

Rozšířený booleovský model

BI-VWM.21

Verze z 28. dubna 2024

Autoři:

Martin Efler (eflermar), Martin Eliáš (eliasma7)

Obsah

1	Popis projektu	3
2	Způsob řešení	3
2.1	Invertovaný seznam	3
2.2	Zpracování dotazu	3
2.2.1	AND	3
2.2.2	OR	4
2.2.3	NOT	4
2.2.4	Složitější dotaz	4
3	Implementace	4
3.1	Dokumenty	4
3.1.1	Získání raw textu	4
3.1.2	Lemmatizace	5
3.2	Invertovaný seznam a term-document matice	5
3.2.1	Výpočet tf-idf	5
3.2.2	Tvorba invertovaného seznamu a term-document matice	5
3.3	Parsování výrazu a výpočet relevance	5
4	Příklady vstupu a výstupu	6
5	Experimentální sekce	7
5.1	Rychlost zpracování výrazu	7
5.2	Rychlost tvorby term-dokument matice a invertovaného seznamu	9
5.3	Rychlost zpracování výrazů v závislosti na počtu dokumentů	10
6	Diskuze	10
7	Závěr	11

1 Popis projektu

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. Aplikace na základě zadaného dotazu vytvoří seznam dokumentů, který je seřazen dle relevance vůči dotazu, a zobrazí ho. Ze seznamu poté lze konkrétní dokument vybrat a prohlížet.

2 Způsob řešení

2.1 Invertovaný seznam

Pro každý term si do slovníku uložíme jeho váhu v jednotlivých dokumentech pomocí následujícího vzorce.

$$tfidf = \frac{f_{t,d}}{\max_t \{f_{t,d}\}} \cdot \log\left(\frac{\#d}{\#d_t}\right)$$

kde $f_{t,d}$ je frekvence termu v daném dokumentu, $\max_t \{f_{t,d}\}$ je nejvyšší frekvence daného termu skrze dokumenty, $\#d$ je počet dokumentů a $\#d_t$ je počet dokumentů obsahující term. Tato hodnota je následně ještě normalizována do intervalu $< 0, 1 >$ pomocí min-max normalizace

$$tfidf_{scaled} = \frac{tfidf - tfidf_{min}}{tfidf_{max} - tfidf_{min}}$$

Pro zjednodušení předpokládáme, že alespoň jeden term se neobjevuje ve všech dokumentech a tudíž bude minimum odpovídat nule. Tím se vzorec ještě zjednoduší a zůstane jen

$$tfidf_{scaled} = \frac{tfidf}{tfidf_{max}}$$

2.2 Zpracování dotazu

Při zpracování dotazu je potřeba spočítat relevanci pro všechny dokumenty a následně je dle ní seřadit. Pokud dotaz obsahuje pouze jeden term, odpovídá relevance jeho váze v daném dokumentu – $tfidf_{term}$. V dotazu lze termy buď řetězit pomocí AND a OR nebo negovat pomocí NOT.

2.2.1 AND

homer AND lisa AND santa

$$1 - \sqrt{\frac{(1 - homer)^2 + (1 - lisa)^2 + (1 - santa)^2}{3}}$$

2.2.2 OR

homer OR lisa OR santa

$$\sqrt{\frac{homer^2 + lisa^2 + santa^2}{3}}$$

2.2.3 NOT

NOT santa

$$1 - santa$$

Takto můžeme počítat negaci díky normalizaci do $\langle 0, 1 \rangle$.

2.2.4 Složitější dotaz

(homer AND lisa) OR NOT santa

$$\sqrt{\frac{1 - \sqrt{\frac{(1-homer)^2 + (1-lisa)^2}{2}} + (1 - santa)^2}{2}}$$

3 Implementace

Pro naši aplikaci jsme použili Python s hlavními knihovnami *nltk* - práce s textem, *flask* - práce s webovou aplikací, a pomocnými *zipfile*, *pysrt*, *matplotlib*.

