Vektorový model vyhledávání dokumentů

Kanstantsin Downar
Arsenii Pogodin

ČVUT-FIT

downakan@fit.cvut.cz
pogodars@fit.cvut.cz

Repozitář projektu

26. března 2025

1 Popis Projektu

Cílem projektu byla implementace webové aplikace, která poskytuje doporučené dokumenty z kolekce na základě jednoho vybraného dokumentu. Aby byla interakce uživatele se systémem co nejjednodušší, uživatel nejprve vybere libovolný dokument z kolekce, poté mu bude poskytnut úplný text vybraného dokumentu a doporučené „podobné“ dokumenty.

2 Způsob řešení

2.1 Zdroj dat

Jako zdroj dat byly zvoleny dokumenty z Wikipedie. Hledali jsme text dokumentů, který byl předem očištěný, tzn. neobsahoval žádné HTML značky a byl v anglickém jazyce. Dataset ke stažení můžete najít zde. Je to CSV soubor, který obsahuje dva sloupce: vnitřní identifikátor dokumentu na Wikipedii a text dokumentu.

2.2 Předzpracování dat

V první řadě se provádějí následující akce:

1. **Extrakce termů:** sestavení slovníku kolekce dokumentů. Všechna slova, která se objevují v dokumentech kolekce, přidávají se do slovníku v jedné instanci.
2. **Odstranění stop slov:** filtrace slov nepotřebných k analýze.
3. **Stemming + lematizace:** získání kmenů slov, tj. částí slov, které se nemění při skloňování nebo časování + úprava na základní tvar slova, např. tvary „go, went, gone“ převedeme na společný tvar „go“.

2.3 Budování datové struktury

V závislosti na použité datové struktuře je postavena jedna z níže uvedených.

Term-by-document matice Na základě slovníku termů a seznamu dokumentů můžeme sestavit term-by-document matici, ve které jsou jednotlivé řádky reprezentovány dokumenty a sloupce představují termy (slova) obsažené v těchto dokumentech. Samotná matice je plná čísel, které vyjadřují váhu konkrétního termu v daném dokumentu. Tyto váhy se vypočítávají pomocí schématu TF-IDF, které kombinuje frekvenci výskytu termu v dokumentu (TF) a inverzní dokumentovou frekvenci (IDF).

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t) = \frac{f(t,d)}{F(t)} \times \log \frac{N}{|\{d \in D : t \in d\}|}$$

kde,
t - term (slovo),
d - dokument,
D - kolekce dokumentů,
$TF(t,d)$ - normalizovaná frekvence termu t v dokumentu d,
$IDF(t)$ - inverzní dokumentová frekvence termu t,
$f(t,d)$ - frekvence termu t v dokumentu d,
$F(t)$ - maximální frekvence termu t v celé kolekci,
N - celkový počet dokumentů v kolekci,
$|\{d \in D : t \in d\}|$ - počet dokumentů obsahujících term t.

Invertovaný seznam V této implementaci je invertovaný index vytvořen na základě TF-IDF matice, kde každý term je spojen se seznamem dokumentů, ve kterých se vyskytuje, spolu s odpovídající váhou TF-IDF.

Nejprve je inicializován invertovaný index jako slovník, kde klíčem je index termu a hodnotou je seznam dvojic (identifikátor dokumentu,TF-IDF pro daný dokument a term). Současně je vytvořena struktura cleaned_queries, která uchovává podobné informace pro jednotlivé dokumenty. Tato struktura slouží pro rychlý přístup k TF-IDF vektoru bez zbytečných termů, tzn, vektoru bez nul.

Nakonec jsou všechny seznamy v invertovaném indexu seřazeny podle identifikátoru dokumentu, což usnadňuje následné vyhledávání.

2.4 Hledání podobných dokumentů

K nalezení podobných dokumentů v kolekci se používá kosinová míra podobnosti, která se vypočítá následovně:

$$CosSim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{|\vec{d_j}| \cdot |\vec{q}|} = \frac{\sum_{i=1}^{m}(w_{ij} \cdot w_{iq})}{\sqrt{\sum_{i=1}^{m} w_{ij}^2 \cdot \sum_{i=1}^{m} w_{iq}^2}}$$

kde,
q - vektor představující dokument vybraný uživatelem,
d_j - vektor představující dokument z kolekce.

