Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

Vyhľadávanie informácií

Údaje o vojne na Ukrajine z pohľadu oboch bojujúcich strán

Adam Žák, Bc.

AIS ID: 103189

E-mail: xzaka@stuba.sk

Course: Information Retrieval

Day and date of the seminar: Monday, 16:00

Teacher: Ing. Igor Stupavský

Winter semester 2022/2023

Informácie

Github

https://github.com/ZakAdam/VINF

Motivácia

Cieľom tejto práce je vytvoriť prehladávač a hlavne porovnávač informácií o vojne na Ukrajine. Vďaka tejto aplikácii bude možné porovnávať informácie a údaje z vojny prezentované z obidvoch bojujúcich strán.

Zdroje

Za účelom porovnania obidvoch bojujúcich strán sme scrapovali 4 zdroje, 2 ukrajinské zdroje a 2 ruské zdroje. Za týmto účelom sme si zvolili anglicky písané zdroje.

Ukrajinské zdroje

1. https://kyivindependent.com/
2. https://www.kyivpost.com/

Ruské zdroje

1. https://www.themoscowtimes.com/
2. https://tass.com/

Riešenie projektu

Crawlovanie + Scrapovanie

V prvej časti projektu sme sa zamerali na crawlovanie a scrapovanie daných zdrojov. Za týmto účelom sme spravili 4 scripty na scrapovanie, pretože pri každej stránke sa použil iný prístup na crawlovanie a scrapovanie.

kyiv_independent.py

Pri spracovaní tejto stránky sme zvolili tradičný prístup, kedy sme na začiatok nahrali zoznam URLs, ktoré navštívime. V rámci nich potom pomocou RegEx-ov hľadáme odkazy na články. Tie vložíme do poľa ktoré potom začneme iterovať. Navštevujeme dané odkazy, v ktorých hľadáme title, date a content a spolu s tým aj ďalšie odkazy. Tieto opätovne pridáme do poľa cez ktoré iterujeme a navštívené odkazy z poľa odstránime a pridáme do set-u, ktorý drží navštívené odkazy. Po zadanom počte iterácii obe tieto polia zapíšeme do .txt súboru pre prípad, že by sa scrapovanie prerušilo. Získané dáta v cykle zapisujeme do dataframu, ktorý taktiež po zadanom počte iterácií uložíme do CSV súboru. Takto iterujeme cez všetky odkazy, až kým nemáme prázdne pole.

kyiv_post.py

Pri spracovaní tohto zdroja sme zvolili iný postup, kedže tento zdroj má URLs v tvare -> kyiv_post.com/post/<ID> teda nám namiesto prehľadávania a hľadania odkazov stačí iterovať cez všetky dostupné ID a navštevovať tieto stránky. Vďaka tomu iterujeme, zvyšujeme ID a počítame počet chýb, teda neexistujúcich ID. Na žačiatok si nastavíme koľko chýb môžeme dostať než ukončíme beh programu. Potom pri každej chybe zvýšime počet chýb a ak tento dosiahne dané číslo, tak beh ukončíme. Počet chýb pri každej úspešne navštívenej stránke zresetujeme na 0. Vďaka tomu sa nám podarí prejsť všetky články, a až ked narazíme na ID, ktoré neexistujú, tak sa nám počet chýb zvyšuje. Získané informácie opäť ukladáme do dataframu, ktorý na záver uložíme do CSV súboru.

moscow_times.py

Pri tomto zdroji postupujeme podobne, ale inak ako pri Kyiv Post. Tuto cez URL: https://www.themoscowtimes.com/ukraine-war/{offset} dokážeme získavať čisté HTML odpovede so zoznamom odkazov na články ktoré spadajú do tejto kategórie, keďže tento zdroj má aj iné kategórie. Teda pri spracovaní tohto zdroja máme dva cykly, prvý prechádza články cez offset, načíta z HTML odpovede všetky href odkazy a tie pridá do poľa. Cez toto pole sa potom iteruje v druhom cykle, kde sa už priamo načítavajú pre nás podstatné informácie (title, date, content) ktoré sa opäť ukladajú do dataframu a následne do CSV súboru.

