Data Mining
&
Meaningful Visualization

by

Usman Ahmad Khwaja
Chew (Jason) Kuang Sheng
Mohamed Helal Elsayed

ABSTRACT

Data Mining **is a data extraction technique used in many *Artificial Intelligence* applications. The purpose of data mining is to obtain data sets from large volumes of complex data. The data sets can be used to generate meaningful information. Functions can be performed on such data for classification, approximation, or prediction purposes that would otherwise be impossible to run on big data. The data sets obtained however are still large and incomprehensible by human eye. This report will explore some scripting methods by which large informative data can be mined using social and market media applications' API. Then we should be able to demonstrate how visually readable models can be generated from the extracted data. This can help in human analysis and visualization of large amounts of data.**

CONTENTS

INTRODUCTION

Text mining is the application of natural language processing techniques and analytical methods to text data in order to derive relevant information. Text mining is getting a lot attention these last years, due to an exponential increase in digital text data from web pages, google's projects such as google books and google ngram, and social media services such as Twitter. Twitter data constitutes a rich source that can be used for capturing information about any topic imaginable. This data can be used in different use cases such as finding trends related to a specific keyword, measuring brand sentiment, and gathering feedback about new products and services. Once the data is collected, we will also use the Google Charts library for python to display the information in a graph

In this report, we will use Twitter data to compare the popularity of 3 programming languages: Python, Javascript and Ruby. First, we will explain how to connect to Twitter Streaming API and how to get the data. Secondly, we will explain how to structure the data for analysis, and lastly paragraph, we will explain how to filter the data and plot the information collected in a chart.

RESEARCH QUESTIONS

1. What is Data Mining
2. What is the application of Data mining
3. How can we use API to extract data
4. How to visualize the extracted data

SYSTEM

• Operating system	Ubuntu 14.04 'Trusty' (x86-64)
• Linux Kernel	3.13.0-37-generic
• Processor	Intel Core i5-3317U CPU @ 1.70GHz x 2
• Memory	3.4 GiB

DATA MINING

Twitter API

API stands for Application Programming Interface. It is a tool that makes the interaction with computer programs and web services easy. Many web services provides APIs to developers to interact with their services and to access data in programmatic way. For this report, we will use Twitter Streaming API to download tweets related to 3 keywords: "python", "javascript", and "ruby". We will be using a Python library called `Tweepy` to connect to Twitter Streaming API and downloading the data.

Date Streaming

We created a file called twitter_streaming.py, and wrote the following script to collect data.

```
#Import the necessary methods from tweepy library
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

#Variables that contains the user credentials to access Twitter API
access_token = "*****"
access_token_secret = "*****"
consumer_key =  "*****"
consumer_secret = "*****"


#This is a basic listener that just prints received tweets to stdout.
class StdOutListener(StreamListener):

    def on_data(self, data):
        print data
        return True

    def on_error(self, status):
        print status


if __name__ == '__main__':

    #This handles Twitter authetification and the connection to Twitter 
Streaming API
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_token, access_token_secret)
    stream = Stream(auth, l)

    #This line filter Twitter Streams to capture data by the keywords: 
'python', 'javascript', 'ruby'
    stream.filter(track=['python', 'javascript', 'ruby'])
```

Sample Data

We ran the script for about an hour on a Saturday afternoon. The data was extracted onto a text file so it can be used later in the program. The file size was about 5 mb. The file contained 1485 tweets.

```
python twitter_streaming.py > twitter_data.txt
```

