Projekt 2: LOL LARRY BIRD

version 0.1

Yuankun LUO, Rena Moehlmann, Carsten Strong

November 25, 2013

Contents

Welcome to hhu_larrybird_lol's documentation! 1

- Intruction for this project 1
- What can our programm do? (Programm workflow) 1
  - Start 1
  - Get Tweets 1
  - Store data 1
  - CLean data 1
  - Computation 2
  - Mathplotlib Plot 2
  - Over 2
- Was ist ein API und wozu ist das gut? 2
- The results of this project(Diagramm Beschreibung) 2
  - Single Bar Diagramms 2
    - Kelly Clarkson's Words TOP 10: 3
  - Double Bar Diagramms Overvie 3
    - Lady GAGA and Kelly Clarkson's Hashtags TOP 5: 4
- Twitter-Accounts and Oauth we use 4
  - Twitter APPlication Info: 4
  - Two Twitter we follow 5
  - Reasons for this two twitters 5
- Diagramms Overview 5
  - Result 5
- Documentation for the Code 8
  - larrybird.main.py 8
    - The workflow of this programm 8
  - larrybird.twitterOauth.py 8
  - larrybird.getalltweets.py 9
  - larrybird.computation.py 9
    - What to do 9
    - How 9
    - Important: 9
  - larrybird.makePicture.py 10

Indices and tables 11

Index 13

Python Module Index 15

Welcome to hhu_larrybird_lol's documentation!

Team: *Yuankun LUO*, *Rena Moehlmann*, *Carsten Strong*

Contents:

Intruction for this project

@Rena

This project is about writing a python programm to download and sort tweets. We want to be able to download tweets from a certain user and later sort them by the amout of words, hashtags and clients used. Furthermore we want to present the numbers in diagrams, using matplotlib.

What can our programm do? (Programm workflow)

@Yuankun

This programm starts with module ***main.py***, all modules are in the package larrybird.

Start

main.start() function is the door of our project. It asks user to input a twitter (screenname). After get the input twitter account, this programm automatcially downloads all tweets of this user, stores this raw_data in disk. Then extracting all usefull data we need. At last, it generates 5 Diagramms for this account.

start() function are listed in blow:

```
def start():
        """To start the programm.

        :param None: No parameter asked
        :type None: None type
        :returns: None
        """
        api = twitterOauth.getApi()
        twitterAccount = raw_input("Please give the account of twitter (screenname) ")
        getalltweets.getAllTweets(api, twitterAccount)
        inputData = twitterAccount
        result = computation.computeAll("data/"+ inputData + ".db")
        makePicture.makeAllPicture(result)
        return result
```

Get Tweets

As you can see, if first call the getApt() function in twitterOauth.py module. After calling that, the api instance was returned. The api is like a key to the door of twitter, but with our Oauthority this key was paged with our application's tokens.

Store data

Then the programm asks user to input the twitter so it can collection tweets. This step calls the ***getAllTweets()*** function in getalltweets.py module. This function use the api and the inputed account to download all statuses of this twitter. It stores this raw data in /data directory with filename suffix _raw.db

CLean data

In getalltweets.py module, after downloading we call a function ***extractUsefulInfo(alltweets)*** to extract for us meaningful data. For this little project, they are:

- status (as String) --> for words statistic

- created_at (as data.time object) --> for time analyse
- hashtags (as list of string) --> for hashingtags ranking
- source (as list of string) --> for client analyse

These meaningful data will be stored also in /data folder with filename suffix .db

Computation

After cleaning data, this programm comes to ***computation.py*** module. This module is maybe the most difficlut part for us. Really, why do you aks so many from us? For God sake!

The complex part is in words statistic. Hier we try to use ***NLTK*** package. NLTK is a very powerful NLP package, we just use two little tools from it.

- reRextokenizer: use this to split the tweets sentenc into words, this dose better as simply use str.split(" ")
- stopWorsSet: we retrieve a set of english stopwords to refine our computation

Our members has problem with the data.time object. So for this part, we dont use that package.

The result will be stored in /data foldre with filename suffix _result.db

Mathplotlib Plot

All ploting things was written in this ***makePicture,py*** module.

The last step's result ist a dict object, like {'xxx_freqOfxxx': {xxx:xxx}}. For detail please read code documentation.

