Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Ilkovičova 2, 842 16 Bratislava 4

Big Picture k inžinierskemu dielu

Tímový projekt 2018/2019

Názov tímu: MonAnt

Členovia tému: Matúš Barabás, Milan Cák, Dung Lam Tuan, Peter Lipták, David Tran Duc, Veronika Žatková Patrik židuliak

Pedagogický vedúci: Ing. Ivan Srba, PhD.

Ak. rok: 2018/2019, zimný semester

Dátum poslednej zmeny: 6.12.2018

1. Úvod

Tento dokument dokumentuje inžinierske dielo tímu 14 vytvárané v rámci predmetu Tímový projekt, v ktorom sa venujeme monitorovaniu antisociálneho správania na webe.

V kapitole číslo 2 sú uvedené globálne ciele pre zimný semester. Kapitola číslo 3 ponúka prehľad vytváraného systému. V kapitole číslo 4 sú uvedené moduly systému spolu s ich opisom. Kapitola 5 obsahuje analázu systému a oblasti (spravodajských, zbierajúcich portálov, porovnanie crawlerov a pod.). V kapitole číslo 6 je uvedený návrh systému. Kapitola číslo 7 uzatvára dokument. Je v nej uvedená implementácia systému spolu s testovaním.

2. Globálne ciele pre zimný semester

Jedným z prvotných cieľov projektu bolo oboznámenie sa s doménou falošných správ a analyzovanie možností získavania dát a následnej realizácie samotného projektu. Po ukončení zimného semestra je v rámci projektu navrhnutá schéma implementovaného systému, identifikované jednotlivé časti a moduly a implementovaná minimálne ich kostra, ktorá umožňuje následne ďalšie škálovanie počas nasledujúceho semestra. Pod základnými časťami rozumieme práve nástroje na získavanie dát, vytvorenie centrálneho úložiska pre poskytovanie zozbieraných dát a prácu s nimi a vytvorenie administrácie.

Výsledkom práce je prvé MVP (minimum viable product), na základe ktorého preukážeme realizovateľnosť projektu v navrhnutej podobe. Vďaka tomuto výsledku rovnako získame cennú spätnú väzbu v rámci projektov Rebelion a Misdeed, ktorých je projekt súčasťou, prípadne ďalších spolupracujúcich partnerov. Získané dáta sa rovnako stanú podkladom pre vznik budúceho monitorovacieho nástroja.