Reading the data

The data that we stored `twitter_data.txt` is in JSON format. JSON stands for JavaScript Object Notation. This format makes it easy to humans to read the data, and for machines to parse it. Below is an example for one tweet in JSON format. You can see that the tweet contains additional information in addition to the main text which in this example: `"Yaayyy I learned some JavaScript today! #thatwasntsohard #yesitwas #stoptalkingtoyourself #hashbrown #hashtag"`.

```
{"created_at":"Tue Jul 15 14:19:30 +0000
2014","id":489051636304990208,"id_str":"489051636304990208","text":"Yaayyy
I learned some JavaScript today! #thatwasntsohard #yesitwas
#stoptalkingtoyourself #hashbrown #hashtag","source":"\u003ca
href=\"http:\/\/twitter.com\/download\/iphone\"
rel=\"nofollow\"\u003eTwitter for
iPhone\u003c\/a\u003e","truncated":false,"in_reply_to_status_id":null,"in_r
eply_to_status_id_str":null,"in_reply_to_user_id":null,"in_reply_to_user_id
_str":null,"in_reply_to_screen_name":null,"user":{"id":2301702187,"id_str":
"2301702187","name":"Toni
Barlettano","screen_name":"itsmetonib","location":"Greater NYC
Area","url":"http:\/\/www.tonib.me","description":"So Full of Art |
\nToni Barlettano Creative Media +
Design","protected":false,"followers_count":8,"friends_count":25,"listed_co
unt":0,"created_at":"Mon Jan 20 16:49:46 +0000
2014","favourites_count":6,"utc_offset":null,"time_zone":null,"geo_enabled"
:false,"verified":false,"statuses_count":20,"lang":"en","contributors_enabl
ed":false,"is_translator":false,"is_translation_enabled":false,"profile_bac
kground_color":"C0DEED","profile_background_image_url":"http:\/\/abs.twimg.
com\/images\/themes\/theme1\/bg.png","profile_background_image_url_https":"
https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile_backgroun
d_tile":false,"profile_image_url":"http:\/\/pbs.twimg.com\/profile_images\/
425313048320958464\/Z2GcderW_normal.jpeg","profile_image_url_https":"https:
\/\/pbs.twimg.com\/profile_images\/425313048320958464\/Z2GcderW_normal.jpeg
","profile_link_color":"0084B4","profile_sidebar_border_color":"C0DEED","pr
ofile_sidebar_fill_color":"DDEEF6","profile_text_color":"333333","profile_u
se_background_image":true,"default_profile":true,"default_profile_image":fa
lse,"following":null,"follow_request_sent":null,"notifications":null},"geo"
:null,"coordinates":null,"place":null,"contributors":null,"retweet_count":0
,"favorite_count":0,"entities":{"hashtags":[{"text":"thatwasntsohard","indi
ces":[40,56]},{"text":"yesitwas","indices":[57,66]},{"text":"stoptalkingtoy
ourself","indices":[67,89]},{"text":"hashbrown","indices":[90,100]},{"text"
:"hashtag","indices":[101,109]}],"symbols":[],"urls":[],"user_mentions":[]}
,"favorited":false,"retweeted":false,"filter_level":"medium","lang":"en"}
```

MAIN PROGRAM

Tools used

For the remaining of this report, we will be using 4 Python libraries `json` for parsing the data, pandas for data manipulation, `matplotlib` for creating charts, and `re` for regular expressions. The `json` and `re` libraries are installed by default in Python. We had to install pandas because it was not installed in our machine.

We will start first by uploading `json` and `pandas` using the commands below:

```
import json
import pandas as pd
import matplotlib.pyplot as plt
```

Next we will read the data in into an array that we call `tweets`.

```
tweets_data_path = '../data/twitter_data.txt'

tweets_data = []
tweets_file = open(tweets_data_path, "r")
for line in tweets_file:
    try:
        tweet = json.loads(line)
        tweets_data.append(tweet)
    except:
        continue
```

Next, we will structure the tweets data into a pandas DataFrame to simplify the data manipulation. We will start by creating an empty DataFrame called `tweets` using the following command.

```
tweets = pd.DataFrame()
```

Next, we will add a column to the `tweets` DataFrame called `text`. `text` column contains the tweet.

```
tweets['text'] = map(lambda tweet: tweet['text'], tweets_data)
```

MINING THE TWEETS

Our main goals in these text mining tasks are: compare the popularity of Python, Ruby and Javascript programming languages. We will add tags to our `tweets` DataFrame in order to be able to manipulate the data easily.