tass.py

Posledný zdroj sme scrapovali s použitím API na ich stránke. Na tejto API sa dá pýtať články v batchoch po 200, ale nie priamo obsah, len metadáta o článkoch, teda title, url, a date. V prvom cykle prechádzame API a v batchoch po 200 článkov si pýtame metadáta. Z nich načítame title a time do dictionary ako pole kde kľúč je url na článok. Takto zaplníme dictionary so všetkými článkami na stránke. Potom s týmito dátami prechádzame v druhom cykle, kde cez URL vždy načítame článok, sprocesujeme jeho obsah a potom do dataframu uložíme práve získaný title (ktorý sme získali už cez API), tak isto dátum, priamo z API a na záver obsah, ktorý sme stiahli už priamo zo stránky, ktorú sme museli navštíviť.

Indexovanie

Na indexovanie som použil PyLucene, ktorý som naplnil dátami cez script indexer_class.py. Tento s využitím Štandardného Analyzéra zapisuje do zadaného priečinka. Pre každý riadok v csv súbore, ktorý čítam po riadkoch, vytvorím nový dokument v indexe, doňho vložím stĺpce dát, takto prejdem celý súbor a potom index commitnem a zatvorím.

Queryovanie

Queriovanie dát spracúva trieda query_class.py. Táto s použitím rovnakého analyzéra na zadaný index zistí, že či sa jedná o date search alebo o string search. Podľa toho upraví query_parser, vyhľadá zadaný vstup pre obe strany, tie uloží do poľa. Následne, ak sa nejedná o testovacie spustenie, chceme aj výpis do terminálu, preto voláme funkciu print_results, ktorá vlastne zistí, či dostala nejaké výsledky, alebo sa žiadne nenašli. Ak sa žiadne nenašli, tak to vypíše a ukončí procesovanie. V inom prípade vytvorí prázdny dict, ktorý má rovnaké polia, ako výsledný, aby keď vypisujeme vstupy vedľa seba a na jednej alebo druhej strane chýba, tak sa dorovná cez tento prázdny dictionary. Vďaka tomu máme výsledky pekne zarovnané, aj keď tam chýbajú dáta. Potom sa vypisujú riadky, kedy je ľavá časť zameraná pre ukrajinské stránky a pravá pre ruské. Cez textwrap.wrap sa text zarovná na jednu alebo druhú stranu.

Distribuované spracovanie

Kód na distribuované spracovanie je v súbore spark_code.py. Taktiež sa v repozitáre nachádzajú ďalšie dva súbory mwxml_code.py a mwxml_spark_code.py ktoré boli pokusy o to použiť knižnicu mwxml na

spracovanie wikipédia dumpu. To mi však prednášajúcim nebolo schválené preto vznikol spark_code.py. Tento kód inicializuje Spark session, zadefinuje schému s dvoma stĺpcami title a text. Potom načíta dáta cez databricks.spark.xml s nami zadanou schémou. Zo všetkých stránok na wikipédii sa vyberú len tie, ktoré majú title like: "Portal:Current events/%". Z týchto pomocou nami zadefinovaných funkcií získame dátum, na ktorý sa events z tej stránky viažu a samotné údaje o udalostiach. Takto získané dáta potom zapíšeme do JSON súborov.

Scripts

Ďalším priečinkom je priečinok s názvom scripts. Tento udržiava pomocné scripty použité primárne na spracovania dát z wikipédie a ich spojenie s stiahnutými dátami. get_wiki_content.py je script, ktorý zistí, ktoré všetky unikátne dátumy máme v datasete, pomocou group_by, a následne pre tieto dáta robí requesty na wikipédiu. Tam potom pomocou RegEx-ov stiahne pre nás relevantné informácie (iné svetové udalosti a informácie o vojne z Wikipédie), ktoré obohatia naše dáta. Ďalším scriptom je wiki_date.py ktorý prejde všetky dáta a pre každý záznam skonvertuje jeho dátum na wikipédia formát dátumu teda YYYY_Month_DD, ktorý sa potom následne používa na groupovanie v scripte ktorý sťahuje dáta spomenutý skorej. merge_csvs.py slúži na spojenie CSV súborov z 4 rôznych stránok do jedného veľkého CSV. Tento script sa spustí ešte pre ostatnými, aby sme každý script nemuseli púšťať 4 krát. Posledným scriptom je add_wiki_to_data.py ktorý prejde JSON všetkých stiahnutých events z wikipédie a pridá ich k stiahnutým dátam, ktoré majú rovnaký dátum ako dané events. Tak získame kompletný súbor dát.