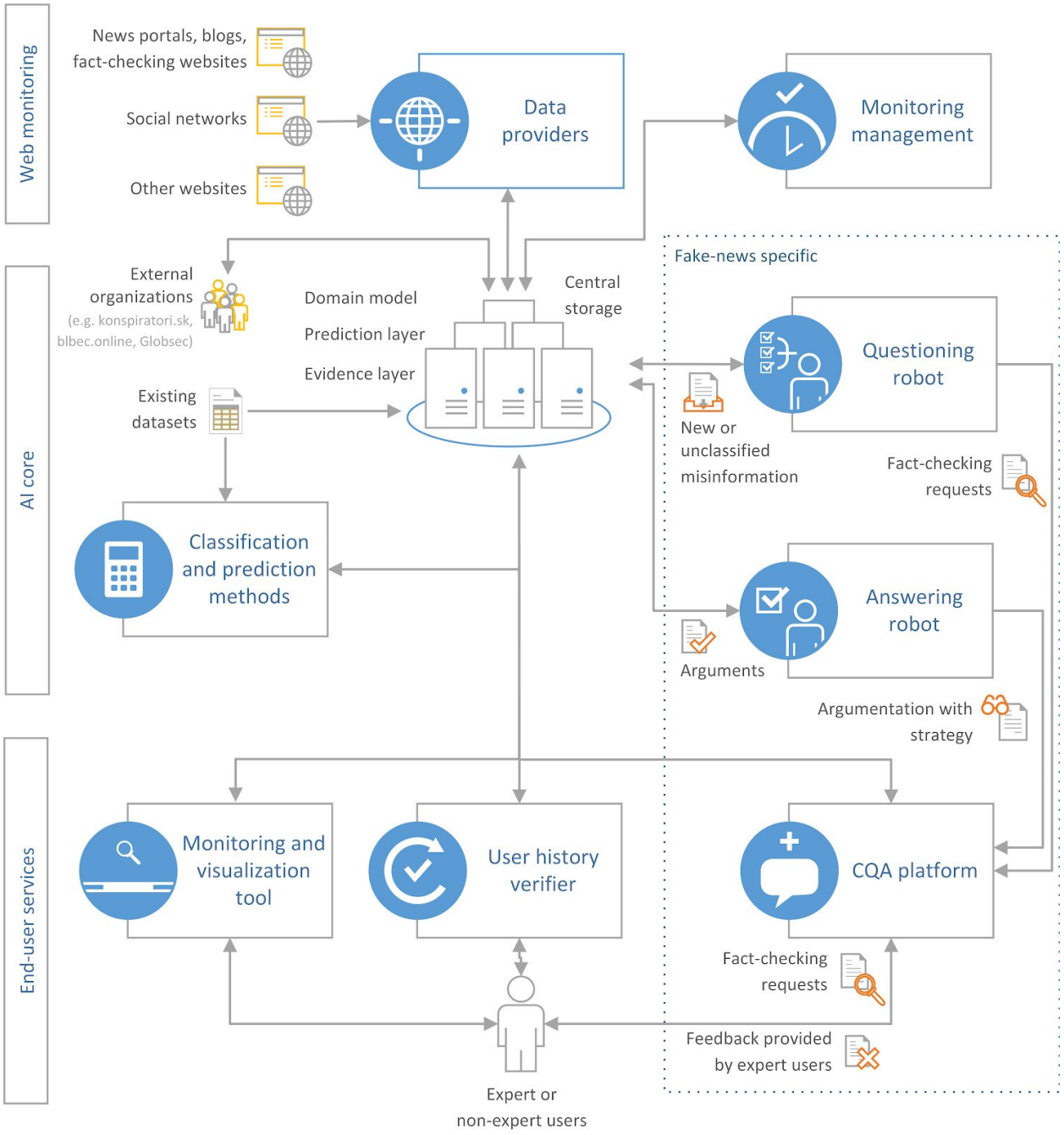
2.1 Ciele pre letný semester

Našim cieľom v letnom semestri je škálovanie existujúceho MVP. V rámci získavania dát sa chceme pozrieť na Google News API a taktiež by sme radi získali dáta nachádzajúce sa v online diskusiách na jednotlivých portáloch. Do procesov parsovania a crawlovania by sme taktiež chceli zakomponovať dostupnú knižnicu *newspaper.* V rámci celkovej štruktúry by sme chceli dať používateľovi možnosť vytvárania monitorovacích kampaní, čo zahŕňa aj spravovanie prístupových účtov. Zameriame sa taktiež na doplnenie autorizácie a autentifikácie API. Radi by sme sprístupnili taktiež viacero možností nastavení spúšťania monitorov a data providerov. Cieľom letného semestra bude taktiež zbieranie a odosielanie štatistík extrakcie. V prípade úspešného zakomponovania knižnice *newspaper* do projektu by sme radi do tejto knižnice pridali možnosť podpory pre Slovenčinu. Taktiež chceme používateľom ponúknuť podporu používania externých crawlerov a parserov, ktorými by dokázali rozšíriť centrálne úložisko o nové dáta. Chceme taktiež pridávať postupne viacero zdrojov, z ktorých môžeme čerpať nové dáta. Možnou alternatívou je napríklad Youtube alebo ruská sociálna sieť VK.

3. Prehľad vytváraného systému

Inžinierske dielo, ktoré je predmetom práce na projekte, je súčasťou projektu Rebelion, ktorý sa zameriava na antisociálne správanie na internete. V rámci neho je okrem monitorovania antisociálneho správania cieľom vytvorenie platformy, ktorá priamo podporí a umožní ďalší výskum zameraný na identifikáciu antisociálneho správania a falošných správ. Monitorovací nástroj, ktorý sleduje portály šíriace antisociálny a falošný obsah, umožní sledovať podozrivé portály, zdieľanie ich obsahu a zároveň používateľom slúži ako spätná väzba ohľadom toho, s akými informáciami sa každodenne stretávajú. Systém slúži taktiež primárne na zbieranie dát v rámci zvolených sledovaných portálov. Podpora výskumu v oblasti antisociálneho správania a falošných správ spočíva najmä vo vytvorení nástroja na zbieranie a uchovávanie dát, ktoré sú základom pre ďalšiu výskumnú činnosť v tejto oblasti. Na projekt Rebelion priamo nadväzuje projekt Misdeed, ktorý je zameraný na sledovanie a rozpoznávanie falošných správ v oblasti medicíny.