3.1 Dokumenty

Naši databázi simulujeme hromadou lokálně uložených textových souborů. Vybrali jsme si titulky ze seriálu The Simpsons, jelikož se nejedná ani o zanedbatelně malou, ani zbytečně velkou kolekci dat. Navíc se v dokumentech budou často objevovat například jména postav, díky čemuž se model dobře testuje.

3.1.1 Získání raw textu

Při získávání samotných textových dokumentů, které by bylo možné dále použít, jsme o spoustu epizod přišli. Nejdříve je nutné soubory odzipovat, což se ne u všech povedlo, dále je potřeba převést původní srt soubor do čistě textové podoby. Tedy bez pořadí stopy, timestampů a podobných dat, které jsou u titulků potřeba, ale s obsahem nemají nic společného. Tam nastala další hromada problémů, takže nám nakonec zůstalo cca 350 použitelných souborů, což, při průměrně 2000 slov na soubor, bohatě stačí.

3.1.2 Lemmatizace

Ze souborů jsme nejdříve odstranili přebytečné mezery a apostrofy. Pomocí knihovny *nltk.word_tokenize* získáme z textu tokeny slov. Z nich odstraníme stopwords pomocí knihovny *nltk.corpus.stopwords*, slova kratší než 2 znaky a zkrácené tvary typu 'll, které knihovna nerozpozná jako *will*. O samotnou lemmatizaci se nám stará knihovna *nltk.WordNetLemmatizer*.

3.2 Invertovaný seznam a term-document matice

3.2.1 Výpočet tf-idf

Před tvorbou invertovaného seznamu a matice si nejdříve předpočítáme hodnoty potřebné k výpočtu tfidf. Postupně projdeme všechny lematizované dokumenty a ukládáme si počty všech nalezených výskytů termů přes všechny dokumenty, pro konkrétní dokument a maximální výskyt termu v dokumentu (realizováno pomocí dvou dictionary). Následně se pro každou kombinaci termu a dokumentu vypočítá jeho hodnota tf-idf (podle výše zmíněného vzorce) a následně jsou tyto hodnoty normalizovány.

3.2.2 Tvorba invertovaného seznamu a term-document matice

Pro invertovaný seznam jsou tyto hodnoty uloženy v dictionary, kde klíč je term a hodnota další dictionary, kde klíč je id dokumentu a hodnota tf-idf pro zadanou kombinaci termu a dokumentu (pro každý term jsou v jeho vnitřním dictionary tf-idf hodnoty pouze pro dokumenty, ve kterých se term vyskytuje). Pro term-document matici jsou hodnoty uloženy ve 2D poli, kde první index odpovídá dokumentům a druhý index odpovídá termům a hodnotou je opět tf-idf (pro každý dokument jsou zde uloženy tf-idf hodnoty pro všechny termy, nezávisle na tom, jestli se v nich term vyskytuje)

3.3 Parsování výrazu a výpočet relevance

Náš parser je rekurzivní funkce, která nejdříve rozdělí zadaný výraz na podvýrazy (těmi jsou buď samotné termy nebo další složené výrazy) a poté se sama zavolá na každý takto nalezený podvýraz. Pokud je podvýraz samotný term, naleznou se v invertovaném seznamu hodnoty tfidf pro všechny dokumenty a funkce vrátí list těchto hodnot. Tyto hodnoty jsou následně dosazeny do vzorců pro "and" nebo "or" (podle toho, který operátor se v podvýrazu vyskytuje) a znegovány, pokud podvýrazu předcházela operátor "not". Postupně se tak z funkce vrátí list výsledných hodnot relevance. List se už pak jen seřadí a zkrátí na požadovanou délku. Parser navíc kontroluje, jestli se nestřídají v jednom podvýrazu operátory, jestli nechybí operátor a jestli sedí počet závorek.