First, we will create a function that checks if a specific keyword is present in a text. We will do this by using regular expressions. Python provides a library for regular expression called `re`. We will start by importing this library

```
import re
```

Next we will create a function called `word_in_text(word, text)`. This function return `True` if a `word` is found in `text`, otherwise it returns `False`.

```
def word_in_text(word, text):
    word = word.lower()
    text = text.lower()
    match = re.search(word, text)
    if match:
        return True
    return False
```

Next, we will add 3 columns to our `tweets` DataFrame.

```
tweets['python'] = tweets['text'].apply(lambda tweet:
word_in_text('python', tweet))
tweets['javascript'] = tweets['text'].apply(lambda tweet:
word_in_text('javascript', tweet))
tweets['ruby'] = tweets['text'].apply(lambda tweet: word_in_text('ruby',
tweet))
```

We can calculate the number of tweets for each programming language as follows:

```
print tweets['python'].value_counts()[True]
print tweets['javascript'].value_counts()[True]
print tweets['ruby'].value_counts()[True]
```

This returns: 252 for python, 563 for javascript and 624 for ruby. We can make a simple comparison chart by executing the following:

```
prg_langs = ['python', 'javascript', 'ruby']
tweets_by_prg_lang = [tweets['python'].value_counts()[True],
tweets['javascript'].value_counts()[True],
tweets['ruby'].value_counts()[True]]
```

```
x_pos = list(range(len(prg_langs)))
width = 0.8
fig, ax = plt.subplots()
plt.bar(x_pos, tweets_by_prg_lang, width, alpha=1, color='g')

# Setting axis labels and ticks
ax.set_ylabel('Number of tweets', fontsize=15)
ax.set_title('Ranking: python vs. javascript vs. ruby (Raw data)',
fontsize=10, fontweight='bold')
ax.set_xticks([p + 0.4 * width for p in x_pos])
ax.set_xticklabels(prg_langs)
plt.grid()
```

