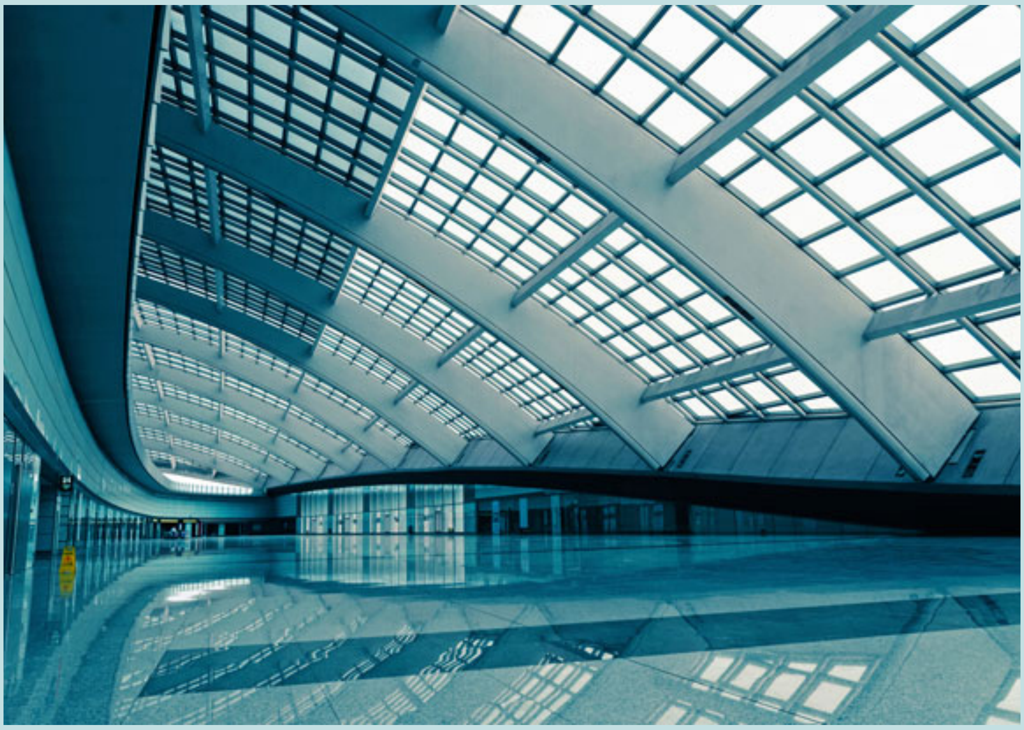
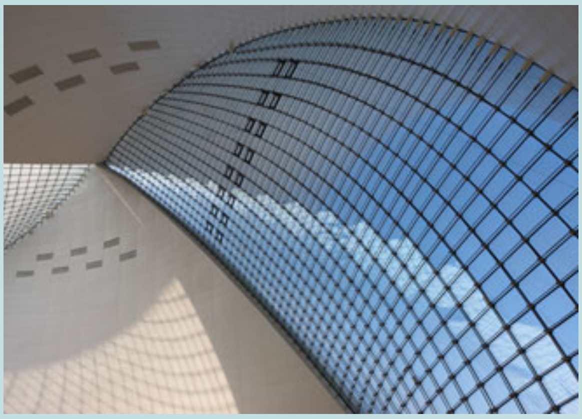
Data Analytics at Scale

Structured abstract

This report mainly analyzes the Twiiter data set(both hadoop 2014 and the data we fetched real time from twitter), finds out the hot activity and preference(like or hate) of the python’s 2 main web framework.

We sampled for one day in 2014 and the changes and comparisons of the sampled one day in 2021.

Table of contents

Introduction

Consider the professional attributes and your own preferences. I am more familiar with python and web development, so I want to study the changes of python-based web frameworks. Because twiiter's data set exceeds 2G in a single day(/data/ProjectDatasetTwitter), it is necessary to use big data cross-machine frameworks such as hadoop and spark.

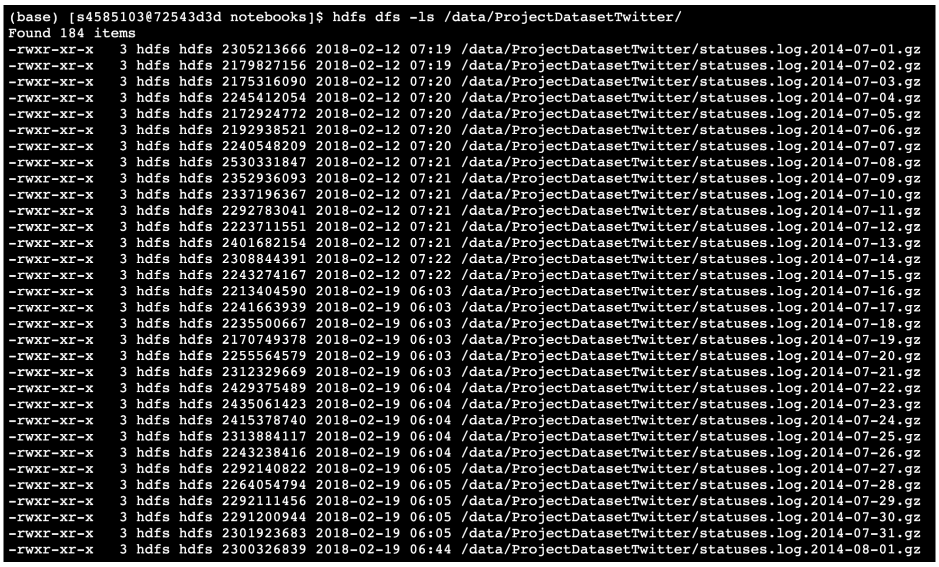
As the advantage of **hadoop** is that it is scalable, ie, any number of systems can be added at any point in time. It works on commodity hardware, so it is easy to keep costs low as compared to other databases. HDFS is mainly designed for large files, and it works on the concept of write once and read many times. In HDFS, individual files are broken into blocks of fixed size (typically 64MB) and stored across a cluster of nodes (not necessarily on the same machine). These files can be more than the size of an individual machine's hard drive. The individual machines are called data nodes. So we use hdfs to store all ProjectDatasetTwitter dataset.

Hadoop was the first open source system that introduced us to the MapReduce paradigm of programming . But **Spark** is the system that made it faster, much much faster(100x).There used to be a lot of data movement in Hadoop as it used to write intermediate results to the file system.This affected the speed at which you could do analysis. Spark provided us with an in-memory model, so Spark doesn't write too much to the disk while working. We use pyspark streaming to do our main analytics.

Then after we get result of our data exploration, we use **python**/js(**D3**)/**html5** to To integrate the results and display the data.

