Dokumentácia k projektu O24 – WikiParser

Krátky popis projektu-problému a motivácia načo je to dobré (15 riadkov)

Témou môjho projektu je „Sparsovanie názvov kníh wikipédie a vytvorenie služby s vrátením zoznamu kníh, ktoré si môže používateľ prečítať (na základe mena autora, roku vydania, počtu strán, …).“

Motivácia: Túto tému som vyberal pre spracovanie z viacerých dôvodov. Jedným z nich je umožnenie používateľom vyhľadávať knihy na základe parametrov, ktoré je možné vidieť v téme tohto projektu. Každý z nás má svoje vlastné preferencie pre výber kníh, no mne osobne chýba služba, ktorá by mi vedela vrátiť tie knihy, ktoré by boli čo najbližšie môjmu štýlu v tomto odvetví. Osobne nemám rád hľadanie kníh v akýchkoľvek online kníhkupectvách, nakoľko veľa krát hľadám len inšpiráciu, poprípade mi tieto kníhkupectvá väčšinou zobrazujú len tie knihy, ktoré sú momentálne v predaji. Týmto spôsobom mám k dispozícií nástroj / službu, ktorá mi vráti zoznam všetkých existujúcich kníh z wikipédie, spomedzi ktorých si môžem vybrať.

Krátky popis problému: Základným problémom pri tejto téme je vytvoriť vlastné parsovanie, ktoré dokáže namapovať len určité časti z datasetu wikipédie, ktorý je vo formáte XML. Je teda potrebné využitie regulárnych výrazov, na základe ktorých vyberiem len mnou hľadanú oblasť, knihy. Z tých potrebujem vyextrahovať informácie o ich vlastnostiach, ako je napríklad typ žánru alebo meno autora. Pre vyriešenie tohto problému som využíval Apache Spark – PySpark, ktorý mi umožňuje prehľadávanie tohto datasetu paralelným spôsobom, pričom sme schopní využívať regulárne výrazy. Výsledné vyparsované dáta bude potrebné zatriediť do viacerých kategórií, pomocou ktorých budeme dané knihy vyhľadávať, teda bude potrebné vytvoriť index.

Prehľad súčasných riešení daného problému - existujúci softvér, algoritmy, vedecké články, linky (0,5 strany)

Pri vyhľadávaní existujúcich riešení, ktoré by sa zaoberali podobným problémom som natrafil na článok https://towardsdatascience.com/wikipedia-data-science-working-with-the-worlds-largest-encyclopedia-c08efbac5f5c. V tom autor rieši spracovanie jednotlivých dump-ov z wikipédie, pričom na parsovanie textu využíva „SAX Parser“ a „mwparserfromhell“. Pomocou týchto importov autor parsuje jednotlivé knihy za pomoci infoboxov. Na parsovanie využíva paralelizáciu či multithreading. Na vyhľadávanie zvolil paralelizáciu a využitie funkcie map, pričom som nezaznamenal využívanie indexov. Ďalším príkladom je modul „wikipedia“, ktorý sa inštaluje pomocou pip (https://towardsdatascience.com/wikipedia-api-for-python-241cfae09f1c). Ten vyhľadáva obsah z wikipédie na základe titulkov. Poprípade je možné využiť import „BeautifulSoup“, ktorý sa pripája na API wikipédie za pomoci importu „requests“ (https://www.jcchouinard.com/wikipedia-api/). Využíva taktiež jednotlivé tagy, pre získavanie obsahu z API. Okrem toho by sme napríklad mohli pri prvom riešení využiť na vyparsovaný text index „lucene“ alebo v prípade jazyku python „pylucene“ (https://pythonhosted.org/lupyne/examples.html#indexers) alebo lupyne (https://coady.github.io/lupyne/examples/). Pomocou takéhoto indexu by sme vedeli

pracovať jednoducho a rýchlo, nakoľko by sme v rámci kódu poukázali na jednotlivé časti v napríklad JSON formáte, pričom by sa nad ním vytvoril index a neskôr by sme pomocou tohto indexu mohli vyhľadávať.

