UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Anže Zabukovšek

Predstavitev trendov na slovenskem Twitterju

Diplomsko delo

Ljubljana, 2017

UNIVERZA V LJUBLJANI

FAKULTETA ZA DRUŽBENE VEDE

Anže Zabukovšek

Mentor: izr. prof. dr. Aleš Žiberna

Predstavitev trendov na slovenskem Twitterju

Diplomsko delo

Ljubljana, 2017

ZAHVALA

Zahvaljujem se mentorju, izr. prof. dr. Alešu Žibernu, za pomoč, nasvete in spodbudo pri izdelavi diplomskega dela.

Predstavitev trendov na slovenskem Twitterju

Twitter je v Sloveniji po številu uporabnikov drugo največje spletno socialno omrežje. Kljub velikemu številu uporabnikov, Twitter slovenskim uporabnikom ne ponuja prikaza trendov kot jih ponuja za nekatera druga območja. Skozi diplomsko nalogo bi rad predstavil spletno aplikacijo za prikazovanje trendov, ki na Twitterju spremlja eno ključnih skupin v družbi: medije. Predstavitev trendov je dinamična, saj so rezultati odvisni od uporabnikove poizvedbe. Ta Python aplikacija je zgrajena iz dveh glavnih komponent. Prvi del s pomočjo Twitter Streaming API in knjižnice Tweepy zbira tvite novinarjev in medijev od 25. Aprila 2017 dalje. Drugi del, ki temelji na Python knjižnici Django, te podatke shranjuje v SQLite bazo in jih glede na uporabnikovo poizvedbo vizualizira. Glavne informacije, ki jih aplikacija posreduje uporabniku, so: koliko (je tvitov), kaj (je njihova vsebina), kdaj (so bili objavljeni) in kdo (jih največ objavlja). Interpretacija je na koncu prepuščena uporabnikom. Ta aplikacija je na voljo vsem, ki jih zanima, o čem tvitajo slovenski mediji in novinarji.

Ključne besede: Twitter, Python, trendi, vizualizacijo podatkov, spletna aplikacija.

A presentation of trends in the Slovenian Twitter space

Based on the number of users, Twitter is the second biggest online social networking service in Slovenia. Despite the high number of users Twitter does not offer users information about trending topics amongst Slovenian users as it does so for other geographical areas. The purpose of this paper is to present a web application that uses one of society's most crucial groups to showcase trends on Twitter: media. The presentation of trends is dynamic since it is based on user query. This Python application has two main components. The first one is built using Twitter Streaming API and Tweepy library to collect Slovenian media and journalist's tweets since 25th April 2017. The second one uses Python library Django to save the tweets in an SQLite database and to visualize the data based on user query. The application offers the user information about how many (tweets), what (are the tweets' contents), when (were they published) and who (published them). The interpretations of the visualizations are left to the user. This application is available to all who are interested in topics the media and journalists tweet about.

Keywords: Twitter, Python, trends, data visualization, web application.

KAZALO VSEBINE

KAZALO SLIK

1 UVOD

V času, ko je uporaba spletnih socialnih omrežij vedno bolj intenzivna in je na voljo vedno več informacij, se soočamo z novim problemom. Kako na najlažji način zbrati podatke in jedrnato ter smiselno predstaviti rezultate.

Ne gre se čuditi, da je v zadnjih letih naraslo zanimanje na področju raziskovanja spletnih socialnih omrežij. Te vrste raziskav so postale velik in pomemben del v smislu zbiranja podatkov. Mobilni podatki in vsebina, ki jo ustvarijo milijoni uporabnikov na spletnih družbenih omrežjih, kot sta Facebook in Twitter, predstavljajo neprekinjene tokove informacij o družbenih dejavnostih (Piña-García in drugi 2016, 1–2).

Twitter je najbolj priljubljena spletna stran za objavljanje t. i. mikroblogov. Uporabniki objavljajo sporočila, ki vsebujejo do 140 znakov. Ta sporočila se imenujejo tviti. V letu 2015 je bilo približno 200 milijonov rednih uporabnikov Twitterja, ki so skupaj dnevno objavili 500 milijonov tvitov (Piña-García in drugi 2016, 2). Po raziskavi Valicona (2016) je imelo v lanskem letu Twitter profil 206.500 različnih slovenskih uporabnikov, kar je na drugem mestu, za Facebookom. Od tega jih je bilo 33.000 aktivnih vsak dan.

Twitter omogoča zbiranje podatkov s pomočjo API-jev. Vendar so ti omejeni s številom računov, ki jim lahko sledimo (Twitter, Inc. 2017). Twitter omogoča sledenje uporabnikom glede na geolokacijo ali glede na unikatni identifikator (Twitter ID). Pomanjkljivost sledenja s pomočjo geoloakcije je ta, da je nimajo vključene vsi uporabniki. Rad bi predstavil, da je sledenje preko Twitter ID dovolj učinkovito v primeru, da se sledi manjši skupini uporabnikov ter da si pri definiranju te ciljne skupine lahko pomagamo s Twitter seznami, ki so jih ustvarili drugi uporabniki.

Moja aplikacija je primerna za spremljanje katerekoli skupine Twitter uporabnikov. Zaradi vsebinske koherentnosti sem se odločil za sledenje novinarjem in medijem. Tradicionalne novinarske hiše imajo pomembno vlogo pri definiranju trendov na Twitterju. To ne pomeni, da so nujno prvi, ki neke ideje, teme objavijo. Ampak prek spremljanja dogajanja na Twitterju, odkrivajo potencialne teme in jih nato delijo med svoje sledilce. Zaradi svojega edinstvenega statusa in navadno velikega števila sledilcev tako občutno pripomorejo k temu, da določene teme dosežejo veliko število Twitter uporabnikov (Chakraborty in drugi 2015).

Kljub zmanjšanju števila spremljanih Twitter uporabnikov je še vedno problem, kako predstaviti bistvene teme, ki se pojavljajo v slovenskem Twitter prostoru. Zato je namen aplikacije, možnost izbire tem prenesti na samega uporabnika. Ta svoboda omogoča, da uporabnik ni omejen na pregled

trendov, ki jih izbere zbiratelj tvitov. Vse analize so nato prikazane s pomočjo modernih orodij za spletno vizualizacijo, ki so uporabniku odprte za interpretacijo.

Cilja te aplikacije sta torej dva. Prikazati, da je mogoče s pomočjo sodobnih orodij za zbiranje, čiščenje, analizo in vizualizacijo podatkov predstaviti vsebino Twitterja na jedrnat in razumljiv način. Poleg tega upam, da bo aplikacija služila kot vpogled vsem tistim, ki jih zanima katere so najbolj vroče teme, o katerih tvitajo slovenski novinarji in medijske hiše.

2 SPLETNA SOCIALNA OMREŽJA

Spletna socialna omrežja so različne storitve, ki jih vsak dan aktivno uporablja več milijonov uporabnikov. Ker ima vsako omrežje zaradi različnih interesov specifične značilnosti, ki povezujejo uporabnike in storitve, ki jih ponujajo posamezne platforme, je spletna socialna omrežja težko opredeliti. Boydova in Ellisonova (2007, 211) sta opredelili spletna socialna omrežja kot spletne storitve, ki uporabnikom omogočajo, da v okviru te storitve ustvarijo javen ali napol javen profil, opredelijo seznam drugih uporabnikov, s katerimi so povezani, in dovolijo vpogled v seznam povezanih uporabnikov na svojem profilu ali profilu nekoga drugega.

Čeprav je mreženje uporabnikov ena izmed storitev, ki jih spletna socialna omrežja omogočajo, to vseeno ni glavni namen. Kar jih loči od ostalih, sorodnih storitev je možnost, da uporabniki ustvarijo in objavijo svoja družbena omrežja. V ta (napol) javna družbena omrežja so lahko vključeni samo tisti, ki so prav tako člani te spletne storitve. Javno deljenje omrežij omogoča, da se povežejo ljudje, ki se sicer ne bi. Kljub temu se večja spletna socialna omrežja v prvi vrsti uporabljajo za komunikacijo med osebami, ki se že poznajo (Boyd in Ellison 2007, 211).

Večina spletnih socialnih omrežij ponuja tudi možnost komunikacije z drugimi uporabniki te storitve. To je lahko preko zasebne komunikacije, ki spominja na elektronsko pošto. Drugi način je (javno) komentiranje in objavljanje sporočil na svojem profilu ali profilu nekega drugega uporabnika. Tu se vzporednice med spletnimi socialnimi omrežji zaključijo, saj vsako omrežje ponuja specifične storitve, ki privlačijo specifične (skupine) ljudi. Omrežja se nato tudi spreminjajo in razvijajo, predvsem glede na to, kdo in kako jih uporabljajo ter koga želijo pritegniti k uporabi (Boyd in Ellison 2007, 213–214).

Vsaka izmed omenjenih značilnosti spletnih socialnih omrežij ima relativno dolgo zgodovino. Vendar je šele spletna stran SixDegrees.com povezala vse v eno spletno storitev in leta 1997 postala prvo spletno socialno omrežje. Kljub temu da je imela milijon uporabnikov, so morali storitev leta 2000 zapreti. Z letom 2001 se je začel nov val spletnih socialnih omrežij. Mnoge storitve so od

takrat propadle, utonile v pozabo ali se usmerile k ožjim skupinam uporabnikov. Na drugi strani so nekatera omrežja do danes zrasla in jih dnevno uporablja več milijonov uporabnikov (Boyd in Ellison 2007, 214–215).

2.1 TWITTER

Twitter je eno izmed najpopularnejših spletnih socialnih omrežij. Ustanovili so go Noah Glass, Biz Stone, Evan Williams in, v prvi vrsti, Jack Dorsey. Danes ima 313 milijonov aktivnih mesečnih uporabnikov in je na voljo v več kot 40 jezikih (Twitter, Inc.).

Uporabniki morajo na svoje Twitter profile vpisati svoje ime – to je lahko daljše, s presledki in ni potrebno, da je resnično. Izbrati morajo svoje kratko ime oziroma »handle«, ki se začne z znakom »@«, in mora biti unikatno (dva uporabnika ne moreta uporabljati istega kratkega imena). To ime se uporablja pri interakciji z drugimi. Po želji si lahko naložijo tudi profilno fotografijo in fotografijo ozadja svojega profila. Te štiri stvari so vidne vsem uporabnikom in neregistriranim obiskovalcem profila.

Ob registraciji je potrebno podati še elektronski naslov, ki ni viden ostalim. Poleg tega lahko uporabniki svoj profil izpolnijo še s kratkim opisom, krajem, spletnim naslovom, datumom rojstva, Vine profilom in barvo profila.

Na Twitterju se povezave med uporabniki kažejo na več načinov. Vsak uporabnik lahko sledi komurkoli in želenega uporabnika s tem doda na svoj seznam *following*. Uporabniku se nato začnejo prikazovati vsi tviti, ki jih objavljajo uporabniki, ki jim ta sledi. Obstajata dve izjemi. Če ima uporabnik, ki mu želimo slediti »zaprt profil«, mora ta uporabnik sledenje odobriti oziroma ga lahko tudi zavrne, če ne želi, da mu sledimo. Druga izjema so ljudje, ki so blokirani. Ko nekdo nekoga blokira, je med njima onemogočena vsa komunikacija na Twitterju, tudi sledenje. To traja, dokler se blokada ne sprosti s strani uporabnika, ki jo je postavil. Istočasno, ko se nekdo odloči, da bo drugemu sledil in ga dodal med *following*, je sledilec samodejno dodan na seznam *followers* na profilu tistega, ki mu je začel slediti. *Followers* je seznam vseh uporabnikov, ki sledijo temu uporabniku.

Zadnji način povezovanja so seznami oziroma *lists*. Na seznam se dodajajo profili pod enakimi pogoji kot pri dodajanju *following*. Nato lahko različni uporabniki sledijo kar temu seznamu namesto vsakemu uporabniku s seznama posebej. Seznami so sicer manj popularni od klasičnega sledenja.