4 Příklady vstupu a výstupu


Query Results

Enter your query:
(bart and lisa) or not homer

How many documents:
20

Results for "(bart and lisa) or not homer"

1. [The Simpsons - 14x01 - Bart vs. Lisa vs. The Third Grade DVDrip.en.txt](#)
2. [The Simpsons - 14x12 - I'm Spelling As Fast As I Can DVDrip.en.txt](#)
3. [The Simpsons - 29x02 - Springfield Splendor HDTV KILLERS.en.txt](#)
4. [The Simpsons - 19x20 - All About Lisa PDTV Proper E7.en.txt](#)
5. [The Simpsons - 19x13 - The Debarfed Caph.en.txt](#)
6. [The Simpsons - 6x21 - The PTA Disbands.en.txt](#)
7. [The Simpsons - 14x16 - Scuse Me While I Miss the Sky DVDrip.en.txt](#)
8. [The Simpsons - 12x14 - New Kids On The Bleech HDTV.en.txt](#)
9. [The Simpsons - 19x17 - Apocalypse Cow PDTV XOR.en.txt](#)
10. [The Simpsons - 19x21 - 24 Minutes.en.txt](#)
11. [The Simpsons - 29x08 - Mr. Lisa's Opus HDTV KILLERS.en.txt](#)
12. [The Simpsons - 7x15 - Bart The Fink.en.txt](#)
13. [The Simpsons - 21x02 - Bart Gets A Z HDTV FOM.en.txt](#)
14. [The Simpsons - 31x14 - Bart the Bad Guy WEB XLF.en.txt](#)
15. [The Simpsons - 21x06 - Pranks and Greens HDTV FOM.en.txt](#)
16. [The Simpsons - 14x07 - Special Edna \(a.k.a. Love And Marking\) DVDrip.en.txt](#)
17. [The Simpsons - 7x07 - Radioactive Man.en.txt](#)
18. [The Simpsons - 30x19 - Gail in the Band WEB TBS.en.txt](#)
19. [The Simpsons - 31x21 - The Hateful Eight-Year-Olds WEB ALiGN.en.txt](#)
20. [The Simpsons - 31x16 - Better Off Ned WEB XLF.en.txt](#)

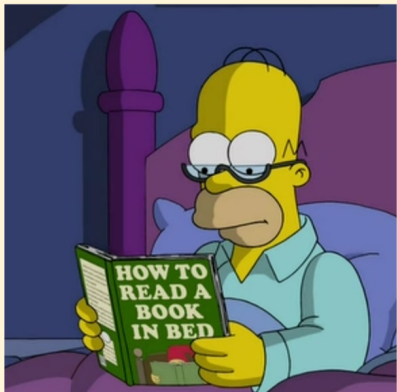


Obrázek 1: Příklad výstupu pro vstup "(bart and lisa) or not homer".

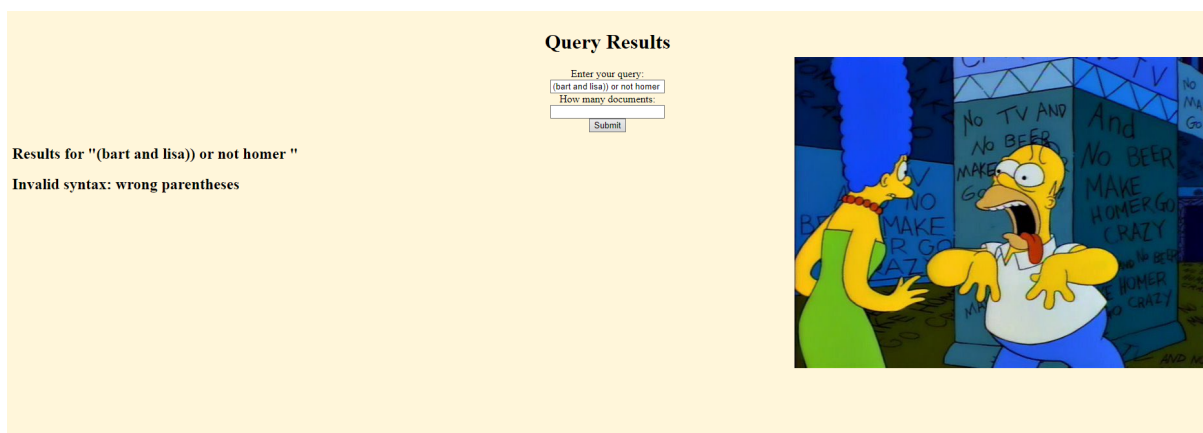
File Content

The PTA Disbands This bus has seen better days. Well, at least it's safer than the old bus. Time to move.

