Vektůrkův domeček

Naďa Fučelová, Petr Laštovička

27. dubna 2023

Obsah

1 O projektu

Práce vznikla jako semestrální práce v rámci předmětu BI-VWM na FIT ČVUT.

1.1 Cíl

Cílem projektu je vytvořit jednoduchý web schopný prohledávat anglickou verzi Wikipedie (její část) na základě uživatelem zadané query a následné hledání podobných dokumentů. K tomu bude využit základní vektorový model.

2 Způsob řešení

Jak již bylo zmíněno, projekt využívá vektorový model. Nejdříve provedeme analýzu a předzpracování textu - lemmatizace a stemming - a vytvoříme invertovaný seznam, kde si pro každou dvojici termu a dokumentu držíme jejich vzájemný vztah formou váhy.

2.1 Výpočet váhy

Používáme *tf-idf* schéma, v něm platí tyto vztahy. Zavedeme:

n ... počet dokumentů
f_{ij} ... počet výskytů termu t_i v dokumentu d_i
tf_i ... relativní frekvence termu t_i v dokumentu d_j, normalizováno na $0 \dots 1$
df_i ... počet dokumentů obsahující term t_i
idf_i ... inverzní počet dokumentů obsahující term t_i
w_{ij} ... váha dvojice t_i a d_j
Samotné váhy spočteme pomocí těchto vztahů

$$tf_{ij} = \frac{f_{ij}}{max_{i \in \hat{n}}\{f_{ij}\}}$$

$$idf_i = log_2(n/df_i)$$

$$w_{ij} = tf_{ij} \cdot idf_i = tf_{ij} \log_2(n/df_i)$$

Nyní si povšimneme, že pokud se term v dokumentu nevyskytuje, výsledná váha je rovna 0. To se nám bude hodit za okamžik. V tuto chvíli tedy máme matici dokumenty-termy.

2.2 Podobnost dokumentů

Každému dokumentu nyní odpovídá vektor vah pro jednotlivé termy. Vektorový model hledá nejbližší vektory v nějaké metrice. Jako naši míru použijeme cosinovu vzdálenost - z vlastnosti skalárního součinu plyne, že pokud se term alespoň v 1 dokumentu nevyskytuje, výsledek dopadne stejně jako kdyby term nebyl ani v jednom dokumentu. Důsledkem je, že musíme iterovat pouze přes dvojice term-dokument, kde váha v obou dokumentech je nenulová. Nuly nás tedy nezajímají, naši řídkou matici budeme moci uložit nějakou formou invertovaného seznamu.

Z našich výsledků vybereme k dokumentů s vektory nejbližšími (nejvyšší vzdálenost) k našemu vstupnímu vektoru q.

$$\mathrm{CosSim}(d_i, q) := \frac{d_i \cdot q}{|d_i| \cdot |q|} \tag{1}$$

2.3 Přijmutí vstupu

Vstupní vektor q získáme buďto z query uživatele - zpracujeme a každému termu v query přiřadíme váhu 1 - nebo v případě hledání podobných dokumentů použijeme vektor srovnávaného dokumentu.

2.4 Rozšíření: Prvních k termů dokumentu

Práce je rozšířena o experiment s přidáním taktiky zpracování pouze prvních k termů dokumentu. Myšlenka vychází z toho, že ta nejpodstatnější informace je nejčastěji někde na začátku dokumentu. Zbytek termů zanedbáme.

2.5 Rozšíření: Top k dokumentů

Druhým rozšířením základního modelu je možnost si pro každý term uložit váhu pouze pro významné dokumenty, tj. když se budou ukládat váhy termu, uložíme jich pouze k nejvyšších. Myšlenka vychází z toho, že v samotném vyhledávání by nám tyto váhy přispěli do cosinovy vzdálenosti zanedbatelně a proto je zanedbáme rovnou. Pro málo používané termy ke změně nedojde, jejich počet výskytů je menší k, ale pro často používané termy by jsme museli kontrolovat mnoho dokumentů, tímto snížíme výkonnostní nároky na vyhledávání s doufáme nízkou újmou na kvalitě.

3 Implementace

K vývoji aplikace byly použity jazyky Python a SQLite. Návod k použití a zpro-

voznění najdete v přiloženém README.md souboru.

3.1 Zpracování dat a vstupu

Jako zdroj dat využíváme data z anglické verze Wikipedie. Na jeho parsování (streamování) při vytvoření indexu využíváme knihovny *wiki_dump_reader* a *mwxml*. Následně odebereme rozcestníky a provedeme lemmatizaci pro angličtinu pomocí nástroje *nltk*. Současně s tím odebereme stopwords. Končíme s termy obsaženými na vstupu - dokumentu či uživatelské query.

3.2 Tvorba indexu

Na místo invertovaného seznamu implementovaného polem spojových listů jsme použili SQLite databázi. Vytvořili jsme 3 tabulky:

- term[termId, name]
- document[docId, title, text]
- weight[termId, docId, value]

Nyní jsme schopni pomocí série SQL příkazů pro sadu termů získat vektor vah pro všechny relevantní dokumenty.