Dataset Analytics

step1: use hadoop command to observe:



```
(base) [s4585103@72543d3d notebooks]$ hdfs dfs -ls /data/ProjectDatasetTwitter/
Found 184 items
-rwxr-xr-x   3 hdfs hdfs 2305213666 2018-02-12 07:19 /data/ProjectDatasetTwitter/statuses.log.2014-07-01.gz
-rwxr-xr-x   3 hdfs hdfs 2179827156 2018-02-12 07:19 /data/ProjectDatasetTwitter/statuses.log.2014-07-02.gz
-rwxr-xr-x   3 hdfs hdfs 2175316090 2018-02-12 07:20 /data/ProjectDatasetTwitter/statuses.log.2014-07-03.gz
-rwxr-xr-x   3 hdfs hdfs 2245412054 2018-02-12 07:20 /data/ProjectDatasetTwitter/statuses.log.2014-07-04.gz
-rwxr-xr-x   3 hdfs hdfs 2172924772 2018-02-12 07:20 /data/ProjectDatasetTwitter/statuses.log.2014-07-05.gz
-rwxr-xr-x   3 hdfs hdfs 2192938521 2018-02-12 07:20 /data/ProjectDatasetTwitter/statuses.log.2014-07-06.gz
-rwxr-xr-x   3 hdfs hdfs 2240548209 2018-02-12 07:20 /data/ProjectDatasetTwitter/statuses.log.2014-07-07.gz
-rwxr-xr-x   3 hdfs hdfs 2530331847 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-08.gz
-rwxr-xr-x   3 hdfs hdfs 2352936093 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-09.gz
-rwxr-xr-x   3 hdfs hdfs 2337196367 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-10.gz
-rwxr-xr-x   3 hdfs hdfs 2292783041 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-11.gz
-rwxr-xr-x   3 hdfs hdfs 2223711551 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-12.gz
-rwxr-xr-x   3 hdfs hdfs 2401682154 2018-02-12 07:21 /data/ProjectDatasetTwitter/statuses.log.2014-07-13.gz
-rwxr-xr-x   3 hdfs hdfs 2308844391 2018-02-12 07:22 /data/ProjectDatasetTwitter/statuses.log.2014-07-14.gz
-rwxr-xr-x   3 hdfs hdfs 2243274167 2018-02-12 07:22 /data/ProjectDatasetTwitter/statuses.log.2014-07-15.gz
-rwxr-xr-x   3 hdfs hdfs 2213404590 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-16.gz
-rwxr-xr-x   3 hdfs hdfs 2241663939 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-17.gz
-rwxr-xr-x   3 hdfs hdfs 2235500667 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-18.gz
-rwxr-xr-x   3 hdfs hdfs 2170749378 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-19.gz
-rwxr-xr-x   3 hdfs hdfs 2255564579 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-20.gz
-rwxr-xr-x   3 hdfs hdfs 2312329669 2018-02-19 06:03 /data/ProjectDatasetTwitter/statuses.log.2014-07-21.gz
-rwxr-xr-x   3 hdfs hdfs 2429375489 2018-02-19 06:04 /data/ProjectDatasetTwitter/statuses.log.2014-07-22.gz
-rwxr-xr-x   3 hdfs hdfs 2435061423 2018-02-19 06:04 /data/ProjectDatasetTwitter/statuses.log.2014-07-23.gz
-rwxr-xr-x   3 hdfs hdfs 2415378740 2018-02-19 06:04 /data/ProjectDatasetTwitter/statuses.log.2014-07-24.gz
-rwxr-xr-x   3 hdfs hdfs 2313884117 2018-02-19 06:04 /data/ProjectDatasetTwitter/statuses.log.2014-07-25.gz
-rwxr-xr-x   3 hdfs hdfs 2243238416 2018-02-19 06:04 /data/ProjectDatasetTwitter/statuses.log.2014-07-26.gz
-rwxr-xr-x   3 hdfs hdfs 2292140822 2018-02-19 06:05 /data/ProjectDatasetTwitter/statuses.log.2014-07-27.gz
-rwxr-xr-x   3 hdfs hdfs 2264054794 2018-02-19 06:05 /data/ProjectDatasetTwitter/statuses.log.2014-07-28.gz
-rwxr-xr-x   3 hdfs hdfs 2292111456 2018-02-19 06:05 /data/ProjectDatasetTwitter/statuses.log.2014-07-29.gz
-rwxr-xr-x   3 hdfs hdfs 2291200944 2018-02-19 06:05 /data/ProjectDatasetTwitter/statuses.log.2014-07-30.gz
-rwxr-xr-x   3 hdfs hdfs 2301923683 2018-02-19 06:05 /data/ProjectDatasetTwitter/statuses.log.2014-07-31.gz
-rwxr-xr-x   3 hdfs hdfs 2300326839 2018-02-19 06:44 /data/ProjectDatasetTwitter/statuses.log.2014-08-01.gz
```

we can see: all twitter dataset is stored in /data/ProjectDatsetTwitter and it’s distributed as many .gz compressed files. Each compressed files is more than 2Gb sizes, it should contain ’s datetime’s tweets from twitter apis.

step2: read twitter api docs to get to know the structure and description of the dataset:

All Twitter APIs that return Tweets provide that data encoded using JavaScript Object Notation (JSON). JSON is based on key-value pairs, with named attributes and associated values. These attributes, and their state are used to describe objects.

At Twitter we serve many objects as JSON, including Tweets and Users. These objects all encapsulate core attributes that describe the object. Each Tweet has an author, a message, a unique ID, a timestamp of when it was posted, and sometimes geo metadata shared by the user. Each User has a Twitter name, an ID, a number of followers, and most often an account bio.
With each Tweet we also generate "entity" objects, which are arrays of common Tweet contents such as hashtags, mentions, media, and links. If there are links, the JSON payload can also provide metadata such as the fully unwound URL and the webpage’s title and description.
So, in addition to the text content itself, a Tweet can have over 150 attributes associated with it.

