize).

ZPŮSOB ŘEŠENÍ:

PREPROCESSING:

V první fázi se všechny dokumenty musí zpracovat a připravit pro rychlé vyhledávání. To zahrnuje odstranění tzv. stop words, což jsou slova, která mají malý informační význam. Typicky jsou to třeba předložky, spojky, zájmena atd. Dále jsou všechna slova „lematizována“, tj. jsou převedena do základního tvaru (např. anglická slova speaking a speaks jsou převedena na speak). Poté je již možné projít všechny dokumenty, spočítat počet výskytů jednotlivých slov a tuto skutečnost zapsat v našem případě do SQLite databáze. Na základě těchto údajů je následně možné spočítat váhu jednotlivých slov v daných dokumentech podle následujících vzorců:

$$f_{ij} = \text{četnost termu } t_i \text{ v dokumentu } d_j$$

$$tf_{ij} = \frac{f_{ij}}{\max_i\{f_{ij}\}}$$

Díky tomuto vzorci získáme normalizované váhy termů – v intervalu $(0;1\rangle$. V dalším kroku spočítáme ještě inverzní frekvenci termu t_i v kolekci dokumentů.

$$df_i = \text{počet dokumentů obsahující term } t_i$$

$$n = \text{celkový počet dokumentů}$$

$$idf_i = \log_2\left(\frac{n}{df_i}\right)$$

Následně tyto hodnoty vynásobíme a dostaneme váhu termu, s níž budeme dále pracovat.

$$w_{ij} = tf_{ij} \times idf_i$$

Tu uložíme do SQLite databáze a do souboru v následujícím JSON formátu:
"název_termu": {"id_dokumentu": váha_termu, "id_dokumentu": váha_termu, ...}


```
{
  "theory": {
    "1": 1.8889686876112561, "4": 0.056387125003321076, "7": 0.08458068750498161, "10": 0.08458068750498161
  },
  "address": {
    "1": 1.434402824145775, "4": 0.35860070603644373, "5": 0.35860070603644373, "7": 0.35860070603644373
  },
  "delivered": {
    "1": 1.1979643381655696, "8": 1.1979643381655696, "9": 1.1979643381655696, "10": 0.5989821690827848
  }
}
```


Obr. 1 Váhy termu

VYHLEDÁVÁNÍ:

Na úvodní stránce webové aplikace si uživatel může vybrat jeden z nabízených dokumentů. Po potvrzení výběru se mu zobrazí stránka, kde je zobrazeno pět nejpodobnějších dokumentů, které jsou nalezeny pomocí kosinové podobnosti. Všechny nezbytné informace již máme z preprocessingu, nyní lze provést samotný výpočet:

$$\frac{\vec{d}_j \times \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{m}(w_{ij} \times w_{iq})}{\sqrt{\sum_{i=1}^{m} {w_{ij}}^2 \times \sum_{i=1}^{m} {w_{iq}}^2}}$$

Hodnoty tohoto výpočtu se mohou pohybovat od $\langle -1;1 \rangle$, kde -1 značí naprosto rozdílné dokumenty a hodnota 1 představuje identický dokument.

IMPLEMENTACE:

První fáze preprocessingu je implementována v Pythonu s využitím následující knihovny:

- NLTK (Natural Language Toolkit), což je knihovna pro práci s lidským jazykem; v našem programu je využita pro odstranění stop words a „lematizaci“

Samotný výpočet vah jednotlivých termů, výpočet nejpodobnějších dokumentů a webové rozhraní je napsáno v jazyce C++. V něm jsme využili následující knihovny:

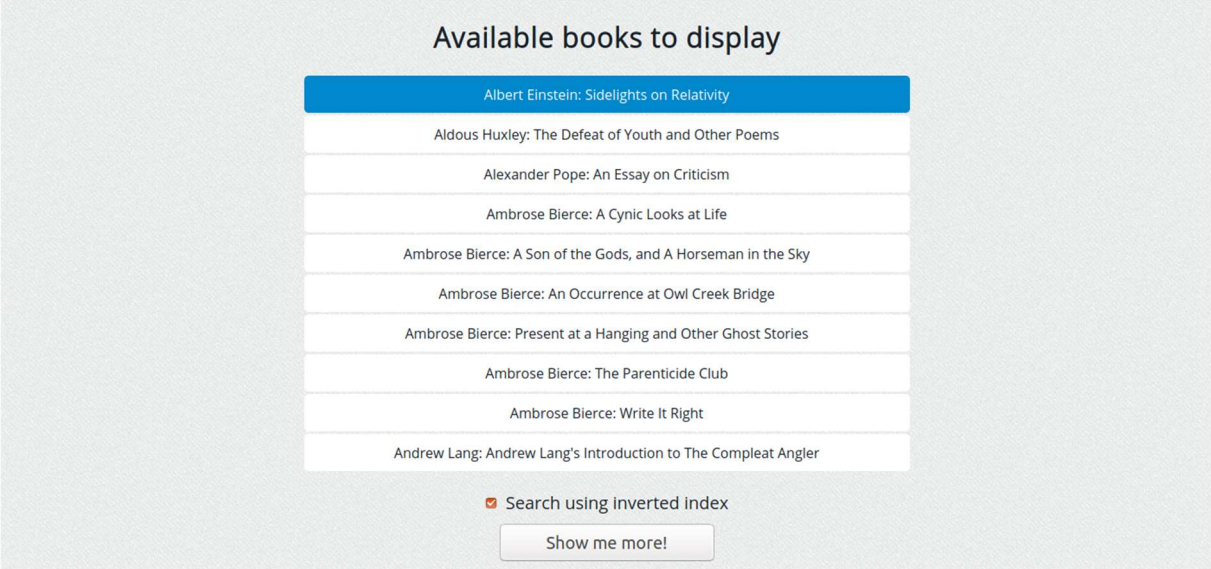
- SQLiteCpp, což je knihovna pro jednodušší práci s databází SQLite, kterou využíváme pro ukládání dat
- Wt; knihovna pro vytváření webových aplikací
- json; knihovnu pro práci se soubory typu JSON