Následně si zjistíme četnosti termů v datasetu a v jednotlivých dokumentech, dopočteme dle algoritmu výše a uložíme do databáze.

3.3 Vyhledávání

Jak již bylo zmíněno, pro query či dokument získáme odpovídající vektor vah pro termy - známe, o jaké termy se jedná. Následně z databáze získáme všechny dokumenty, které obsahují tyto termy. Postupně počítáme cosinovu vzdálenost a vždy držíme seznam 10 nejbližších dokumentů. Po projití vrátíme a zobrazíme tento seznam.

3.4 Uživatelská interakce

Na zpracování CLI argumentů pro práci s indexem používáme knihovnu *click*, na zobrazení webového grafického rozhraní využíváme knihovnu *Streamlit*.

3.5 Sekvenční průchod

Ve vyhledávání využíváme chytrost databáze, která je schopna rychle projít "invertovaný seznam". Pokud ovšem v databázi odstraníme indexy nad sloupci *termId* a *docId*, databázový stroj bude nucen sekvenčně projít celou databázi. Dopady na výkon budou zhodnoceny později.

4 Ukázka běhu programu

4.1 CLI

Aplikace disponuje základním CLI rozhraním pro spouštění subrutin jako třeba info o indexu, generování indexu, zapnutí sekvenčního průchodu či spuštění benchmarku.

Pomocí přepínačů jde nastavit tvorba indexu - velikost indexu, limit slov a top k dokumentů.

```
Usage: python -m vector_house [OPTIONS] COMMAND [ARGS]...

  Vector house command line interface Manage indexes and more.

Options:
  --help  Show this message and exit.

Commands:
  benchmark  Benchmark databases
  db-index   Handle database column indexes
  index      Creates index (DB)
  info       Show info about an index (DB)
```

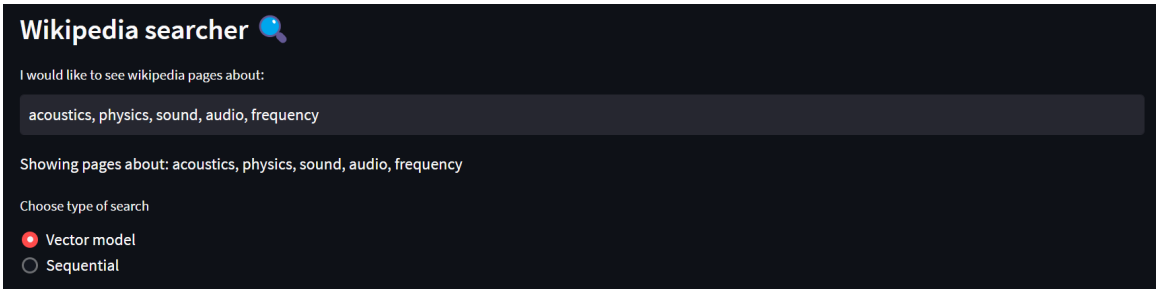
Obrázek 1: CLI nápověda

```
Hello, vector home is starting...
Running in /mnt/d/Dokumenty/Dokumenty/CVUT/BI-VWM/semestral
Showing index stats
Terms: 114710
Documents: 2000
Values: 1070265
Indexes created: False
```

Obrázek 2: CLI info příkaz

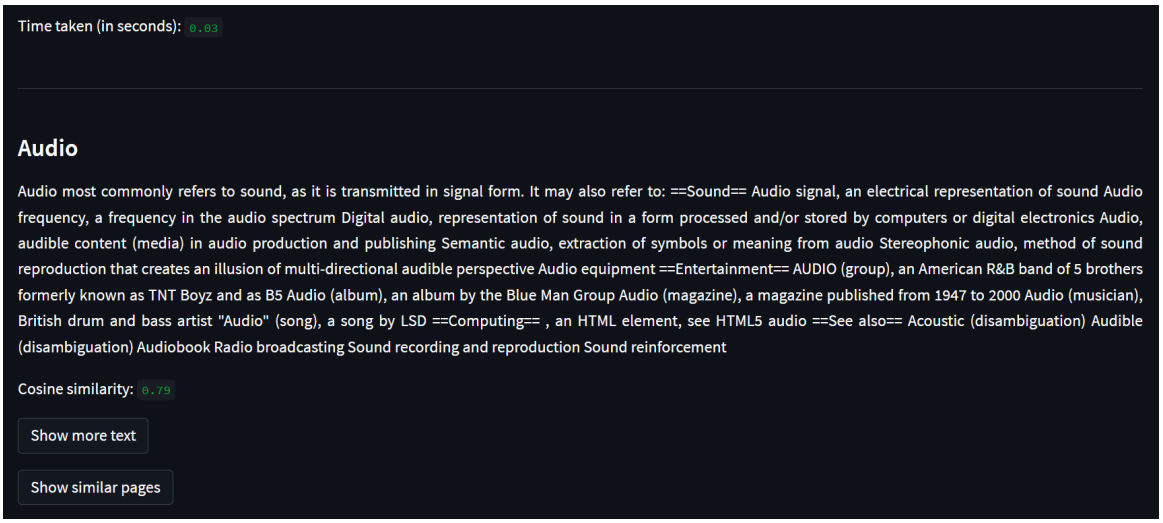
4.2 Vyhledávání

Po spuštění webové aplikace se uživateli zobrazí vyhledávací okno. Do tohoto okna může zadat jak klíčová slova tak souvislý text, na jehož základě chce dokumenty vyhledat. Text je následně lemmatizován a jsou z něj odstraněny tzv. stopwords. Uživatel si dále může zvolit, jestli chce pro vyhledávání použít výchozí vektorový model nebo chce využít sekvenčního průchodu.