So each single record should look like following JSON , which illustrates the structure for these objects and some of their attributes:

```
{
  "created_at": "Thu Apr 06 15:24:15 +0000 2017",
  "id_str": "850006245121695744",
  "text": "1\/ Today we\u2019re sharing our vision for the future of the Twitter API platform!\nhttps:\/\/t.co\/XweGngmxlP",
  "user": {
    "id": 2244994945,
    "name": "Twitter Dev",
    "screen_name": "TwitterDev",
    "location": "Internet",
    "url": "https:\/\/dev.twitter.com\/",
    "description": "Your official source for Twitter Platform news,
updates & events. Need technical help? Visit https:\/\/
twittercommunity.com\/ \u2328\ufe0f #TapIntoTwitter"
  },
  "place": {
  },
  "entities": {
    "hashtags": [
    ],
    "urls": [
      {
        "url": "https:\/\/t.co\/XweGngmxlP",
        "unwound": {
          "url": "https:\/\/cards.twitter.com\/cards\/18ce53wgo4h\/
3xo1c",
          "title": "Building the Future of the Twitter API Platform"
        }
      }
    ],
    "user_mentions": [
    ]
  }
}
```

step3: we choose random day’s giz file from hadoop, use pyspark to get the sample data to explore:

inputFile = '/data/ProjectDatasetTwitter/statuses.log.2014-12-04.gz'

conf = SparkConf().setAppName("SparkSQLTwitter")

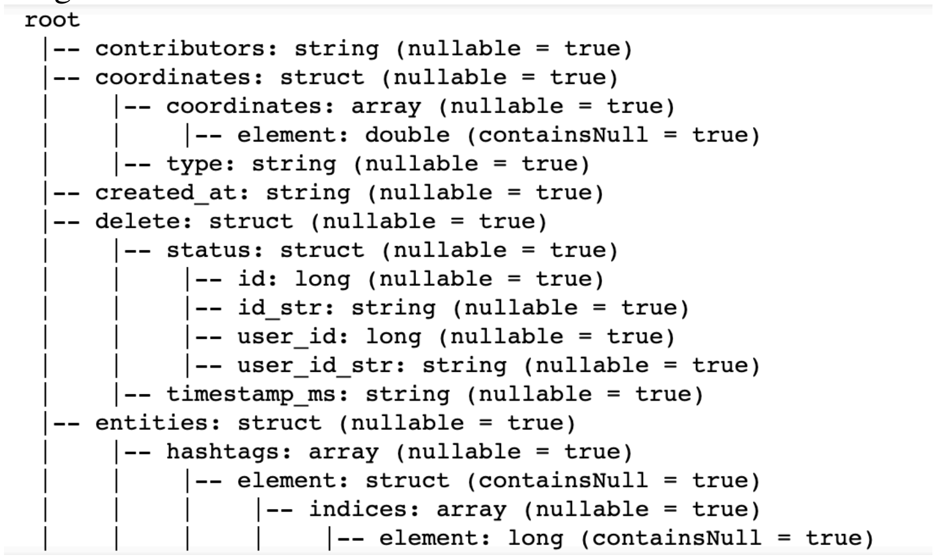
sc = SparkContext()

input = hiveCtx.read.json(inputFile)

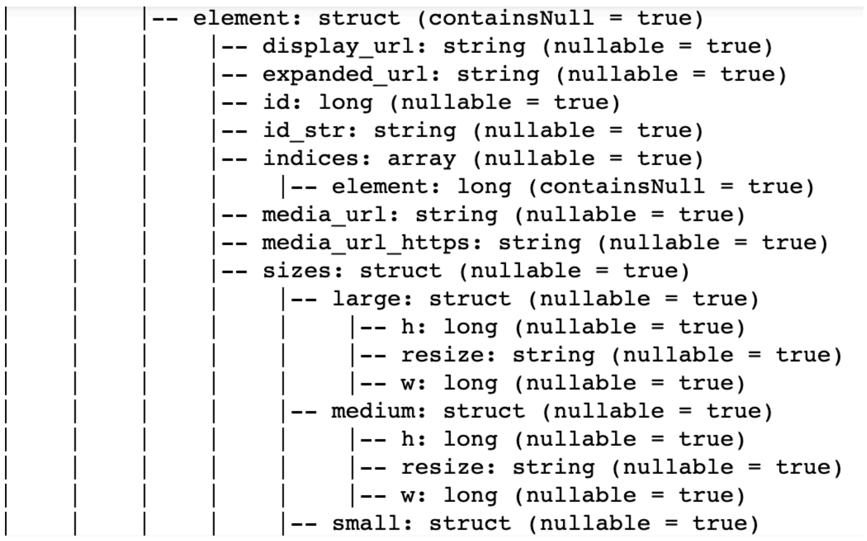
input.show(n=20, truncate=False)

input.printSchema()

we get real schema from data:



```
root
 |-- contributors: string (nullable = true)
 |-- coordinates: struct (nullable = true)
 |    |-- coordinates: array (nullable = true)
 |    |    |-- element: double (containsNull = true)
 |    |-- type: string (nullable = true)
 |-- created_at: string (nullable = true)
 |-- delete: struct (nullable = true)
 |    |-- status: struct (nullable = true)
 |    |    |-- id: long (nullable = true)
 |    |    |-- id_str: string (nullable = true)
 |    |    |-- user_id: long (nullable = true)
 |    |    |-- user_id_str: string (nullable = true)
 |    |-- timestamp_ms: string (nullable = true)
 |-- entities: struct (nullable = true)
 |    |-- hashtags: array (nullable = true)
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- indices: array (nullable = true)
 |    |    |    |    |-- element: long (containsNull = true)
```



```
 |    |    |-- element: struct (containsNull = true)
 |    |    |    |-- display_url: string (nullable = true)
 |    |    |    |-- expanded_url: string (nullable = true)
 |    |    |    |-- id: long (nullable = true)
 |    |    |    |-- id_str: string (nullable = true)
 |    |    |    |-- indices: array (nullable = true)
 |    |    |    |    |-- element: long (containsNull = true)
 |    |    |    |-- media_url: string (nullable = true)
 |    |    |    |-- media_url_https: string (nullable = true)
 |    |    |    |-- sizes: struct (nullable = true)
 |    |    |    |    |-- large: struct (nullable = true)
 |    |    |    |    |    |-- h: long (nullable = true)
 |    |    |    |    |    |-- resize: string (nullable = true)
 |    |    |    |    |    |-- w: long (nullable = true)
 |    |    |    |    |-- medium: struct (nullable = true)
 |    |    |    |    |    |-- h: long (nullable = true)
 |    |    |    |    |    |-- resize: string (nullable = true)
 |    |    |    |    |    |-- w: long (nullable = true)
 |    |    |    |    |-- small: struct (nullable = true)
```

sample record is:

row= Row(contributors=None, coordinates=None, created_at=None, delete=Row(status=Row(id=56428245058457600, id_str='56428245058457600', user_id=276700097, user_id_str='276700097'), timestamp_ms='1417647600084'), entities=None, extended_entities=None, favorite_count=None, favorited=None, filter_level=None, geo=None, id=None, id_str=None, in_reply_to_screen_name=None, in_reply_to_status_id=None, in_reply_to_status_id_str=None, in_reply_to_user_id=None, in_reply_to_user_id_str=None, lang=None, place=None, possibly_sensitive=None, retweet_count=None, retweeted=None, retweeted_status=None, scopes=None, source=None, text=None, timestamp_ms=None, truncated=None, user=None, withheld_in_countries=None)

step4: data clean

we want to see tweet with content , and related retweet_count, fovorite_count is not null.