Popis riešenia, použitý softvér, použité existujúce riešenia, popis problémov ktoré sa vyskytli, popis prác na projekte (1 strana)

Môj priebeh implementácie sa odrážal od zverejnených syláb a konzultácií pre tento projekt, ktoré sú uverejnené na wiki predmetu. Úvodná časť obsahovala vytvorenie pseudokódu, podľa ktorého som neskôr postupoval aj v rámci nasledujúcich častí a v ktorej som si vytvoril približnú logiku pre toto zadanie. Ďalšia časť sa zameriavala na prácu s malým datasetom, kde som si vybral jeden menší dump od wikipédie. Nad týmto datasetom som si vytvoril parsovanie a nad jeho výsledkami som následne implementoval vyhľadávanie, ktoré mi vrátilo knihy na čítanie. Nasledujúca časť už zahŕňala prácu s celým wikipedia dump-om. Obohatil som parsovanie dump-u o regulárne výrazy, pri ktorých som sa snažil o vyberanie každej s mnou žiadaných kategórií, nad ktorými som neskôr tvoril vyhľadávanie. Ich využitie rieši aj prípady rôznych zápisov informácií, konkrétne mám na mysli zápis informácií v Infoboxe, kde sme potrebovali ošetriť rôzne typy oddeľovačov. Ďalšou časťou bolo zapracovanie paralelizácie. Tá je v projekte zapracovaná dvojakým spôsobom. Jeden využíva modul „multiprocessing“, ktorý je zapracovaný do riešenia mimo využívania pyspark-u. Tam som si definoval 3 procesy. Prvý slúžil na čítanie riadkov z xml dump-u. Druhý dané riadky parsoval a tretí mal za úlohu zápis vyparsovaných záznamov do súborov. Druhý zo spôsobov využíval spark, konkrétne pyspark. To so sebou prinieslo radu problémov. Prakticky sme projekt implementovali nanovo, akurát sme využili základnú logiku celého parsera v jazyku Python. Najprv sme sa snažili čítať dataset ako textový súbor, čo so sebou prinášalo radu problémov. Výsledkom nášho snaženia bolo správne vyparsovanie výsledku, no s tým, že sa program neukončil, ale padol následkom error-u. Zaujímavé je však poznamenať, že výsledok z parsovania nám ostal zachovaný v súboroch. Po konzultácií tohto problému sme sa snažili pristúpiť k čítaniu datasetu iným spôsobom. Vytvorili sme si schému pre čítanie XML súborov, čo ale taktiež samo o sebe nestačilo. Pre správne fungovanie pyspark-u bolo potrebné zladiť verzie pre spracovanie xml (konkrétne sme zvolili spark-xml_2.12-0.15.0.jar a staršiu verziu sme zmazali). Po tejto zmene sa nám podarilo daný súbor čítať správne a program zbehol bez akýchkoľvek problémov. Ďalšou významnou zmenou bolo samotné využívanie regex-ov. Tie bolo potrebné úplne pozmeniť, nakoľko keď sme pracovali mimo pyspark-u, súbor bol čítaný riadok po riadku. V prípade pyspark-u sme dostali na vstup (v našom chápaní v rámci jedného riadku) celú stránku wikipédie. To spôsobilo pretvorenie regex-ov, kde sme si najprv potrebovali vyparsovať „Infobox book“, ktorý popisuje nami hľadanú knihu a na daný vyparsovaný výsledok sme následne potrebovali opakovane ďalšie regulárne výrazy. Tie už riešili parsovanie danej knihy podľa jednotlivých kategórií. Po týchto úpravách sme potrebovali vyriešiť ďalší problém a tým je paralelizácia ako taká. V rámci nej sme si zadávali počet paralelných procesov, ktoré daný vstupný súbor roztriedili do viacerých „task-ov“. Ich výsledkom bolo n-súborov. Teda bolo taktiež potrebné vyriešiť čítanie obsahu n-súborov s tým, aby sme nenarušili index. Ďalšou súčasťou riešenia bolo vytvorenie indexu, na ktorý som nepoužil žiaden z importov (ako je napríklad lucene / pylucene / lupyne). Môj vlastný index pracuje nad každou z mnou zadaných kategórií. Teda každý z kategórií má vlastný index, pričom jej obsahom sú jednotlivé záznamy. Príkladom je žáner „fantasy“, pričom tento žáner má okrem jeho názvu v rámci index-u priradené tie knihy, ktoré túto informáciu o knihe

spĺňajú. Pri napĺňaní index-u prechádzame jednotlivé vyparsované záznamy len raz, pričom sa pozeráme či daný prvok kategórie existuje v indexe. Tu riešime 2 prípady. Ak existuje, priradíme mu id danej knihy, ktorú parsujeme. Ak daný prvok v indexe neexistuje, pridáme ho na koniec a priradíme nami parsovanú knihu. Tento postup sa deje pre všetky z indexov pri len jednom prechádzaní celého vyparsovaného zoznamu kníh. Taktiež sme pred spracovaním každej knihy daný vyparsovaný záznam pretransformovali do JSON formátu. Po vytvorení indexov sme pristúpili k vyhľadávaniu danej knihy na základe nami zadaných parametrov.