There are three main functions in this module:

- sortDict --> just for sorting Dict with ***OrderedDict*** datatype, because python dict can not be sorted.
- makeBar --> plot a bar diagramm for one result, such like a bar for wordsfrequence of Lady GAGA.
- makeCompBar --> plot a bar diagramm with two bars in it, to show the diffrerece of two twitter account.

Then we have in this module also some shortcut function, so that we can ust input result from last step automatically generate all plot and store them in /image folder.

Over

That's all.

I hope next time, we have less work. Please. The weise man make other man's life easy!

Team of Y,R,C

Was ist ein API und wozu ist das gut?

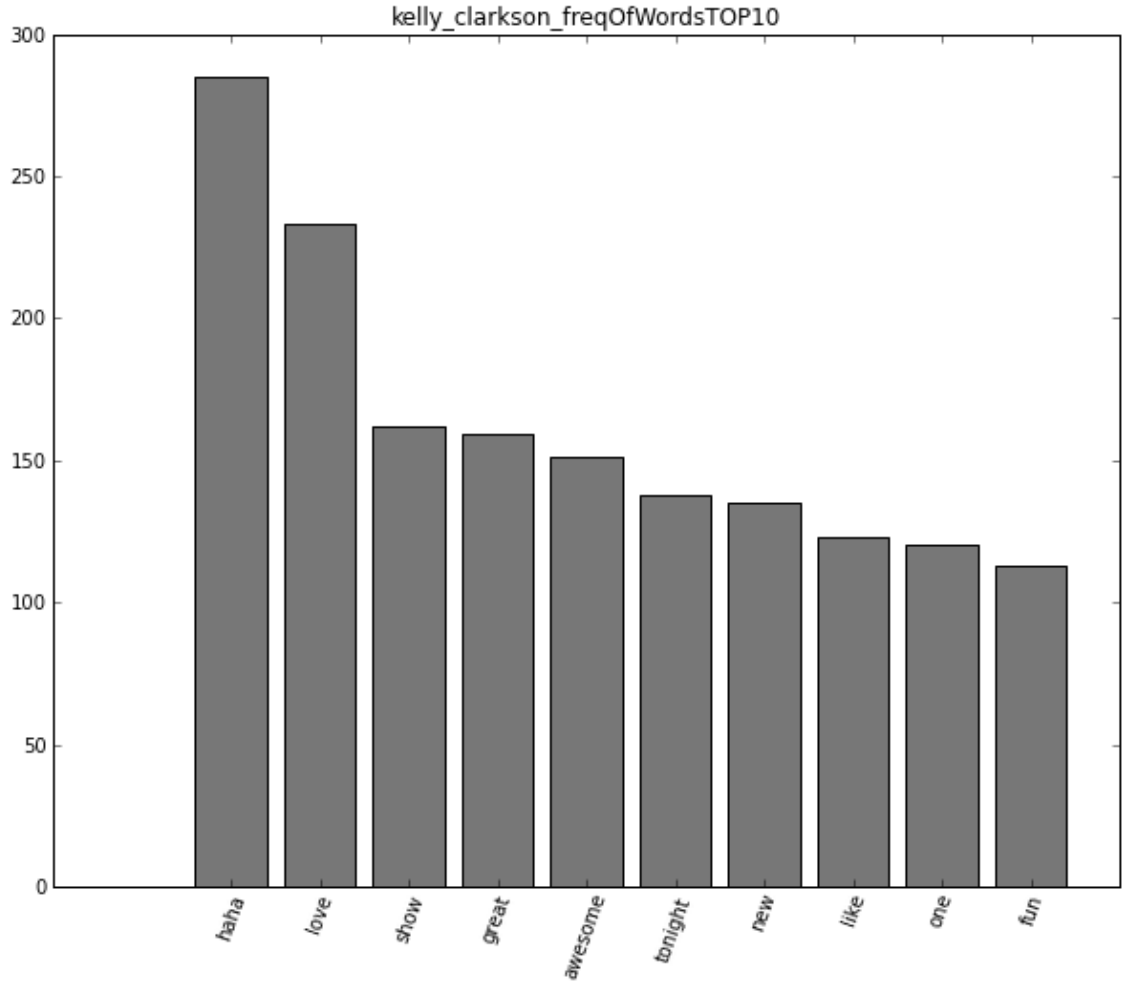
@Carsten

Sorry, Carsten has caught cold this week. So his text will be separately mailed to you.