So wo have to remove useless records from datatset:

```
 input.na.drop(subset=[“text"])

 input.na.drop(subset=[“retweet_count"])

input.na.drop.na.drop(subset=[“entities”])
```

step5: data explore

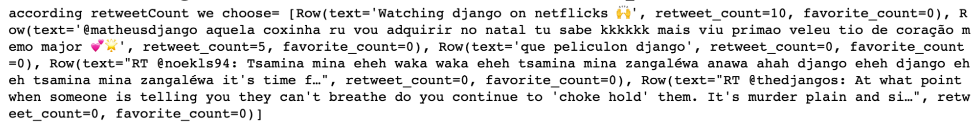
```
df.registerTempTable("tweets")

t = sqlContext.sql("SELECT distinct id,entities.hashtags FROM tweets").rdd

related_hashtags= t.map(lambda t: map(lambda t0: t0[2].lower(), t[1])) \

                .map(lambda t: list(itertools.combinations(t, 2))) \

                .flatMap(lambda t: t) \

                .map(lambda t: sorted(t)) \

                .map(lambda x: '\t'.join(unicode(i) for i in x)) \

                .map(lambda t: (t, 1)) \

                .reduceByKey(lambda x,y: x+y) \
                .filter(lambda t: t[1]>=min_occurs) \
                .sortByKey(False) \
                .map(lambda x: '\t'.join(unicode(i) for i in x)) \
                .repartition(1)
print('here', related_hashtags)
```

When the processing is finished , then we save some result from tweets to hdfs temperally

related_hashtags.saveAsTextFile("%s/%s" % ("/tmp/tests/00", “related_hashtags”))

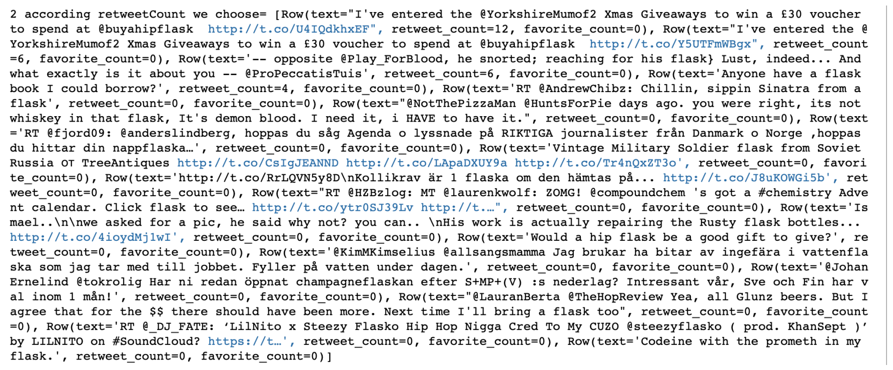
we explore the top 20 tweets by all angles: like favorite_count / retweet_count , something like that:

```
topTweets = hiveCtx.sql("SELECT text, retweet_count,favorite_count FROM tweets where text IS NOT NULL and \
                text like '%django%' LIMIT 20")
print('according retweetCount we choose=', topTweets.collect() )
```



```
according retweetCount we choose= [Row(text='Watching django on netflicks 🙌', retweet_count=10, favorite_count=0), R
ow(text='@matheusdjango aquela coxinha ru vou adquirir no natal tu sabe kkkkkk mais viu primao veleu tio de coração m
emo major 💕🌟', retweet_count=5, favorite_count=0), Row(text='que peliculon django', retweet_count=0, favorite_count
=0), Row(text="RT @noekls94: Tsamina mina eheh waka waka eheh tsamina mina zangaléwa anawa ahah django eheh django eh
eh tsamina mina zangaléwa it's time f…", retweet_count=0, favorite_count=0), Row(text="RT @thedjangos: At what point
when someone is telling you they can't breathe do you continue to 'choke hold' them. It's murder plain and si…", retw
eet_count=0, favorite_count=0)]
```

```
topTweets = hiveCtx.sql("SELECT text, retweet_count,favorite_count FROM tweets where text IS NOT NULL and \
                text like '%flask%' LIMIT 20")
print('2 according retweetCount we choose=', topTweets.collect() )
```



```
2 according retweetCount we choose= [Row(text="I've entered the @YorkshireMumof2 Xmas Giveaways to win a £30 voucher
to spend at @buyahipflask  http://t.co/U4IQdkhxEF", retweet_count=12, favorite_count=0), Row(text="I've entered the @
YorkshireMumof2 Xmas Giveaways to win a £30 voucher to spend at @buyahipflask  http://t.co/Y5UTFmWBgx", retweet_count
=6, favorite_count=0), Row(text='-- opposite @Play_ForBlood, he snorted; reaching for his flask} Lust, indeed... And
what exactly is it about you -- @ProPeccatisTuis', retweet_count=6, favorite_count=0), Row(text='Anyone have a flask
book I could borrow?', retweet_count=4, favorite_count=0), Row(text='RT @AndrewChibz: Chillin, sippin Sinatra from a
flask', retweet_count=0, favorite_count=0), Row(text="@NotThePizzaMan @HuntsForPie days ago. you were right, its not
whiskey in that flask, It's demon blood. I need it, i HAVE to have it.", retweet_count=0, favorite_count=0), Row(text
='RT @fjord09: @anderslindberg, hoppas du såg Agenda o lyssnade på RIKTIGA journalister från Danmark o Norge ,hoppas
du hittar din nappflaska…', retweet_count=0, favorite_count=0), Row(text='Vintage Military Soldier flask from Soviet
Russia от TreeAntiques http://t.co/CsIgJEANND http://t.co/LApaDXUY9a http://t.co/Tr4nQxZT3o', retweet_count=0, favori
te_count=0), Row(text='http://t.co/RrLQVN5y8D\nKollikrav är 1 flaska om den hämtas på... http://t.co/J8uKOWGi5b', ret
weet_count=0, favorite_count=0), Row(text="RT @HZBzlog: MT @laurenkwolf: ZOMG! @compoundchem 's got a #chemistry Adve
nt calendar. Click flask to see… http://t.co/ytr0SJ39Lv http://t.…", retweet_count=0, favorite_count=0), Row(text='Is
mael..\n\nwe asked for a pic, he said why not? you can.. \nHis work is actually repairing the Rusty flask bottles...
http://t.co/4ioydMj1wI', retweet_count=0, favorite_count=0), Row(text='Would a hip flask be a good gift to give?', re
tweet_count=0, favorite_count=0), Row(text='@KimMKimselius @allsangsmamma Jag brukar ha bitar av ingefära i vattenfla
ska som jag tar med till jobbet. Fyller på vatten under dagen.', retweet_count=0, favorite_count=0), Row(text='@Johan
Ernelind @tokrolig Har ni redan öppnat champagneflaskan efter S+MP+(V) :s nederlag? Intressant vår, Sve och Fin har v
al inom 1 mån!', retweet_count=0, favorite_count=0), Row(text="@LauranBerta @TheHopReview Yea, all Glunz beers. But I
agree that for the $$ there should have been more. Next time I'll bring a flask too", retweet_count=0, favorite_count
=0), Row(text='RT @_DJ_FATE: ‘LilNito x Steezy Flasko Hip Hop Nigga Cred To My CUZO @steezyflasko ( prod. KhanSept )’
by LILNITO on #SoundCloud? https://t…', retweet_count=0, favorite_count=0), Row(text='Codeine with the prometh in my
flask.', retweet_count=0, favorite_count=0)]
```