Po výpočtu kosinové míry podobnosti mezi dokumentem vybraným uživatelem a každým dokumentem z kolekce (s výjimkou samotného sebe) je vybráno několik dokumentů s největší hodnotou kosinové míry podobnosti.

3 Implementace

Vzhledem k tomu, že zdroj dat byl použit k určení matice TF-IDF a obsahuje čistý text bez HTML, tento text se nehodí pro zobrazení dokumentu v přijatelném formátu. Abychom zobrazili dokument v čitelné podobě, používáme jeho Wikipedia ID, které umožňuje stáhnout obsah skutečného Wikipedie článku. Tento obsah je poté zobrazen na stránce dokumentu v čitelné podobě.

Komponenty architektury projektu:

1. **Načítání a předzpracování dat:** Čtení dokumentů z CSV souboru, tokenizace, odstranění stop-slov, lemmatizace a stemming.
2. **Vytvoření TF-IDF matice:** Výpočet váhových hodnot TF-IDF pro každý termín v každém dokumentu.

3. **Invertovaný index:** Ukládá termíny s odkazem na dokumenty a jejich TF-IDF váhy pro urychlení vyhledávání.
4. **Vyhledávání podobných dokumentů:** Výpočet kosinové podobnosti mezi vektory dokumentů.
5. **Optimalizace výkonu:** Měření doby provedení operací s využitím třídy `SpeedTester`.

Použité knihovny:

- **pickle** - pro serializaci a deserializaci dat (ukládání a načítání objektů).
- **numpy** - pro práci s poli a maticemi (TF-IDF matice), pro matematické operace.
- **pandas** - pro zpracování a analýzu dat (čtení a práce s CSV soubory).
- **nltk** - pro zpracování přirozeného jazyka (tokenizace, lemmatizace, odstranění stop-slov).
- **collections** - pro práci s datovými kontejnery (počítadla a slovníky s frekvencemi).
- **fastapi** - pro vytvoření serverové části.
- **bs4** - pro zpracování reálných HTML stránek.

Všechny potřebné závislosti, stejně jako způsob spuštění programu, lze nalézt v repozitáři projektu.

4 Příklady vstupu a výstupu

Tato sekce poskytuje příklady vstupu a výstupu pro aplikaci, která zobrazuje seznam dokumentů a umožňuje zobrazení podrobností o konkrétním dokumentu, včetně podobných dokumentů a výkonových metrik.

4.1 Hlavní stránka

Na hlavní stránce jsou zobrazeny všechny dokumenty, které jsou uloženy v databázi. U každého dokumentu je uveden jeho název a možnost přechodu na stránku s podrobnostmi o daném dokumentu. Existuje možnost výběru náhodného dokumentů z celého seznamu. Příklad výstupu můžete vidět na Obrázku 1