Vytvárané inžinierske dielo pokrýva funkcionalitu nástrojov pre zbieranie dát a monitorovanie zvolených portálov a zdrojov správ, vytvorenie a udržiavanie databázy (centrálneho úložiska), ktoré uchováva a umožňuje prístup k dátam a tiež nástroje pre manažovanie a monitorovanie zvoleného (antisociálneho) obsahu na internete.



Obrázok 1: Prehľad vytváraného systému

Z pohľadu návrhu samotného systému, ktorý pokrýva funkcionalitu opísanú vyššie a je znázornený na obrázku č. 1, sa systém nami implementovaný v rámci projektu dá rozdeliť na časti:

- Data provider: obsahuje logiku na crawlovanie stránok a na parsovanie potrebných informácií z týchto stránok.
- Manažment crawlerov: Jedná sa o Django aplikáciu, ktorá spravuje spúšťania crawlerov
- Centrálne úložisko: Skladá sa z databázy a REST API pre prácu s dátami v databáze.
- Frontend pre vizualizáciu: Aplikácia vizualizuje štatistiky o nacrawlovaných webových stránkach.

4. Moduly systému

Nasledujúca kapitola obsahuje priblíženie jednotlivých častí systému a ich funkcionality.

4.1 Data provider

Modul poskytuje primárne nástroje pre získavanie dát (falošných článkov a príspevkov) z rôznych typov internetových zdrojov. Komunikuje s centrálnym úložiskom, kam zozbierané dáta následne zasiela pre uchovanie.

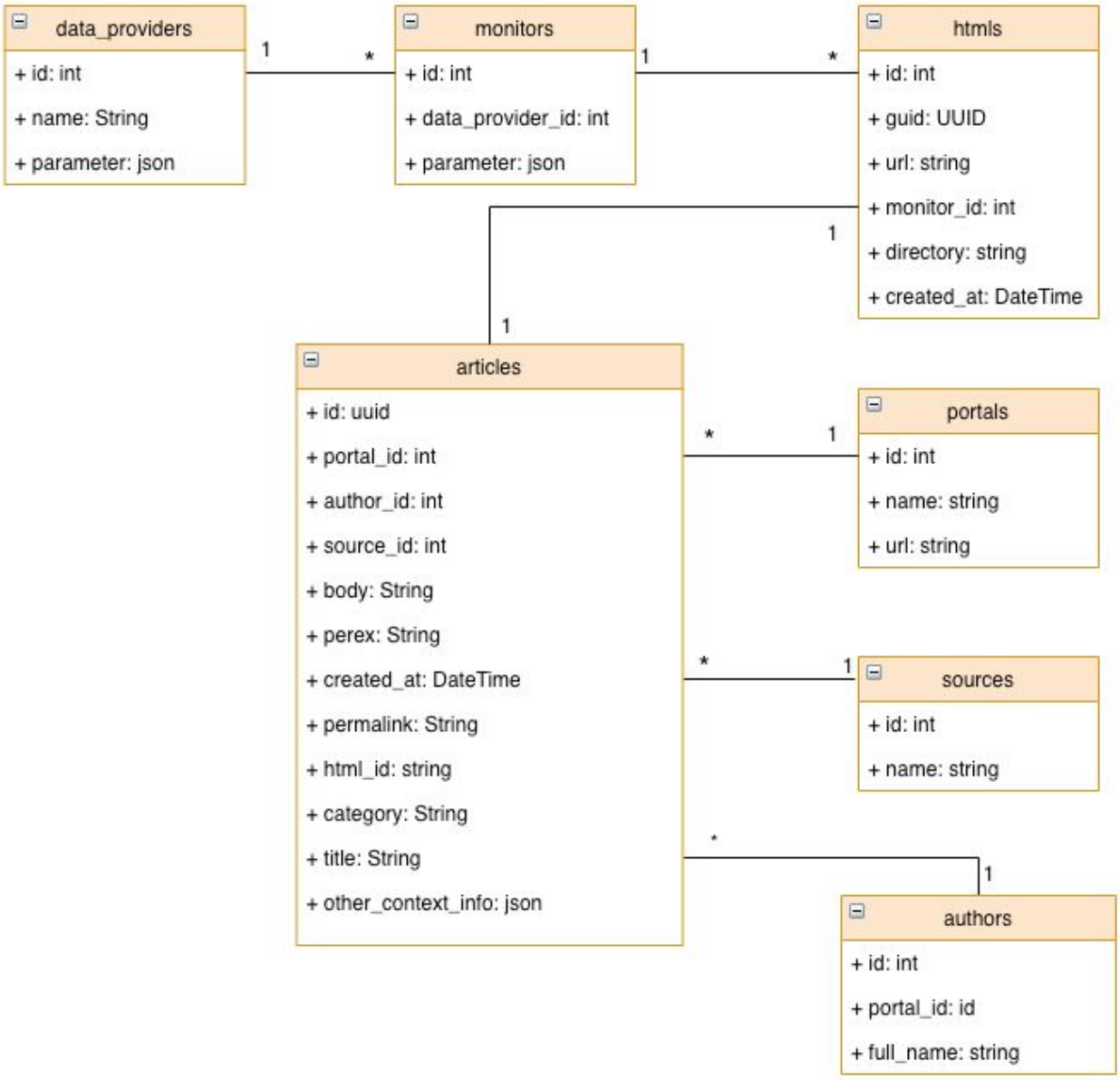
- Crawler: Tento modul zabezpečí implemtácie crawlovania pre všetky webové stránky, ktoré monitorujeme v rámci projektu. Implementácie sú napísané v programovacom jazyku Python 3.7 s použitím knižnice Scrapy vo verzii 1.5.1.Nacrawlované webové stránky vo forme HTML sú potom posielané na centrálne úložisko, kde sa ukladajú na disk.