Glavna aktivnost in prevladujoči način komunikacije, ki jo ponuja Twitter, je objavljanje tvitov. Tvit je vsako sporočilo, ki je objavljeno na Twitterju. Vsebuje lahko slike, video, hiperpovezave oziroma 140 znakov besedila. V tvite lahko vključimo tudi *hashtage*. To so vse besede v tvitih, ki se začnejo z znakom # oziroma s *hashem*. Ta znak lahko postavimo pred katerokoli besedo. Sicer je prvoten namen ta, da se znak postavi pred ključnimi besedami v tvitu. Ker omogoča Twitter iskanje po *hashtagih,* lahko z dobrim *hashtagom* dosežemo več ljudi. Najbolj uporabljeni *hashtagi* lahko postanejo tudi trendi, ki jih Twitter še posebej izpostavi. Uporabnik lahko tvite tudi retvita. To pomeni, da že napisan tvit deli med svoje sledilce. Če uporabnik želi, lahko retvita na način, da izvirni tvit citira in ga dopolni še s svojim tvitom (Twitter, Inc.).

Twitter uporabniku omogoča, da nastavi vidljivost svojih tvitov. Privzete nastavitve so takšne, da so tviti vidni vsem. Kar ne pomeni, da tviti dosežejo vse ljudi. Tisti, ki uporabniku ne sledijo in ne obiščejo njegovega profila, tvitov verjetno ne bodo videli. Lahko pa uporabnik nastavi, da so tviti zaščiteni (angl. *protected*). Ti tviti so nato vidni samo tistim, ki jih uporabnik potrdi za sledilce (Twitter, Inc.).

3 ZBIRANJE PODATKOV

Naše možnosti in zmožnosti zbiranja podatkov se izboljšujejo iz desetletja v desetletje, iz leta v leto. Informacije in znanje, ki jih pridobimo z rudarjenjem podatkov, se lahko uporabljajo za različne aplikacije za vodenje podjetij, nadzor proizvodnje in tržne analize, inženirsko načrtovanje in raziskovanje znanosti.

Podatkovno rudarjenje (angl. *data mining*) je interdisciplinarno področje, ki med drugim združuje tehnologije podatkovnih baz, umetne inteligence, nevronskih mrež, statistike, prepoznavanje vzorcev, sistemov, ki temeljijo na znanju visoko zmogljivega računalništva, in vizualizacije podatkov. Za začetek podatkovnega rudarjenja, kot ga poznamo danes, smatramo konec osemdesetih oz. začetek devetdesetih let prejšnjega stoletja (Han in Kamber 2000).

Podatkovno rudarjenje je proces, v katerem se iz velikih količin podatkov izvleče oz. r*udari* znanje. Han in Kamber (2000, 6) sta sicer mnenja, da bi bil pravilnejši izraz *rudarjenje znanja* oz. *rudarjenje znanja iz podatkov*, saj je znanje tisto kar iščemo in ne podatki. *Rudarjenje* je sicer precej primeren izraz, saj iščemo relativno majhne kose (znanja) v velikem viru (podatkov). Za ta proces obstaja sicer še več različnih izrazov, ki imajo podoben pomen. Med strokovnjaki je pogosto uporabljen tudi izraz, ki je podoben terminu podatkovno rudarjenje – *odkrivanje znanja v*

podatkovnih bazah ali *KDD* (angl. *Knowledge Discovery in Databases*). Odkrivanje znanja je proces, ki ima 7 ponavljajočih se korakov:

- **čiščenje podatkov**: odstranjevanje nepotrebnih podatkov,
- **integracija podatkov**: v primeru, da je v proces združenih več različnih virov,
- **izbor podatkov**: priklic podatkov, ki so relevantni za analizo, iz podatkovne baze,
- **preoblikovanje podatkov**: preoblikovanje in poenotenje oblike podatkov, ki omogoča izvedbo analiz,
- **podatkovno rudarjenje**: v ožjem pomenu – uporaba naprednih metod za iskanje vzorcev,
- **ocenjevanje vzorcev**: identificiranje najpomembnejših vzorcev, ki so se pokazali v prejšnjem koraku in
- **predstavitev znanja**: predstavitev odkritega znanja uporabniku s pomočjo različnih tehnik vizualizacije in predstavitev znanja.

V praksi se izkaže, da se vsi koraki ne izvedejo oziroma se ne izvedejo na način in v obsegu, kot sta to predvidela Han in Kamber. V primeru moje aplikacije je precej specifična povezava med zadnjimi tremi koraki, saj je uporabnik tisti, ki s pomočjo poizvedbe išče vzorce, jih ocenjuje in jih nato "predstavi" – interpretira s pomočjo različnih vizualizacij, ki jih aplikacija ponuja. Prav tako je pomemben še en korak, ki poteka pred čiščenjem. To je sam proces zbiranja podatkov.

Vendar je zbiranje podatkov danes zelo enostavno (Witten in Hall 2011, 3–4). Na voljo imamo visoko zmogljive računalnike, cenovno ugodne pomnilne enote in hitro internetno povezavo. V procesu podatkovnega rudarjenja se podatki zbirajo samodejno – ali vsaj s pomočjo računalnikov.

Zbiranje podatkov za predstavitev trendov na slovenskem Twitterju se je začelo aprila 2017. Ker je namen, prikazati trende daljšega časovnega obdobja, je tisti del aplikacije, ki pobira oziroma *streama* tvite, nenehno aktiven. Twitter je eno od najbolj priljubljenih spletnih socialnih omrežij. Poleg tega so izdali lasten vmesnik, ki omogoča, da se preko njega povežemo na Twitter.

3.1 PYTHON

Python je razširjeni, prosto dostopni programski jezik, primeren za začetnike in napredne uporabnike. Čeprav so nekateri drugi programski jeziki morda zmogljivejši, je Python gotovo eden izmed hitrejših, ko je govora o samem razvijanju aplikacij. Pythonova sintaksa omogoča, da želeno dosežemo z manj vrsticami kode. Hkrati je Python sintaksa lažje berljiva v primerjavi s sintaksami

drugih programskih jezikov. Pomembni lastnosti, ki sta pripomogli k priljubljenosti Pythona, sta tudi odprtokodnost in brezplačnost. Python je sicer objektno orientiran programski jezik. Pythonova koda potrebuje interpreter (Python).

Zaradi svoje priljubljenosti je za Python na voljo veliko različnih knjižnic, ki omogočajo razvijanje programskih aplikacij, in skript za veliko različnih namenov. Dve najpomembnejši knjižnici, ki sem ju uporabil pri predstavitvi trendov slovenskih medijev na Twitterju sta Tweepy in Django. O Djangu bom več povedal v nadaljevanju, ko bo govora o spletnem delu aplikacije.

3.2 TWEEPY IN TWITTER API

Twitterjevi API-ji so primerni za uporabo v več kot desetih različnih programskih jezikih. Za spremljanje tvitov sta predvidena dva API-ja. Prvi je REST API, ki s pomočjo vnesenih parametrov poišče tvite, ki tem parametrom ustrezajo. Najde tudi starejše tvite, vendar z nekaterimi omejitvami. Klici te aplikacije so časovno omejeni in ne najdejo tvitov v nedogled nazaj. Zato sem se raje odločil za Streaming API. To je aplikacija, ki prav tako išče tvite glede na vhodne parametre. Sicer aplikacija ni sposobna iskanja tvitov za nazaj, ampak najde samo tiste, ki so ustvarjeni takrat, ko aplikacija teče. Ker aplikacija teče daljše obdobje, to ni predstavljalo težave, da ne bi zajel dovolj velikega časovnega obdobja. Hkrati Streaming API nima časovnih omejitev in je lahko aktivna brez prekinitev (Twitter, Inc. 2017).

Na Streaming API sem se priključil preko Pythonove knjižnice Tweepy. Tweepy knjižnica zahteva, kot podaljšek Twitterjevega API-ja, OAUth preverjanje pristnosti. To v praksi pomeni, da se mora vsaka aplikacija, ki želi spremljati tvite ali pridobiti kakšne druge podatke s pomočjo Twitterjevega API-ja, registrirati na spletni strani Twitterja. Registrirajo jo lahko samo registrirani uporabniki Twitterja. Uporabnik nato prejme 4 različne ključe, ki mu omogočajo potrditev pristnosti aplikacije in nato spremljanje tvitov oziroma drugih podatkov s Twitterja (Tweepy).

3.3 PROCES ZBIRANJA PODATKOV

Zbiranje podatkov oziroma shranjevanje tvitov se je začelo 13. marca 2017. To so bili še testni podatki v času, ko sem odstranjeval napake v aplikaciji, ki je zbirala podatke. 25. aprila 2017 se je nato začelo shranjevanje podatkov, ki so primerni za spletno aplikacijo. Pred tem je bilo potrebno še zbrati uporabnike oziroma skupine uporabnikov Twitterja. Ker ima Twitterjev API omejitve, koliko uporabnikov lahko spremlja, sem se odločil, da se omejim na slovenske medije. Obstaja precej seznamov, v katerih so zbrani slovenski novinarji in novičarski portali. Po primerjavi različnih seznamov, sem se odločil, da uporabim dva. Prvi seznam sledi petdesetim različnim slovenskim

uporabnikom Twitterja, s področja medijev in novinarstva. To se večinoma večji novičarski portali, novinarske hiše, časopisi, glavne slovenske informativne oddaje in podobno. Seznam je ustvaril uporabnik *Evropa moja dežela* in se imenuje *SLO mediji*. Ta seznam sem nato nadgradil s seznamom *slo-novinarji*, ki ga je ustvaril uporabnik *kompl33kator*. Ta seznam vsebuje 321 uporabnikov – v njem so, poleg novičarskih portalov, časopisov, informativnih oddaj in podobnega, še slovenski novinarji.

Seznama sem nato združil in odstranil podvajanja. Tako sem na koncu izluščil 335 unikatnih Twitter profilov s področja slovenskih medijev ter novinarjev. Nato sem s pomočjo spletne aplikacije *gettwitterid.com* pridobil Twitter ID-je od vseh uporabnikov v svojem seznamu. Tweepy oziroma Twitter API kot vhodni podatek ne sprejme uporabniških imen, ampak te unikatne ID-je, prek katerih nato sledi ljudem in zbira njihove tvite.

Glavni namen te aplikacije ni, da se predstavi specifična skupina slovenskih uporabnikov Twitterja. Pomembno je, da se predstavi trende. Zato sem se zadovoljil s tem združenim seznamom. Pomemben aspekt pri spremljanju trendov je njihova popularnost oz. uporaba skozi čas. Zaradi tega je bilo pomembno, da se začne tvite zbirati čim prej in posledično zajame daljše časovno obdobje. Novinarji in medijske hiše so nenazadnje pomembni pri oblikovanju javnega mnenja in tudi eni izmed pokazateljev trendov na slovenskem Twitterju. Aplikacija je sicer primerna za katerokoli skupino uporabnikov Twitterja. Zamenjati bi bilo potrebno le vhodne podatke – ID-je njihovih Twitter računov.

Podatki se zbirajo brez prestanka. Izjema so prekinitve zaradi izpada električnega/internetnega omrežja, vzdrževalnih del na strani Twitterja, drugih napak na omrežju Twitter ali nekih drugih napak. Po vsaki napaki se zbiranje podatkov za nekaj časa ustavi – odvisno od vrste napake. Sicer bi prišlo do preobremenitve, zaradi prepogostega poskušanja vzpostavljanja nove povezave. Uporabniki bodo tako lahko preverili uporabo nekega niza v tvitih za obdobje od začetka zbiranja podatkov – 13. marca 2017, do dneva, ko obiščejo spletno stran in vpišejo želeno poizvedbo. Za lažje upravljanje s podatki, se tviti zbirajo v datoteke, ki so ločene po dnevih.

Na začetku so se vsi tviti shranjevali v eno datoteko. Izkazalo se je, da velikost te datoteke relativno hitro narašča. Na dan se shrani nekje med 4 in 10 megabajti tvitov. Zaradi lažjega upravljanja, nadzora in shranjevanja tvitov v bazo spletne aplikacije, sem skripto prilagodil tako, da se vsak dan ustvari nova datoteka, kjer so shranjeni tviti tistega dneva.

