Instruction Manual

Team #4

12146323 Jeong Sang-heon

14146320 Lee Jun-ha

14146326 Hong Joon-ki

16102278 Sung Chang-keu

Contents

1. Twitter Crawling

2. Sentiment Analysis

3. Implement Web

4. DataBase

Twitter Crawling

1. System requires to install tweepy module.
   `$pip install tweepy`

2. Twitter Oauth can be granted through the developer registration.
   https://developer.twitter.com/en.html

3. Function **`crawl( )`** defines input query and limit of the number of tweets.
   - Keyword would return the user input string value.
   - Num_limit will limit the maximum lookup counts.

4. Function **`tweets_json( )`** actually crawls the tweets and store it as JSON files.
   - Files will be stored separately under the date filtering.
   - Date result will be transformed in YYYY-MM-DD format.
   - To remove the text redundant issue, it filters retweeted tweets.

```python
# -*- coding: utf-8 -*-
from tweepy import OAuthHandler
import tweepy
import json
from datetime import datetime
import re

####트위터 개발자 정보를 입력해 줍니다.
consumer_key = '@@@@@@' #보안
consumer_secret = '@@@@@@' #보안
access_token = '@@@@@@' #보안
access_token_secret = '@@@@@@' #보안

#개발자 인증을 받습니다.
auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True, retry_delay=10)

#트윗 크롤링을 날짜 기준으로 저장합니다.
def tweets_json(keyword, num_limit):
    i=0
    for tweet in tweepy.Cursor(api.search, q=keyword, since='2018-11-01', lang="ko").items(num_limit):
        if (not tweet.retweeted) and ('RT @' not in tweet.text):
            StatusObject = tweet._json
            dict1 = {
                    'id': StatusObject['id_str'],
                    'permalink':"",
                    'username':StatusObject['user']['name'],
                    'text': StatusObject['text'],
                    'date':StatusObject['created_at'],
                    'retweets': StatusObject['retweet_count'],
                    'favorites': StatusObject['favorite_count'],
                    'mentions':StatusObject['entities']['user_mentions'],
                    'hashtags':StatusObject['entities']['hashtags'],
                    'geo':StatusObject['geo']
                    }
            ####날짜 형태 변환하기####
            unformatted = StatusObject['created_at']
            remove_ms = lambda x:re.sub("\+\d+\s","",x)
            mk_dt = lambda x:datetime.strptime(remove_ms(x), "%a %b %d %H:%M:%S %Y")
            my_form = lambda x:"{:%Y-%m-%d}".format(mk_dt(x))
            formatted = my_form(unformatted)
            ####날짜 형태 변환하기####
            with open("/Users/ck/Desktop/aaa/{}.json".format(str(formatted)),'a',encoding="utf-8") as
make_file:
                twitter = json.dumps(dict1, ensure_ascii=False, indent=2)
                make_file.write(twitter+',')
            i += 1

#크롤링 대상을 명시합니다.
def crawl():
    keyword = input()
    num_limit = 100
    tweets_json(keyword, num_limit)

#메인에서 작업 진행을 합니다.
if __name__ == '__main__':
    crawl()
```

Twitter Crawling

1. To do the sentiment analysis, JSON file should be enclosed in square brackets(‘[’ and ‘]’)

2. To refine all the JSON files in the same directory(same name database),
   - Define the path.
   - Execute the code.

```python
# -*- coding: utf-8 -*-

#폴더 내 각각의 .json파일에 대괄호로 묶기.
import os
import glob

path = '/Users/ck/Desktop/aaa/'
extension = 'json'
os.chdir(path)
result = [i for i in glob.glob('*.{}'.format(extension))]

for file in result:
    open_file = open(file,'r')
    read_file = open_file.read()
    new_content = "[" + read_file + "]"
    write_file = open(file,'w')
    write_file.write(new_content)
    write_file.close()
```

Sentiment Analysis

Overview: After receiving the json format file as a result of the data crawl, morphology Analysis, Sensitivity Analysis, and finally, R to draw a sensitivity trend graph of the data.

