

**Универзитет Св. Кирил и Методиј-Скопје**

**Факултет за Електротехника и информациски технологии - ФЕИТ**

**Предмет:**

**Мрежно програмирање**

**Тема:**

**Twitter client за поврзување со Twitter, превземање на твитови, нивно процесирање и враќање одговор**

Ментори: Изработиле:

Проф. д-р Марко Порјазоски Жанета Тренчева Проф. д-р Марија Календар Симона Симоновскa

**Содржина**

1. Twitter API..........................................................................................................................3
   1. TwitterAppFeit ...................................................................................................................3
2. Автентикација за Twitter...................................................................................................3
   1. Регистрирање на апликацијата за Twitter API.................................................................3
   2. Барања за автентикација...................................................................................................4
3. Twitter клиент.....................................................................................................................4
4. Превземање твитови..........................................................................................................6
5. Испраќање одговор............................................................................................................7
6. Главна програма.................................................................................................................7
7. Ограничувања.....................................................................................................................8
   1. Ограничувања на ратата....................................................................................................8
   2. Twitter streaming API........................................................................................................10

Користена литература......................................................................................................11

1. **Twitter API**

Twitter API обезбедува пристап до сите функции коишто би сакале еден Twitter клиент да ги користи. Со Twitter API можеме да создадеме клиенти што ќе пребаруваат актуелни твитови (според hashtag, mention и сл.), откриваат корисници и ги следат нивните профили, а дури и да пишуваат твитови во име на корисник којшто е автентициран и да испраќаат пораки до други корисници.

Во овој проект ќе користиме Twitter API везија 1.1.

* 1. **TwitterAppFeit**

За да ги илустрираме некои од функционалностите на Twitter API, направивме едноставна апликација. Нашата апликација пребарува твитови по одреден hashtag, го филтрира текстот и корисничкото име од нив и генерира нов твит како одговор на твит по избор на корисникот на апликацијата.

1. **Автентикација за Twitter**

Пред да почнеме со програмата во Python, треба да одредиме на кој начин ќе се врши автентикацијата. Треба да се регистрираме и потоа да увидиме како Twitter очекува да ги автентицираме нашите барања.

* 1. **Регистрирање на апликацијата за Twitter API**

Треба да направиме Тwitter account, да ја регистрираме нашата апликација преку тој account и потоа ќе ги добиеме акредитивите (credentials) за автентикација за нашата апликација.

Креираме account на следниов линк: <http://www.twitter.com>. Со веќе постоечки account, можеме да ја регистрираме апликацијата следејќи ги следниве чекори:

1. Најавување на <http://apps.twitter.com> со нашиот Twitter account и креирање нова апликација
2. Пополнување на формата за нова апликација (името на апликацијата мора да биде уникатно)
3. Во подесувањата на апликацијата да се дозволат пермисии за читање и пишување. Доколку апликацијата користи испраќање одговор на твит, треба да се регистрира и телефонски број.

Следно е добивање на акредитивите за пристап:

1. Во делот Keys and Access Tokens се наоѓаат Consumer Key и Access Secret.
2. Генерирање на Access Token.
3. Access Token и Access Secret ќе се користат во апликацијата.
   1. **Барања за автентикација**

Сега ги имаме сите информации за барањата за автентикација. Twitter користи стандард за автентикација наречен oAuth, верзија 1.0a. Детално е опишан на: <http://oauth.net/core/1.0a/>.

Автентикацискиот стандард oAuth може да биде незгоден за користење, но за среќа Requests модулот има придружна библиотека наречена requests-oauthlib, која може да се справи со комплексноста на стандардот. Библиотеката е достапна на PyPi, па можеме да ја превземеме и инсталираме со pip.

***$ pip install requests-oauthlib***

***Downloading/unpacking requests-oauthlib***

***...***

Сега ни е овозможено да додадеме автентикација на нашите барања и да ја пишуваме нашата апликација.

1. **Twitter клиент**

На почеток ги верифицираме акредитивите. На местото на полињата <CONSUMER\_KEY>, <CONSUMER\_SECRET>, <ACCESS\_TOKEN> и <ACCESS\_SECRET> треба да се внесат добиените вредности од претходно регистрираната апликација на Twitter API. Тие се уникатни за секоја апликација.