Popis dát na ktorých ste testovali, nejaká dáta aj ako zip na wiki.

V rámci projektu som pracoval na dump-och z EN Wikipédie. V prvom prípade šlo o čiastočný dump o veľkosti približne 260 mb (https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles1.xml-p1p41242.bz2). V ďalšej práci na väčšej množine som využíval celý dump EN Wikipédie, ktorého veľkosť je viac ako 34 GB (https://dumps.wikimedia.org/enwiki/20220920/enwiki-20220920-pages-meta-current.xml.bz2).

Vyhodnotenie slovné subjektívne na nejakých konkrétnych príkladoch. Vo väčšine projektov aj vyhodnotenie pomocou presnosti a pokrytia (precision a recall) (0,5-1 strana)

Tento projekt hodnotím ako úspešne dokončený vzhľadom na fakt, že sa mi podarilo aplikovať využitie regulárnych výrazov, ako aj paralelizácie v dvojakej podobe (lokálne cez python s využitím modulu multiprocessing a taktiež v rámci pyspark-u), či indexovania (ktoré som si naprogramoval sám). Po vyparsovaní a zaindexovaní kníh je taktiež možné vyhľadávať, čo spĺňa parametre zadania, ktoré som si na začiatku stanovil. Služba / projekt mi vráti zoznam kníh na základe mnou určených parametrov. Program taktiež funguje ako na malých datasetoch týkajúcich sa wikipédie, tak aj na celom dump-e. V rámci hodnotenia sa mi podarilo zaznamenať časy jednotlivých verzií tohto programu, pričom ich výsledky sú zaznamenané v tabuľke nižšie:

Prvotné riešenie	*Technológia*	*Popis*	*Čas*	*Percentuálne zlepšenie voči prvotnému riešeniu*
Áno	Python bez využitia Sparku	Bez paralelizácie	7273 sekúnd	-
Nie	Python s využitím Sparku	S použitím paralelizácie (multiprocessing)	5821 sekúnd	19,96%
Nie	Python s využitím Sparku	S použitím paralelizácie (Spark)	4033 sekúnd	44,55%

Príklad vyhľadania kníh od autora Graeme Base pomocou tohto parsera:

```
Welcome to WikiParser! :)
Creating indexes...
Trying to find a book...

We can recommend you this set of books. Enjoy :)
The Sign of the Seahorse
Unos Garden
The Eleventh Hour
The Worst Band in the Universe
Animalia
```

Github repository link: https://github.com/Hurtuk18/VINF_Information_Retrieval

Príklad vyparsovaného súboru:


```
{'name': ['Adventures of Huckleberry Finn'], 'author': ['Mark Twain'], 'country': ['United States'],
'language': ['English'], 'series': ['Tom Sawyer'], 'genre': ['Picaresque novel'], 'pub_date': '1884',
'pages': '366'}
{'name': ['I Robot'], 'author': ['Isaac Asimov'], 'country': ['United States'], 'language':
['English'], 'series': ['Robot series (Asimov)'], 'genre': ['Science fiction'], 'pub_date': '1950',
'pages': '253'}
{'name': ['Ivanhoe'], 'author': ['Walter Scott'], 'country': ['Scotland'], 'language': ['English'],
'series': ['Waverley Novels'], 'genre': ['Historical novel', 'chivalric romance'], 'pub_date':
'1819', 'pages': '401'}
{'name': ['Johnny Got His Gun'], 'author': ['Dalton Trumbo'], 'country': ['United States'],
'language': ['English'], 'series': [], 'genre': ['Anti-war novel'], 'pub_date': '1939', 'pages':
'309'}
{'name': ['Kalevala'], 'author': ['Elias Lönnrot'], 'country': ['Grand Duchy of Finland'],
'language': ['Finnish language'], 'series': ['Keith Bosley'], 'genre': ['Epic poetry', 'National
epic'], 'pub_date': '1835', 'pages': '1'}
{'name': ['Icehenge'], 'author': ['Kim Stanley Robinson'], 'country': ['United States'], 'language':
['English'], 'series': [], 'genre': ['Science fiction'], 'pub_date': '1984', 'pages': '262'}
{'name': ['Known Space'], 'author': ['Larry Niven'], 'country': ['United States'], 'language':
['English'], 'series': [], 'genre': ['Science fiction'], 'pub_date': '1964', 'pages': []}
```