step6: We check the result , Exclude tweets with non-frame information with the same name (determined based on the context of the tweet sentence)， from anlysing we can see: in a random day in 2014, the django/flask Heat comparison of events： **15: 4**

step7: We use scrapy/python to download

athering tweets URL by searching through hashtags

For searching for tweets we will be using the legacy Twitter website. Let's try searching for :

mobile.twitter.com/hashtag/flask

mobile.twitter.com/hashtag/django

We want to use this legacy version for gathering tweets URLs as it loads the data without using javascript which makes our job easy.

The first parse() function bears the brunt of the task. First, items are introduced as lists, so that they can be used in the function.

Then, using the response.xpath() method from before to collect links, we collect every link on the page.

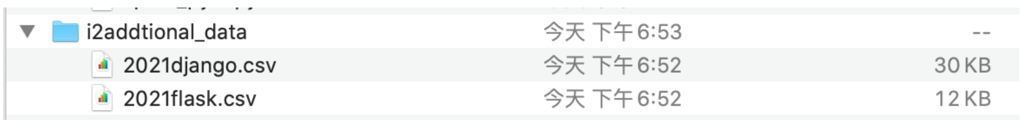
We then filter those links into twitterlink, which collects links that include "twitter.com," which can be truncated into handles, and domainlink, or links from that website, which we use to crawl the website for further handles.

Then, as a way of keeping a running list throughout the spider, we add twitterlink to twitterlinku, a cumulative list of Twitter links.

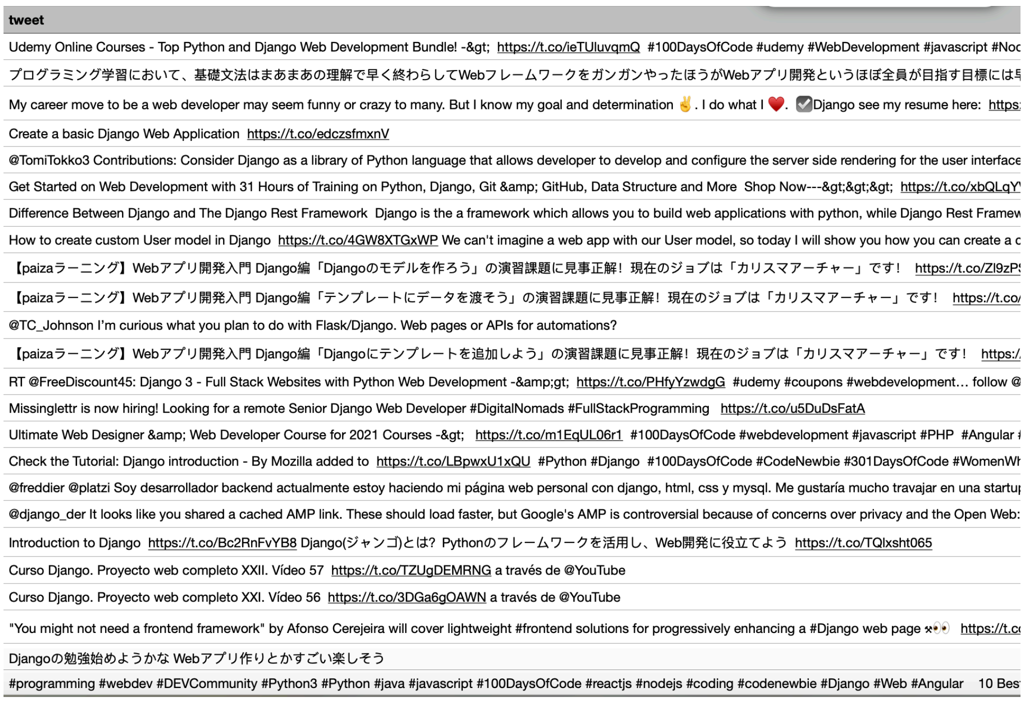
At this point, we change gears from scraping to crawling. Since in many cases scraping an entire website will take too much time and memory, we'll only do this to a secondary level: only the first page and any page that the first page links to will be scraped. This is useful for most instances where there is a list of sites that include Twitter links.

The method scrapy.Request() allows us to call parse2(). However, before we yield this request, we use .meta to ensure that the lists of items are maintained as the same throughout the process.

After all downloading is done, we got today's all tweets about "Flask/ Django", we save into 2 seperate csvs.



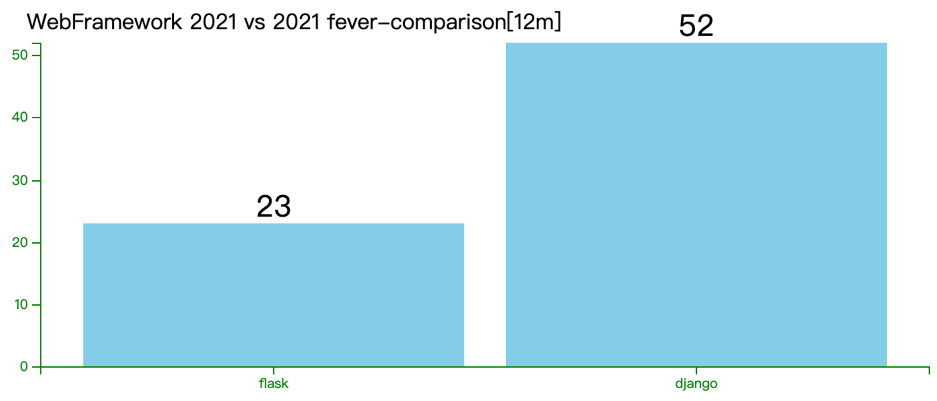
▼ i2addtional_data	今天 下午6:53	--
2021django.csv	今天 下午6:52	30 KB
2021flask.csv	今天 下午6:52	12 KB