The results of this project(Diagramm Beschreibung)

We use these two images to show what our progamm dose. All images were automatically genereted.

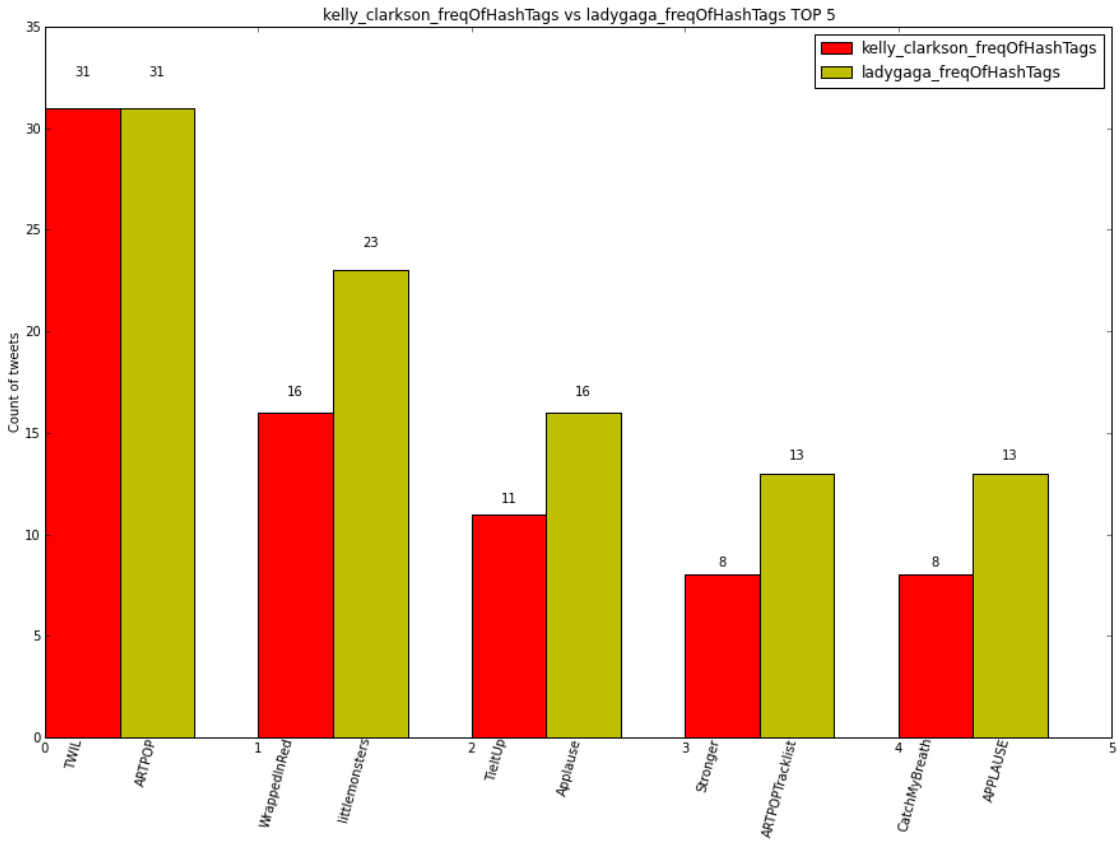
Single Bar Diagramms



Kelly Clarkson's Words TOP 10:

Need description? I dont think so.

Double Bar Diagramms Overvie



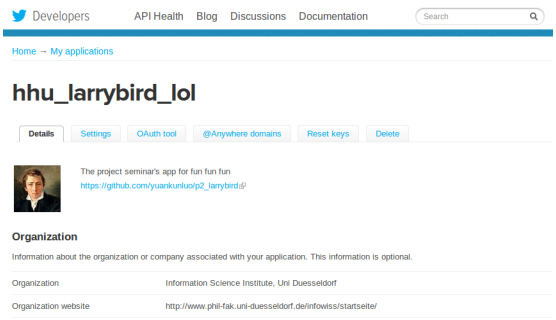
Lady GAGA and Kelly Clarkson's Hashtags TOP 5:

Need description? I dont think so.