Pseudokód

1. Spusti sťahovanie stránok cez 4 crawlery (vypísané crawlery)
2. Spoj 4 csv data súbory do jedného veľkého súboru (merge_csvs.py)
3. Pridaj k dátam stĺpec s formátovaným dátumu (wiki_date.py)
4. Stiahni dáta z wikipédie k daným dátumom a ulož ich do JSONu (get_wiki_content.py)
5. Pridaj dáta z wikipédie k dátam ako nový stĺpec (add_wiki_to_data.py)
6. Spusti skript na indexovanie, ktorý vytvorí nový index nad dátami (indexer_class.py)
7. Spusti program cez príkaz: **python3 war_news_comparer.py**
8. Zvol želanú operáciu a využívaj program!

Dáta

Dáta sú sťahované z 4 rôznych zdrojov a preto je pre nás dôležité ich držať v unifikovanje forme pre jednotné spracovanie. V rámci nášho spracovania sme sa rozhodli získavať 5 základných atribútov:

1. **title** - titulok článku, dôležitý pre nás lebo stručne opisuje obsah
2. **link** - odkaz na článok pre ďalšie použitie, prípadnú kontrolu
3. **country** - krajina ktorá dané informácie publikovala
4. **date** - dátum a čas, kedy daný článok vznikol
5. **content** - samotný obsah, ktorý je v článku

Tieto dáta sme potom obohatili o dáta z wikipédia. Vďaka tomu sme pridali stĺpce ktoré nám pomohli s týmto pridávaním.

6. **wiki_date** - dátum preformátovaný na wikipédia formát
7. **wiki_data** - dáta načítané z wikipédie

Stĺpec wiki_date a date sa môžu zdať duplicitné, a pôvodne som chcel premazať stĺpec date hodnotou z wiki_date, ale nakoniec som radšej pridal nový stĺpec, pretože pri niektorých date údajoch je napríklad čas, ktorý môže byť dôležitá informácia a bola by škoda o ňu prísť. Pamätajúc na slová pána prednášajúceho: “Nikdy žiadne dáta nemažte” som teda rozhodol pre takýto prístup.

Samotné dáta teda na konci vyzerajú nasledovne:

ID	10205
title	Families of 131 missing Russian servicemen contact SBU – advisor to SBU chief
link	https://www.kyivpost.com/post/10678
country	Ukraine
date	October 16, 2014, 8:40 pm
content	“We have received reports on 131 servicemen of the Armed Forces of the Russian Federation having gone missing; their fate remains unknown and they might have actually become victims of the Russian regime through having been sent to war against Ukraine,” the SBU press center quoted Lubkivsky on Oct. 15. Adviser to the SBU chief urged the mothers’ unions and Russian families who are searching for their relatives to keep addressing the Security Service of Ukraine to look for their

	next of kin. “We are ready to fully assist them in this search,” he added.
wiki_date	2014_October_16
wiki_data	Armed conflicts and attacks War in North-West Pakistan - The Pakistan Air Force claims to have killed at least 27 militants in overnight airstrikes. (AFP via Hindustan Times) …

Dlhé stĺpce (content, wiki_data) sú skrátené kvôli ukážke, aby nezabrali celú prácu.