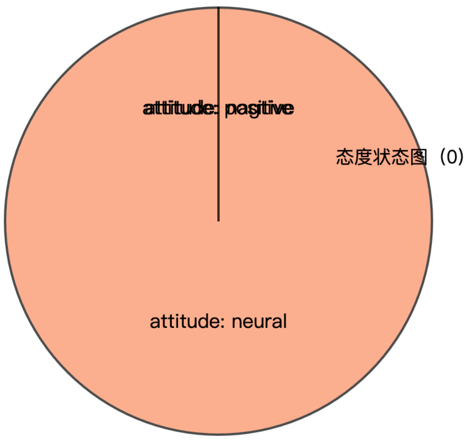
tweet
Udemy Online Courses - Top Python and Django Web Development Bundle! -&gt; https://t.co/ieTUluvqmQ #100DaysOfCode #udemy #WebDevelopment #javascript #Noc
プログラミング学習において、基礎文法はまあまあの理解で早く終わらしてWebフレームワークをガンガンやったほうがWebアプリ開発というほぼ全員が目指す目標には
My career move to be a web developer may seem funny or crazy to many. But I know my goal and determination ✌️. I do what I ❤️. ☑️Django see my resume here: https:
Create a basic Django Web Application https://t.co/edczsfmxnV
@TomiTokko3 Contributions: Consider Django as a library of Python language that allows developer to develop and configure the server side rendering for the user interface
Get Started on Web Development with 31 Hours of Training on Python, Django, Git &amp; GitHub, Data Structure and More Shop Now---&gt;&gt;&gt; https://t.co/xbQLqY
Difference Between Django and The Django Rest Framework Django is the a framework which allows you to build web applications with python, while Django Rest Framew
How to create custom User model in Django https://t.co/4GW8XTGxWP We can't imagine a web app with our User model, so today I will show you how you can create a c
【paizaラーニング】Webアプリ開発入門 Django編「Djangoのモデルを作ろう」の演習課題に見事正解！現在のジョブは「カリスマアーチャー」です！ https://t.co/Zl9zP
【paizaラーニング】Webアプリ開発入門 Django編「テンプレートにデータを渡そう」の演習課題に見事正解！現在のジョブは「カリスマアーチャー」です！ https://t.co/
@TC_Johnson I'm curious what you plan to do with Flask/Django. Web pages or APIs for automations?
【paizaラーニング】Webアプリ開発入門 Django編「Djangoにテンプレートを追加しよう」の演習課題に見事正解！現在のジョブは「カリスマアーチャー」です！ https:/
RT @FreeDiscount45: Django 3 - Full Stack Websites with Python Web Development -&gt; https://t.co/PHfyYzwdgG #udemy #coupons #webdevelopment… follow @
Missinglettr is now hiring! Looking for a remote Senior Django Web Developer #DigitalNomads #FullStackProgramming https://t.co/u5DuDsFatA
Ultimate Web Designer &amp; Web Developer Course for 2021 Courses -&gt; https://t.co/m1EqUL06r1 #100DaysOfCode #webdevelopment #javascript #PHP #Angular #
Check the Tutorial: Django introduction - By Mozilla added to https://t.co/LBpwxU1xQU #Python #Django #100DaysOfCode #CodeNewbie #301DaysOfCode #WomenWh
@freddier @platzi Soy desarrollador backend actualmente estoy haciendo mi página web personal con django, html, css y mysql. Me gustaría mucho travajar en una startu
@django_der It looks like you shared a cached AMP link. These should load faster, but Google's AMP is controversial because of concerns over privacy and the Open Web:
Introduction to Django https://t.co/Bc2RnFvYB8 Django(ジャンゴ)とは? Pythonのフレームワークを活用し、Web開発に役立てよう https://t.co/TQlxsht065
Curso Django. Proyecto web completo XXII. Vídeo 57 https://t.co/TZUgDEMRNG a través de @YouTube
Curso Django. Proyecto web completo XXI. Vídeo 56 https://t.co/3DGa6gOAWN a través de @YouTube
"You might not need a frontend framework" by Afonso Cerejeira will cover lightweight #frontend solutions for progressively enhancing a #Django web page ⚒👀 https://t.c
Djangoの勉強始めようかな Webアプリ作りとかすごい楽しそう
#programming #webdev #DEVCommunity #Python3 #Python #java #javascript #100DaysOfCode #reactjs #nodejs #coding #codenewbie #Django #Web #Angular 10 Bes

step8: We compare the downloaded yesterday's data heat comparation, and we use d3 to visualize it:

in 2021's ramdom day , the django/flask hot comparation: **23:52**



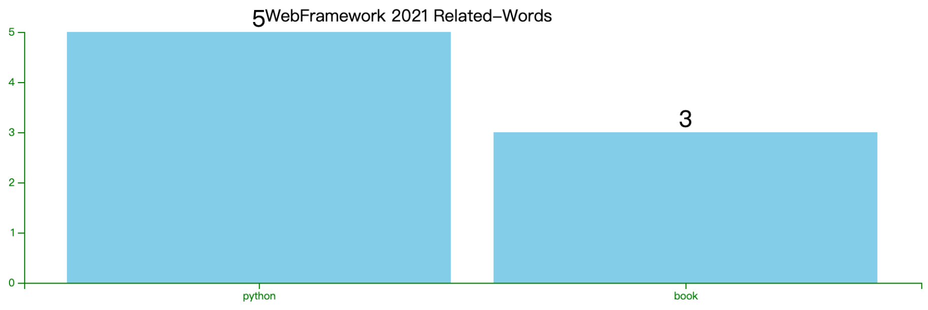
we all use maching learning to get to know people's attitude about these web framework:



most of twitter use is neural about the web framework.

Then we use python scripts to explore the related words:

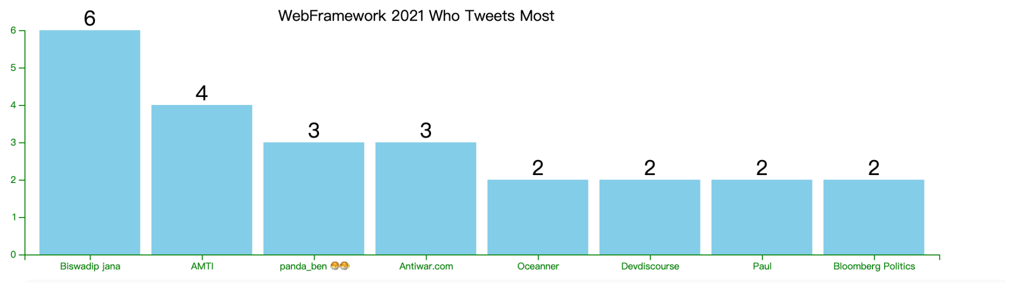
```
a1_sorted_keys = sorted(wordfreq, key=wordfreq.get,
reverse=True)

for r in a1_sorted_keys:

    if wordfreq[r] > 1:

        print(r, wordfreq[r])

        if index < 10:

            js_txt += "'" + r + "':" + str(wordfreq[r]) + ','

            index += 1
```



We can see: the most people tweet about the 2 framework is when you also tweet python or books .