- Parser: Parsery zabezpečujú extrakciu jednotlivých častí dát (napr. textu článkov, komentárov) z dokumentov získaných crawlovaním.

4.2 Centrálne uložisko

Centrálne úložisko je tvorené databázou, ktorá uchováva všetky získané dáta (či už prostredníctvom crawlovania alebo ako pomocné dáta). Ukladanie a pristupovanie k dátam sa vykonáva na základe vytvoreného API.

Aktuálny dátový model



4.3 Administrácia

S touto časťou systému interaguje priamo používateľ. Bude v nej možné nastaviť crawlovanie, parsovanie, monitorovanie článkov a mnoho ďalšieho. Na jej vytvorenie sme použili template AdminLTE, ktorý sme integrovali do frameworku Django.

5. Analýza systému a skúmanej oblasti

V tejto kapitole je uvedená analýza oblasti antisociálneho správania a domény týkajúcej sa vytváraného systému. Obsahuje informácie o tom, aké možné zdroje pre dáta existujú a akým spôsobom ich možno získať, vrátane ich získavania zo sociálnych sietí ako Facebook, alebo Twitter. Jej obsahom je taktiež analýza spravodajských a zbierajúcich portálov a napokon porovnanie ručne implementovaného crawlera, crawlera využívajúceho python knižnicu scrappy a crawlera apache nutch z rôznych uhlov pohľadu. Analýza predchádzala následnému návrhu implementácie systému.

5.1 Analýza Facebook API

Graph API

- primárne api na získavanie dát z facebooku
- postavený na HTTP protokole
- zložený z
    - nodes - používateľ, fotka, stránka, komentár
    - edges - spojenie medzi kolekciou objektov a jedným objektom (komentáre na fotke, fotky na stránke)
    - fields - dáta o objekte (narodeniny používateľa, meno stránky)
- na väčšinu dotazov sa vyžaduje token a permission, prípadne feature
- formát dát - JSON

Čo potrebujeme:

- “Page Public Content Access” (200 dopytov za hodinu * počet používateľov)
    - schvaľuje Facebook

Čo vieme získať:

- stránky
    - kategóriu
    - počet fanúšikov
    - link
    - stránky, ktoré na danú stránku dali like
- príspevky
    - obsah
    - čas vytvorenia
    - priamy link
    - počet zdieľaní
    - počet reakcií
- komentáre
    - obsah

- čas pridelenia
- priamy link

5.2 Analýza Twitter API

Twitter ponúka viaceré úrovne API. Konkrétne tieto:

- Standard APIs - zdarma
- Premium APIs - platené mesačne, ale Twitter ponúka free sandbox
- Enterprise APIs - platené členstvo

Streaming API

Toto API slúži na získavanie nových tweetov v reálnom čase. Pre nás je to v tejto fáze projektu nezaujímavé nakoľko budeme pracovať s tweetmi z minulosti.