4.2 Jednotlivé dokumenty

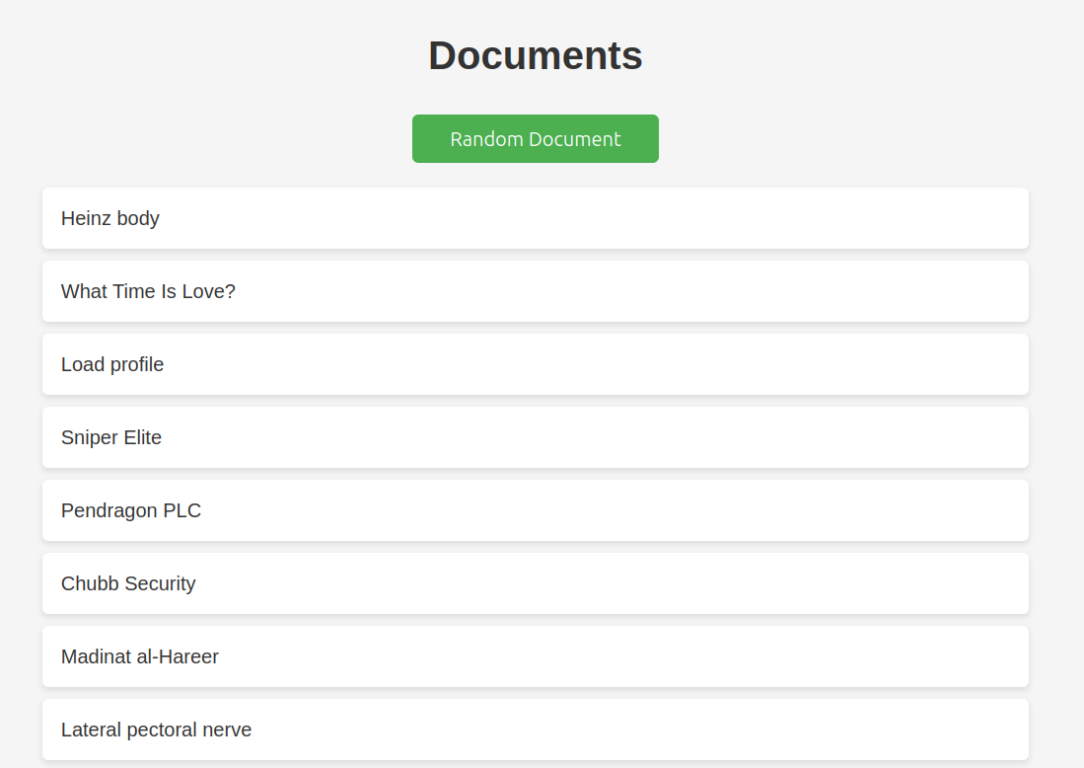
Po kliknutí na konkrétní dokument z hlavní stránky bude uživatel přesměrován na stránku s podrobnostmi o tomto dokumentu. Tato stránka obsahuje název dokumentu, obsah Wikipedie a seznam podobných dokumentů.

Pro každý dokument jsou rovněž zobrazeny metriky výkonu, jako je čas potřebný k načtení obsahu Wikipedie a rychlost vyhledávání podobných dokumentů, která je měřena dvěma způsoby: sekvenčním vyhledáváním a vyhledáváním s použitím inverzního indexu.°Příklad výstupu můžete vidět na Obrázku 2

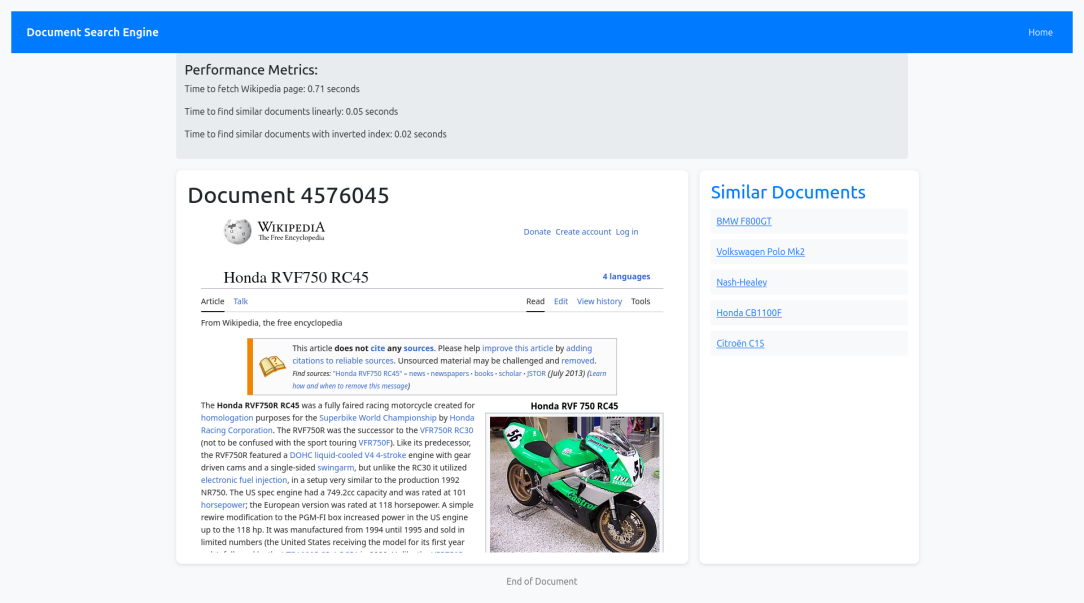
5 Experimentální sekce

Cílem našich experimentů bylo zjistit, jak velikost datasetu TF-IDF hodnot ovlivňuje výkon aplikace. Z původního zdroje dat byly vytvořeny různé velikosti datasetů, začínající od 100 dokumentů až po 12 500 dokumentů.