Pri začetku razvoja aplikacije je največkrat prišlo do napake, ko je bilo premalo tvitov. Streaming API je naravnan tako, da prekine povezavo, ko v nekem obdobju ni novih tvitov. Do napake pride,

ko se začne *stream* zapirati ravno takrat, ko je na poti nov tvit in aplikacija prejme nepopolne podatke. Podobno je tudi, ko pride do težav v omrežju. Obe napaki sem rešil tako, da sem relevantne dele kode dal v *try*. S tem sem omogočil, da ni prišlo do popolne zaustavitve skripte. Te napake se zgodijo nekajkrat na teden. Ampak ker je več kot 10.000 tvitov na teden, teh nekaj izgubljenih tvitov bistveno ne spremeni rezultatov. Toda preden sem ta problem rešil, se je skripta nekajkrat ustavila in sem jo moral nato ročno pognati, ko sem napako opazil po nekaj urah. Izjema je napaka, ki se je zgodila 14. marca. Te napake nisem opazil do 24. marca. Zato v tem obdobju ni shranjenih tvitov. Od tega dne dalje, se shranjevanje tvitov ni več ustavilo. Iz tega razloga je 25. april najstarejši datum tvitov, ki so shranjeni v spletni aplikaciji.

Izvleček kode 3.1: Koda, ki zbira in shranjuje tvite.

```
timeout420 = 60

class listener(StreamListener):

    def on_data(self, data):
        try:
            fn= 'twitDB-'+datetime.date.today().strftime("%Y-%m-%d")+'.txt'
            saveFile = open(fn, 'a')
            saveFile.write(data)
            saveFile.write('\n')
            saveFile.close()
            return True
        except BaseException:
            #print('Napaka')
            time.sleep(5)
        global timeout420
        timeout420 = 60

    def on_error(self, status):
        print(status)
        if status == 420:
            global timeout420
            time.sleep(timeout420)
            timeout420 = timeout420*2
        else:
            time.sleep(320)
```

4 OBDELAVA PODATKOV

Twitter Streaming API posreduje podatke v JSON. JSON (Javascript Object Notation) je format datoteke za izmenjavo podatkov (Introducing JSON). Prednost te oblike je, da je ljudem enostavna za branje. Na drugi strani je za stroje enostavna za branje in razčlenjevanje ter hitra za shranjevanje in zapis podatkov. Kot pove že ime, JSON oblika zapisa temelji na programskem jeziku JavaScript. Kljub temu je ta tekstovni zapis neodvisen od programskega jezika, saj uporablja dve univerzalni podatkovni strukturi. Ti strukturi sta:

- Nabor parov ime – vrednost. Ta struktura je v Pythonu prisotna kot slovar ter vsebuje ključ in vrednost ter
- urejen seznam vrednosti. Tudi seznami so prisotni v Pythonu.

Podobno kot v Pythonovih podatkovnih strukturah je tudi v JSON formatu mogoče te strukture gnezditi (Introducing JSON).

4.1 ČIŠČENJE PODATKOV

Čiščenje podatkov je prvi ponavljajoči se korak, ki ga Han in Kamber (2000, 4) prepoznata v okviru odkrivanja znanja v podatkovnih bazah. Čiščenje podatkov je velikokrat korak, ki se opravi v času *predobdelove* oz. preden se podatki shranijo v podatkovno bazo. Tako je tudi v tem primeru, ko za vizualizacijo trendov na slovenskem Twitterju potrebujemo relativno malo podatkov v primerjavi s količino podatkov, ki nam je naj voljo.

V vsaki vrstici, ki nam jo posreduje Twitter Streaming API, je veliko podatkov o tvitu, osebi, ki je tvit objavila in drugem. Skupaj prejmemo 31 glavnih ključev in njihovih vrednosti. Ti ključi so:

- **text** - vrednost tega ključa je besedilo posameznega tvita. Je vrednost, ki je najpomembnejša za to aplikacijo. Primer: *"text":"@barjanski Meni je predvsem huda ta kognitivna disonanca.",*
- **user** - pod tem ključem *user* so shranjeni vsi podatki o uporabniku, ki je tvit objavil. Za to aplikacijo je najbolj pomembna vrednost, ki je znotraj uporabnika shranjena pod ključem *screen_name* - to je kratko ime uporabnika. Ker je to ime unikatno za vsakega uporabnika posebej, je to dovolj, da lahko razlikujemo med posameznimi uporabniki. Primer: *"screen_name":"nad_bogom",*
- **created_at** - pod tem ključem je shranjen čas objave tvita. Iz vrednosti lahko razberemo datum, do sekunde točen čas in ime dneva v tednu. Primer: *"created_at":"Tue May 02 22:21:22 +0000 2017" in*
- ostali ključi, ki jih Streaming API posreduje, a niso uporabni za našo aplikacijo: *coordinates, entities, favorite_count, favorited, filter_level, id, id_str, in_reply_to_screen_name, in_reply_to_status_id, in_reply_to_status_id_str, in_reply_to_user_id, in_reply_to_user_id_str, lang, place, possibly_sensitive, quoted_status_id, quoted_status_id_str, quoted_status, scopes, retweeted_count, retweeted,*

retweeted_status, source, truncated, withheld_copyright, withheld_in_countries, withheld_scope.

Nekateri ključi so lahko tudi brez vrednosti, saj vsi elementi in lastnosti niso prisotni v vseh tvitih (Twitter, Inc. 2017).

4.1.2 BESEDILO

Besedilo posameznega tvita je shranjeno pod ključem *text*. Iz besedila se zaradi lažjih nadaljnjih zapisov odstranijo nove vrstice in se jih nadomesti s presledki. Zaradi lažje primerjave besed se celotno besedilo preoblikuje tako, da vsebuje samo male črke. Twitter pri procesiranju tvitov nekatere elemente, kot so hiperpovezave, zavije v svoje hiperpovezave (Twitter, Inc. 2017). Pri razvoju se je izkazalo, da je takšnih hiperpovezav veliko ter da motijo prikaz najpogostejših besed. Te hiperpovezave, ki so značilne za Twitter, se nahajajo na koncu besedila, ki je shranjeno pod ključem *tekst*. Zato sem se pri ponovnem vnosu v bazo odločil, da uporabim samo del besedila, ki je shranjen pred temi hiperpovezavami.

Izvleček kode 4.1: Iskanje in priprava izvirnega teksta tvita za vnos v bazo.

```
t_text = json_load[u'text']
text_list = []
text_list.append(t_text)
if any('\n' in s for s in text_list):
    text_list = [w.replace('\n', ' ') for w in text_list]
tweet_text = str(text_list)
if 'https:\/\/' in tweet_text:
    tweet_text = tweet_text.split('https:\/\/')[0]
elif 't.co' in tweet_text:
    tweet_text = tweet_text.split('https://t.co')[0]
tweet_text_orig = tweet_text.lower()[2:-2]
```

Poleg izvirnega besedila, se shrani še lematizirano besedilo in besedilo, ki ne vsebuje motečih besed oz. *stopwords*.

Lematizacija

Pri prikazu pogosto uporabljenih besed je zelo pomemben korak lematizacija. Lematizacija je proces iskanja nevtralne slovarske oblike besede, ki ji pravimo lema. Na primer, lema besede *volkovi* je *volk*, lema besede *živim*, je *živeti*. Lematizacija tako abstrahira pregibne različice posameznih besed in je kot taka pomemben proces pri obdelavi teksta (Plisson in drugi 2008, 15).

Programski paket za lematizacijo slovenskega jezika (ter 11 drugih evropskih jezikov) so razvili na Inštitutu "Jožef Štefan". Projekt LemmaGen (Python) je primeren za lematizacijo teksta in lematizacijo vsake besede posebej. Je izjemno učinkovit, saj lahko obdela več milijonov besed na sekundo. Poleg tega je odprtokoden, brezplačen ter na voljo tudi za Python.

S pomočjo tega paketa sem lahko izboljšal poizvedbo. Lematizira se tudi vsaka iskana beseda in zato je posledično več zadetkov. V nasprotnem primeru bi bilo zadetkov manj, saj iskalnik ne bi našel tvita z besedo *ministri*, če bi v iskalnik vnesli besedo *minister*.

Izvleček kode 4.2: Lematizacija besed v tvitu.

```
tweet_text_re = re.findall(r'\w+', tweet_text)

lemmatizer = Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
tweet_text_lem = ''

for word in tweet_text_re:
    tweet_text_lem += ' '
    tweet_text_lem += lemmatizer.lemmatize(word).lower()
```

Stopwords

Stopwords so besede, ki nimajo vsebinskega pomena in se hkrati pogosto pojavljajo. Pri vizualizaciji bi bile tako samo v napoto, zato jih odstranim iz ene različice besedila tvita. Seznam slovenskih stopwords besed je na portalu *github* zbral uporabnik *genediazjr* (GitHub, Inc. 2017b). Ob testiranju sem ta seznam nekoliko prilagodil.

Pri prvi zasnovi aplikacije sem stopwords besede odstranil šele v koraku vizualizacije. To je sicer pomenilo, da je imel model v bazi en atribut manj in je bila zato velikost baze občutno manjša. Vendar je proces odstranjevanja stopwords besed zelo upočasnil pripravo besed za vizualizacijo. V nekem povprečnem tednu je dobrih 10.000 tvitov. To pomeni, da je trajalo več kot dve minuti, preden so se tviti prečistili in se je stran naložila. Sedaj, ko so tviti v bazi shranjeni brez stopwords besed, se nalaganje strani zaključi v manj kot sekundi.

Izvleček kode 4.3: Odstranjevanje *stopwords* besed iz besedila tvita.

```
Stopwords = [...] #seznam z vsemi 'stopwords' besedami

for stopword in stopwords:
    while stopword in tweet_text_re:
        tweet_text_re.remove(stopword)

tweet_text_stop = ''
for word in tweet_text_re:
    tweet_text_stop += ' '
    tweet_text_stop += word.lower()
```

4.1.2 UPORABNIK TER URA IN DATUM

Pridobivanje uporabniškega imena je najbolj enostavno. Najprej je potrebno poiskati podatke o uporabniku, ki so zapisani pod ključem *user*, in nato poiskati še njegovo kratko ime, ki se skriva pod ključem *screen_name*.

Pridobivanje časa nastanka tvita sem poiskal s ključem *created_at*. Pridobljeni čas se mora nato pretvoriti v obliko, ki je primerna za Djangovo bazo. Nazadnje se nastavi še lokalni, torej slovenski, čas. Streaming API namreč vse tvite vrne na čas, ki je nastavljen na UTC – Univerzalni koordinirani čas (ang. Coordinated Universal Time), ki za našim zaostaja za eno uro v zimskem času oz. dve uri v poletnem času.

Izvleček kode 4.4: iskanje in priprava časa objave tvita za vnos v bazo.

```
tweet_time = json_load['created_at']
tweet_time = time.strftime('%Y-%m-%d %H:%M:%S', time.strptime(tweet_time, '%a %b
%d %H:%M:%S +0000 %Y'))
slo = timezone('Europe/Ljubljana')
utc = timezone('UTC')
tweet_time = utc.localize(tweet_time)
tweet_time = tweet_time.astimezone(slo)
```

5 UPORABNIŠKA APLIKACIJA IN VIZUALIZACIJA PODATKOV

Spletna aplikacija je tisti del, ki bo v interakciji z uporabniki. Je končna postaja podatkov in omogoča dinamično vizualizacijo podatkov, ki je odvisna od uporabnikove poizvedbe.

5.1 SPLETNE APLIKACIJE

Internet raste tako hitro, da je postal nepogrešljiv del našega vsakdana, spletne aplikacije pa so neločljiv del programskih sistemov. Naše naprave, še posebej mobilne naprave, so omejene s procesno močjo in pomnilniškim prostorom, zato so z razvojem interneta začele spletne aplikacije prevzemati primat namiznim aplikacijam. Hyun in Soo (2010) sta prepričani, da so pomembno vlogo pri tem v zadnjem desetletju igrali tudi pametni telefoni in tablice.