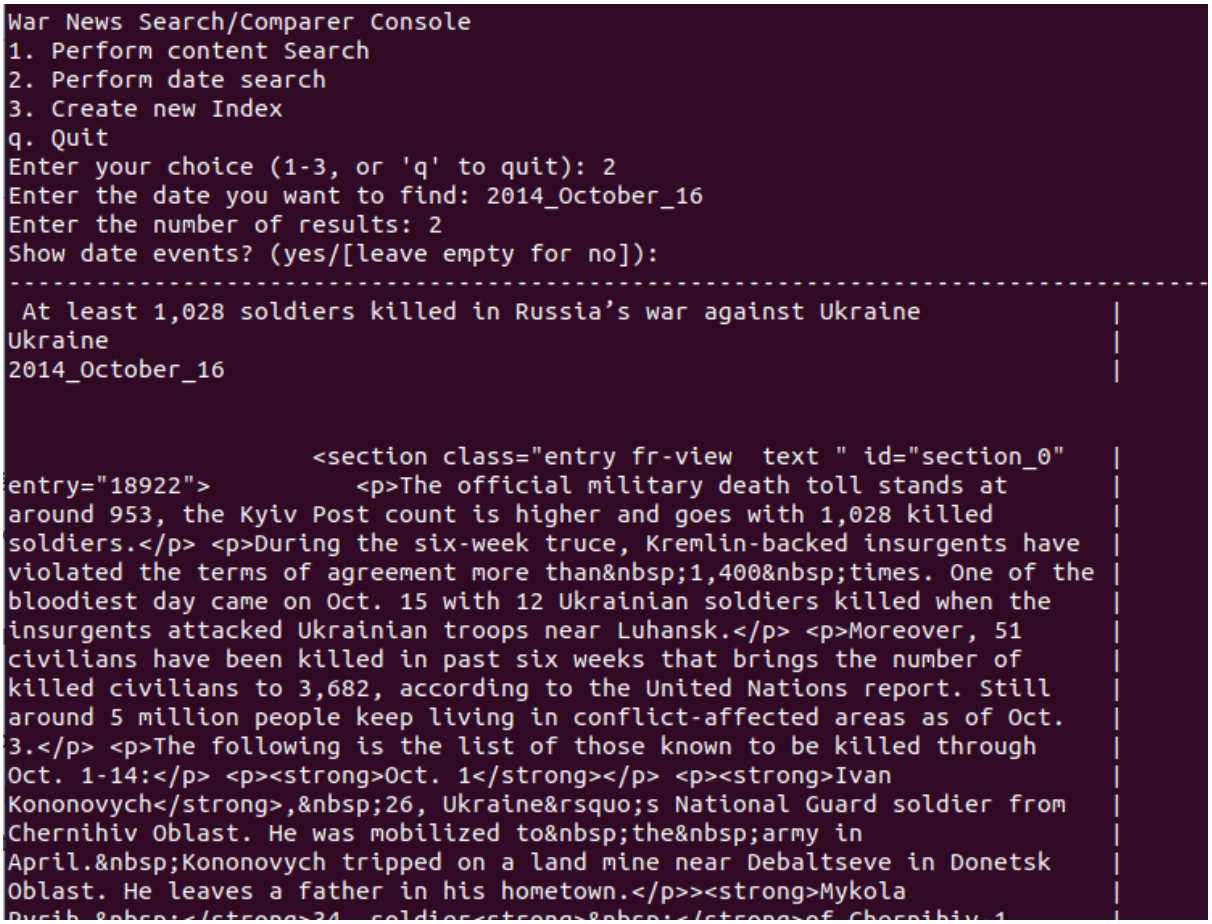
Používateľské rozhranie

V rámci používateľského rozhrania sa pracuje s konzolovou aplikáciou, ktorá si pýta vstupy podľa zadaného príkazu.

Na obrázku vidíme spustenie programu, kedy sa nás program najprv pýta, ktorú akciu chceme vykonať, pri zvolení čísla 1 (teda vyhľadávania v dátach) sa nás program opýta doplňujúce informácie: *String ktorý chceme vyhľadať*, *Počet výsledkov ktoré chceme dostať*, *A či chceme zobraziť aj wikipédia dáta*.