The hole's getting bigger. Seymour, the children are playing in the hole again. Shouldn't you get that fixed? Edna, you know they just cut the school's budget. If I had the money, I'd fix the exhaust leak. I think it's causing some of our low test scores. The battlefield is just a half-mile ahead. Begin braking procedure! This Civil War cannon has been fully restored and is in ready-to-fire condition. But it's a good thing we're not firing it... because it happens to be aimed at the main support leg of that lookout tower. People don't realize that these cannons are very sensitive... and the slightest jolt could set them off. Of course, for safety reasons... we don't keep the cannon loaded. It's just common sense. Otto, why don't you get some more gas. Here's the credit card. And a mint for afterwards. Five dollars a child? Last year it was free! New ownership. But we don't have that kind of money. In fact, no school could afford... Here's the admission, plus something for you. See that they get a little extra education. Yes, sir, Principal Vallant! He thinks he's so hot ever since he swept the Princel Awards. Those things are rigged. On May 21, 1864, the men of the 9th Bearded Infantry... were sunning and fluffing their beards in the sun. Suddenly, enemy troops crested that hill over there. Fort Springfield, we surrender unconditionally! We're sick! We need leeches... and hawksaws to saw off our gangrenous limbs. But the Springfield brigade was too brave to accept their surrender. Come on, boys. Those white flags are no match for our muskets! Charge! And the Springfielders heroically slaughtered their enemies... as they prayed for mercy. It's hard to see what's going on. I can only make out the fat soldiers. All right, children. Switch. Hey, they're trying to learn for free! - Get them! - Use your phony guns at clubs! Run, children. Start the bus! Otto! Start the bus! Damn! I shouldn't have eaten the mint first. Okay, hop on. Wait! Wait! Seymour, because of your penny-pinching... we're coming back from a field trip with the fewest children yet. God bless the man who invented permission slips. I think I got your lunch. I didn't think this was for me. Seymour, the teachers are fed up. You have to put money back into the school. You've cut back on everything. Salaries, supplies, the food. I don't care what you say, I can taste the newspaper. Shredded newspapers add much-needed roughage and essential inks. Besides, you didn't notice the old gum mats. There's very little meat in these gym mats. Our demands are very reasonable. By ignoring them, you're selling out these children's futures. Oh, come on, Edna! We both know these children have no future! Prove me wrong, kids. Prove me wrong. I've never seen them fight like that. I'm worried that all this posturing



Obrázek 2: Zobrazení obsahu souboru "The Simpsons - 6x21 - The PTA Disbands.en.txt".