***import requests, requests\_oauthlib, sys, json***

***consumer\_key = '<CONSUMER\_KEY>'***

***consumer\_secret = '<CONSUMER\_SECRET>'***

***access\_token = <ACCESS\_TOKEN>'***

***access\_secret = '<ACCESS\_SECRET>'***

***def init\_auth():***

***auth\_obj = requests\_oauthlib.OAuth1(***

***consumer\_key, consumer\_secret,***

***access\_token, access\_secret)***

***if verify\_credentials(auth\_obj):***

***print('Validated credentials OK')***

***return auth\_obj***

***else:***

***print('Credentials validation failed')***

***sys.exit(1)***

***def verify\_credentials(auth\_obj):***

***url = 'https://api.twitter.com/1.1/' \***

***'account/verify\_credentials.json'***

***response = requests.get(url, auth=auth\_obj)***

***return response.status\_code == 200***

Consumer secret и access secret играат улога на лозинка за нашиот Twitter account, па добра практика е да бидат превземени од безбедна надворешна локација наместо да бидат hard-code-ирани во кодот.

Во горенаведениот код, креираме автентикациска инстанца од OAuth1, auth\_obj, во функцијата init\_auth() користејќи ги нашите акредитиви за пристап. Инстанцата ја предаваме на Requests кога треба да испратиме HTTP барање и преку неа Requests се справува со автентикацијата. Пример за ова е функцијата verify\_credentials().

Во verify\_credentials() тестираме дали Twitter ги препознава нашите акредитиви. URL-то што го користиме овде е крајна точка што Twitter ја обезбедува чисто за тестирање дали нашите акредитиви се валидни. Враќа статусен код HTTP 200 ако се валидни или 401 ако не се.

Откако ќе се увериме дека апликацијата функционира правилно (добиеме порака Validated credentials OK), продолжуваме со превземање на твитови.

1. **Превземање твитови**

Следно ќе додадеме функција get\_hashtag() после фунцијата verify\_credentials() која ќе ја пребарува почетната страница на нашиот Twitter account за твитови со одреден хаштаг. Во листата на превземени модули (imports) треба да се додаде json.

***def get\_hashtag(URL,since\_id, auth\_obj):***

***params = {'count': 200, "since\_id":since\_id,***

***'include\_rts': 0, 'include\_entities': 'true'}***

***url = URL***

***response = requests.get(url, params=params, auth=auth\_obj)***

***response.raise\_for\_status()***

***return json.loads(response.text)***

Со функцијата get\_hashtag() пребаруваме и ги превземаме сите твитови со одреден hashtag поврзувајќи се со крајната точка search/tweets.json?q=%23<HASHTAG> (%23 се користи за прилепување на hashtag). Предаваме неколку параметри кои Requests ги предава како string за барањето. Овие параметри се специфицирани од Twitter и тие контролираат како твитовите ќе ни бидат вратени. Тие се следниве:

* ‘count’: Го специфицира максималниот број на твитови коишто ќе бидат вратени. Twitter дозволува да бидат примени 200 твитови на едно барање до крајната точка
* ‘include\_entities’: Ова се користи за отсекување на додатни информации од примените твитови
* ‘include\_rts’: Ова му кажува на Twitter да не вклучува re-твитови.
* ‘since\_id’: Ова му кажува на Twitter да ги врати само твитовите кои имаат ID поголемо од вредноста на since\_id. Секој твит има уникатно 64-битно ID и подоцнежните твитови имаат поголемо ID од претходните. Запомнувајќи го ID-то на последниот твит што сме го процесирале и пренесувајќи го како параметар, Twitter ќе ги исфилтрира твитовите коишто веќе сме ги виделе.

1. **Испраќање одговор**

За да испратиме твит како одговор, ќе ја употребиме функцијата post\_reply(). За ова, ќе ја користиме крајната точка statuses/update.json.

***def post\_reply(reply\_to\_id, text,auth\_obj):***

***params = {'status':text,'in\_reply\_to\_status\_id': reply\_to\_id}***

***url = 'https://api.twitter.com/1.1/statuses/update.json'***

***response = requests.post(url, params=params, auth=auth\_obj)***

***response.raise\_for\_status()***

Оваа функција ја повикува крајната точка користејќи ги следниве параметри за да го информира Twitter што да објави:

При тестирање на ова, можеме да добиеме грешки со статусен код 403. Тоа е во ред, се случува поради тоа што Twitter одбива да ни дозволи да објавиме два твита со идентичен текст по ред, што може да ни се случи.

* status: Ова е текстот на твитот што ќе се испрати како одговор
* in\_reply\_to\_status\_id: Ова е ID-то на твитот кому му испраќаме. Го испраќаме за Twitter да може да ги поврзе твитовите како разговор.

1. **Главна програма**

Во главната програма се повикуваат горенаведените функции.