We then use python scripts to check the most frequently tweets such subjects users, find out the most active twitter users in this field, and find out who are most concerned about flask/django keywords:

```
a2_sorted_keys = sorted(usernamefreq, key=usernamefreq.get, reverse=True)

for r in a2_sorted_keys:

    if usernamefreq[r] > 1:

        print(r, usernamefreq[r])

        if index < 10:

            js_txt += "'" + r + "':" + str(usernamefreq[r]) + ','

            index += 1
```



Discussion and conclusions of the analysis

From the aboving , we can see:

1. the flask/django's absolute popularity has risen. No matter which framework it is, the absolute number of people and news discussed in 2021 will increase compared with 2014. More people care about the python web framework. The total num from 19 jump to 75 in a random days.

2. The relative gap between flask and django has been greatly reduced, and the ratio has been reduced from "15: 4" to "52： 23" I digged the possible reasons: some of world-famous companies such as Airbnb and Reddit use Flask. Flask gives you more control over your project, since you can choose which components to use and how you interact with them. Also, you can plug in any extension you need.

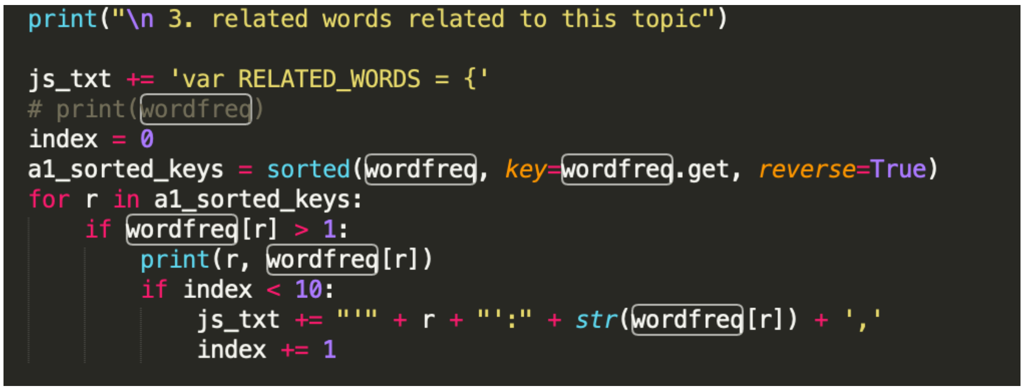
3. If the purpose is very clear, the demand for data processing capabilities can be greatly reduced. But in the exploratory stage, when the purpose is not clear, it is more advantageous to have the ability to process big data. In our case, the day of tweets is 2Gb, but tweets with 'flask/django' hastags in it, it's just less than 1Mb.

4. Use hadoop or local file system to store intermediate calculation results, which can be very convenient for subsequent visualization and participation of machine learning libraries. In our case , we use hadoop to save template hashtags , and use local csvs to save 2021's tweets data which compain 'flask/django hashtags, it helps the processing procedure.

Appendix

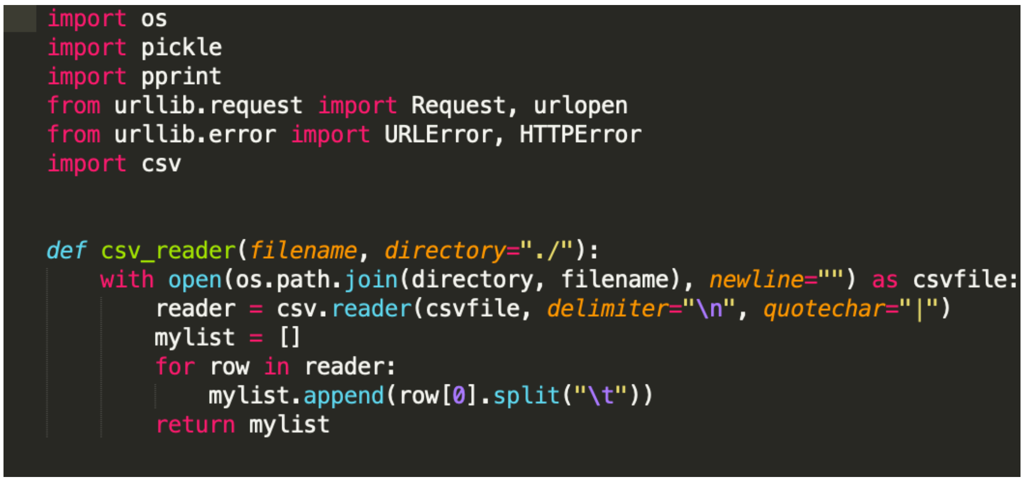
```python
import itertools
# int_num = df.count()
# print('total num=', int_num)
df.na.drop(subset=["entities"])

df.registerTempTable("tweets")
t = sqlContext.sql("SELECT distinct id,entities.hashtags FROM tweets").rdd


#related_hashtags= t.map(lambda t: map(lambda t0: t0[2], t[1])).map(lambda t: list(itertools.co
related_hashtags= t.map(lambda t: map(lambda t0: t0[2].lower(), t[1])) \
                     .map(lambda t: list(itertools.combinations(t, 2))) \
                     .flatMap(lambda t: t) \
                     .map(lambda t: sorted(t)) \
                     .map(lambda x: '\t'.join(unicode(i) for i in x)) \
                     .map(lambda t: (t, 1)) \
                     .reduceByKey(lambda x,y: x+y) \
                     .filter(lambda t: t[1]>=min_occurs) \
                     .sortByKey(False) \
                     .map(lambda x: '\t'.join(unicode(i) for i in x)) \
                     .repartition(1)
print('here', related_hashtags)
## save stats from tweets to hdfs
related_hashtags.saveAsTextFile("%s/%s" % ("/tmp/tests/00", "related_hashtags"))
```

```python
input.registerTempTable("tweets")
# topTweets = hiveCtx.sql("SELECT text, retweet_count,favorite_count FROM tweets where retweet_count!= NULL ORDER BY re
# print('according retweetCount we choose=', topTweets.collect() )
topTweets = hiveCtx.sql("SELECT text, retweet_count,favorite_count FROM tweets where text IS NOT NULL and \
                        text like '%django%' LIMIT 20")
print('according retweetCount we choose=', topTweets.collect() )
```