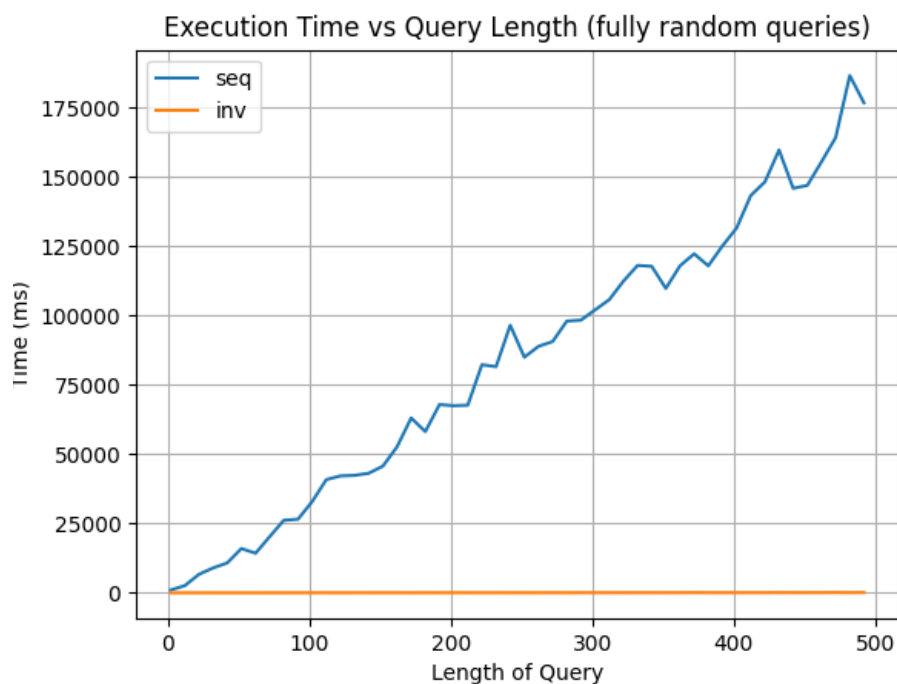


Obrázek 3: Chybová hláška pro špatně použité závorky.

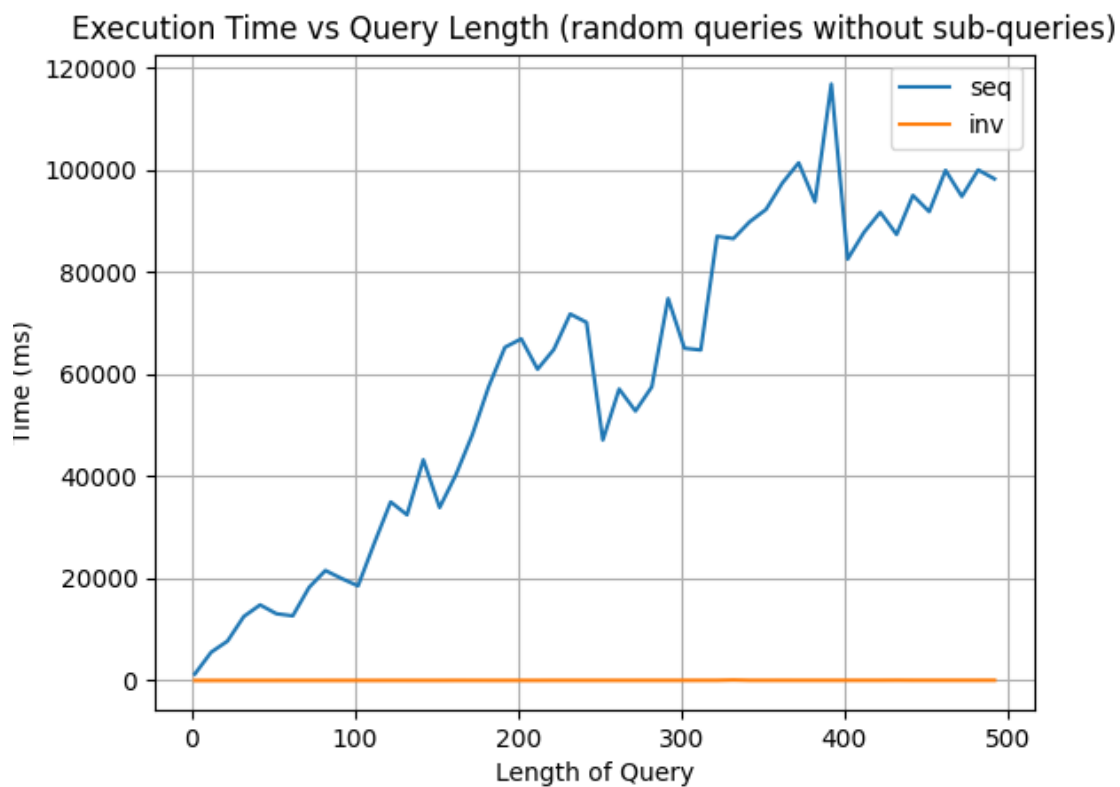
5 Experimentální sekce

5.1 Rychlost zpracování výrazu

Pro experimentování s výrazy jsme naimplementovali jednoduchý generátor náhodných výrazů zadané délky. Výrazy jsou náhodně generovány pouze z termů ze slovníku a mohou obsahovat vnořené výrazy a záporny.