Spustenie, inštalácia softvéru, použitie softvéru. (0,5-1 strana)

V prvom kroku je potrebné mať stiahnutý dump z Wikipédie EN a umiestnené v priečinku /src. Odkazy na používané dumpy sú v časti „Popis dát na ktorých ste testovali, nejaká dáta aj ako zip na wiki.“

Pre spustenie projektu je potrebné mať nainštalovaný Python vo verzií 3.10, a mať naimportované nasledujúce moduly:

- modul PySpark,
- modul json,
- modul re (ide o modul pracujúci s regulárnymi výrazmi),
- modul time (pre meranie trvania chodu daného projektu).

Taktiež je potrebné stiahnuť verziu pre spracovanie xml formátu v pysparku. Najprv je potrebné nastaviť sa do priečinka /pyspark/jars (v mojom prípade /Users/adam/mambaforge/envs/GitHub/lib/python3.10/site-packages/pyspark/jars)
a stiahnuť si verziu xml: spark-xml_2.12-0.15.0.jar. Využijeme na to nasledujúci príkaz, ktorý zadáme do konzoly:

- curl -O https://repo1.maven.org/maven2/com/databricks/spark-xml_2.12/0.15.0/spark-xml_2.12-0.15.0.jar

Tiež treba myslieť na to, aby sme mali vytvorený priečinok /src, do ktorého uložíme dump-y Wikipédie.

Výsledné vyparsované súbory budú uložené v ./final_books.
Pri opätovnom spustení je potrebné vymazať priečinok /final_books, alebo prepísať miesto uloženia v súbore spark_parser.py.

```
237     wiki_rdd.saveAsTextFile("./final_books")
```

Pre zmenu údajov pre vyhľadávanie je potrebné upraviť vstup pre vyhľadávanie. Predpripravené sú 2 (riadok 306 a riadok 307), pre iné prepíšte riadok 3O7 v súbore spark_search.py.

```
306     #test_books_list = find_my_books('-', "Stephen King", '-', '-', '-', "dark fantasy", '-', '-')
307     test_books_list = find_my_books('-', "Graeme Base", '-', '-', '-', '-', '-', '-')
```

Github repository link: https://github.com/Hurtuk18/VINF_Information_Retrieval

Je potrebné dodržať formát vstupu, ktorý je vysvetlený nižšie a zadať ho do riadku 307 ako to vidíme na obrázku nad tým.

```
Hello in WikiParser!

I've created an app that will search for the perfect set of books for you based on input parameters from the console!

So LET'S GOO!!!

Please insert the input from the keyboard, following these rules:
- To separate multiple entries for a category use ',' and after this symbol use Space ' '
- To separate categories use ';' and after this symbol use Space ' '
- To skip a category, type '-'
- !!! It is NECESSARY to enter all categories !!!

The order of the categories is:
- 'name'
- 'author'
- 'country'
- 'language'
- 'series'
- 'genre'
- 'pub_date'
- 'pages'

Thank you for using WikiParser!
```

Pre spustenie parsovania:

- spark-submit --jars /Users/adam/mambaforge/envs/GitHub/lib/python3.10/site-packages/pyspark/jars/spark-xml_2.12-0.15.0.jar spark_parsing.py

Pre spustenie indexovania a prehľadávania vyparsovaných súborov:

- spark-submit --jars /Users/adam/mambaforge/envs/GitHub/lib/python3.10/site-packages/pyspark/jars/spark-xml_2.12-0.15.0.jar spark_searching.py

Pre spustenie testov:

- potrebné nastaviť premennú test na True v súbore spark_search.py (riadok 296)

```
292 ▶ if __name__ == '__main__':
293       print("Welcome to WikiParser! :)")
294
295       # If you want to run unit tests set variable 'test' as True, otherwise False
296       test = True
```

Github repository link: https://github.com/Hurtuk18/VINF_Information_Retrieval