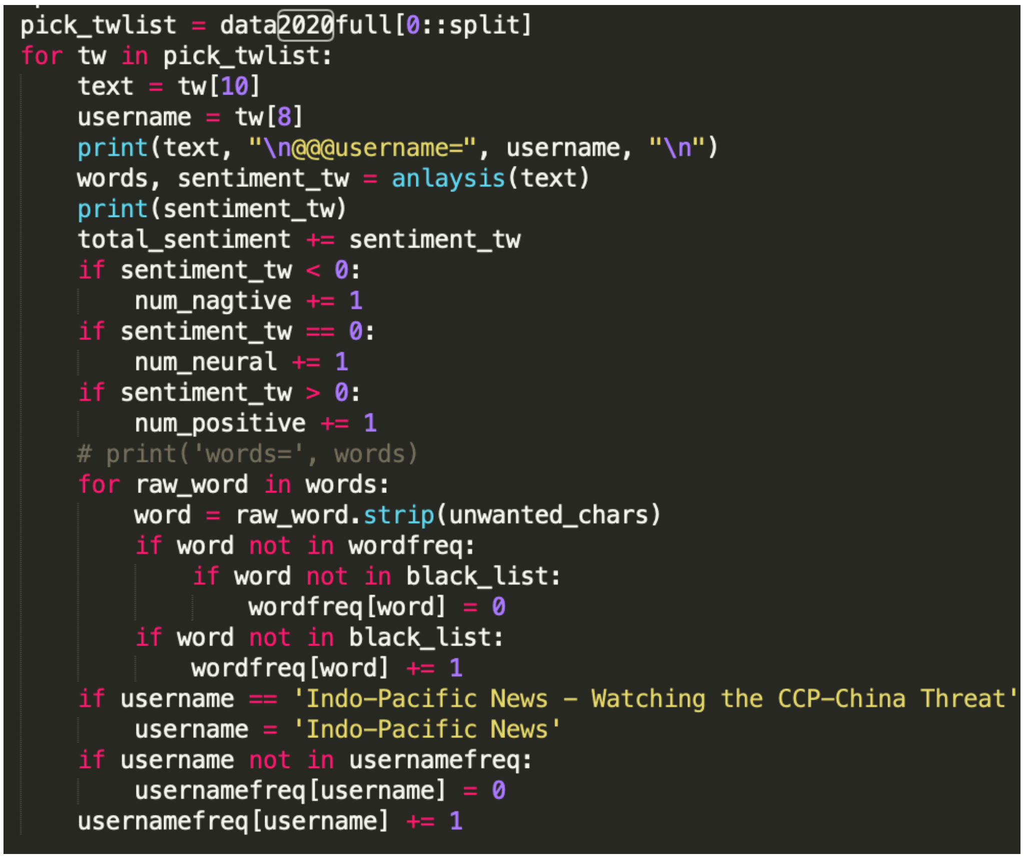


```python
print("\n 3. related words related to this topic")

js_txt += 'var RELATED_WORDS = {'
# print(wordfreq)
index = 0
a1_sorted_keys = sorted(wordfreq, key=wordfreq.get, reverse=True)
for r in a1_sorted_keys:
    if wordfreq[r] > 1:
        print(r, wordfreq[r])
        if index < 10:
            js_txt += "'" + r + "':" + str(wordfreq[r]) + ','
            index += 1
```

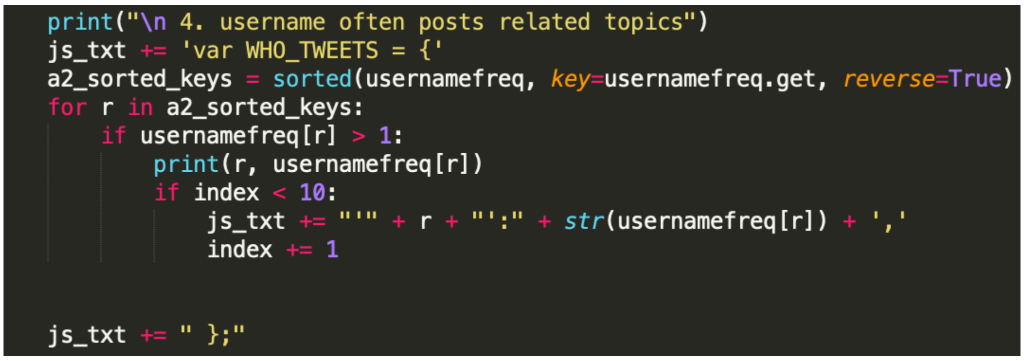
```python
topTweets = hiveCtx.sql("SELECT text, retweet_count,favorite_count FROM tweets where text IS NOT NULL and \
                        text like '%flask%' LIMIT 20")
print('2 according retweetCount we choose=', topTweets.collect() )
```



```python
import os
import pickle
import pprint
from urllib.request import Request, urlopen
from urllib.error import URLError, HTTPError
import csv


def csv_reader(filename, directory="./"):
    with open(os.path.join(directory, filename), newline="") as csvfile:
        reader = csv.reader(csvfile, delimiter="\n", quotechar="|")
        mylist = []
        for row in reader:
            mylist.append(row[0].split("\t"))
        return mylist
```



```python
pick_twlist = data2020full[0::split]
for tw in pick_twlist:
    text = tw[10]
    username = tw[8]
    print(text, "\n@@@username=", username, "\n")
    words, sentiment_tw = anlaysis(text)
    print(sentiment_tw)
    total_sentiment += sentiment_tw
    if sentiment_tw < 0:
        num_nagtive += 1
    if sentiment_tw == 0:
        num_neural += 1
    if sentiment_tw > 0:
        num_positive += 1
    # print('words=', words)
    for raw_word in words:
        word = raw_word.strip(unwanted_chars)
        if word not in wordfreq:
            if word not in black_list:
                wordfreq[word] = 0
        if word not in black_list:
            wordfreq[word] += 1
    if username == 'Indo-Pacific News - Watching the CCP-China Threat'
        username = 'Indo-Pacific News'
    if username not in usernamefreq:
        usernamefreq[username] = 0
    usernamefreq[username] += 1
```



```python
print("\n 4. username often posts related topics")
js_txt += 'var WHO_TWEETS = {'
a2_sorted_keys = sorted(usernamefreq, key=usernamefreq.get, reverse=True)
for r in a2_sorted_keys:
    if usernamefreq[r] > 1:
        print(r, usernamefreq[r])
        if index < 10:
            js_txt += "'" + r + "':" + str(usernamefreq[r]) + ','
            index += 1


js_txt += " };"
```