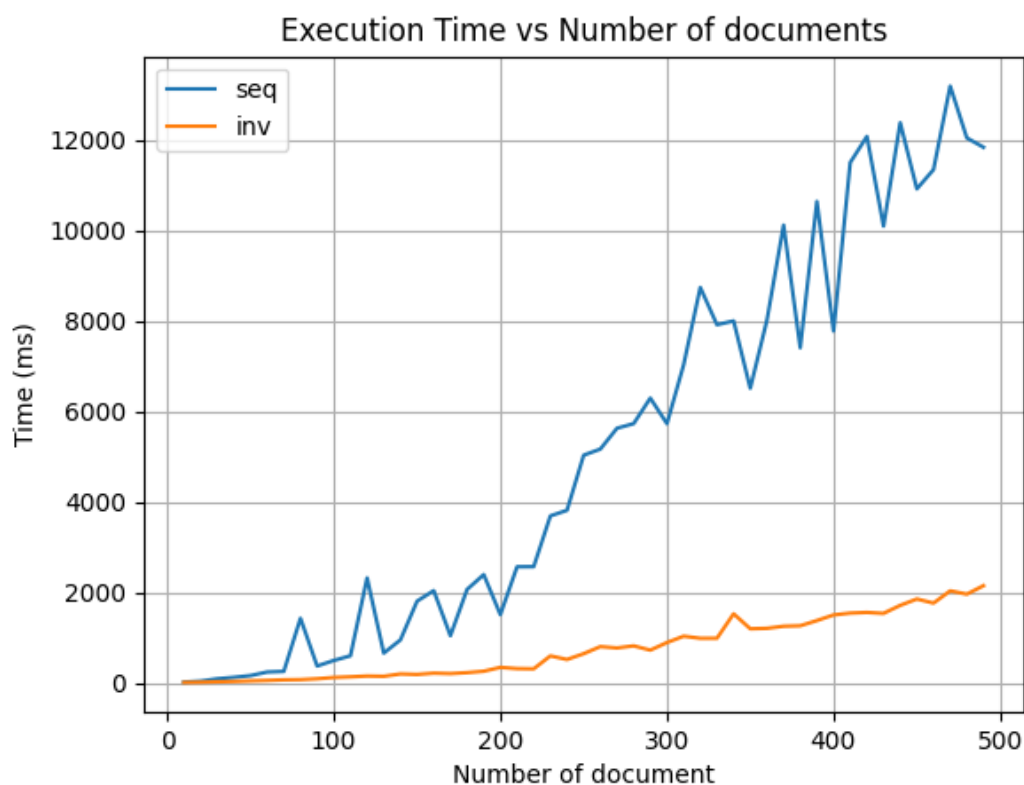
Obrázek 4: Graf rychlosti zpracování výrazu v závislosti na jeho délce (modrá sekvenční, oranžová invertovaný seznam)



Obrázek 5: Graf rychlosti zpracování výrazu v závislosti na jeho délce (modrá sekvenční, oranžová invertovaný seznam)

Z předchozích dvou grafů můžeme vidět, že je skutečně použití term by document matice výrazně pomalejší, než použití invertovaného indexu. Skoky v rychlosti jsou nejspíš způsobeny náhodností výrazů. V první verzi jsou použity výrazy s podvýrazy a ve druhém pouze nezanořené výrazy.