Twitter-Accounts and Oauth we use

Our programm use this app and Oauth to get tweets:

Twitter APPlication Info:



OAuth settings

```
Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should neve

Access level     Read, write, and direct messages
About the application permission model
Consumer key    2isZanDsKimDcBLMcHg
Consumer secret PoIxTGCUyrTxcdughdiIjaffVKrFuLnQf2QTp5aS7M
Request token URL       https://api.twitter.com/oauth/request_token
Authorize URL   https://api.twitter.com/oauth/authorize
```

```
Access token URL        https://api.twitter.com/oauth/access_token
Callback URL    None
Sign in with Twitter    No
Your access token
Use the access token string as your "oauth_token" and the access token secret as your "oauth

Access token    390104604-DOmwvmszefTMFIFWYe9sdM73NSuQ4HDYe6gRRmbh
Access token secret     HWg95QP0IKR5zoLhZ1qgqITOVaLPNzbSjP9JnrG2NOGfZ
Access level    Read, write, and direct messages
```

Two Twitter we follow

1. https://twitter.com/kelly_clarkson
2. https://twitter.com/ladygaga

Reasons for this two twitters

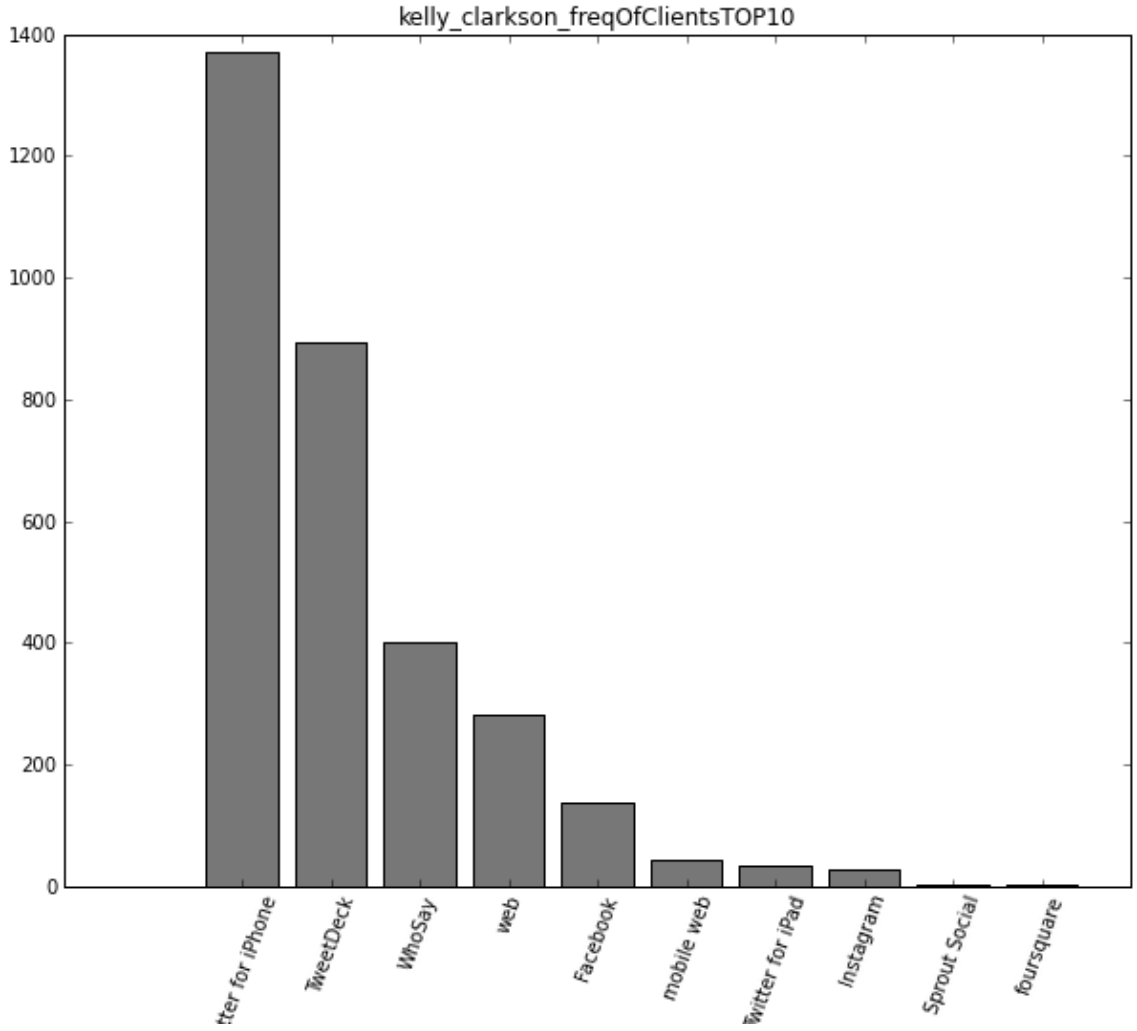
For testing we use ZDNet as our raw data, but it turns out this twitter use few hashtags, so we decide to use pop siger's twitter.