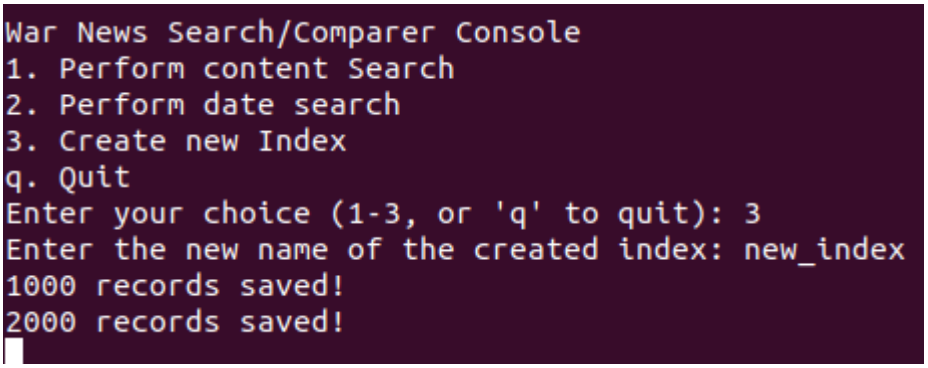
Následne dostaneme odpoveď rozdelenú na dve strany, kedy naľavo sú vždy výsledky z Ukrajinskej strany a napravo sú vždy výsledky z Ruskej strany. Vidíme, že pri hľadanom stringu “SBU missing” dostaneme prvú odpoveď článok ktorý sme videli už v ukážke dát v tabuľke vyššie. Na ruskej strane dostaneme iný článok, ktorý najviac súvisí s naším vstupom.

Pri zvolení hodnoty 2 dostaneme nasledovný výstup:



Kedy vidíme, že sa systém pýta na hľadaný dátum, počet výsledkov a zobrazenie wikipédia dát. Následne nám zobrazí článok z daného dátumu a čo je zaujímavé, tak vidíme, že len pre Ukrajinskú stranu a že z Ruskej strany pre daný dátum neboli nájdené výsledky.
Pri všetkých search možnostiach sa zobrazia aj wikipédia dáta, pri správne zvolenej možnosti vstupu. Tie sú zobrazené nasledovne:

Pri tretej možnosti create new index dostaneme:



Zadáme názov nového indexu a spustí sa indexovanie.

Testy

Na testovanie bola použitá knižnica *import unittest*, ktorá nám umožňuje porovnávanie prijatých hodnôt a očakávaných hodnôt. Testov je celkovo 8 a testuje sa korektnosť získaných odpovedí. Názvy testov odzrkadľujú to čo testujú. Testy sú v súbore: test_class.py

tests_results	sleduje či sa v odpovediach nachádza hľadané slovo
test_number_of_results	sleduje či sedí počet vrátených výsledkov s zadaným počtom želaných výsledkov
test_only_ukr	sleduje či sa vracajú výsledky aj keď sú výsledky len pre Ukrajinskú stranu
test_only_rus	sleduje či sa vracajú výsledky aj keď sú výsledky len pre Ruskú stranu
test_empty_input	sleduje správne fungovanie programu aj v prípade prázdneho hľadaného stringu
test_wiki_data_enabled	sleduje či pri nastavení zobraz wikipedia dáta sú tieto naozaj súčasť výsledku
test_date_search	sleduje či funguje vyhľadávanie podľa dátumu a vrátané výsledky sú zo zadaného dátumu
test_everything	sleduje či sa pri správnom nastavení vráti korektne krajina, zobrazí wikipédia dáta a content obsahuje hľadaný keyword

Testy zbehli úspešne:

```
/usr/bin/python3.10 /home/adam/Desktop/FIIT/9_semester/VINF/lucene_classes/test_class.py
..Empty string is not valid input
......
----------------------------------------------------------------------
Ran 8 tests in 0.541s

OK

Process finished with exit code 0
```

Konzultácie

Konzultácie 1.

Dohodnuté zadanie projektu

Konzultácia 2.

16.10.2023 - Doplní pseudokód. Crawluje dáta (2 ukrajinské a 2 ruské stránky). Zatiaľ používa čiastočne regex.

Konzultácia 3.

6.11.2023 - Spravil indexer, doplní cieľ. Do budúcej konzultácie spojí dáta s wikipédiou a urobí prípravu na prezentáciu.

Konzultácia 4. + Prezentácia

20.11.2023 - Má indexáciu, parser, search. Prezentácia OK. Do budúcej konzultácie dokončí spojenie s wikipédiou a paralelné spracovanie dát, kde už má kód. Potrebuje to iba spustiť na celých dátach.

Konzultácia číslo 5.

7.12.2023 - Prezentácia OK.