Obrázek 3: Ukázka webové aplikace

Po dokončení vyhledávání se uživateli zobrazí 10 nejrelevantnějších dokumentů vzhledem k jeho dotazu. Na začátku stránky se také zobrazí čas, který trvalo než se výsledné dokumenty našly. Uživatel pak může postupně procházet stránku, přičemž u jednotlivých dokumentů je zobrazen jeho název, úryvek a kosinová vzdálenost vzhledem k dotazu. Uživatel má možnost u každého dokumentu zobrazit celý jeho text nebo najít 10 k němu nejpodobnějších dokumentů. Takto může uživatel pokračovat dál a prohledávat Wikipedii přes podobné dokumenty.



Obrázek 4: Ukázka výstupu aplikace

4.3 Výkon, rychlost

4.3.1 Způsob měření

Jako benchmark byl měřen čas potřebný na vykonání 6 query o až 20 termech a následné pro každou z nich hledání 3 podobnýchpro každou z nich dokumentů k některým z výsledků. Ačkoliv díky použití jiných indexů - jiné výsledky vyhledávání - nejsou tyto výsledky stejné (z podstaty věci), stále je považuji za dostatečně reprezentativní pro měření rychlosti operací.

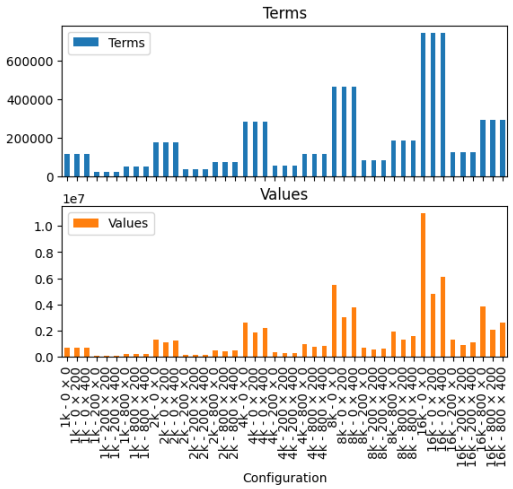
4.3.2 Parametry

Testy byly provedeny pro počty dokumentů 1000, 2000, 4000, 8000 a 16000, limity na počet zpracovaných slov 200, 800, ∞ a top dokumenty pro 200, 400, ∞.

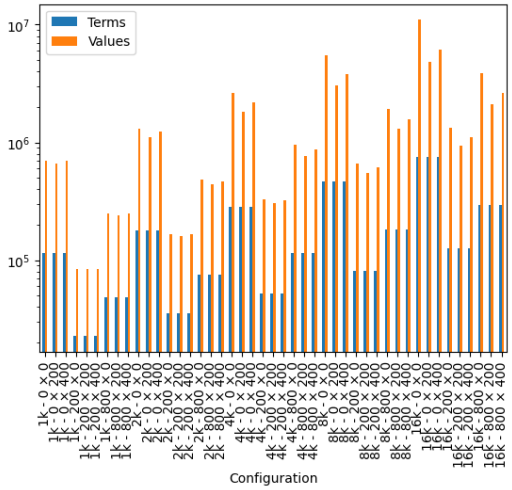
4.4 Počet termů a nenulových vah

Grafy níže vyjadřuje počet termů a nenulových vah v indexech v závislosti na počtu naindexovaných dokumentů.

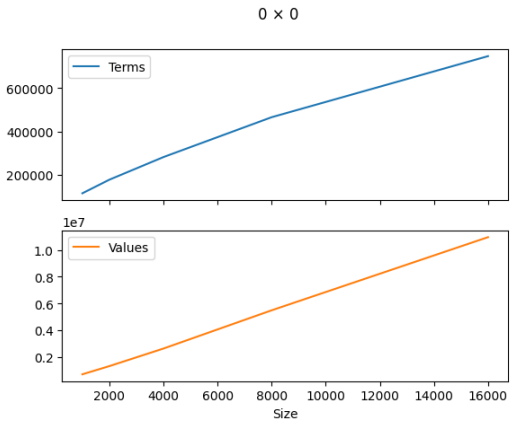
Popisky chápejte jako počet dokumentů - limit - top dokumenty. 0 znamená, že daný experiment nebyl použit.