Razširjen model, ki se ga poslužujejo razvijalci spletnih aplikacij, je MVC. *Model-View-Controller* (MVC) je model, ki temelji na treh glavnih komponentah:

- **Model**: najosnovnejši nivo, ki je zadolžen za upravljanje s podatki,
- **View**: modul, ki je v neposredni interakciji z uporabnikom,
- **Controller**: most med zgornjima moduloma. "Prošnjo", ki jo uporabnik zahteva preko *view*, posreduje v *model*. *Model* nato da odgovor in *controller* uporabniku posreduje primeren *view*.

5.2 DJANGO

Django je druga velika in pomembna Pythonova knjižnica, ki je ključna za to aplikacijo. Je napredno ogrodje za spletne aplikacije in spodbuja hiter razvoj ter čisto, pragmatično obliko. Django so izkušeni razvijalci zgradili z namenom, da se lahko spletni razvijalci posvetijo razvoju aplikacije. Poleg tega, da omogoča hiter razvoj spletnih aplikacij, je glavna prednost Djanga še visok nivo varnosti spletnih aplikacij in dejstvo, da je primeren za spletne aplikacije vseh velikosti – od preprostih, z majhno bazo podatkov in malo obiskovalci, do zelo kompleksnih z veliko bazo podatkov in velikim številom rednih uporabnikov (Django).

Django je knjižica, ki temelji na MVC modelu, vendar so si ustvarjalci ta model interpretirali nekoliko po svoje. *Model* je več ali manj še vedno *model*. *View* vidijo kot podatke, ki so predstavljeni uporabniku. Pri tem se ne ozirajo na to, kako podatki izgledajo, ampak kateri podatki so predstavljeni. Predstavitev podatkov je smiselno ločiti od same vsebine in zato to počnejo v ločenem delu, ki mu pravijo *template*. Zato je ta model bolj MTV, *Model-Template-View*. *Controller*, kot ga poznamo v MVC modelih, je tu vpet v samo strukturo in delovanje celotnega Django sistema (Django).

5.3 BAZA PODATKOV

Za shranjevanje in dostop do podatkov spletne aplikacije sem uporabil podatkovno bazo. Standardni jezik za delo z relacijskimi bazami je jezik SQL oz. *Structured Query Language*. Ime izhaja iz osnovne funkcije, ki jo jezik omogoča, to je poizvedba (angl. Q*uery*). Čeprav to še zdaleč ni edina stvar, ki jo SQL omogoča. Med drugim omogoča delo s podatki in z metapodatki (Jaklič 2002, 125).

Django podpira več različnih različic SQL podatkovnih baz. Privzeta knjižnica je SQLite oziroma najnovejša različica te knjižnice, ki je SQLite3. SQLite omogoča upravljanje z bazo brez strežnika, saj zapisuje neposredno v datoteke na sistemu. Vse tabele, indekse in poglede hrani v eni sami datoteki. Poleg tega je za uporabo na voljo takoj, brez predhodnih konfiguracij. Oblika datoteke, kamor SQLite shranjuje podatke, je podprta na različnih operacijskih sistemih. SQLite knjižnica je v javni domeni, in je posledično brezplačna ter na sistemih zasede relativno malo prostora – odvisno od verzije operacijskega sistema in količine vključenih dodatnih lastnosti, toda vedno manj od 1 MB. To so nekateri izmed razlogov, zakaj je danes SQLite najbolj razširjena podatkovna baza na svetu (SQLite 2017).

Vseeno veliko znanja SQL sintakse ni potrebnega, saj Django omogoča, da se delo s podatki in metapodatki opravi preko sintakse, ki je bolj "pythonovska" po naravi. Django nato to sintakso prevede v SQL sintakso, ki se izvede na bazi.

5.3.1 MODEL

Django in SQL pri aplikacijah omogočata ustvarjanje kompleksnih relacijskih modelov podatkov, ki jih definiramo v Djangovi datoteki models.py. Vendar sem za potrebe te aplikacije uporabil preprost model baze z enim entitetnim tipom. Ime tega entitetnega tipa v naši aplikaciji je Tweet, ki sem ga definiral s Pythonovim razredom in mu nato določil posamezne atribute. Te atribute sem nato dalje definiral s tipom atributa (besedilo, število, datum ...) različnimi omejitvami in dodatnimi informacijami, ki Djangu in posledično SQLitu povedo, kako naj se ti atributi vedejo. Vsi atributi so obvezni in morajo imeti vrednost. Atributi so naslednji:

- **tweet_text**: izvirna vsebina tvita – ker je to polje z besedilom, ga je potrebno omejiti s številom znakov. Omejitev je 500 znakov. Maksimalna dolžina posameznega tvita je 140 znakov, vendar se v teh 140 znakov ne štejejo hiperpovezave, citiranje drugih tvitov in podobno, zato je omejitev v bazi višja.
- **tweet_text_lem**: vsebina tvita, kjer so vse besede lematizirane. Omejitev je 500 znakov.
- **tweet_text_stop**: vsebina tvita, kjer so odstranjene vse stopwords besede. Omejitev je 500 znakov.
- **tweet_date**: datum objave tvita. V obliki Python stringa je v model dodan tudi opis tega atributa.
- **tweet_user**: kratko ime uporabnika, ki je tvit objavil. Tudi to je polje z besedilom, omejitev je 100 znakov.
- **ID**: unikaten identifikator vsake entitete ali ključ, ki ga Django oz. SQLite doda samodejno.

Izvleček kode 5.1: Definiranje modela in atributov.

```
class Tweet(models.Model):
    tweet_text_orig = models.CharField(max_length=500)
    tweet_text_lem = models.CharField(max_length=500)
    tweet_text_stop = models.CharField(max_length=500)
    tweet_date = models.DateField('date tweet published')
    tweet_user = models.CharField(max_length=100)
```

5.3.2 SHRANJEVANJE

Shranjevanje podatkov v SQLite bazo se izvede enkrat dnevno, s pomočjo po meri narejene skripte populatedb.py. V bazo se shranijo podatki, ki so bili zbrani prejšnji dan. To je precej enostavno, saj

se ob prvotnem zbiranju podatkov, neobdelani podatki shranjujejo v datoteke, ki so ločene po dnevih. Zato ta skripta samo poišče datoteko, ki se je ustvarila prejšnji dan, jo odpre in prebere vsako vrstico posebej.

Skripta pregleda vsako vrstico posebej in izloči podatke na način, kot je predstavljeno v točki 4.1. Vse vrednosti atributov se shranijo in zapišejo v bazo. Django oz. SQLite ob tem vsakemu shranjenemu tvitu doda še unikaten ključ.

Izvleček kode 5.2: Shranjevanje enega tvita.

```
t = Tweet(tweet_user=tweet_user, tweet_date=datetime.date(tweet_time),
tweet_text_orig=tweet_text_orig, tweet_text_lem=tweet_text_lem,
tweet_text_stop=tweet_text_stop)
t.save()
```

Število tvitov, ki so dnevno shranjeni, precej variira. Za vikend je občutno manj tvitov – dobrih tisoč na dan. V času delovnih dni jih je praviloma več – tudi po 1.700, 1.800 na dan. To pomeni, da je navadno dobrih 10.000 tvitov vsak teden.

5.3.3 POIZVEDBA

Uporabnik lahko ob obisku spletne aplikacije sproži dve poizvedbi v podatkovni bazi. Tudi tu Django omogoča, da se poizvedba napiše v Pythonovi sintaksi, ki jo Django prevede v SQL sintakso. Prva poizvedba se izvede, ko obišče vstopno stran in je uporabnik ne nadzoruje. Ta poizvedba temelji na "današnjem" datumu. To je datum, ko uporabnik obišče stran. SQLite nato vrne vse tvite v bazi, ki so bili objavljeni v zadnjem tednu. Ker se baza osvežuje z enodnevno zamudo, je zato začetek tega tedna pred 7 dnevi in konec tedna 1 dan nazaj. Nato se v datoteki views.py na vrnjenih tvitih oziroma njihovih atributih izvedejo posamezne operacije, ki pripravijo podatke za posamezne vizualizacije.

Izvleček kode 5.3: Poizvedba v bazi za tviti, ki so bili objavljeni v preteklem tednu.

```
enddate = timezone.now() - timedelta(days = 1)
startdate = enddate - timedelta(days = 6)
Tweet.objects.filter(tweet_date__range=[startdate, enddate])
```

Druga poizvedba se izvede, ko uporabnik vnese želeni niz v iskalnik. Aplikacija najprej poišče vse besede, ki so bile vpisane v iskalnik. Nato vsako besedo posebej lematizira in lematizirano besedo shrani v zapisu z majhnimi črkami. Nato se izvede poizvedba, kjer so pogoji iskanja takšni, da mora biti v lematiziranih tvitih vsaka lematizirana beseda, ki jo je uporabnik vpisal v iskalnik. Na tvitih, ki ustrezajo tem pogojem se nato izvedejo operacije za posamezne vizualizacije.

Izvleček kode 5.4: Poizvedba v bazi za tviti, ki vsebujejo iskane besede.

```
query = request.GET.get('query')
```

```
query_words = re.findall(r'\w+', query)
query_lem = []
for word in query_words:
    query_lem.append(Lemmatizer.lemmatize(word).lower())
Tweet.objects.filter(functools.reduce(operator.and_,
(Q(tweet_text_lem__icontains=x) for x in query_words_lem)))
```

5.4 VIZUALIZACIJA PODATKOV

Vizualizacija podatkov je končna postaja naše aplikacije. Je rezultat uporabniške poizvedbe ter tisti del, ki uporabniku pove zgodbo. Vizualizacija podatkov (angl. *data visualization*) je ožji pojem, ki spada pod pojem *vizualizacije informacij* (angl. *information visualization*). Vse kar je dovolj organizirano, je neke vrste informacija. Tabele, grafi, zemljevidi in celo teksti, bodisi statični, bodisi dinamični, podajajo sredstva, da najdemo odgovore na vprašanja. Vprašanja, ki se nam včasih lahko pojavijo samo, ko vidimo podatke v grafični podobi (Friendly 2009, 2).

Vizualizacijo podatkov, kot ožji pojem, Friendly (2009, 2) razume kot vizualno predstavitev podatkov. Ta vizualna predstavitev je informacija, ki je bila abstrahirana določenih elementov. Cilj je vizualna predstavitev za raziskovanje in odkrivanje.

Tufte (2009) pravi, da je odlična statistična grafika sestavljena iz kompleksnih idej, ki se izražajo jasno, natančno in učinkovito. Grafični podatki so dobro zasnovana predstavitev zanimivih podatkov in so stvar vsebine, statistike in oblikovanja. Grafični prikazi morajo odkrivati podatke. Grafični prikazi so lahko celo bolj natančni in povedni kot običajni statistični izračuni. Grafični prikaz naj:

- prikaže podatke,
- izzove bralca, da razmišlja o vsebini in ne metodologiji, grafični obliki, tehnologiji grafične produkcije ali čem podobnem,
- se izogiba izkrivljanju zgodbe, ki jo naj podatki povedo,
- predstavi veliko številk na malem prostoru,
- da veliki zbirki podatkov pomen,
- spodbuja bralca, da primerja različne dele podatkov,
- razkrije podatke na več ravneh podrobnosti – od celostne slike do natančnih podrobnosti,
- služi jasnemu namenu: opisovanju, odkrivanju, tabeliranju ali okrasu in
- bo tesno povezan s statističnimi in vsebinskimi opisi nabora podatkov.

Preden se podatki vizualizirajo, je pomembno, da izberemo pravo vrsto vizualizacije za izbrane podatke in sporočilo, ki ga hočemo prenesti bralcem. Few (2004) pravi, da kvantitativne podatke vedno spremljajo kategorični podatki, ki kvantitativne podatke osmišljajo. Kvantitativni podatki nam povedo *koliko*, kategorični podatki pa *kaj*. Iz kategoričnih podatkov moramo ugotoviti, za kakšne podatke gre: nominalne, ordinalne, razmernostne ali intervalne. Razdelitev na kvantitativne in kategorične podatke je zelo pomembna, saj je prepoznavanje vrste podatkov prvi korak za izbor prave vizualizacije. Tudi v modelu moje aplikacije je shranjenih več različnih tipov podatkov. Sicer več o izboru primernega grafičnega prikaza glede na posamezno vrsto podatkov sledi v nadaljevanju, ko je predstavljena vsaka vizualizacija posebej.