Search API

V našom projekte budeme používať práve toto API. História tweetov je dostupná od marca 2006. Sú v nej dostupné všetky verejne dostupné tweety.

Standard API - nám nepostačujúce

- posledných 7 dní
- vyhľadáva iba vo vzorke
- iba relevantné tweets na základe parametrov
- nie je určené na zdroj všetkých tweetov
- podobné výsledky ako search na webe alebo v aplikácii

Premium API - vyhovujúce

- posledných 30 dní / úplná história tweetov
- všetky tweety za dané obdobie
- stránkovanie po 100

Historical PowerTrack

- dostupné len v Enterprise API
- slúži na získavanie veľkých dávok tweetov
- 8 mil tweetov za 6 hodín

Engagement API

Toto API slúži na informácie o počte zobrazení a počte interakcií s tweetmi na autorizovaných/vlastnených účtoch. Rovnako poskytuje metriky “počet obľúbení”, “počet

retweetov" a "počet odpovedí" pre akýkoľvek verejný tweet. Tieto metriky je možné získať za posledných 28 hodín, posledné 4 týždne a celé obdobie. Toto API je však dostupné výlučne na Enterprise úrovni.

Autentifikácia

Používa sa klasická Auth 2.0 autentifikácie. Na prístup do API je potrebné získať prístupové údaje a na ich základe token.

Technické parametre API

Všetky API volania opísané v tomto dokumente sú realizované cez REST. Volania sa vykonávajú cez HTTP protokol s certifikátom SSL.

Technické limity API

Limit Premium API - sandbox je 10 volaní/s, 30 volaní/min a 250 volaní/mesiac (pri platenej verzii je to 500-10,000 / mesiac). Maximálny počet znakov na definovanie požiadavky je 128 a maximálny počet vrátených tweetov je 100.

5.3 Analýza spravodajských portálov

Z dôvodu formátovania veľkej tabuľky je analýza spravodajských portálov v samostatnom dokumente, tabuľka spravodajských portálov.

5.4 Analýza zbierajúcich portálov

Týka sa portálov, ktoré sa venujú podobným projektom, ako je cieľ nášho projektu a istým spôsobom monitorujú antisociálne správanie na internete.

Konšpirátori

- https://www.konspiratori.sk/zoznam-stranok.php
- dá sa stiahnuť csv, ktoré obsahuje 100 stránok s najviac sporným obsahom spolu s ohodnotením, ktoré vykonala komisia

Lovci šarlatánov

- rss na články - http://www.lovcisarlatanov.sk/feed/
    - nadpis článku
    - link na článok
    - dátum publikovania
    - kategória článku

        - text článku
        - autor článku
        - ďalšie stránky pridaním parametra “?paged=2” do url
- rss na komentáre - http://www.lovcisarlatanov.sk/comments/feed/
    - titul článku, ku ktorému bol komentár napísaný
    - link na komentár
    - dátum pridania komentáru
    - obsah komentáru

Blbec.online

- parsovať môžeme všetko
- všetky dáta vieme dostať cez Facebook API

5.5 Reddit

API

Je voľne prístupná, bez poplatkov. Dokumentácia:https://www.reddit.com/dev/api/

Získavanie článkov zo subredditu:

GET [/r/subreddit]/top

Príklad:

https://www.reddit.com/r/news/top.json?count=20

Získavanie komentárov z článku:

GET [/r/subreddit]/comments/article

Príklad:

https://www.reddit.com/r/news/comments/9lyfnq.json

Výstup je JSON

Je taktiež možné sa pripojit na WS, pre live feed:

https://www.reddit.com/dev/api/#section_live

Reddit API Access:

https://www.reddit.com/wiki/api

RSS

Formát:

[/r/subreddit]/.rss

Príklad:

https://www.reddit.com/r/news.rss

Výstup je XML

API je omnoho lepšia, poskytuje všetko, čo by sme mohli chcieť

5.6 Porovnanie crawlerov

Pre účely vytvorenia crawlerov, ktoré zbierajú dokumenty z rôznych zdrojov, sme implementovali a porovnali 3 varianty: vlastnoručne implementovaný crawler, crawler implementovaný s využitím knižnice Scrapy a Apache Nutch Crawler.