POŽADAVKY NA BĚH:

- Python verze 3.6 a vyšší
- Knihovny třetích stran pro Python: NLTK
- Kompilátor podporující minimálně standart C++17, CMake 3.1 nebo vyšší
- Knihovny pro C++: SQLiteCpp, JSON, Wt, Boost (vyžaduje knihovna Wt)

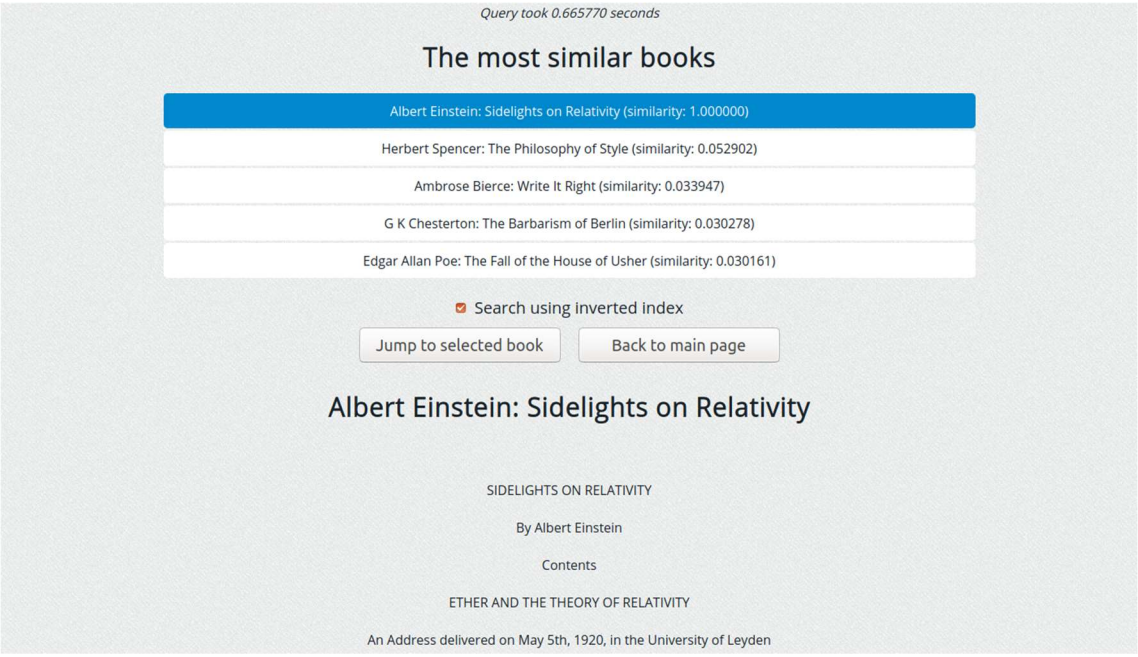
PŘÍKLAD VÝSTUPU:

Na obrázku 2 níže vidíme úvodní obrazovku celé webové aplikace. Zde je možné vybrat jednu z deseti nabízených knih, zvolit zda se při hledání má využít invertovaný index a následně potvrdit svůj výběr kliknutím na tlačítko „Show me more!“.