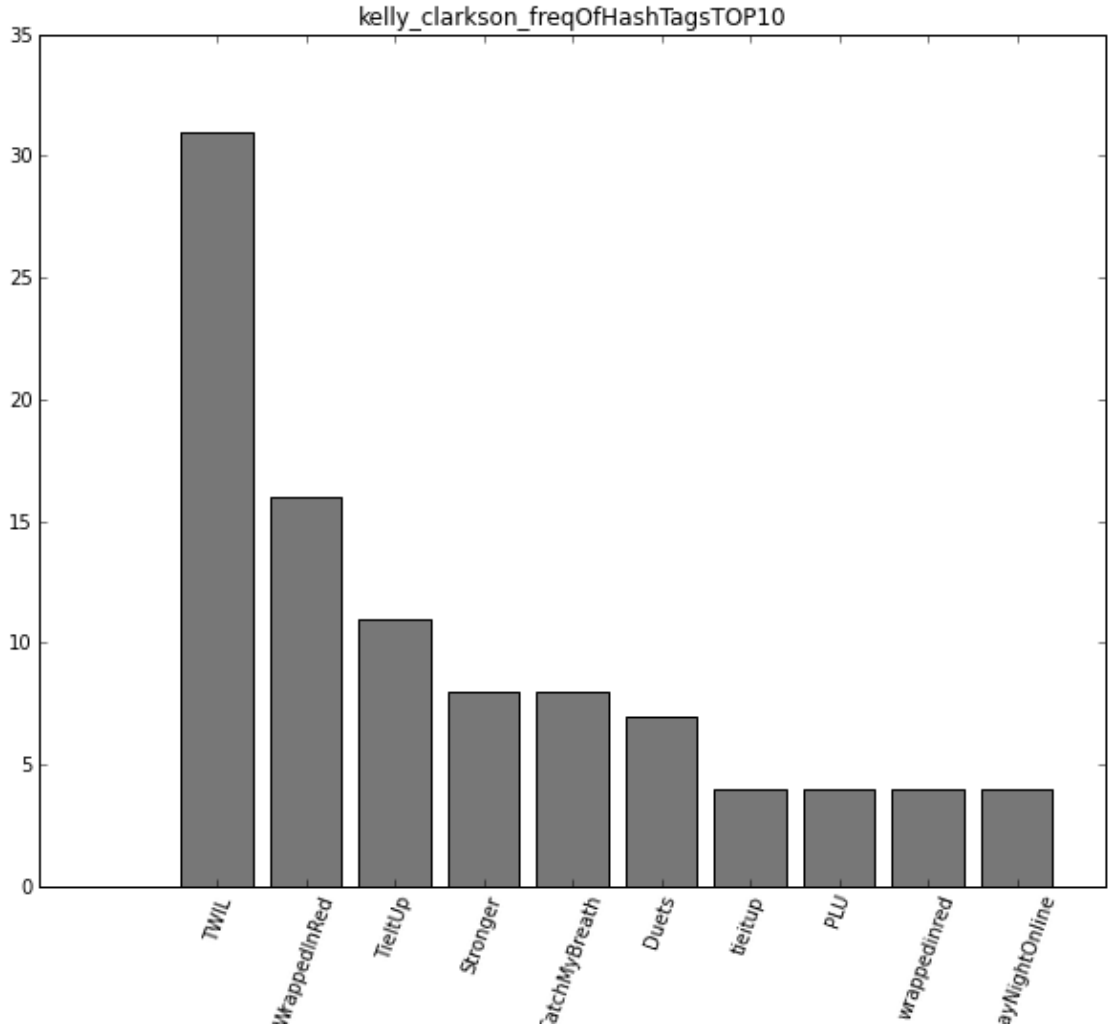
Diagramms Overview

should i put all pics unter /image folder hier? I dont think so. :) If you want more, please go to larrybird package's folder, hier lives this ***image*** folder. It concludes all pics, too many!!

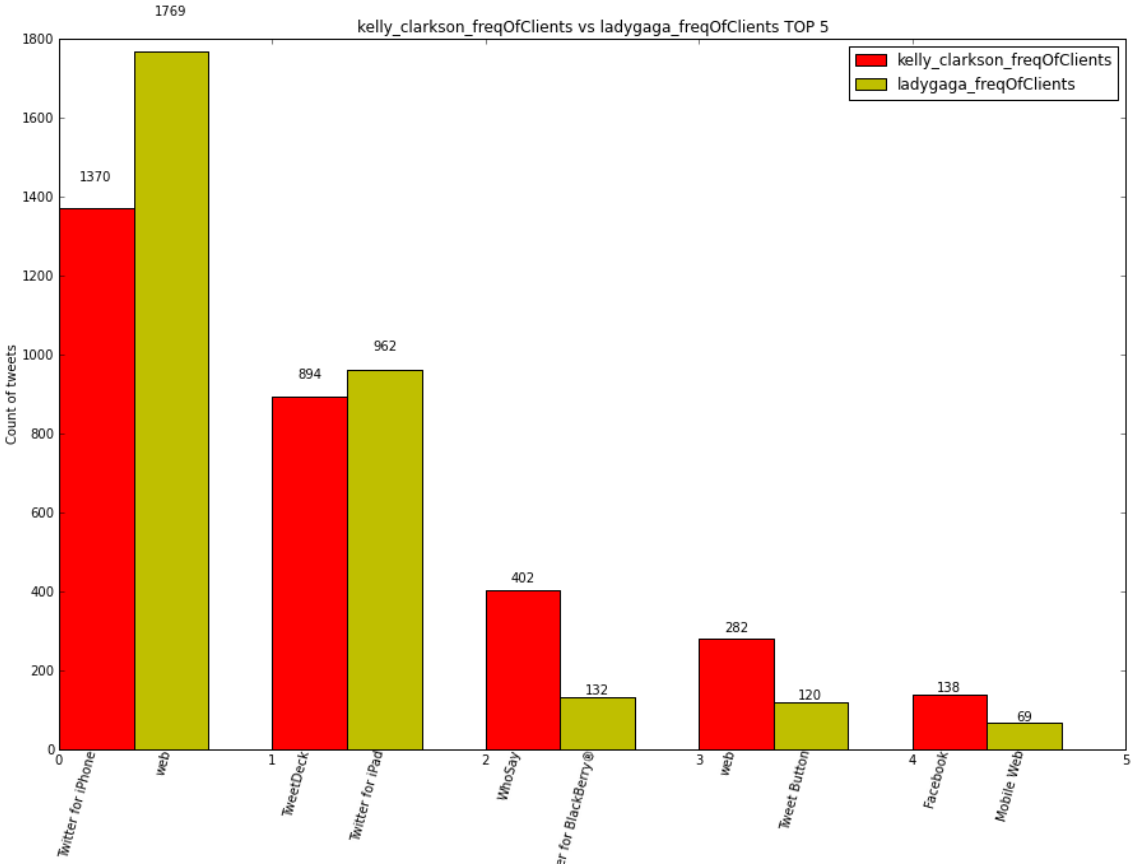
Result

After the user inputs "kelly_clarkson" to the **main.start()**, this programm usese mathplotplot to create images.


kelly_clarkson_freqOfClientsTOP10
1400
1200
1000
800
600
400
200
0
tter for iPhone
TweetDeck
WhoSay
web
Facebook
mobile web
Twitter for iPad
Instagram
Sprout Social
foursquare