Used package: KoNLPy

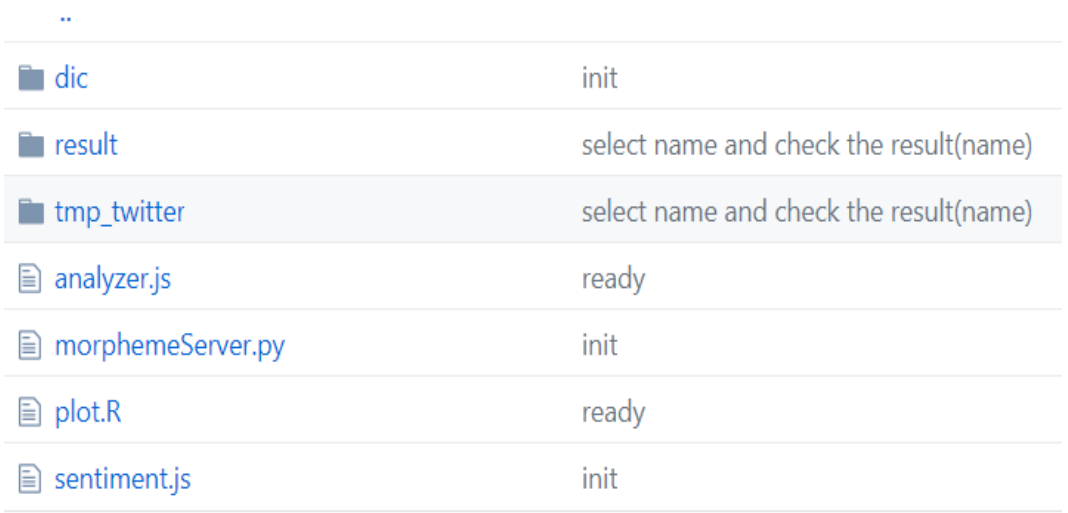
Requirement: npm, python, pip3, jdk8

Module

- morphemeServer.py : Since it takes a lot of time to retrieve a dictionary after the first run, the first dictionary is called up when a request is made by creating a morpheme analyzer server, and then the next request post uses the loaded dictionary data.

- analyzer.js : Save the results of sensitivity analysis. The file(json) in ./result directory in the form of json file.(directory = result/YYYY-MM-DD.json.)

- plot.R : draw a graph with the result(after execute analyzer.js) and save graphical data



<https://github.com/wnsgk91/name_/tree/master/sentiment>

Sentiment Analysis

Order of execution

1. Installization

```
~$ npm install -g korean-sentiment-analyzer
```

Install Korean-sentiment-analyzer to analyze morphology

2. Server open

```
pip3 install konlpy
$ ksa-server localhost 7000
```

Install KoNLPy package and open ksa-server

3. Morphology analysis

```
~$ pip3 intall JPype1
$ python analyzer.js
```

Execute analyzer(sentiment analysis). Also, install JPype1 package.

4. Draw a graph

```
~$ Rstudio plot.R
```

All module are executed in result.js (server)

..

dic	init
result	select name and check the result(name)
tmp_twitter	select name and check the result(name)
analyzer.js	ready
morphemeServer.py	init
plot.R	ready
sentiment.js	init

<https://github.com/wnsgk91/name_/tree/master/sentiment>

Implement Web

1. System requires to install **NPM(Node Package Manager)** to implement nodejs, also some modules are needed : **npm install (express, body-parser, ejs, mysql)**

2. Set the port number as 5000.

3. Routing the address to show screen :

   localhost:5000/home – Home page of our web

   localhost:5000/result – check the result of sentiment analysis

app.js | home.ejs | result.ejs

```
var express = require('express');
var path = require('path');
var bodyParser = require('body-parser');
var app = express();


// ejs 사용
app.set('view engine', 'ejs');
app.engine('html', require('ejs').renderFile);

// mysql 연결
var mysql = require('mysql');
var con = mysql.createConnection({
 host: 'localhost',
 user: 'root',
 password: 'dlwnsgk94', //변경
 database : 'sad' //변경
 });

con.connect();

var staticResource = path.join(__dirname, '/public');
app.use(express.static(staticResource));
app.use(bodyParser.urlencoded({ extended: false }));

app.listen(5000, function() {
    console.log('Connected');
});

app.get('/', function(req,res){
    res.render('header');
});

app.get('/home', function(req,res){
    res.render('home/home');
});

app.get(['/result',"'/result?name="], function(req, res){

    var sql = "SELECT * FROM name";

    con.query(sql, function(err, names, fields){

        var name = req.query.name;

        if(name){
            var sql = "SELECT * FROM name WHERE name='"+name+"'";
            con.query(sql,[name], function(err, name, fields){
                res.render('result/result', {name:name})
            })
        }else{
            res.render('result/result',{name:names});
        }
    })
})
```

DataBase

1. To implement our prototype, your database has to have one table.

```
mysql> show tables;
+---------------+
| Tables_in_sad |
+---------------+
| name          |
+---------------+
1 row in set (0.00 sec)
```

2. The column of table is as follow :

```
mysql> desc name;
+-------+--------------+------+-----+---------+----------------+
| Field | Type         | Null | Key | Default | Extra          |
+-------+--------------+------+-----+---------+----------------+
| num   | int(10)      | NO   | PRI | NULL    | auto_increment |
| name  | varchar(10)  | YES  |     | NULL    |                |
| img   | varchar(100) | YES  |     | NULL    |                |
+-------+--------------+------+-----+---------+----------------+
3 rows in set (0.02 sec)
```

3. Also, it needs to store the result graph of the sentiment analysis as a image file.

```
mysql> select * from name;
+-----+-----------+----------------------+
| num | name      | img                  |
+-----+-----------+----------------------+
|   1 | 아이린    | /images/아이린.png   |
|   2 | 양진호    | /images/양진호.png   |
+-----+-----------+----------------------+
2 rows in set (0.00 sec)
```