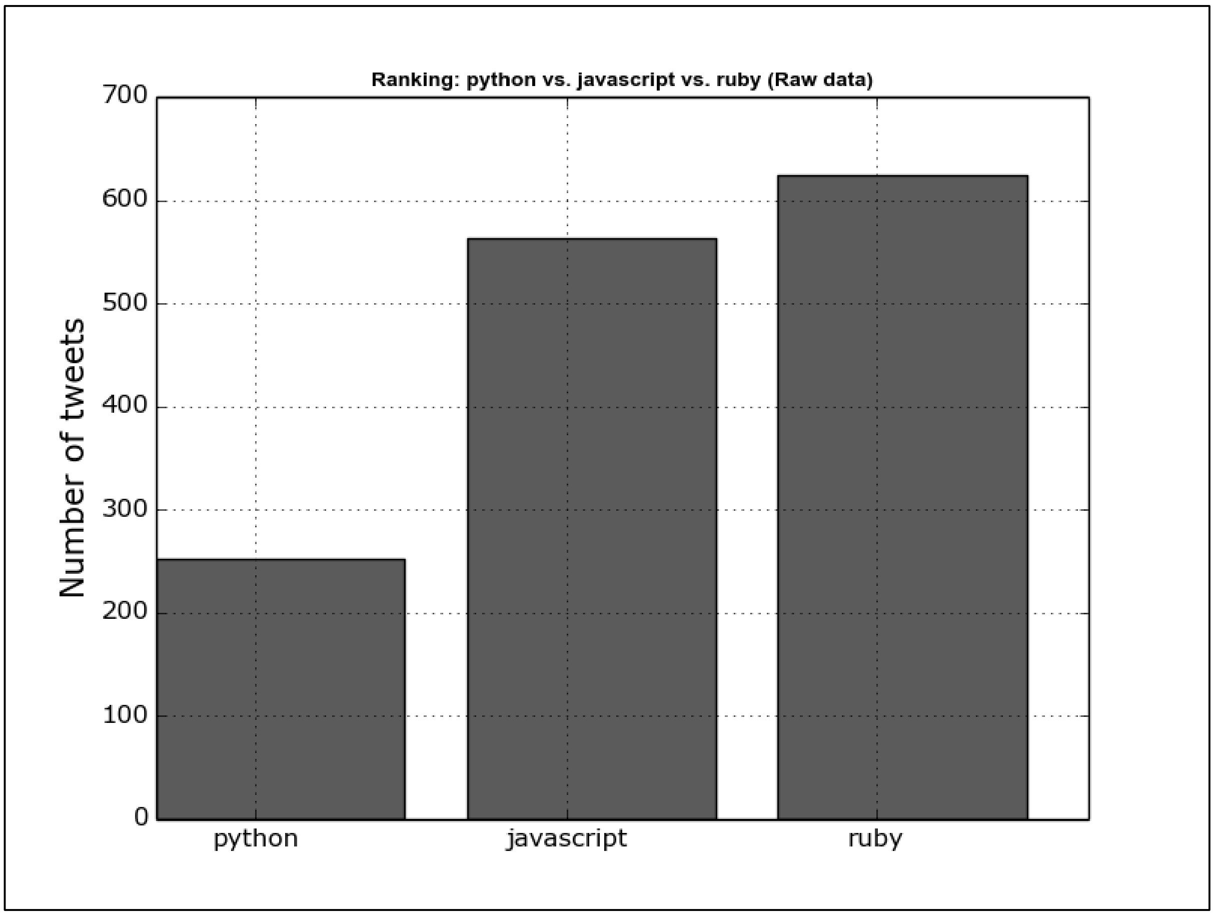

Figure 1 Test results png

This shows, that the keyword ruby is the most popular, followed by python then javascript. However, the tweets DataFrame contains information about all tweets that contains one of the 3 keywords and doesn't restrict the information to the programming languages. For example, there are a lot tweets that contains the keyword ruby and that are related to a political scandal called Rubygate. In the next section, we will filter the tweets and re-run the analysis to make a more accurate comparison.

Targeting relevant tweets

We are intersted in targetting tweets that are related to programming languages. Such tweets often have one of the 2 keywords: "programming" or "code". We will create 2 additional columns to our `tweets` DataFrame where we will add this information.

```
tweets['programming'] = tweets['text'].apply(lambda tweet: 
word_in_text('programming', tweet))
tweets['code'] = tweets['text'].apply(lambda tweet: word_in_text('code', 
tweet))
```

We will add an additional column called `relevant` that take value `True` if the tweet has either "programming" or "tutorial" keyword, otherwise it takes value `False`.

```
tweets['relevant'] = tweets['text'].apply(lambda tweet: 
word_in_text('programming', tweet) or word_in_text('code', tweet))
```

We can print the counts of relevant tweet by executing the commands below.

```
print tweets['programming'].value_counts()[True]
print tweets['tutorial'].value_counts()[True]
print tweets['relevant'].value_counts()[True]
```

This returns, 45 for `programming` column, 60 for `tutorial` column, and 89 for `relevant` column.

We can compare now the popularity of the programming languages by executing the commands below.

```
print tweets[tweets['relevant'] == True]['python'].value_counts()[True]
print tweets[tweets['relevant'] == True]['javascript'].value_counts()[True]
print tweets[tweets['relevant'] == True]['ruby'].value_counts()[True]
```

We can see a drastic change in results. This way we can apply advanced filters on tweets to get most relevant results. Javascript is the most popular with a count of 49, followed by python by a count of 36, and ruby by a count of 2. We can make a comparaison graph by executing the commands below:

```
tweets_by_prg_lang = [tweets[tweets['relevant'] == 
True]['python'].value_counts()[True], 
                      tweets[tweets['relevant'] == 
True]['javascript'].value_counts()[True], 
                      tweets[tweets['relevant'] == 
True]['ruby'].value_counts()[True]]
```

```
x_pos = list(range(len(prg_langs)))
width = 0.8
fig, ax = plt.subplots()
plt.bar(x_pos, tweets_by_prg_lang, width,alpha=1,color='g')
ax.set_ylabel('Number of tweets', fontsize=15)
ax.set_title('Ranking: python vs. javascript vs. ruby (Relevant data)',
fontsize=10, fontweight='bold')
ax.set_xticks([p + 0.4 * width for p in x_pos])
ax.set_xticklabels(prg_langs)
plt.grid()
```