Poleg vseh osnovnih elementov, ki služijo oblikovanju vsebin na spletu (HTML, CSS in JavaScript) sem za pomoč pri vizualizaciji uporabil JavaScript knjižnico D3. D3 oziroma *Data-Driven Documents* omogoča vizualizacijo podatkov z uporabo sodobnih spletnih standardov. D3 združuje vizualizacijo z interaktivnostjo. Omogoča veliko svobode pri oblikovanju in hkrati izkorišča vse zmogljivosti modernih brskalnikov (Data-Driven Documents).

Vse vizualizacije se pojavijo na dveh straneh. Na vstopni strani, glede na časovno obdobje preteklega tedna, ter ob uporabnikovem vnosu iskalnega niza.

Slika 5.1: Primer vstopne strani na dan 29. junija 2017.

Spletna stran sicer nima veliko elementov. Ima glavo in nogo za osnovne informacije ter navigacijo. V glavi na desni strani ima iskalnik, v katerega se vpiše želeno poizvedbo. Nato sledi glavni del – prikaz podatkov. Najprej se nam izpiše število vseh tvitov, ki so jih slovenski novinarji objavili v zadnjih sedmih dneh. Nato sledi izpis vseh besed in uporabniških imen, ki so bila najpogosteje omenjena v izbranih tvitih. Večja kot je beseda, pogosteje je bila omenjena. Tretja vizualizacija

nam prikaže število izbranih tvitov in različnih uporabnikov po dnevih. Sledi še zadnja vizualizacija, ki prikaže pet novinarjev oz. medijev, ki so najpogosteje tvitali v preteklem tednu. V primeru, da v iskalnik vpišemo nek niz, se na enak način izpišejo vizualizacije, ki se nanašajo na iskani niz.

V nadaljevanju bom prikazal vse oblike vizualizacije, ki jih ponuja spletna aplikacije, z dvema primeroma. Ob iskanju besede "magna" in "arbitraža".

5.4.1 ŠTEVILO VSEH TVITOV

Število vseh tvitov je najenostavnejša vizualizacija, če ji lahko tako sploh rečemo. Glede na vneseno poizvedbo, Python prešteje število vseh tvitov, ki jih je SQLite vrnil. Nato se ta številka izpiše, kot prva informacija na spletni strani. Zaradi preprostosti informacije in vizualizacije ni tu uporabljen noben dodaten vtičnik. Vse je oblikovano s pomočjo osnovnih HTML in CSS tehnologij.

Slika 5.2: Prikaz skupnega števila tvitov ob iskanem nizu "magna" na dan 4. julija 2017.

Beseda "magna" se je od začetka zbiranja podatkov do 3. julija (podatki na dan poizvedbe še niso shranjeni v bazi aplikacije) pojavila v 1749 tvitih. Sama številka ne pove dosti. Ob pogostejši uporabi te aplikacije in poizvedbi večih besed, ugotovimo, da je ta med bolj pogostimi besedami v tem obdobju.

Slika 5.3: Prikaz skupnega števila tvitov ob iskanem nizu "arbitraža" na dan 4. julija 2017.

Na drugi strani se beseda "arbitraža" pojavlja v veliko manj tvitih kot beseda "magna". 833 tvitov do 3. julija pomeni, da je razmerje med besedama približno 2:1.

5.4.2 BESEDE, KI SE POJAVLJAJO POLEG IZBRANE

Za prikaz besed, ki se v tvitih najpogosteje uporabljajo ob iskanem nizu sem uporabil vizualizacijo v obliki *wordcloud* oz. oblaka besed. Wordcloud so za vizualizacijo vsebine na twitterju uporabljali že v preteklosti. Eden največjih problemov, pri delu s takšnimi podatki, je preobremenjenost z informacijami. V našem primeru se lahko v enem dnevu v bazo shrani par tisoč različnih tvitov. Kaptein (2012) se je osredotočila na uporabo wordclouda pri analizi rezultatov iskanja po Twitterju. Pri tem si je s to vizualizacijo pomagala pri navigaciji in pri prikazu povzetka iskanja.

V naši aplikaciji je uporaba wordclouda namenjena temu, da prikaže najpogostejše besede, ki se pojavljajo v tvitih, ki jih vrne poizvedba. Torej gre tudi tu za nekakšne vrste povzetek.

Tudi tu je uporabljena lematizacija. To pomeni, da se v končnem izpisu vsaka beseda pojavi samo enkrat, npr. *tir*. V nasprotnem primeru bi se lahko ta beseda pojavila tudi večkrat v različnih oblikah. Npr. *tiru*, *tira*, *tirom*.

Od vseh vizualizacij je ta najbolj kompleksna in vključuje največ dodatnih knjižnic za vizualizacijo. Priprava podatkov je sicer precej preprosta. Iz vseh tvitov, ki jih poizvedba vrne, funkcija poišče vse besede in prešteje, kolikokrat se katera pojavlja. Nato se 30 najpogostejših shrani v seznam. Med temi tridesetimi se nato za vsako v ločen seznam shrani še delež pogostosti pojavljanja, ki se pomnoži z 2000. Tako dobimo besede in njihove velikosti, ki so primerne za vizualizacijo.

Izvleček kode 5.5: Priprava podatkov za wordcloud vizualizacijo.

```
for tweet in Tweet.objects.filter(tweet_date__range=[startdate, enddate]):
    words += ' '
    words += str(tweet.tweet_text_stop)

tweet_text_words = re.findall(r'\w+', words)

lemmatizer = Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
lem_words = []
lem_words_count = defaultdict(list)
for word in tweet_text_words:
    lem_words.append(lemmatizer.lemmatize(word))
    lem_words_count[lemmatizer.lemmatize(word)].append(word)

word_count = Counter(lem_words).most_common(30)
word_count_sum = 0
for word in word_count:
    word_count_sum += word[1]

tweet_text_list = []
for word in word_count:
    word_dict = {}
    word_dict['size'] = word[1]*2000//word_count_sum
    word_dict['text'] = Counter(lem_words_count[word[0]]).most_common(1)[0][0]
    tweet_text_list.append(word_dict)
```

Za vizualizacijo sem uporabil kodo, ki jo je spisal Benny Bottema (2017). Bottema je svoj wordcloud zasnoval na wordcloudu Jasona Davisa (GitHub, Inc. 2017a), ki je za osnovo uporabil JavaScript knjižnico D3. Kodo sem prilagodil tako, da sem odstranil elemente, ki jih ne potrebujem, v prvi vrsti iskalnik, ki išče po wordcloudu.

Kljub temu, da so besede različnih velikosti, se tudi pobarvajo različno, in sicer z osmimi različnimi barvami. Barve sicer niso povezane z analizo, ampak pomagajo pri razločitvi različnih besed.

Izvleček kode 5.6: Definiranje barv s katerimi se obarvajo izpisane besede.

```
var fill = d3.scaleOrdinal()
  .range(["#226666", "#aa6c39", "#0d4d4d", "#804515", "#003333", "#552600",
"#407f7f", "#d49a6a"]);

d3.select("#cloud").append("svg")
  .style("fill", function(d, i) {
      return fill(i);
  })
```

Davisov wordcloud je zasnovan tako, da najprej izriše največjo besedo. Nato ji doda drugo največjo, nato tretjo itd. Problem nastane, ko za besedo ne najde prostora. V tem primeru to besedo izpusti in poskuša izrisati naslednjo. Bottema je zato dodal funkcijo, ki v tem primeru besedo nekoliko zmanjša v velikosti in jo poskuša še enkrat izrisati. Poskušanja sem omejil na 6, kar se je pri testiranju izkazalo za dovolj. Na manjših zaslonih je število poskusov omejeno na 3.

Izvleček kode 5.7: Definiranje števila možnih poskusov, da se beseda izpiše.

```
var MAX_TRIES = (width > 400) ? 6 : 3;

d3.layout.cloud()
    .size([width, height])
    .on("end", function(fittedWords) {
        if (fittedWords.length == wordsToDraw.length) {
            drawSkillCloud(fittedWords);
        }
        else if (!retryCycle || retryCycle < MAX_TRIES) {
            generateSkillCloud((retryCycle || 1) + 1);
        }
        else {
            drawSkillCloud(fittedWords);
        }
    })
    .start();
```

Slika 5.4: Prikaz besed, ki se najpogosteje pojavljajo ob besedi "magna" na dan 4. julija 2017.

Ob izpisu besed, ki se pojavljajo v tvitih, ki vsebujejo besedo "magna" ugotovimo, da imamo eno izrazito dominantno besedo. To je "vladars". "vladaRS" je sicer uradni Twitter profil Vlade Republike Slovenije. Glede na to, da je to en izmed odmevnejših projektov te vlade, ne gre čuditi, da se ravno oni največkrat pojavljajo v teh tvitih. Tudi profil največje koalicijske stranke in predsednika vlade se pojavljata v teh tvitih. Poleg tega tudi kaj se bo gradilo v tem projektu – "lakirnico", kraj, kjer se bo lakirnica gradila – "hoče" ter tudi stvari, ki so bolj sporne v tej zadevi: "zemljo" in "kmetom".

Slika 5.5: Prikaz besed, ki se najpogosteje pojavljajo ob besedi "arbitraža" na dan 4. julija 2017.

V tvitih, ki vsebujejo besedo "arbitraža" ni tako izrazito dominantnih besed. Največkrat se pojavlja "hrvaške", "meje", "sloveniji". Torej besede, ki nekako opišejo bistvo problema arbitraže: meja med Slovenijo in Hrvaško. Nekoliko manj izrazite so besede "odprtim", "morjem" in "piranski". Besede, ki znotraj arbitraže ponazarjajo najbolj kočljiv problem: razdelitev Piranskega zaliva in dostop Slovenije do odprtega morja. Relativno pogosto se pojavlja tudi "nova24tv", to je Twitter profil istoimenskega medija. Sklepamo lahko, da so bili zelo aktivni pri spremljanju arbitraže.

5.4.3 UPORABA BESEDE SKOZI ČAS

Prikaz pogostosti uporabe iskanega niza skozi čas je spremenljivka z razmernostno mersko lestvico. Namen vizualizacije je prikaz trendov skozi čas. Imamo 2 vrsti podatkov: število tvitov, ki vsebujejo iskani niz ter število različnih uporabnikov, ki so objavili tvit, ki vsebuje iskani niz. Podatki so zbrani v enakomernih intervalih – vsak dan posebej. Za ta namen Few (2004, 5) prepoznava časovnico kot najboljši primer vizualizacije, kjer je spremenljivka čas vedno prikazana na horizontalni osi. Predlaga točke z vrednostmi za vsako časovno točko merjenja, da se izpostavijo posamezne dnevne vrednosti. Prav tako predlaga, da se točke povežejo s črto, da se izpostavi splošni vzorec.