Zaměřili jsme se především na časové metriky sestavení celkové TF-IDF matice a sestavení invertovaného seznamu pomocí této matice, paměťovou složitost obou řešení, rychlost vyhledávání a rychlost předzpracování dat.



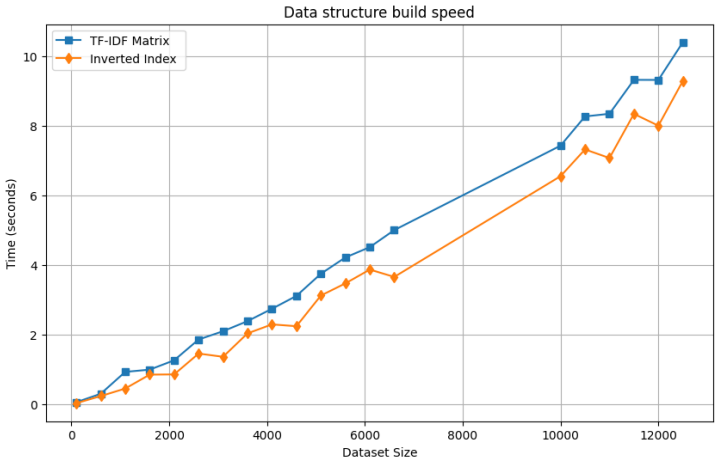
Obrázek 1: Při přístupu na hlavní stránku aplikace (/), server načte seznam všech dokumentů, který bude zobrazen na stránce.



Obrázek 2: Při přístupu na stránku dokumentu (/document/id) server načte podrobnosti o dokumentu s daným ID (například id = 1), stáhne obsah Wikipedie a vyhledá podobné dokumenty. Zobrazí se navíc metriky výkonu vyhledávání.

Sestavení TF-IDF a Invertovaného seznamu Obrázek 3 ukazuje rychlost sestavení TF-IDF matice z předzpracovaných dokumentů. Jak můžeme vidět, s rostoucí velikostí datasetu TF-IDF hodnot se zvyšuje průměrná doba vytváření TF-IDF matice, což je očekáváné.

Důvodem nižšího času na vytvoření invertovaného seznamu je to, že nemusíme znovu počítat všechny hodnoty TF-IDF, což by zabralo více času, ale můžeme využít již vypočítané nenulové hodnoty.



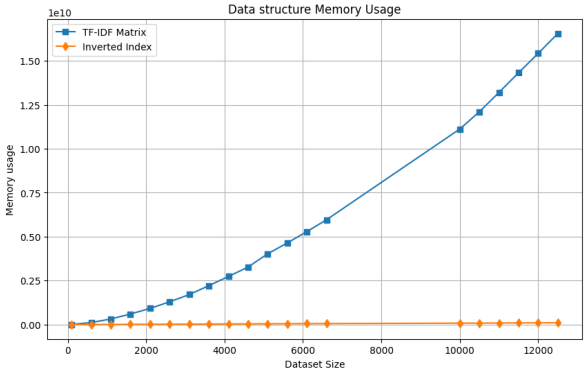
Obrázek 3: Rychlost sestavení TF-IDF matice a invertovaného seznamu v sekundách v závislosti na velikosti datasetu.

Celkově čas sestavení struktur roste skoro lineárně se zvětšující se velikostí datasetu.