Obr. 2 Úvodní uživatelské rozhraní

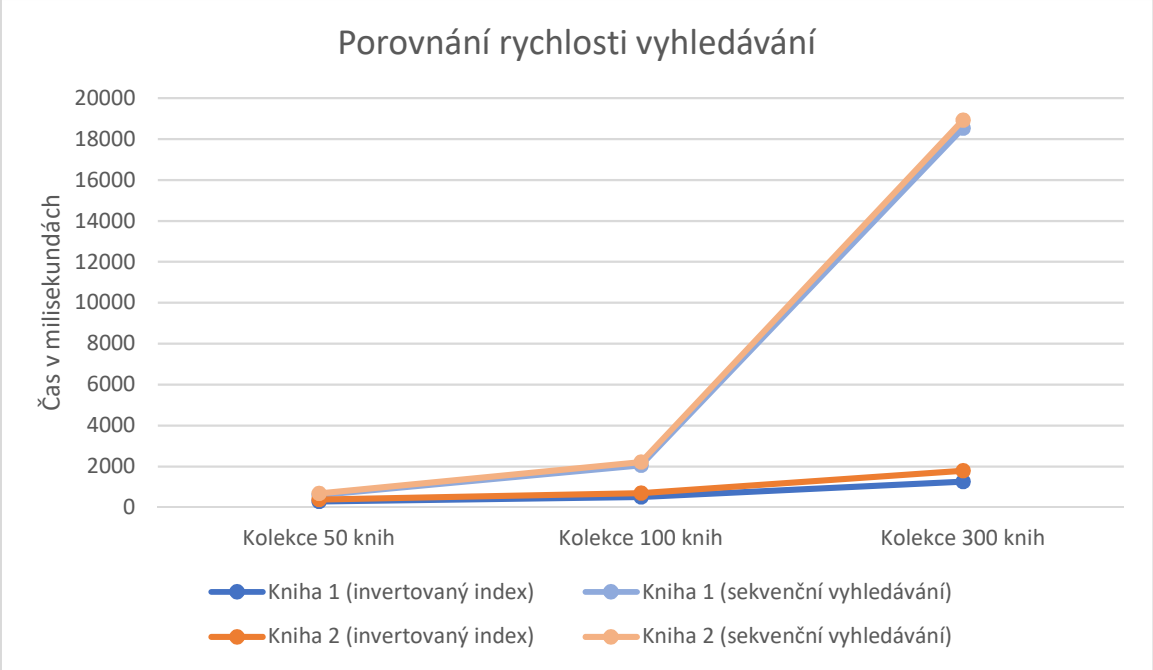
Na obrázku 3 níže již vidíme výsledek akce z úvodní obrazovky. V horní části stránky je zobrazeno pět nejpodobnějších knih k vybrané knize na úvodní stránce a čas, který byl potřeba pro jejich nalezení. Nechybí zde ani informace o tom, jak jsou si knihy podobné. Je umožněno na kteroukoliv z nich přejít (opět je možné zvolit, zda využít invertovaný index při hledání), případně se vrátit zpět na úvodní stránku. V druhé části stránky můžeme vidět samotný obsah vybrané knihy.



Obr. 3 Výsledek dotazu

EXPERIMENTÁLNÍ SEKCE:

Na grafu níže můžeme vidět porovnání mezi vyhledáváním pomocí invertovaného indexu a sekvenčním vyhledáváním u dvou různých knih. Tyto dva typy vyhledávání jsme navíc porovnávali i při různě velkých kolekcích dokumentů (kolekce 50, 100 a 300 knih). Měření bylo několikrát opakováno a do grafu jsou vyneseny průměrné hodnoty.



Obr. 4 Porovnání rychlosti sekvenčního vyhledávání a invertovaného indexu

Z grafu na obrázku 4 výše můžeme vidět, že s větší kolekcí knih začal exponenciálně narůstat čas, který je potřebný pro sekvenční vyhledávání. Naopak čas vyhledávání pomocí invertovaného indexu narůstá lineárně vzhledem k velikosti kolekce. Rozdíly v časech mezi jednotlivými knihami při stejném typu vyhledávání jsou minimální a tento rozdíl je způsoben rozdílným počtem odlišných slov v dané knize.

DISKUZE:

Naše práce nebude dokonalým řešením zadaného problému, jelikož by se pravděpodobně našly algoritmy, které by naši aplikaci dokázaly zrychlit, případně přinést úsporu ve využité paměti. Jako základní vyhledávací engine nad rozumně velkou kolekcí dokumentů by ale bez pochyby šla využít. Při práci jsme se setkali s tím, že není úplně jednoduché najít volně dostupný dataset, který by se hodil na demonstraci naší aplikace. Knihy jím pravděpodobně nebudou z důvodu velké pestrosti slov v nich obsažených, neboť naše aplikace neřeší problém se synonymy.

ZÁVĚR:

Při vypracovávání projektu jsme se detailně seznámili s problematikou indexování dat v dokumentech a následným vyhledáváním ve vytvořeném indexu. Práce nám jistě byla přínosem ať už nabytím nových znalostí, tak ověřením těchto znalostí při následné implementaci a optimalizaci naší aplikace.

ZDROJE:

Prezentace k předmětu BI-VWM
https://moodle-vyuka.cvut.cz/pluginfile.php/213162/course/section/33535/BIVWM_lecture03.pdf

Zadání projektu
https://moodle-vyuka.cvut.cz/pluginfile.php/213181/mod_page/content/21/i-3.pdf