Setup

Vlastný crawler: rozbehanie programovacieho prostredia (Python + knižnice) a naprogramovanie celej logiky crawlovania aj scrapovania (<= 1 MD)

Scrapy: Rozbehanie programovacieho prostredia (Python + knižnice), vytvorenie scrapy projektu (https://docs.scrapy.org/en/latest/intro/tutorial.html) a zadefinovanie správania pavúka a logiky scrapovania (<= 0.5MD)

Apache Nutch: Inštalácia prostredia (Java 8 + JDK), inštalácia Apache Ant a inštalácia Apache Nutch (1-2 MD), definovanie parametrov crawlovania a spustenie crawlera

Vytvorenie samotného crawlera

Vlastný crawler: naprogramovať

Scrapy: Napísanie regexov a nastavenie parametrov pavúka, implementovanie logiky pipeline

Apache Nutch: Napísanie regexov pre crawlovanie, nastavenie parametrov

Škálovateľnosť

Vlastný crawler: nutnosť všetko vlastnoručne implementovať, “pain”

Scrapy: jednoduché pridanie ďalších pavúkov, optimalizované

Apache Nutch: pre crawlovanie viacerých portálov jednoducho pridať viac regexov a seedov, ak príliš veľký overhead, tak nový nutch process

Optimalizácia

Vlastný crawler: čo nenaprogramujeme, nemáme, pomalé

Scrapy: optimalizovaný, nepodporuje viacvláknovosť na 1 pavúka (ale viacero pavúkov == viacero procesov) - pre mňa bol ale najrýchlejší (najviac docs per second)

Apache Nutch: podpora viacvláknovosti, optimalizovaný

Javascript a dynamicky generovaný obsah

Vlastný crawler: napr. použitie Selenium

Scrapy: nastavenie použitia ScrapyJS modulu

Apache Nutch: Nutch Selenium plugin (nepokúšala som sa ani rozbehať)

Parametrizovateľnosť

Vlastný crawler: čo nenaprogramujeme, nemáme

Scrapy: veľmi široké nastavenia, nenašla som nič obmedzujúce, čo by som nevedela prekonať

Apache Nutch: nastavenie pre robots.txt mi nefungovalo, inak parametrizovateľné, iba komplikované nastavenia (dlhšie na pochopenie a nastavenie, než Scrapy) + JAVA REGEX!!!

Robots.txt

Vlastný crawler: čo je to? :)

Scrapy: zmena jednej hodnoty v nastaveniach

Apache Nutch: buď prepísať Java kód, alebo nastaviť whitelists - https://wiki.apache.org/nutch/WhiteListRobots , nepodarilo sa mi to, crawlovala som preto iný portál

Prístup k raw dokumentom (html)

Vlastný crawler: requests by default vráti celý html dokument, ktorý si môžem vložiť do DB (nutnosť implementovať rozhranie)

Scrapy: Pridanie vkladania dokumentu do pipeline (možnosť spojiť so scrapovaním entít)

Apache Nutch: Nutnosť medzikroku - urobiť dump z Nutch interného dump formátu do priečinka s html súbormi

Scrapovanie entít

Vlastný crawler: čo nenaprogramujeme, nemáme

Scrapy: Výber itemov v pipeline, napr. pomocou BeautifulSoup, pomerne jednoduché, prípadne napísanie scrapera

Apache Nutch: Nutch vracia iba html dokumenty, s viac nevie pracovať. Nutnosť implementovať scraper, ktorý načíta z priečinku html súbory a scrapuje ich a ukladá do DB, resp. existuje Apache Tika

Usability a reusability

Vlastný crawler: :(môžem duplikovať kód

Scrapy: Ak mám pavúka, viem ho použiť stále (filtruje duplicitné dokumenty), viem podľa neho vytvoriť ďalších pavúkov, môžu mať jednotnú pipeline - alebo pre každý iný typ dát definujem vlastnú pipeline (https://stackoverflow.com/questions/8372703/how-can-i-use-different-pipelines-for-different-spiders-in-a-single-scrapy-proje), viem, čo sa kedy deje

Apache Nutch: stačí zmeniť nastavenia a prispôsobiť crawler tomu, čo práve chceme crawlovať, avšak menšia transparentnosť (príšerné logy)

Jazyk

Vlastný crawler: aký chcem

Scrapy: Python

Apache Nutch: implementovaný v Java

Na základe daného porovnania je na majoritu dokumentov ideálne použiť práve Scrapy. Avšak v prípade, že crawlujeme iterované urls (http://domain.com/?id=1, 2, 3, ...), je veľmi triviálne použiť aj vlastný crawler.