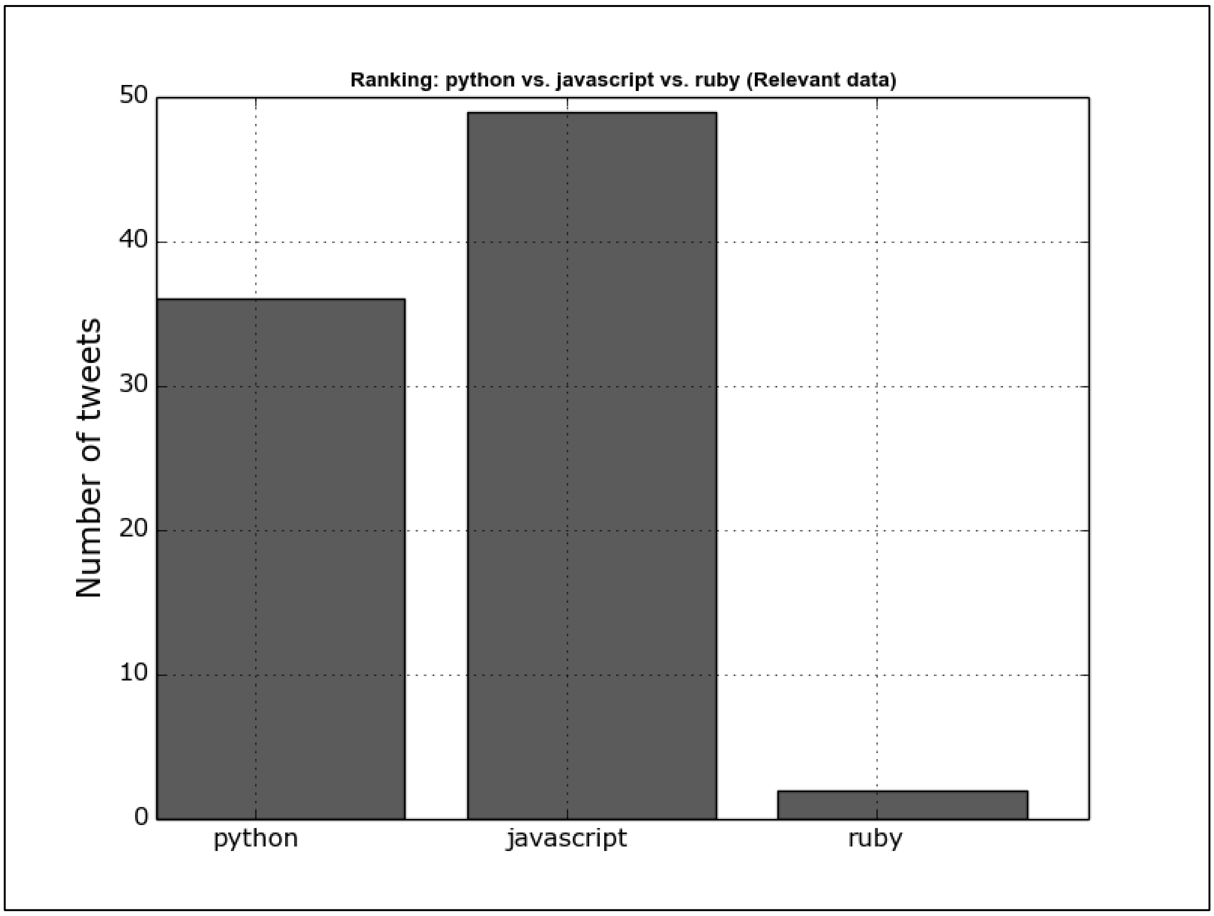

Figure 2 relevant tweets only

CONCLUSION

In this report, we covered many techniques used in text mining. The code we used can be modified to create a deeper analysis or could be adapted to another use case. We can use this to compare tweets by location or region. We can filter them by language. Or even extract links from the tweets. The charting then help us to visualize the scenario we were looking for.

REFERENCES