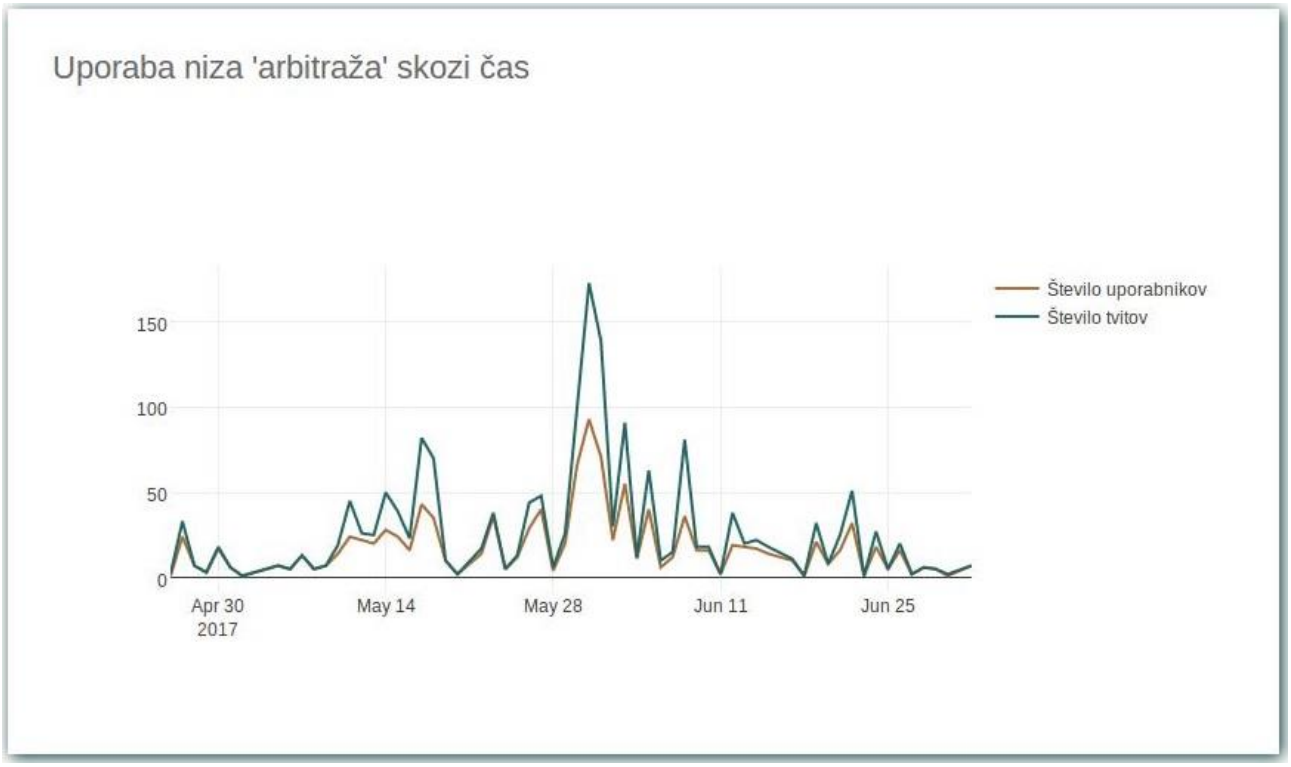
Izvleček kode 5.8: Priprava podatkov za izris časovnice.

```
dates = []
x = []
```

```
trace1 = []
trace2 = []
datesXusers = defaultdict(list)

for tweet in Tweet.objects.filter(tweet_date__range=[startdate, enddate]):
    dates.append(str(tweet.tweet_date))
    if tweet.tweet_user not in datesXusers[str(tweet.tweet_date)]:
        datesXusers[str(tweet.tweet_date)].append(tweet.tweet_user)

dates_counted = Counter(dates)
for key in sorted(dates_counted):
    x.append(key)
    trace1.append(dates_counted[key])

for key in sorted(datesXusers):
    trace2.append(len(datesXusers[key]))
```

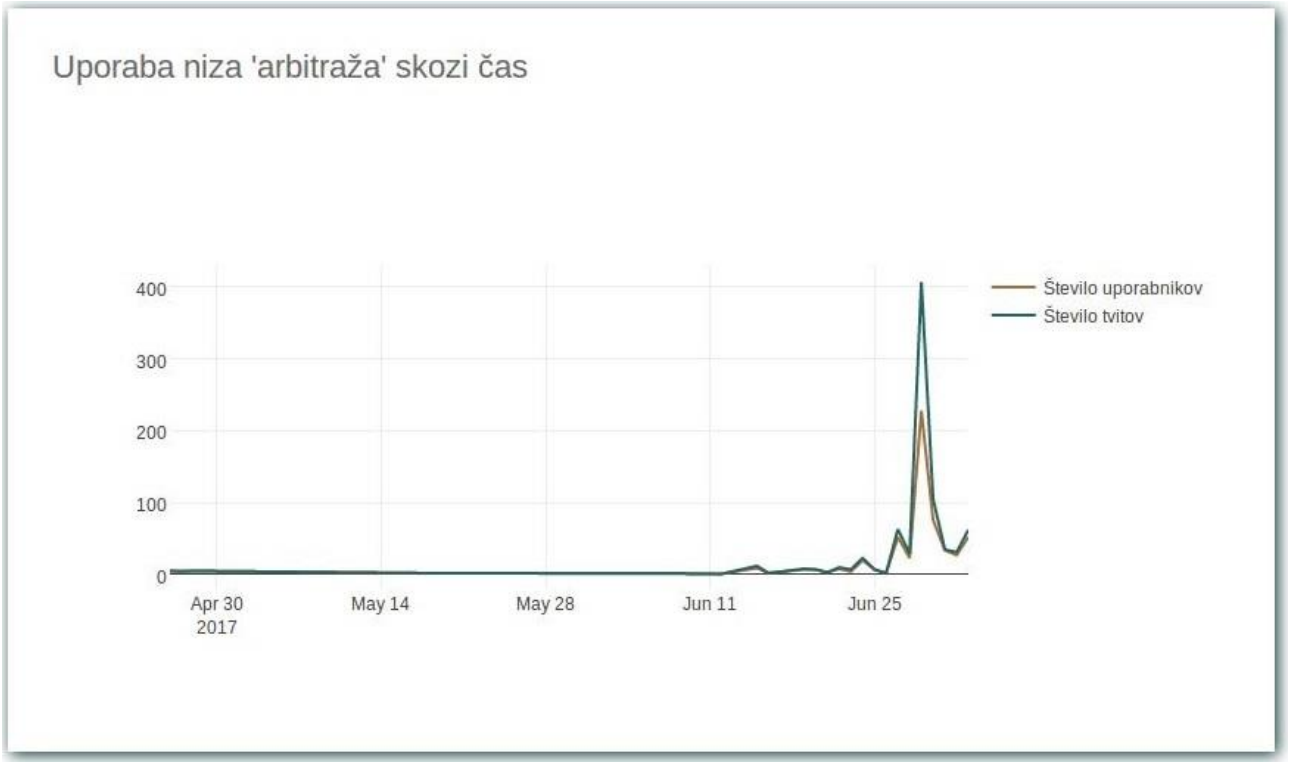
Za časovni prikaz sem uporabil JavaScript knjižnico *plotly.js*. Plotly.js knjižnica je zgrajena na osnovi že omenjene knjižnice D3 (Plotly 2017). Čeprav je to zelo visoko-nivojska knjižnica, ki ponuja 20 različnih tipov grafov, sem uporabil le del njenih zmogljivosti. Ko se jo enkrat vključi v projekt, je za izris časovnice potrebno relativno malo vrstic kode. Potrebno je samo definirati lastnosti posameznih linij, postavitev in vhodne podatke. Vhodni podatki so v obliki seznama. Za vsako linijo je potrebno v ločenem seznamu podati podatke za x os – to so dnevi, in vrednosti y – to je število tvitov oziroma število različnih uporabnikov za posamezen dan. Te podatke sem nato dinamično vnesel s Pythonovimi seznami.

Slika 5.6: Prikaz pogostosti uporabe "magna" skozi čas na dan 4. julija 2017.



Pri časovnem prikazu besede "magna" v tvitih vidimo, da se o tem projektu govori vsaj od začetka zbiranja podatkov. Sicer je ta beseda postala pravi trend šele s koncem aprila. Prvi vrh je dosegla 18. maja. Predvidevam, da je k temu precej pripomogla takratna oddaja Tarča na TV Slovenija na to temo. Največ tvitov je bilo 31. maja, ko je bilo v enem dnevu objavljenih več kot 200 tvitov s to besedo. Na tem primeru je tudi dobro razvidno, kako med novinarji pade uporaba Twitterja med vikendi.

Slika 5.7: Prikaz pogostosti uporabe "arbitraža" skozi čas na dan 4. julija 2017.



Na drugi strani ima beseda "arbitraža" precej mirnejšo zgodovino. Šele v drugi polovici junija vidimo resnejši poskok uporabe te besede. Ampak ima na dan razglasitve sodbe arbitraže 29. junija precej višji vrh, saj je bilo objavljenih več kot 400 tvitov s to besedo.

5.4.4 TVITERAŠI, KI NAJVEČKRAT UPORABLJAJO TO BESEDO

Zadnja vizualizacija, ki jo ponuja aplikacija, je prikaz uporabnikov tviterja, ki so največkrat objavili tvit, ki vsebuje iskani niz. To je spremenljivka z nominalno mersko lestvico. Namen vizualizacije je prikaz uporabniških imen in število objavljenih tvitov, ki vsebujejo iskani niz. Zaradi lažje preglednosti, je prikazanih samo 5 najbolj "aktivnih" uporabnikov. V ta namen Few (2004, 5) predlaga vizualizacijo z rangiranjem. Ker je namen izpostaviti najbolj "aktivne" uporabnike, predlaga horizontalni stolpčni diagram, kjer vrednosti padajo.

Izvleček kode 5.9: Priprava podatkov za priakz najaktivnejših uporabnikov.

```
users = []
top5 = []
top = {}

for tweet in Tweet.objects.filter(tweet_date__range=[startdate, enddate]):
    users.append(tweet.tweet_user)
top_users = Counter(users)

for key in (top_users):
    top[top_users[key]] = key
```

```
user1 = sorted(top, reverse=True)[:1]

for key in (sorted(top, reverse=True)[:5]):
    i = []
    i.append(top[key])
    i.append(key)
    i.append(100 * int(key) // int(sorted(top, reverse=True)[0]))
    top5.append(i)
```

Za to vizualizacijo nisem uporabil nobenih dodatnih knjižnic. Samo osnovne HTML in CSS tehnologije za osnovno obliko in barvo horizontalnih stolpcev. Pri vsaki vrstici se s pomočjo Pythona dinamično vnesejo najprej ime uporabnika (levo), širina stolpca (sredina) in število tvitov (desno). Velikost stolpcev sem omejil z absolutno dolžino 400 pikslov. Pri najbolj aktivnemu tviterašu je potem tudi moder stolpec vedno maksimalne dolžine – 400 pikslov. Dolžina stolpcev vseh ostalih je bila nastavljena relativno – glede na delež tvitov v primerjavi z najbolj aktivnim uporabnikom. Če bi na primer *uporabnik1* objavil 200 tvitov in *uporabnik2* 120 tvitov, bi bila dolžina stolpca *uporabnik1* 400 pikslov (100 %) in *uporabnik2* 240 pikslov (60 %).

Slika 5.8: Prikaz uporabnikov, ki so največkrat uporabili besedo "magna" na dan 4. julija 2017.

Vidimo, da je bil v obdobju od začetka iskanja do 3. julija 2017 uporabnik BojanPozar daleč najaktivnejši pri objavljanju tvitov s to besedo. To je Twitter profil odgovornega urednika medija Požareport. Sicer so o tej temi pisali še uporabniki Pertinacal, MatijaStepišnik, petra_jansa in SrecniLuka.

Slika 5.9: Prikaz uporabnikov, ki so največkrat uporabili besedo "arbitraža" na dan 4. julija 2017.

Tudi besedo "arbitraža" je v največ tvitih objavil BojanPozar. Sicer so razlike med prvimi petimi uporabniki manjše kot pri besedi "magna". Spet sta med prvimi petimi uporabniki tudi MatijaStepisnik in Pertinacal. Nova24TV je na petem mestu. Da je med prvimi petimi verjetno ni presenetljivo, saj se je to ime pojavilo že v *wordcloud* vizualizaciji ob tej poizvedbi.

6 ZAKLJUČEK

Z izdelavo te aplikacije sem hotel doseči dve stvari. Prvič: hotel sem prikazati, da je mogoče s sodobnimi orodji uporabniku na razumljiv in jedrnat način prikazati vsebino Twitterja. Drugi namen je, da se slovenskim uporabnikom Twitterja poda orodje, ki jim omogoča lažje spremljanje dogajanja na slovenskem Twitterju. Storitev Twitter že sama ponuja nekatere analize na področju trendov. Vendar je to prva, prosto-dostopna spletna aplikacija, ki poskuša spremljati trende slovenskih medijev in novinarjev na Twitterju. Poleg najpogostejših besed, ponuja tudi najaktivnejše uporabnike in uporabo besed skozi čas. To je nekaj, česar Twitter ne ponuja. Pomembno je tudi poudariti, da je uporaba te aplikacije, na naslovu http://webales.fdv.uni-lj.si/tweet, na voljo vsem. Tako lahko trende spremljajo tudi osebe, ki nimajo svojega Twitter računa.