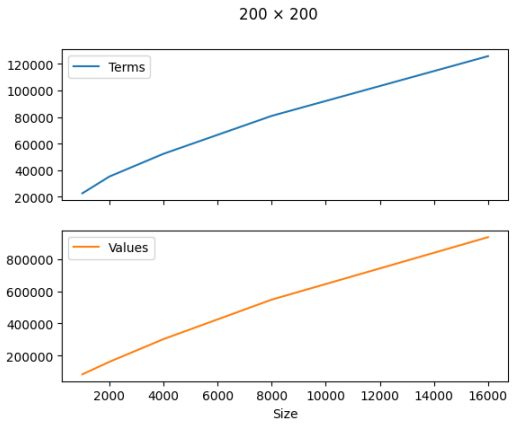
Obrázek 5: Počet termů a vah pro data



Obrázek 6: Logaritmická škála



Obrázek 7: Bez optimalizací



Obrázek 8: Nejvíce optimalizováno

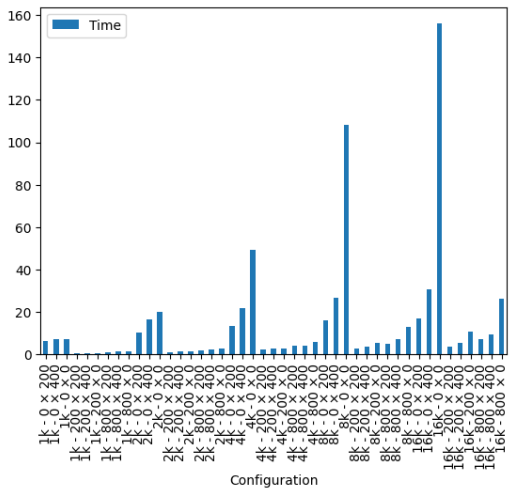
Z dat můžeme vidět mnoho zajímavých vzorů:

- První překvapivý fakt je, že počet termů roste s počtem dokumentů lineárně. Mohly bychom si říci, že na $400k$ vyčerpáme možnosti jazyka, není tomu tak.

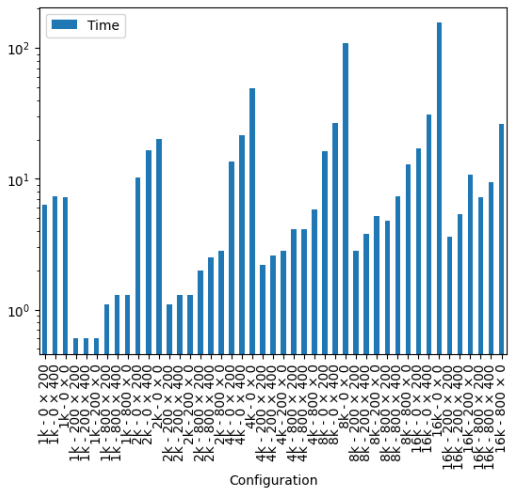
- Následně je hezky vidět, že limitování počtu termů opravdu funguje a že v úvodech článků se nenachází dost různých slov, aby oseknutí přebilo.
- U vah můžeme vidět, že jsme pro větší data dokázali srazit počet uložených vah na polovinu a to pro obě volby.
- Následně i ve vahách je vidět významný vliv úbytku termů.
- U vah i termů vidíme lineární růst.
- Z log grafů můžeme vidět, že pokud srazíme exponenciální růst počtu dokumentů, dostáváme obdobné relativní "zlepšení"pro všechny velikosti.

4.5 Rychlost

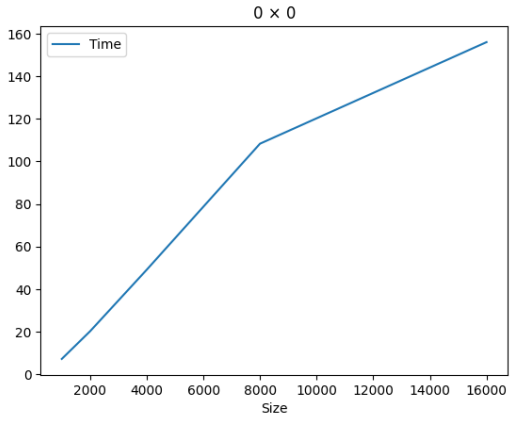
V naměřených datech můžeme sledovat lineární až logaritmickou závislost času velikosti indexu. Časový nárůst je způsoben mimo jiné tím, že termy se vyskytují v stále více dokumentech, které musíme následně projít.