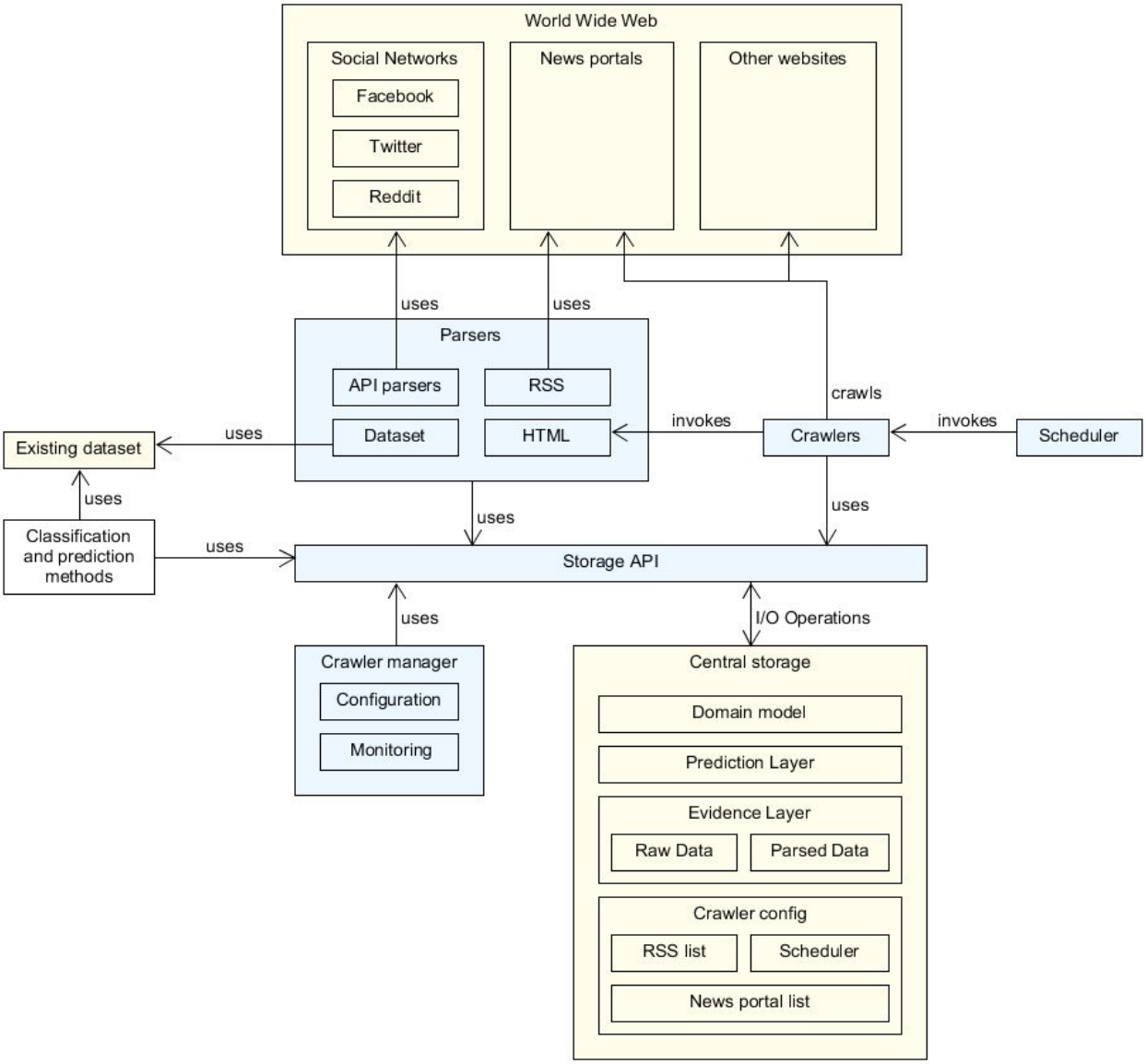
Rovnako, Scrapy môže byť rozšírený o pár nutch crawlerov.