***if \_\_name\_\_ == '\_\_main\_\_':***

***auth\_obj = init\_auth()***

***since\_id = 1***

***hashtag = input("Po koj hashtag sakas da prebaruvas tvitovi? -> ")***

***url = 'https://api.twitter.com/1.1/search/tweets.json?q=%23'+hashtag***

***for x in range(0, 3):***

***tweet=get\_hashtag(url,since\_id,auth\_obj)***

***username=tweet['statuses'][x]['user']['screen\_name']***

***message=tweet['statuses'][x]['text']***

***y=x+1***

***mes=str(y)+'. '+'@'+username+': '+message***

***print(mes)***

***brr=input("Na koj tvit sakas da odgovoris? Vnesi broj, za izlez vnesi -1")***

***br=int(brr)***

***while br!=-1:***

***username=tweet['statuses'][br-1]['user']['screen\_name']***

***repl = input("Sto sakas da odgovoris na "+username+"? -> ")***

***print(repl)***

***post\_reply(tweet['statuses'][br-1]['id'],repl,auth\_obj)***

***brr=input("Na koj tvit sakas da odgovoris? Vnesi broj, za izlez vnesi -1")***

***br=int(brr)***

Најпрво се повикува init\_auth() и вредноста што ја враќа функцијата се зачувува во промеливата auth\_obj. Потоа програмата побарува од нас да внесеме hashtag по кој сакаме да пребарува твитови. Програмата ни ги печати последните корисничките имиња и текстовите на последните 3 твита коишто го користат тој hashtag. Корисникот може да избере дали сака да врати на одреден твит и доколку сака, може да избере на кој/кои твит/ови. Со тоа ќе се објави твит со текстот кој го внел корисникот како reply на избраниот.

1. **Ограничувања**
   1. **Ограничувања на ратата**

Доколку ја извршиме програмата повеќе пати, можно е да престане да функционира. Или акредитивите привремено ќе престанат да бидат валидни или HTTP барањето во get\_hashtag() ќе престане да работи.

Ова се случува затоа што Twitter воведува ограничувања на ратата на неговото API, што значи дека нашата апликација смее да направи само одреден број на барања до некоја крајна точка во одреден временски период. Ограничувањата се наведени во документацијата на Twitter и тие варираат во зависност од автентикациската рута и од крајната точка. Ние користиме search/tweets крајна точка, па нашиот лимит е 180 барања на секои 15 минути. Доколку го надминеме овој лимит, Twitter ќе одговори со статусен код 429 Тоо many requests. Тоа би не ограничило да чекаме 15 минути за да нашата апликација биде повторно функционална.

Ограничувањата на ратата се заедничка карактеристика на сите web АPI-ја, па корисно е да се има начини на ефикасно тестирање кога се користат. Еден пристап на тестирање со податоци од API-ja со ограничување на ратата е да се превземат (download) поголем број податоци одеднаш и да се зачуваат локално. После ова, се земаат од датокетата наместо да се влечат од API-то со ново барање. Превземањето на податоци за тестирање користејќи го Python интерпретерот може да се изведе на следниов начин:

***>>> from twitterAppFeit import \****

***>>> auth\_obj = init\_auth()***

***Credentials validated OK***

***>>> hashtags = get\_hashtag(1, auth\_obj)***

***>>> json.dump(hashtags, open('test\_hashtags.json', 'w'))***

На овој начин се креира датотека по име test\_hashtags.json, која ги содржи нашите твитови по hashtag во JSON формат. Овде, json.dump() функцијата ги запишува добиените податоци во датотека наместо да ги врати како string.

Наместо да го повикуваме API-то, можеме да ги користиме овие податоци со мала модификација на главниот дел на нашата програма:

***if \_\_name\_\_ == '\_\_main\_\_':***

***hashtags = json.load(open('test\_hashtags.json'))***

* 1. **Twitter streaming API**

Постои проблемот дека клиентот може да извлече максимум 200 твита при едно барање. Во секое барање, Twitter ги враќа најновите твитови. Ова значи дека ако помеѓу две последователни барања се објават повеќе од 200 твитови, најпрво објавените трајно ќе ги изгубиме.

Решението на Twitter за овој проблем е да обезбеди алтернативен тип на API, кој е наречен streaming API. Кога ќе се поврземе на ваков тип API, HTTP конекцијата за одговор останува отворена и твитовите коишто непрестајно доаѓаат континуирано поминуваат низ конекцијата. Requests пакетот обезбедува добра функционалност за справување со ова. Објектите за одговор од Requests имаат метода iter\_lines(), која се извршува непрекинато. Таа е способна за прикажување податоци секогаш кога серверот ќе испрати, кои потоа ние можеме да ги процесираме. Како се користи ова може да се најде на следниов линк: http://docs.python-requests.org/en/latest/user/ advanced/#streaming-requests.

**Користена литература**

Learning Python Network Programming  M. O. - Faruque Sarker and Sam Washington

<https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets>

<https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json>