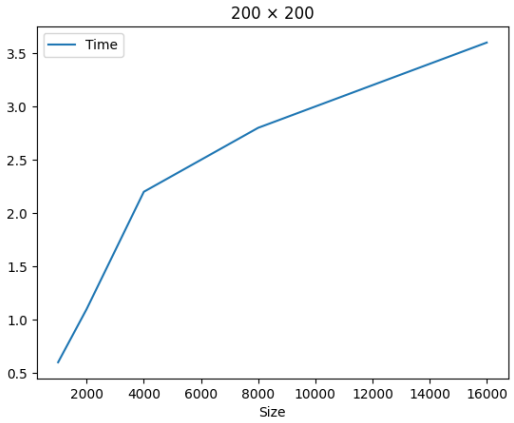
Obrázek 9: Naměřená doba běhu



Obrázek 10: Logaritmická škála



Obrázek 11: Bez optimalizací

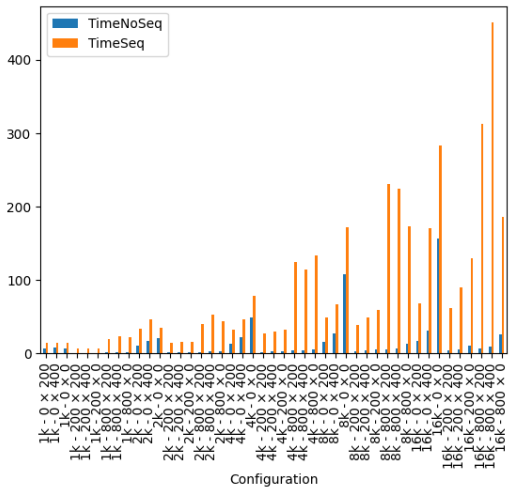


Obrázek 12: Nejvíce optimalizováno

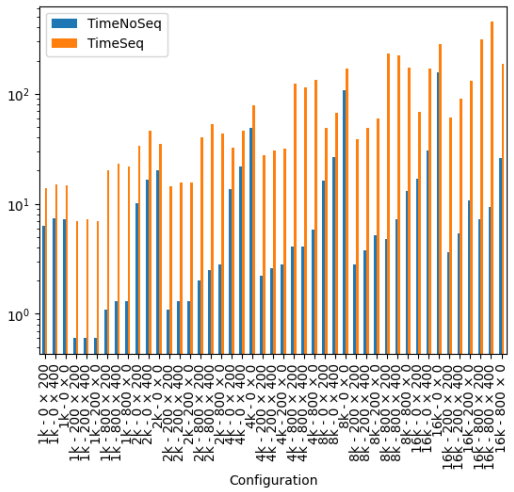
- Velice zajímavé pozorování je, že čas roste pouze v případě, kdy nepoužíváme limit nebo top documents. Dává to smysl - oba parametry limitují maximální počet vazeb dokumentu na ostatní. Jakmile je tento počet dovršen, budeme vždy porovnávat s "konstantním"(omezeným) počtem dokumentů.
- Jak by se dalo čekat, nejrychlejší jsou nejvíce osekané varianty.

4.5.1 Sekvenční průchod

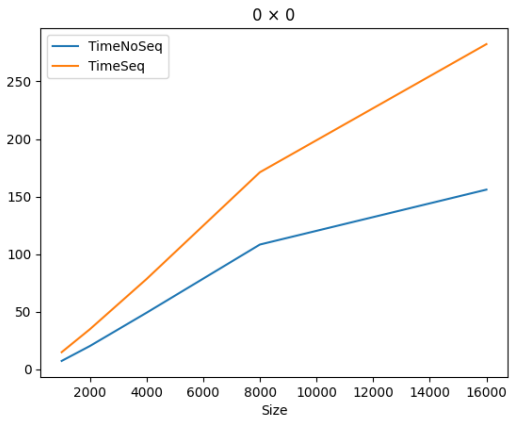
Sekvenční průchod potřebujeme zanalyzovat v dvou případech - pro žádné a použité optimalizace.



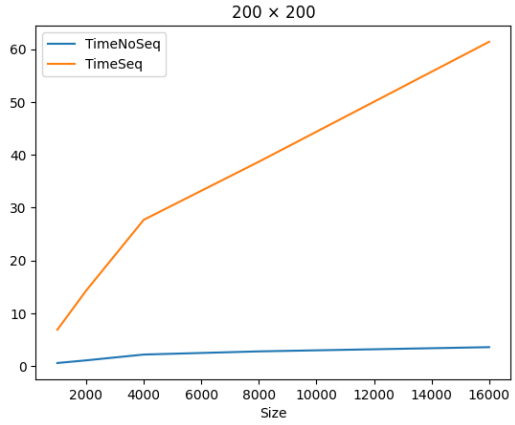
Obrázek 13: Naměřená doba běhu



Obrázek 14: Logaritmická škála



Obrázek 15: Bez optimalizací



Obrázek 16: Nejvíce optimalizováno

Pokud máme případ bez optimalizací, sekvenční průchod je zhruba 2x pomalejší. Aplikace má ale nezanedbatelnou režii na přečtení SQL cursoru (Python a for cykly), proto se nám samotná práce databáze na průchodu neprojevuje tolik, jak bychom možná mohli čekat.

Pokud se podíváme na ostatní případy, je krásně vidět, jak omezením vnitřních relací v databázi efektivně využijeme indexy. Samotný sekvenční průchod se také částečně zrychlí (méně vah a termů), ale nestačí to na rychlost práce s malými indexy. Ty se drží jak jsme si řekli v minulém bodě na zhruba konstantním výkonu.