6. Návrh systému

V tejto kapitole je uvedený návrh časti systému, ktorý bude vytváraný počas dvoch semestrov na predmete Tímový projekt. V procese návrhu bol vytvorený diagram architektúry vytváraného systému, ktorý obsahuje všetky potrebné komponenty. Skladá sa z týchto komponentov:

- **Scheduler -** komponent, ktorý umožňuje naplánovať a spúšťať crawlovanie stránok v určitých časových intervaloch.
- **Crawler -** komponent, ktorý prehľadáva web a získava html stránok, ktoré posúva na spracovanie parserom.
- **Parsers** - komponent, ktorý slúži na získavanie dát z rôznych zdrojov. Dokáže získavať dáta z existujúcich API, ktorými disponujú najmä sociálne siete. Dáta novinových portálov vie získať prostredníctvom parsovania stránok. Niektoré stránky ponúkajú RSS feed, ktorý je ďalšou možnosťou získavania dát. Parser taktiež umožňuje využívať dáta z už existujúcich datasetov.
- **Crawler manager -** komponent, ktorý slúži na monitorovanie a konfiguráciu systému.
- **Storage API -** API, ktoré poskytuje možnosť pracovať s centrálnym úložiskom. Ponúka možnosť kontrolovať všetky dopyty, ktoré sa vykonávajú nad dátovým úložiskom.
- **Dátové úložisko -** komponent, ktorý slúži na ukladanie dát. Je rozdelený do viacerých vrstiev. Každá z týchto vrstiev obsahuje dáta na rôznej úrovni spracovania. Okrem získavaných dát z procesu parsovania obsahuje aj konfiguráciu schedulera. Dátové úložisko je kľúčovým bodom architektúry navrhovaného systému. Prostredníctvom vystaveného API k nemu pristupujú všetky vyššie spomínané komponenty systému.
- **Classification and prediction methods -** komponent, ktorý umožňuje klasifikáciu a predikciu antisociálneho správania buď na základe dát z dátového úložiska ale dát získaných z existujúcich datasetov. Tento komponent je síce uvedený v diagrame architektúry ale nebude s veľkou pravdepodobnosťou súčasťou implementácie prvotnej fázy systému.

Ako je možné vidieť na obrázku č. 2, jednotlivé komponenty komunikujú s centrálnym úložiskom - crawlery a parsery doň ukladajú dáta, ktoré sú následne prístupné prostredníctvom API používateľom alebo monitorovaciemu nástroju.



Obrázok 2: Diagram architektúry vytváraného systému

7. Implementácia systému a testovanie

7.1 Implementácia

V prvom kroku sme implementovali jednoduchý prototyp na zahodenie, kde sme crawlovali a parsovali náhodný článok z portálu hlavnespravy.sk. Pomocou jednoduchých vytvorených REST endpointov a jednoduchého frontendu sme vyskúšali celý cyklus fungovania medzi modulmi. Celý cyklus sa skladá z nasledujúcich krokov:

1. Používateľ zadá link, ktorý kde crawlovať.
2. Crawler spustí proces pre crawlovanie daného linku.
3. Pomocou vystaveného endpointu posielame nacrawlovaný HTML na centrálne uložisko.
4. Parser spustí proces pre parsovanie HTML súboru, ktorý dostane zo centrálneho úložiska.
5. Parsované elementy sa uložia do databázy pomocou REST endpointov.

Detailná dokumentácia k centrálnemu úložisku, resp. k API (vrátane zoznamu endpointov), je generovaná automaticky pomocou nástroja Swagger a dostupná je na adrese (po spustení servera): SERVER_ADDRESS/apidocs/.

V ďalších krokoch sa vytvorený prototyp neustále škáluje a rozširuje práve tak, aby pokrýval aspekty zbierania dát na rôznych portáloch a ich následného poskytovania a spracovania.

8. Testovanie

Testovanie bolo v tejto fáze vykonávané len na úrovni unit testov. Testy, ktoré boli vytvorené pokrývajú najmä funkcionalitu dátového úložiska - najdôležitejšieho komponentu architektúry systému. Postupom času a reálnym nasadením systému sa predpokladá vykonávanie testov aj na integračnej a akceptačnej úrovni v kombinácii s nástrojom pre kontinuálnu integráciu.

Samotné testy sú implementované prostredníctvom knižnice Pytest a Pytest-Flask, ktorá je určená pre programovací jazyk Python a vyznačuje sa priamou podporou pre Flask API.