Kljub temu je pri aplikaciji še precej možnosti za izboljšave in razširitev ponudbe. Za res natančno spremljanje dogajanja trendov na slovenskem Twitterju, bi bil prvi korak natančnejši izbor spremljanih uporabnikov. To je sicer naloga, ki bi potrebovala ločeno, poglobljeno kvantitativno in

kvalitativno analizo. V isti sapi bi bila dobrodošla tudi natančnejša določitev slovenskih stopwords besed. Na tem področju je zaenkrat malo takšnih seznamov.

Trenutno lahko uporabnik išče tvite samo po vsebini. V prihodnosti bi bilo dobro dodati še iskanje tvitov po datumu, po osebah in kombinaciji vseh treh. Z natančnejšim izborom spremljanih uporabnikov in njihovo analizo, bi v prihodnosti dodal še analizo glede na skupine tviterašev, npr. spletni mediji, tiskani mediji, konservativni mediji, liberalni mediji …

Trenutno je prikaz celotnega števila tvitov zgolj informativna številka, ki ob redki uporabi aplikacije ne pove veliko. Rešitev za izboljšavo tega bi bila tudi relativizacija te številke. Vendar je problem, s čim primerjati. Če bi izračunali delež teh tvitov v primerjavi s številom vseh tvitov, ki so v bazi, potem bi v večini primerov dobili relativno nizke deleže, ki zopet ne bi veliko povedali. Tedensko število tvitov bi lahko primerjali s tednom, ko je bilo največ tvitov. Število tvitov, ki vsebujejo iskani niz bi lahko primerjali z najpogostejšo besedo tistega obdobja. Vendar je problem, ker bi takšni izračuni zahtevali veliko procesne moči oz. veliko časa.

Poizvedba, lematizacija in vizualizacija *wordcloud* trenutno obravnavajo vsako besedo posebej. To je po eni strani pospešilo razvoj aplikacije ter pospešilo poizvedbe, ki se izvedejo. Slaba lastnost tega je, da aplikacija ne prepoznava besednih zvez. V *wordcloudu* bi bilo bolj smiselno, da se besedna zveza *drugi tir* izpiše skupaj in ne vsaka beseda posebej. Če vpišemo *drugi tir* v iskalnik, potem je med analiziranimi tviti hipotetično tudi tvit z besedilom: »To je že druga oseba, ki dela na tirih,« kar nas v tem primeru verjetno ne zanima. Trenutno iskalnik deluje tako, da zajame širok nabor tvitov, ki ustrezajo iskalnim pogojem in ni primeren za zelo specifična iskanja, ki temeljijo na besednih zvezah.

Ena izmed značilnosti slovenskega jezika so tudi šumniki. Vendar se predvsem pri digitalni komunikaciji večkrat kaže, da vsi šumnikov ne uporabljajo. Moja aplikacija trenutno loči med besedami, ki so pisane s šumniki od tistih, ki so pisane s sičniki. Tudi tu bi bilo v prihodnosti dobro, če bi aplikacijo optimiziral tako, da bi enačila besede, ki so pisane s šumniki s tistimi, ki so pisane s sičniki. Tako bi lahko pri iskanju zajeli večje število tvitov. S tem bi se verjetno predvsem izboljšala vizualizacija *wordcloud*. Problem te optimizacije je sicer podoben kot pri lematizaciji. Če bi se operacije izvedle ob uporabnikovi poizvedbi, bi se s tem podaljšal čas nalaganja strani. Če bi shranjevali vse besede v obeh oblikah, torej enkrat s šumniki in enkrat sičniki, bi se s tem občutno povečala podatkovna baza spletne aplikacije.

Pomembna analiza, ki bi v prihodnosti predstavljala dodano vrednost aplikacije, je analiza sentimenta. To je vsekakor nekaj, kar imam v mislih za v prihodnost, vendar je analiza sentimenta element, ki je verjetno vsaj tako zahteven, kot že obstoječe analize in vizualizacije skupaj.

Za boljšo uporabniško izkušnjo vidim možnost za napredek pri wordcloudu, kjer bi besede spremenil v hiperpovezave. Te hiperpovezave bi nato vodile do analize, ki se nanašajo na to besedo. Tudi iskalnik ima možnosti za napredek. Zaenkrat poišče samo tiste tvite, ki vsebujejo vse iskane besede. V začetku je bila tudi ideja, da se tvite prikaže tudi v njihovi celotni, izvirni podobi. Morda je to nekaj, kar bo postalo del aplikacije v prihodnosti.

Kljub tem možnostim za napredek in razširitev ponudbe menim, da mi je uspelo uresničiti glavna cilja te aplikacije.

7 LITERATURA

1. Boyd, B. Danah in Nicole B. Ellison. 2007. Social Network Sites: Definition, History, and Scholarship. *Journal of Computer-Mediated Communication* 13 (1): 210–230.
2. Chakraborty, Abhijnan, Johnnatan Messias, Fabricio Benevenuto, Saptarshi Ghosh, Niloy Ganguly in Krishna P. Gummadi. 2017. *Who Makes Trends? Understanding Demographic Biases in Crowdsourced Recommendations*. Dostopno prek: https://arxiv.org/pdf/1704.00139.pdf (9. junij 2017).
3. GitHub, Inc. 2017a. *D3-Cloud*. Dostopno prek https://github.com/jasondavies/d3-cloud (5. maj 2017).
4. --- 2017b. *Stopwords-sl*. Dostopno prek: https://github.com/stopwords-iso/stopwords-sl/blob/master/stopwords-sl.txt (20. maj 2017).
5. *Data-Driven Documents*. Dostopno prek: https://d3js.org/ (19. marec 2017).
6. *Django*. Dostopno prek: https://www.djangoproject.com/ (12. marec 2017).
7. Few, Stephen. 2004. *Eenie, Meenie, Minie, Moe: Selecting the Right Graph for Your Message*. Dostopno prek https://www.perceptualedge.com/articles/ie/the_right_graph.pdf (10. maj 2017).
8. Friendly, Michael. 2012. *Milestones in the history of thematic cartography, statistical graphics, and data visualization*. Dostopno prek: http://www.math.yorku.ca/SCS/Gallery/milestone/milestone.pdf (10. maj 2017).
9. *Introducing JSON*. Dostopno prek: http://www.json.org/ (19. marec 2017).
10. Jaklič, Jurij. 2002. *Upravljanje in uporaba podatkov*. Ljubljana: Ekonomska Fakulteta.
11. Kaptein, Rianne. 2012. Using Wordclouds to Navigate and Summarize Twitter Search Results. *CEUR Workshop Proceedings* (909): 67–70.
12. Python Software Foundation. 2017. *Lemmagen 1.2.0*. Dostopno prek: https:pypi.python.org/pypi/Lemmagen (8. julij 2017).
13. Han, Jiawei in Micheline Kamber. 2000. *Data Mining: Concepts and Techniques*. Dostopno prek: http://www.cs.wmich.edu/~yang/teach/cs595/han/ch01.pdf (10. maj 2017).

14. Hyun, Jung La in Kim Soo Dung. 2010. *Balanced MVC Architecture for Developing Service-based Mobile Applications*. Dostopno prek: http://ieeexplore.ieee.org.nukweb.nuk.uni-lj.si/stamp/stamp.jsp?tp=&arnumber=5704330 (13. Julij 2017).

15. Piña-García, C. A., Dongbing Gu in Mario Siqueiros-García. 2016. Towards a standard sampling methodology on online social networks: collecting global trends on Twitter. *Applied Network Science* 1 (1): 1–19.

16. Plisson, Joel, Nada Lavrač, Dunja Mladenić in Tomaž Erjavec. 2008. Ripple Down Rule learning for automated word lemmatisation. *AI Communications* (21): 15–26.

17. Plotly. 2017. *Plotly.js*. Dostopno prek: https://plot.ly/javascript/ (20. april 2017).

18. *Python*. Dostopno prek: https://www.python.org/ (26. junij 2017).

19. SQLite. 2017. *About SQLite*. Dostopno prek: https://www.sqlite.org/about.html (26. april 2017).

20. Tufte, Edward Rolfe. 2009. *The visual display of quantitative information*. 2nd edition, 6th printing. Cheshire (Connecticut): Graphics.

21. *Tweepy*. Dostopno prek: http://www.tweepy.org (19. marec 2017).

22. *Twitter, Inc*. Dostopno prek: https://twitter.com (20. marec 2017).

23. Twitter, Inc. 2017. *Twitter Developer Documentation*. Dostopno prek: https://dev.twitter.com/overview/api/tweets (20. marec 2017).

24. Valicon. 2016. *Uporaba družbenih omrežij v Sloveniji v številkah*. Dostopno prek: http://www.valicon.net/files/Sporocilo%20za%20javnost%202016-06-23%20(1).pdf (26. junij 2017).

25. Witten, Ian H, Eibe Frank in Mark A. Hall. 2011. *Data Mining: Practical Machine Learning Tools and Techniques*. Third Edition. Massachusetts: Morgan Kaufmann Publishers.

26. Benny Bottema. 2017. *Showcase your skillset with an interactive colorful D3.js tag cloud*. Dostopno prek: http://www.bennybottema.com/2016/03/02/showcase-your-skillset-with-an-interactive-colorful-d3-js-tag-cloud (5. maj 2017).

PRILOGE

PRILOGA A: KODA – ZBIRANJE TVITOV

```
from tweepy import Stream
from tweepy import OAuthHandler
from tweepy.streaming import StreamListener
import time
import datetime

ckey= '**********'
csecret= '**********'
atoken= '**********'
asecret= '**********'

timeout420 = 60

class listener(StreamListener):

    def on_data(self, data):
        try:
            fn= 'twitDB-'+datetime.date.today().strftime("%Y-%m-%d")+'.txt'
            saveFile = open(fn, 'a')
            saveFile.write(data)
            saveFile.write('\n')
            saveFile.close()
            return True
        except BaseException:
            #print('Napaka')
            time.sleep(5)
        global timeout420
        timeout420 = 60

    def on_error(self, status):
        print(status)
        if status == 420:
            global timeout420
            time.sleep(timeout420)
            timeout420 = timeout420*2
        else:
            time.sleep(320)

media_list = [...] # seznam vseh uporabnikov, ki jim skripta sledi

while True:
    try:
        auth = OAuthHandler(ckey, csecret)
        auth.set_access_token(atoken, asecret)
        twitterStream = Stream(auth, listener())
        twitterStream.filter(follow=media_list)
    except BaseException as e:
        with open("dnevnikNapak.txt","a") as errFile:
            errFile.write("{}\n{}\n\n".format(time.ctime(), str(e)))
```

PRILOGA B: KODA – MODEL IN ATRIBUTI

```
from django.db import models

class Tweet(models.Model):
    tweet_text_orig = models.CharField(max_length=500)
    tweet_text_lem = models.CharField(max_length=500)
    tweet_text_stop = models.CharField(max_length=500)
    tweet_date = models.DateField('date tweet published')
    tweet_user = models.CharField(max_length=100)
```