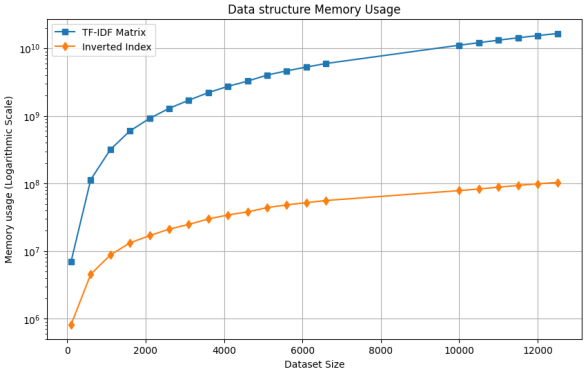
Paměťová složitost Obrázky 4 a 5 ukazují, kolik místa zabírá TF-IDF matice a invertovaný seznam v bajtech.

Na první pohled z Obrázku 4 se může zdát, že invertovaný seznam má velikost 0, což není pravda. Proto na Obrázku 5 byla použita logaritmická škála pro zobrazení rozdílu.

Je vidět, že použití invertovaného seznamu má obrovskou výhodu nejen z důvodu rychlosti vyhledávání, ale i z pohledu paměti.



Obrázek 4: Paměťová složitost TF-IDF matice a invertovaného seznamu v bajtech v závislosti na velikosti datasetu.



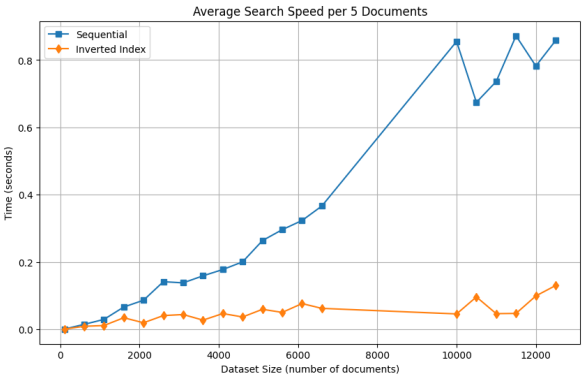
Obrázek 5: Paměťová složitost TF-IDF matice a invertovaného seznamu za použití logaritmické škály v závislosti na velikosti datasetu.

Rychlost vyhledavani Obrázek 6 ukazuje rychlost vyhledávání podobných dokumentů. U menších datasetů (např. 100, 500 dokumentů) není rozdíl v rychlostech vyhledávání příliš patrný, ale u větších datasetů je vidět několikanásobné zrychlení při použití invertovaného seznamu.

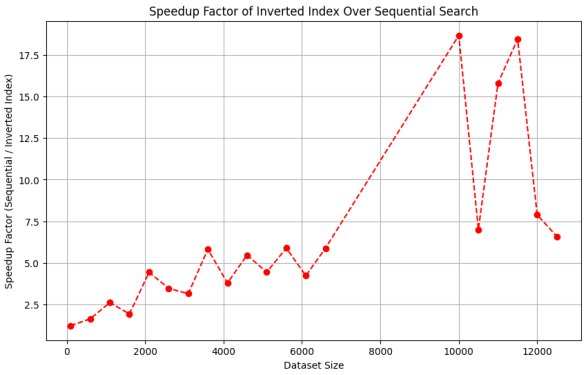
Pro měření času bylo vygenerováno 5 náhodných indexů pro vyhledávání a pak byla spočítána průměrná doba vyhledávání.

Obrázek 7 ukazuje faktor zrychlení, který byl vypočítán jako podíl časů pro vyhledávání dokumentu sekvenčně, tedy pomocí TF-IDF matice, a časů pro vyhledávání dokumentu pomocí invertovaného seznamu.

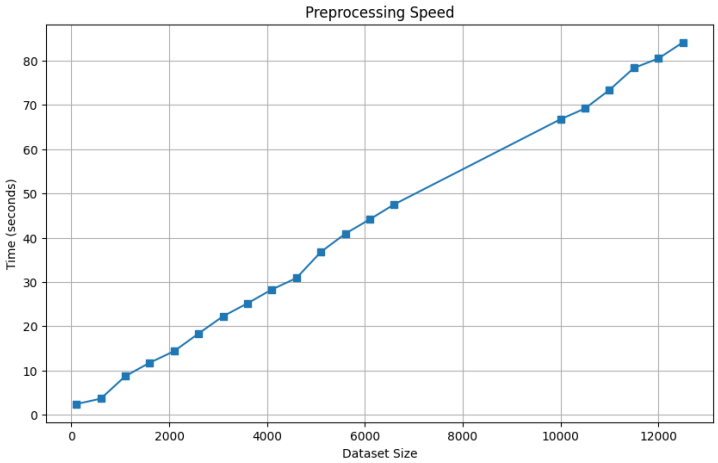
Rychlost předzpracování Obrázek 9 ukazuje rychlost předzpracování dokumentů, tzn. odstranění stop slov (např. předložky), převod slova na základní tvar (lemmatizace) a následně na kmen slova (stemming).