We also have a function in makePicture.py module to create comparison diagramm for two twitters. Hier ist one of them for Lady Gaga and Kelly Clarkson's Clients Comparision



Documentation for the Code

All the source code are in the larrybird package.

larrybird.main.py

The main module to start this project. Created on Wed Nov 6 19:57:49 2013

The workflow of this programm

1. ask the twitter name (screen)
2. download the raw twitter data from twitter.com
3. extract our useful data fro raw _data
4. store this two data locally. (raw_data and clean data)
5. compute the frequence
6. store this compute result data locally (xxx_result.db)
7. use the result data to make pictures

`larrybird.main.`**`start`** ()

To start the programm.

Parameters: **None** (*None type*) -- No parameter asked

Returns: None

larrybird.twitterOauth.py

We use the module to store all import credential for twitter Oauth.

Created on Wed Nov 6 19:57:39 2013

`larrybird.twitterOauth.`**`getApi`** ()
Use tweepy to handle the creepy but lovly twitter Oauth system, then return a api instance to continue our work.

Parameters:
- **name** (*None.*) -- No param.
- **state** -- None.

Returns: tweppy.Api instance

larrybird.getalltweets.py

Created on Sat Nov 9 14:56:56 2013

`larrybird.getalltweets.`**`extractUsefulInfo`** (*`alltweets`*)
Extra usefull Information for our project. Such like created time, hashtags, source(client), text Erery Item hast a tweet_id as key, a tuple consists of (text, hashtags, source, created_at)

Parameters: **alltweets** (*List*) -- a List of all retrievaled tweets

Returns: a List full of usefull infomation as Tuple

`larrybird.getalltweets.`**`getAllTweets`** (*`api`*, *`screen_name`*)
Return all tweets for a given screenname.

Parameters:
- **api** (*tweepy.API.*) -- the api instance of tweepy api class.
- **screen_name** (*String*) -- The screen name of twitter.

Returns: A List full of all tweets of this given screenname.

`larrybird.getalltweets.`**`saveTweets`** (*`alltweets`*)
Save all given tweets to two local file. One is a raw file, that stores the orginal data from api. The other one is a refined file for using. Both file use the user's screen_name als filename. Extention is .db. Attention: Both files are stored in data directory

Parameters: **alltweets** (*List*) -- a List of all retrievaled tweets

Returns: True if successful, False if not

larrybird.computation.py

We use the module to compute all computational work for our collected data.

What to do

1. Anzahl je Jahr, Monat, Tag, etc. (Analyse mit Hilfe des datetime-Moduls)
2. Menge und Häufigkeit der verwendeten Wörter
3. Menge und Häufigkeit der verwendeten Hashtags
4. Menge und Häufigkeit der verwendeten Twitter-Clients

How

we use tree functions to do this computation, every function use the clean data,

that we collected as input, then output something for matloplib.

Important:

1. Every output data is in form of dict and stored with pickle in local.

2. **NLTK** was used to clean stopwords and toknizer sentences.

Created on Wed Nov 6 19:57:39 2013

`larrybird.computation.`**`computeAll`** (*`inputData`*)
A shorcut to call all functions in this module

Parameters: **inputData** (*String*) -- The name of the local data.
Returns: A dict of all informations

`larrybird.computation.`**`computeFreqOfClient`** (*`inputData`*)
Compute die Frequence of Client, eg, iPad, web. Then return a dict as putput and store the result dict in a local data.

Parameters: **inputData** (*String*) -- The name of the clean data (list objects).
Returns: A Dict, whoes key ist the client name and value ist interger.

`larrybird.computation.`**`computeFreqOfHashtags`** (*`inputData`*)
Computes the frequence of hashtags used. Then returns a dict as output and stores the result dict in a local data.

Parameters: **inputData** (*String*) -- The name of the clean data (list objects).
Returns: A dict, whose key is the hashtag and value ist integer.

`larrybird.computation.`**`computeFreqOfWords`** (*`inputData`*)
Computes the frequence of words used. Then returns a dict as output and stores the result dict in a local data. Try to import **NLTK** package to throw out those stop-word, then we can get more intressting data. And use NLTK to tokenize words, and clean the shorturl or something not import.