4.6 Kvalita

Tato sekce se zabývá subjektivním zhodnocením kvality nalezených výsledků. Na základě několika různých dotazů jsme vyhledávali 10 nejpodobnějších dokumentů pomocí jinak sestavených indexů a snažili jsme se zhodnotit, jak relevantní jsou. Ve všech případech jsme měli k dispozici celkem 8000 různých dokumentů.

4.6.1 Základ

Ve výchozí formě vyhledávače, kdy nelimitujeme jak počet termů dokumentu, tak ani počet dokumentů pro jednotlivé termy jsme našli následujících 10 nejrelevantnějších výsledků k zadanému dokumentu:

1. **Asteroid**
   Asteroid, Interplanetary spaceflight, Exoplanet, Double planet, Absolute magnitude, Friedrich Bessel, International Space Station, Galileo project, Eclipse, Black hole

2. **Galileo project**
   Galileo project, International Space Station, Apollo 13, Asteroid, Apollo 15, Galilean moons, Comet Shoemaker–Levy 9, Apollo 12, Apollo 14, Apollo 16

3. **Albert Einstein**
   Albert Einstein, Copenhagen interpretation, David Hilbert, Friedrich Hayek, Enrico Fermi, Hannibal, H. G. Wells, George Washington, Alessandro Volta, Felix Bloch

4. **Apple Inc.**
   Apple Inc., Intel, David Beckham, Compaq, General Electric, Adobe Inc., Microsoft Windows version history, AOL, IKEA, Digital Equipment Corporation

5. **IKEA**
   IKEA, Intel, Apple Inc., General Electric, Dalek, Apple II series, E-commerce, Jennifer Lopez, BMW, Batman Returns

6. **Bentley**
   Bentley, Bugatti, Holden, Ferrari, Audi, International Astronomical Union, Aston Martin, Chrysler, Jeep, National Alliance (Italy)

7. **Cookie**
   Cookie, Dessert, Almond, Haggis, August, Glossary of French words and expressions in English, Cream, French fries, Chocolate, Industrial Revolution

V prvních třech případech dostáváme všech 10 dokumentů poměrně dobře souvisejících se zadaným. O něco horší výsledky dostáváme ve zbývajících čtyřech případech, kdy při hledání podobného dokumentu o společnosti Apple dostaneme na třetím místě dokument o bývalém anglickém fotbalistovi Davidu Beckhamovi nebo u řetězce IKEA máme ve výsledku dokument o americké zpěvačce Jeninifer Lopez. Navzdory těmto několika nesouvisejícím dokumentům však celkově dostáváme relativně dobré výsledky.

4.6.2 Prvních k termů v dokumentu

Vyhledávač modifikovaný do podoby, kde z každého dokumentu bereme v úvahu jen jeho prvních 400 termů nám vrací následujících 10 nejrelevantnějších dokumentů k danému dotazu:

1. **Asteroid**
   Asteroid, Comet Shoemaker–Levy 9, BC, Transport in Bosnia and Herzegovina, Allan Dwan, Hani Hanjour, J. G. Ballard, E2, Aarhus, Bolventor

2. **Galileo project**
   Galileo project, Interplanetary spaceflight, Ionosphere, Galilean moons, Apollo 12, Deep Space 1 , Giant planet, Adrastea (moon), Gun safety, Earth

3. **Albert Einstein**
   Albert Einstein, Battle of Ramillies, August 1, Heart of Oak, Eugene Wigner, Charles Proteus Steinmet, Copenhagen interpretation, Bohr model, Islamism, Caesium

4. **Apple Inc.**
   Apple Inc., General Electric, Intel, Borland, General Motors, Dispersion, Apple I, G4, Bill Gates, Blitz BASIC

5. **IKEA**
   IKEA, Chrysler, Economy of Italy, Austria-Hungary, Foreign relations of Iraq, Carl Linnaeus, Bald eagle, Carolina parakeet, Advertising, Bank of China Tower (Hong Kong)

6. **Bentley**
   Bentley, Chrysler, Ducati Motor Holding, Audi, Glass, David Deutsch, CD-R, Fatah, Geography of the Comoros, Franz Schmidt (composer)

7. **Cookie**
   Cookie, French fries, Banacek, Phase II feature requests/Cookies and privacy, Bourbon, Brownie McGhee, Building society, Grape, Chocolate, Achilles

V tomto případě dostáváme o něco méně související výsledky oproti předcházejícím. Ani při jednom vyhledávání nelze říci, že by všechny nalezené dokumenty byly relevantní k zadanému dotazu. Jako příklad lze uvést vyhledávání dokumentů podobných tomu o značce aut Bentley, kde se na poměrně vysoké místo dostal dokument zabývající se sklem, nebo vyhledávání dokumentů podobných tomu o teoretickém fyzikovi Albertu Einsteinovi, kde hned na druhém místě dostáváme bitvu u Ramillies, co byla válka o španělské dědictví.