Obrázek 6: Rychlost vyhledávání podobných dokumentů pomocí TF-IDF matice a invertovaného seznamu v sekundách v závislosti na velikosti datasetu.



Obrázek 7: Faktor zrychlení invertovaného seznamu oproti sekvenčnímu vyhledávání v závislosti na velikosti datasetu.



Obrázek 8: Rychlost předzpracování datasetů (odstranění nevýznamnych slov, stemming a lemmatizace) v sekundach v závilsoti na velikosti datasetu.

6 Diskuze

Hlavním cílem bylo vytvoření funkčního prototypu, který umožňuje uživatelům prohlížet wikipedia stránky a získávat doporučení na základě kosinové podobnosti jejich TF-IDF vektorů. Nicméně, během vývoje a testování aplikace bylo zjištěno, že existují určité nedostatky, které by mohly být vylepšeny:

- Invertovaný index je vyroben jako slovník seznamů, což může být náročné na paměť.
- Současná implementace získává obsah Wikipedie na letu. Může to být pomalé a nestabilní. Často přistupované stránky Wikipedie je možné nahradit cachem.
- Je možné použit multiprocessing nebo multithreading pro úlohy jako předzpracování dokumentů a výpočet podobnosti, jelikož předzpracování zabere vetšinu času.
- Současná implementace načte potřebné proměnné (TF-IDF matici, texty apod.) do paměti. I když to umožňuje rychlý přístup k datům, limituje to počet dokumentů, které lze použít. Je možné využít rychlou cache, jako je například Redis, v kombinaci s databází pro ukládání dokumentů a TF-IDF vektorů.

7 Závěr

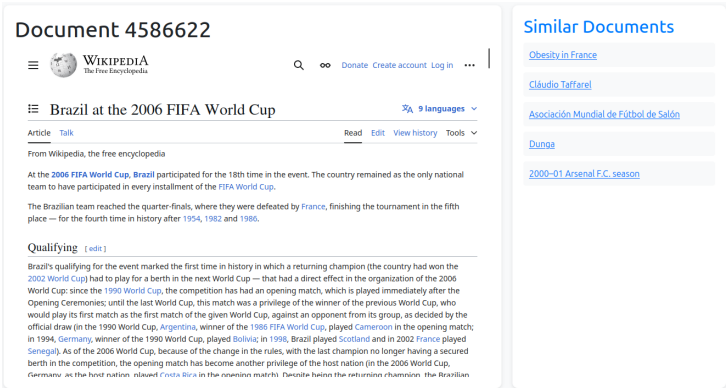
V rámci našeho semestrálního projektu jsme se zaměřili na vývoj aplikace, která umožňuje uživatelům prohlížet dokumenty z Wikipedie a získávat podobné dokumenty na základě analýzy kosinové podobnosti

TF-IDF vektorů.

Projekt byl úspěšný v tom, že jsme dokázali vytvořit funkční prototyp, který splňuje základní požadavky specifikace. Během testování aplikace jsme však narazili na několik problémů týkajících se výkonu při zpracování většího množství dat a rychlému přístupu k reálným stránkám Wikipedie.

Aplikace byla následně nasazena s datasetem o velikosti 2000 dokumentů. I když některé doporučené dokumenty jsou velmi relevantní k prohlíženému dokumentu, některá doporučení nejsou optimální z důvodu menšího rozměru dokumentové databáze.

Příklad takového doporučení je zobrazen na obrázku níže:



Obrázek 9: Příklad divného doporučení pro dokument "Brazil at the 2006 FIFA World Cup", kde je jeden z doporučených dokumentů "Obesity in France".