PRILOGA C: KODA – VNOS TVITOV V BAZO

```
from django.core.management.base import BaseCommand
from tweet.models import Tweet
import datetime as dt
from datetime import datetime, timedelta
import re
import lemmagen.lemmatizer
from lemmagen.lemmatizer import Lemmatizer
import time
import schedule
import json
import os
from urllib.request import urlopen
from pytz import timezone

class Command(BaseCommand):
    args = '<foo bar>'
    help = 'help string'

    def _create_tweets(self):

        baseurl = 'http://webales.fdv.uni-lj.si/TwitterAnze/'
        yesterday = str(dt.date.today() - timedelta(days=1))
        fn= 'twitDB-'+yesterday+'.txt'
        url = baseurl+fn

        print('start', url)
        filename = urlopen(url)

        for line in filename:
            line = line.decode('ISO-8859-1')
            line = line.strip()
            if line:
                try:

                    json_load = json.loads(line)

                    # parsamo tekst tvita
                    if 'text' in json_load:
                        t_text = json_load[u'text']
                        text_list = []
                        text_list.append(t_text)
                        if any('\n' in s for s in text_list):
                            text_list = [w.replace('\n', ' ') for w in
text_list]
                        tweet_text = str(text_list)
                        if 'https:\/\/' in tweet_text:
```

```
                            tweet_text = tweet_text.split('https:\/\/')[0]
                        elif 't.co' in tweet_text:
                            tweet_text = tweet_text.split('https://t.co')[0]
                        tweet_text_orig = tweet_text.lower()[2:-2]
                        tweet_text_re = re.findall(r'\w+', tweet_text)

                        lemmatizer =
Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
                        tweet_text_lem = ''

                        for word in tweet_text_re:
                            tweet_text_lem += ' '
                            tweet_text_lem += lemmatizer.lemmatize(word).lower()


                        stopwords = [...] # seznam z vsemi 'stopwords' besedami
                        for stopword in stopwords:
                            while stopword in tweet_text_re:
                                tweet_text_re.remove(stopword)

                        tweet_text_stop = ''
                        for word in tweet_text_re:
                            tweet_text_stop += ' '
                            tweet_text_stop += word.lower()

                        # parsamo čas
                        tweet_time = json_load['created_at']
                        tweet_time = time.strftime('%Y-%m-%d
%H:%M:%S',time.strptime(tweet_time, '%a %b %d %H:%M:%S +0000 %Y'))
                        tweet_time = datetime.strptime(tweet_time, "%Y-%m-%d
%H:%M:%S")
                        # nastavimo lokalni čas
                        slo = timezone('Europe/Ljubljana')
                        utc = timezone('UTC')
                        tweet_time = utc.localize(tweet_time)
                        tweet_time = tweet_time.astimezone(slo)

                        #parsamo uporabnika
                        tweet_user = json_load['user']['screen_name']

                        #shranimo
                        t = Tweet(tweet_user=tweet_user,
tweet_date=datetime.date(tweet_time), tweet_text_orig=tweet_text_orig,
tweet_text_lem=tweet_text_lem, tweet_text_stop=tweet_text_stop)
                        t.save()

                except Exception as e:
                    print('!error!: ',e)
        print('end', url, '\n', filename)

    def handle(self, *args, **options):
            self._create_tweets()

schedule.every().day.at("00:43").do(Command().handle)

while True:
    schedule.run_pending()
    time.sleep(1)
```

PRILOGA Č: KODA – POIZVEDBA IN PRIPRAVA PODATKOV ZA VIZUALIZACIJO

```
from django.shortcuts import render, render_to_response
from django.http import HttpResponse
from tweet.models import Tweet
from datetime import datetime, timedelta
import datetime as dt
import re
from collections import Counter, defaultdict
import lemmagen.lemmatizer
from lemmagen.lemmatizer import Lemmatizer
import operator
import functools
from django.db.models import Q
from django.utils import timezone
from django.views.decorators.cache import cache_page

def index(request):

    try:
        tweet_count = 0
        words = ''
        users = []
        top5 = []
        top = {}
        dates = []
        datesXusers = defaultdict(list)
        enddate = dt.date.today() - timedelta(days = 1)
        startdate = enddate - timedelta(days = 6)

        for tweet in Tweet.objects.filter(tweet_date__range=[startdate,
enddate]):
            # število vseh tvitov
            tweet_count += 1
            # top users
            users.append(tweet.tweet_user)
            # timeseries
            dates.append(str(tweet.tweet_date))
            if tweet.tweet_user not in datesXusers[str(tweet.tweet_date)]:
                datesXusers[str(tweet.tweet_date)].append(tweet.tweet_user)
            # wordcloud
            words += ' '
            words += str(tweet.tweet_text_stop)

        # top users
        top_users = Counter(users)
        for key in (top_users):
            top[top_users[key]] = key
        user1 = sorted(top, reverse=True)[:1]
        for key in (sorted(top, reverse=True)[:5]):
            i = []
            i.append(top[key])
            i.append(key)
            i.append(100*int(key)//int(sorted(top, reverse=True)[0]))
            top5.append(i)

        # timeseries
        dates_counted = Counter(dates)
```

```
        x = []
        trace1 = []
        trace2 = []
        for key in sorted(dates_counted):
            x.append(key)
            trace1.append(dates_counted[key])
        for key in sorted(datesXusers):
            trace2.append(len(datesXusers[key]))

        # wordcloud
        tweet_text_words = re.findall(r'\w+', words)

        lemmatizer = Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
        lem_words = []
        lem_words_count = defaultdict(list)
        for word in tweet_text_words:
            lem_words.append(lemmatizer.lemmatize(word))
            lem_words_count[lemmatizer.lemmatize(word)].append(word)

        word_count = Counter(lem_words).most_common(30)
        word_count_sum = 0
        for word in word_count:
            word_count_sum += word[1]

        tweet_text_list = []
        for word in word_count:
            word_dict = {}
            word_dict['size'] = word[1]*2000//word_count_sum
            word_dict['text'] =
Counter(lem_words_count[word[0]]).most_common(1)[0][0]
            tweet_text_list.append(word_dict)

        return render_to_response('index.html', {'tweet_count': tweet_count,
'top5': top5, 'x': x, 'trace1': trace1, 'trace2': trace2, 'top5': top5,
'tweet_text_list': tweet_text_list, 'word_count': word_count})

    except Exception:
        return render_to_response('db-update.html')


def results(request):

    try:
        tweet_count = 0
        words = ''
        users = []
        top5 = []
        top = {}
        dates = []
        datesXusers = defaultdict(list)

        query = request.GET.get('query')
        query_words = re.findall(r'\w+', query)
        query_lem = []
        lemmatizer = Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
        for word in query_words:
            query_lem.append(lemmatizer.lemmatize(word).lower())

        for tweet in Tweet.objects.filter(functools.reduce(operator.and_,
(Q(tweet_text_lem__icontains=x) for x in query_lem))):
            # število vseh tvitov
```

```
            tweet_count += 1
            # top users
            users.append(tweet.tweet_user)
            # timeseries
            dates.append(str(tweet.tweet_date))
            if tweet.tweet_user not in datesXusers[str(tweet.tweet_date)]:
                datesXusers[str(tweet.tweet_date)].append(tweet.tweet_user)
            # wordcloud
            words += ' '
            words += str(tweet.tweet_text_stop)

        # top users
        top_users = Counter(users)
        for key in (top_users):
            top[top_users[key]] = key
        user1 = sorted(top, reverse=True)[:1]
        for key in (sorted(top, reverse=True)[:5]):
            i = []
            i.append(top[key])
            i.append(key)
            i.append(100*int(key)//int(sorted(top, reverse=True)[0]))
            top5.append(i)

        # timeseries
        dates_counted = Counter(dates)
        x = []
        trace1 = []
        trace2 = []
        for key in sorted(dates_counted):
            x.append(key)
            trace1.append(dates_counted[key])
        for key in sorted(datesXusers):
            trace2.append(len(datesXusers[key]))

        # wordcloud
        tweet_text_words = re.findall(r'\w+', words)

        lemmatizer = Lemmatizer(dictionary=lemmagen.DICTIONARY_SLOVENE)
        lem_words = []
        lem_words_count = defaultdict(list)
        for word in tweet_text_words:
            lem_words.append(lemmatizer.lemmatize(word))
            lem_words_count[lemmatizer.lemmatize(word)].append(word)

        word_count = Counter(lem_words).most_common(30)
        word_count_sum = 0
        for word in word_count:
            word_count_sum += word[1]

        tweet_text_list = []
        for word in word_count:
            word_dict = {}
            word_dict['size'] = word[1]*2000//word_count_sum

            word_dict['text'] =
Counter(lem_words_count[word[0]]).most_common(1)[0][0]
            tweet_text_list.append(word_dict)

        return render_to_response('results.html', {'tweet_count': tweet_count,
'top5': top5, 'x': x, 'trace1': trace1, 'trace2': trace2, 'top5': top5,
'tweet_text_list': tweet_text_list, 'word_count': word_count, 'query': query})
```

```
    except Exception:
        return render_to_response('results-neg.html')
```

PRILOGA D: KODA – VIZUALIZACIJA ŠTEVILA TVITOV IN UPORABNIKOV SKOZI ČAS

```
  <script>
    var trace1 = {
        y: {{ trace1|safe }},
        x: {{ x|safe }},
        type: 'scatter',
        name: 'Število tvitov',
        marker: {
          color: '#226666',
        }
    };
    var trace2 = {
        y: {{ trace2|safe }},
        x: {{ x|safe }},
        type: 'scatter',
        name: 'Število uporabnikov',
        marker: {
          color: '#aa6c39',
        }
    };
    var layout = {
        title: "",
        showlegend: true,
        annotations: {
          color: 'black',
        },
      };
    var data = [trace2, trace1];
    Plotly.newPlot('time-series', data, layout, {displayModeBar: false});
  </script>
```

PRILOGA E: KODA – VIZUALIZACIJA WORDCLOUD

```
<script>

      var minfont = 8;
      var maxfont = 80;

      var width = 800;
      var height = 400;

      var fill = d3.scaleOrdinal()
        .range(["#226666", "#aa6c39", "#0d4d4d", "#804515", "#003333",
"#552600", "407f7f", "#d49a6a"]);

      var MAX_TRIES = (width > 400) ? 6 : 3;

      generateSkillCloud();

      function generateSkillCloud(retryCycle) {
          var wordsToDraw = {{ tweet_text_list|safe }};;
          d3.layout.cloud()
```

```
            .size([width, height])
            .words(wordsToDraw)
            .rotate(function() {
                return ~~(Math.random() * 0) * 90;
            })
            .font("Impact")
            .fontSize(function(d) {
                return d.size;
            })
            .on("end", function(fittedWords) {
                if (fittedWords.length == wordsToDraw.length) {
                    drawSkillCloud(fittedWords);
                }
                else if (!retryCycle || retryCycle < MAX_TRIES) {
                    generateSkillCloud((retryCycle || 1) + 1);
                }
                else {
                    drawSkillCloud(fittedWords);
                }
            })
            .start();

        function toFontSize(years, relevancy, retryCycle) {
            var lineairSize = (((years - minyears) / (maxyears - minyears)) *
(maxfont - minfont) * relevancy) + minfont;
            var polarizedSize = Math.pow(lineairSize / 8, 3);
            var reduceSize = polarizedSize * ((MAX_TRIES - retryCycle) /
MAX_TRIES);
            return ~~reduceSize;
        }

        function drawSkillCloud(words) {
            d3.select("#cloud svg").remove();
            d3.select("#cloud").append("svg")
                .attr("width", width)
                .attr("height", height)
                .append("g")
                .attr("transform", "translate(" + ~~(width / 2) + "," +
~~(height / 2) + ")")
                .selectAll("text")
                .data(words)
                .enter().append("text")
                .style("font-size", function(d) {
                    return d.size + "px";
                })
                .style("-webkit-touch-callout", "none")
                .style("-webkit-user-select", "none")
                .style("-khtml-user-select", "none")
                .style("-moz-user-select", "none")
                .style("-ms-user-select", "none")
                .style("user-select", "none")
                .style("cursor", "default")
                .style("font-family", "Impact")
                .style("fill", function(d, i) {
                    return fill(i);
                })
                .attr("text-anchor", "middle")
                .attr("transform", function(d) {
                    return "translate(" + [d.x, d.y] + ")rotate(" + d.rotate +
")";
                })
```

```
                .text(function(d) {
                    return d.text;
                });

            var svg = document.getElementsByTagName("svg")[0];
            var bbox = svg.getBBox();
            var viewBox = [bbox.x, bbox.y, bbox.width, bbox.height].join(" ");
            svg.setAttribute("viewBox", viewBox);
        }
    }
</script>
```