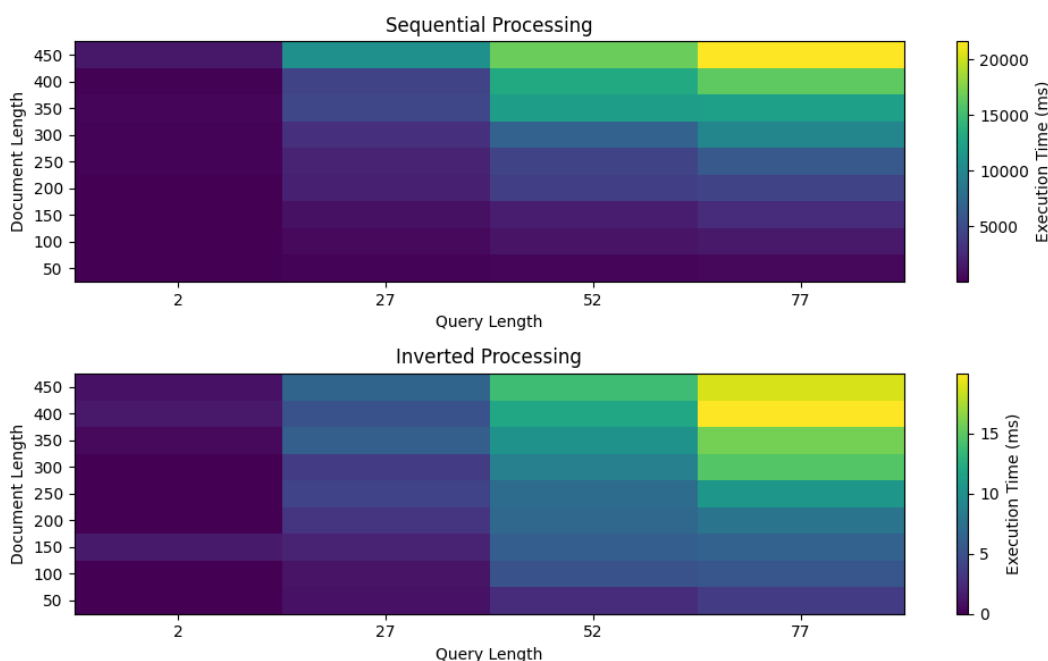
5.2 Rychlost tvorby term-dokument matice a invertovaného seznamu



Obrázek 6: Graf rychlosti přípravy term-dokument matice a invertovaného seznamu v závislosti na počtu dokumentů (modrá matice, oranžová invertovaný seznam)

Zde opět můžeme vidět, že je použití invertovaného indexu výrazně efektivnější. Pokoušeli jsme se měřit i množství využití paměti, ale vnitřní implementace Pythonu nám nedokázala vrátit použitelná data (Python si ukládání zdánlivě náhodně optimalizuje a na různých zařízeních zabíral stejný slovník jinak velkou část paměti).

5.3 Rychlost zpracování výrazů v závislosti na počtu dokumentů



Obrázek 7: Heatmapa rychlosti zpracování výrazu v závislosti na délce výrazu a počtu dokumentů

Z heatmap jde vidět, že je opět řádově lepší invertovaný seznam. Tam je doba zpracování závislá hlavně na délce výrazu. U sekvenčního přístupu pak ovlivňují efektivitu oba faktory výrazněji.

6 Diskuze

Během implementace jsme narazili na několik problémů:

- Vzhledem k předchozí nezkušenosti s webovými aplikacemi se nám nakonec podařilo udělat jen velmi základní interface. Aplikace běží pouze lokálně a na reálné čtení nalezených dokumentů, je prakticky nepoužitelná, jelikož původní textové soubory nejsou nijak upravené.
- Trochu práce navíc by si rozhodně zasloužilo ošetření vstupu do parseru, to máme jen velmi základní.
- Nejspíš by se vyplatilo mít někde uložený invertovaný seznam (případně term-document matici), v naší implementaci se vždy počítá při spuštění aplikace. Přitom by stačilo ho přepočítávat jenom při změně dokumentů.

- Naše provedení experimentů způsobilo velké množství duplicitního kódu. Přitom by stačilo jen nějak parametrizovat původní třídu, ale vzhledem k tomu, že šlo jen o vytvoření grafů, tak nám to přišlo zbytečné.
- Na poslední chvíli jsme opravovali parser, ten se původně volal zvlášť pro každý dokument, přitom ho stačilo zavolat jednou a počítat všechny dokumenty naráz (což výrazně urychlilo výpočet).
- Naše reprezentace databáze není ideální. Pro větší množství souborů je naše lokální uložení prakticky nepoužité a aplikace jako taková je nenasaditelná.

7 Závěr

Celkově si myslíme, že se nám podařilo splnit velkou část zadání projektu. Vylepšení by si rozhodně zasloužila experimentální část práce, protože jsme pouze porovnávali sekvenční a ne-sekvenční přístup. Projekt nás bavil a jsme rádi, že jsme se s danou problematikou blíže seznámili.