Note

Not every pc has NLTK, in this function if NLTK was not successful imported,we just compute word frequence over raw inputdata.

Parameters: **inputData** (*String*) -- The name of the clean data (list objects).
Returns: A dict, whose key is the word and value is integer.

`larrybird.computation.`**`computeTime`** (*`inputData`*)
Tweets selected by year, month and day. Use the cleanded data from the getallweets.py module, this function loads the data and compute the frequrence for days, months and years. Use tree dicts to store the output with pickle into locale storage for makePicture.py.

Parameters: **inputData** (*String*) -- The name of the clean data (list objects).
Returns: A tuple, whose elements are the dicts of years, months, days.

larrybird.makePicture.py

Use **matplotlib** to generate pics.

`larrybird.makePicture.`**`makeAllCompare`** (*`result1`*, *`result2`*)
A short to make all compare bar diagramms, using the module makePicture.py

Parameters:
- **result1** -- The first result data that stored locally after computation.py processing.
- **result2** -- The second result data that stored locally after computation.py processing.

Returns: None

`larrybird.makePicture.`**`makeAllPicture`** (*`result`*)
A shortcut to make all bar diaagramms.

Parameters: **result** (*Dict.*) -- The result data that stored locally after computation.py processing.
Returns: None

larrybird.makePicture.**makeBarPlot** (*inputDict*, *filename*, *rank=10*, *title=None*, *ylabel='Count'*, *output='image/'*)

Plot the bar diagramm of the input dict. After call of computation module's functions, we have a inputDict als result. This function use this result to make diagramm (plot) then store the plot locally. ..Note: If the rank parameter is None, then output all data ranking.

Parameters:
- **inputDict** (*Dict, the key is String, the value is interger*) -- A dict of (key:value) paire.
- **rank** (*Integer*) -- Indicate the top N (ranking), **default** is None
- **output** (*String*) -- The folder path where our plot diagramms were stored, **default** is the image folder.

Returns: None

larrybird.makePicture.**makeCompareBar** (*inputDict1*, *inputDict2*, *fileName1*, *fileName2*, *rank=5*, *output='image/'*)

This function use two InputDict as input parameter, then plot a two bars diagramm.

Parameters:
- **inputDict1** (*A dict object of items in form of (String,int)*) -- The first inputDict.
- **inputDict2** (*A dict object of items in form of (String,int)*) -- The second inputDict.
- **fileName1** (*String.*) -- The name that appears as a part of the title of this diagramm
- **fileName2** (*String.*) -- The name that appears as a part of the title of this diagramm
- **rank** -- Indicate how many items will be showed (ranking for top N), **default is 5**
- **output** (*String*) -- The directory , where the output png files will be stored locally

Returns: None

larrybird.makePicture.**sortDict** (*inputDict*, *rank=None*)

Sort the inputDict.

Parameters:
- **inputDict** (*dict*) -- the Dict to sort
- **rank** (*Int*) -- A number to indicate how many items will be ranked

Returns: OrderedDict

Indices and tables

- *genindex*
- *modindex*
- *search*

Index

C

computeAll() (in module larrybird.computation)

computeFreqOfClient() (in module larrybird.computation)

computeFreqOfHashtags() (in module larrybird.computation)

computeFreqOfWords() (in module larrybird.computation)

computeTime() (in module larrybird.computation)

E

extractUsefulInfo() (in module larrybird.getalltweets)

G

getAllTweets() (in module larrybird.getalltweets)

getApi() (in module larrybird.twitterOauth)

L

larrybird (module)

larrybird.computation (module)

larrybird.getalltweets (module)

larrybird.main (module)

larrybird.makePicture (module)

larrybird.twitterOauth (module)

M

makeAllCompare() (in module larrybird.makePicture)

makeAllPicture() (in module larrybird.makePicture)

makeBarPlot() (in module larrybird.makePicture)

makeCompareBar() (in module larrybird.makePicture)

S

saveTweets() (in module larrybird.getalltweets)

sortDict() (in module larrybird.makePicture)

start() (in module larrybird.main)

Python Module Index

l

larrybird

larrybird.computation

larrybird.getalltweets

larrybird.main

larrybird.makePicture

larrybird.twitterOauth