Namísto prvních 400 termů jsme ještě zkusili prvních 600 termů. Výsledky, které jsme obdrželi jsou následující:

1. **Asteroid**
   Asteroid, Interplanetary spaceflight, Johann Elert Bode, Evolutionism, Son of Godzilla, Corona Australis, August 13, Transport in Bosnia and Herzegovina, BC, Allan Dwan

2. **Galileo project**
   Galileo project, Interplanetary spaceflight, Ionosphere, Adrastea (moon), Galilean moons, Deep Space 1, January 7, December 3, European Space Operations Centre, Ephemeris time

3. **Albert Einstein**
   Albert Einstein, Copenhagen interpretation, Charles Proteus Steinmetz, Islamism, Dieting, Bohr model, Eugene Wigner, Alexander I of Serbia, Georg Cantor, August 1

4. **Apple Inc.**
   Apple Inc., Apple I, Intel, Borland, Apple II, DC Comics, G4, Ibanez, Bill Gates, BearShare

5. **IKEA**
   IKEA, Dot-com bubble, Bald eagle, Carolina parakeet, Transport in the Faroe Islands, Galeon, GnuCash, Transport in Cuba, Telecommunications in Georgia (country), Armed Forces of Honduras

6. **Bentley**
   Bentley, Aston Martin, Chrysler, Hard disk drive, Clerihew, Glass, Honda, Ford GT40, Charge-coupled device, David Deutsch

7. **Cookie**
   Cookie, Hakka cuisine, French fries, Banacek, Gambling, Food, Brownie McGhee, Balalaika, Grape, Chocolate

Pozorujeme, že relevance mírně stoupla, ale oproti výchozímu stavu dosahujeme stále mnohem horších výsledků. Tedy co se relevance výsledků týká se nezdá být vhodným nápadem brát v úvahu jen prvních k termů.

4.6.3 Prvních k dokumentů pro term

Další modifikací kterou jsme zkoušeli byla taková, kde pro každý term ukládáme jen maximálně 800 dokumentů s nejvyšší váhou. Výsledky, které z takového vyhledávače dostáváme jsou následující:

1. **Asteroid**
   Asteroid, Interplanetary spaceflight, Exoplanet, Absolute magnitude, Double planet, Friedrich Besse, International Space Station, Galileo project, Eclipse, Black hole

2. **Galileo project**
   Galileo project, International Space Station, Apollo 13, Asteroid, Apollo 15, Comet Shoemaker–Levy 9, Galilean moons, Apollo 12, Apollo 14, Apollo 16

3. **Albert Einstein**
   Albert Einstein, Copenhagen interpretation, David Hilbert, Friedrich Hayek, Enrico Fermi, Hannibal, H. G. Wells, George Washington, Alessandro Volta, Felix Bloch

4. **Apple Inc.**
   Apple Inc., Intel, David Beckham, Compaq, General Electric, Adobe Inc., Microsoft Windows version history, AOL, Digital Equipment Corporation, IKEA

5. **IKEA**
   IKEA, Apple Inc., General Electric, Dalek, Apple II series, E-commerce, Jennifer Lopez, BMW, Batman Returns, Buckminster Fuller

6. **Bentley**
   Bentley, Bugatti, Holden, Ferrari, Audi, International Astronomical Union, Aston Martin, Chrysler, Jeep, National Alliance (Italy)

7. **Cookie**
   Cookie, Dessert, Almond, Haggis, August, Glossary of French words and expressions in English, Cream, Chocolate, French fries, Industrial Revolution

I v tomto případě se mezi dokumenty vyskytují případy, které nejsou relevantní vyhledávání, avšak jedná se spíše o menší počet. Porovnáme-li tyto výsledky s výchozí podobou vyhledávače, zjistíme, že se mezi sebou liší jen velmi málo.

V případě vyhledávání o značce aut Bentley či v případě teoretického fyzika Alberta Einsteina dostáváme výsledky naprosto totožné s výchozím vyhledávačem. U švédského řetězce s nábytkem IKEA se vyhledávání liší v jednom dokumentu. Ve výchozí podobě dostáváme ve výsledku výrobce mikroprocesorů Intel, zatímco v této podobě se tam tento dokument nevyskytuje vůbec a místo něj tam nacházíme Buckminstra Fullera, amerického architekta, což zdá se více odpovídá zadanému dotazu. Ve zbývajících případech dostáváme stejné dokumenty jen v jiném pořadí. Lze si všimnout, že v případě vyhledávání sušenky Cookie je oproti výchozímu vyhledávači

čokoláda umístěna před hranolky, což se jeví více související. Stejně tak v případě vyhledávání dokumentu o technologické společnosti Apple Inc. dostáváme ve výsledku firmu z počítačového průmyslu DEC před řetězcem s nábytkem IKEA.

Celkově by se dalo říci, že takto modifikovaný vyhledávač poskytuje obdobně relevantní výsledky jako jeho výchozí podoba.

Jelikož jsme v tomto případě dosáhli poměrně dobrých výsledků zkusili jsme ještě jak to dopadne pokud budeme pro jednotlivé termy uchovávat jen maximálně prvních 400 dokumentů s největší váhou. Toto jsou výsledky, které jsme obdrželi:

1. **Asteroid**
   Asteroid, Interplanetary spaceflight, Exoplanet, Absolute magnitude, Double planet, Friedrich Bessel, International Space Station, Galileo project, Eclipse, Black hole

2. **Galileo project**
   Galileo project, International Space Station, Apollo 13, Asteroid, Apollo 15, Comet Shoemaker–Levy 9, Galilean moons, Apollo 12, Apollo 14, Apollo 16

3. **Albert Einstein**
   Albert Einstein, Copenhagen interpretation, David Hilbert, Friedrich Hayek, Enrico Fermi, H. G. Wells, Hannibal, George Washington, Alessandro Volta, Felix Bloch

4. **Apple Inc.**
   Apple Inc., Intel, David Beckham, Compaq, General Electric, Adobe Inc., Microsoft Windows version history, AOL, Digital Equipment Corporation, IKEA

5. **IKEA**
   IKEA, Apple Inc., General Electric, Dalek, Apple II series, Jennifer Lopez, BMW, E-commerce, Batman Returns, Economy of Armenia

6. **Bentley**
   Bentley, Bugatti, Holden, Ferrari, Audi, International Astronomical Union, Aston Martin, Chrysler, Jeep, National Alliance (Italy)

7. **Cookie**
   Cookie, Dessert, Almond, Haggis, August, Glossary of French words and expressions in English, Cream, Industrial Revolution, Chocolate, French fries

Dokumenty, které dostáváme jsou stále relativně relevantní k vyhledávanému, avšak pozorujeme zhoršení. Například v případě vyhledávání sušenky Cookie se dostala průmyslová revoluce před čokoládu i hranolky, což není žádoucí. Rovněž v

případě řetězce s nábytkem IKEA byl nahrazen dokument o americkém architektu Buckminstrovi Fullerovi za dokument o ekonomii Arménie, který je méně relevantní. Celkově by se v tomto případě dalo říci, že tento vyhledávač dosahuje horších výsledků oproti výchozímu.

4.6.4 Zhodnocení

Na základě pozorování tedy lze říci, že pokud zvolíme vhodně počet dokumentů lze dosáhnout podobně relevantních výsledků, ale třeba být při volbě parametrů opatrný, protože může dojít jak ke zlepšení, tak i ke zhoršení. Velkou výhodou však dále zůstává fakt, že takto modifikovaný vyhledávač bude méně výkonnostně náročný a tak by mohlo stát za zvážení se touto formou vyhledávače zabývat dále.

5 Diskuze

5.1 Statický index

Naše implementace nepodporuje přidávání dokumentů do již vytvořeného indexu. Pokud by totiž přidávaný dokument obsahoval alespoň 1 term, který se vyskytl dříve, musíme až celý index přepočítat.

5.2 Váhy v query

Naše implementace v uživatelské query dá všem termům stejnou váhu 1. Další alternativou by bylo query zpracovat jako každý jiný dokument a podle toho termů přiřadit váhy. To by se dalo implementovat pomocí přidání sloupce s četností termu a sérií SQL dotazů.

5.3 Efektivnější jazyk/databáze

Jak Python, tak SQLite nepatří ani zdaleka k nejefektivnějším v svém oboru. Python jsme využili kvůli snadné implementaci lemmatizace a načítání dat z Wikipedie a protože nebylo mnoho dalších jazyků, které ovládáme oba. SQLite byla jasnou volbou pro rychlou a snadnou implementaci - stačí soubor, nemusí se zprovozňovat databázový server.

I tento kód by šel na úkor čitelnosti a good practices dále optimalizovat. Dalšího zrychlení by šlo dosáhnout například paralelizací.

5.4 LSI vektorový model

Pokud bychom chtěli významně vylepšit relevanci našich výsledků, použijeme LSI vektorový model. To ovšem nebylo zadáním naší práce, pokud ale v budoucnu budu implementovat vlastní hobby vyhledávač, využiji právě tento.

5.5 Přenositelnost na datasety

Pokud bychom chtěli vyhledávat například v datasetu novinových článků, musela by se změnit pouze úvodní část indexační pipeline, kdy se parsuje Wikipedia XML dump.

5.6 Produkční nastavení limitu a top documents

Z našich pozorování vyplývá, že nemá dobrý smysl limitovat počet termů, které pro každý dokument bereme v úvahu. Takto sestavený vyhledávač bude poskytovat horší výsledky, protože podstatné informace se vyskytují v celém dokumentu, ne jen na jeho začátku. Například pro zpravodajské články by mohl být výsledek značně odlišný.

Co se týče uchovávání jen top k dokumentů pro jednotlivé termy je třeba být velmi opatrný při konkrétní volbě parametru k, protože je-li špatně zvolen má výrazný vliv na kvalitu výsledku. V našem vyhledávači ve výchozím nastavení z tohoto důvodu ani tento parametr neomezujeme, protože přece jen relevance se testuje poměrně obtížně a nelze na základě pár výsledků usuzovat zásadnější závěry.

6 Závěr

Ačkoli naše řešení není nejrychlejší, vyhledané výsledky jsou zpravidla relevantní (vzhledem k počtu indexovaných dokumentů) a až zkrachuje Google a ChatGPT vynutí 10. tier předplatného, budeme ještě rádi za vlastní vyhledávač.