Appendix 1

NLTK Installation and Setup

The NLTK module can be installed either by downloading the package through the NLTK website. Its preset text repositories should be downloaded as well. Some of the features can be tried out more easily by typing the following Python command line:

>>> import nltk

>>> nltk.download()

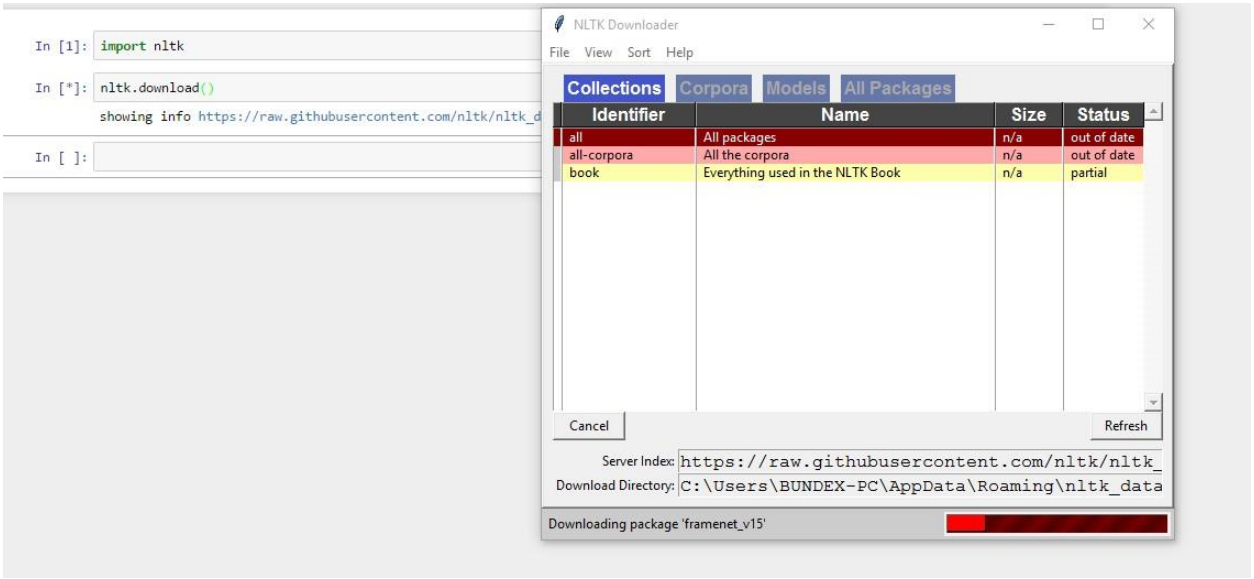


Figure 1: Downloading nltk

Twitter OAuth 1.0a Flow with Ipython Notebook

Twitter implements OAuth 1.0A as its standard authentication mechanism, and in order to use it to make requests to Twitter's API, you'll need to go to https://dev.twitter.com/apps and create a sample application. There are three items you'll need to note for an OAuth 1.0 A workflow, a consumer key and consumer secret that identify the application as well as the oauth_callback URL that tells Twitter where redirect back to after the user has authorized the application. One also needs an ordinary Twitter account in order to log in, create an app, and get these credentials. For development purposes or for accessing your own account's data, one can simply use the OAuth token and OAuth token secret that are provided the application settings to authenticate as opposed to going through the steps here. The process of obtaining and the OAuth token and OAuth token secret is fairly straight forward (especially with the help of a good library (Russell, 2013)

You must ensure that your browser is not blocking pop-ups in order for this script to work.

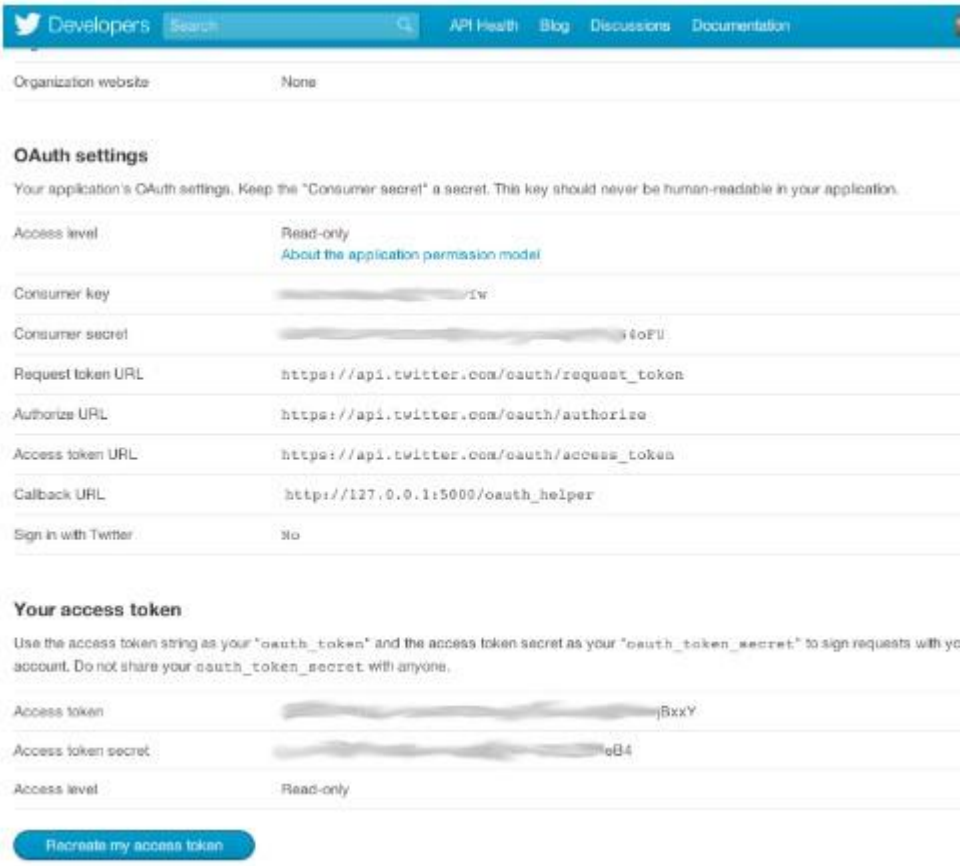


Figure 2: Twitter OAUTH dance

Anaconda installation

Anaconda is a FREE enterprise-ready Python distribution for data analytics, processing, and scientific computing. Anaconda comes with Python 2.7 and 100+ cross-platform tested and optimized Python packages. All of the usual Python ecosystem tools work with Anaconda.

Additionally, Anaconda can create custom environments that mix and match different Python versions (2.6, 2.7 or 3.3) and other packages into isolated environments and easily switch between them using conda, our innovative multi-platform package manager for Python and other languages.

For Detailed Anaconda Installation Instructions check out http://docs.continuum.io/anaconda/install.html

INSTALLATION

System Requirements		
Linux	Windows	Mac OS X
32/64 bit x86 processor	32/64 bit x86 processor	64-bit x86 processor

Download Anaconda

Figure 3: Anaconda Installation

Tweepy Installation

Tweepy supports OAuth authentication. Authentication is handled by the tweepy.AuthHandler class.

Tweepy can be installed from command line by this command

->Pip install tweepy

OAuth Authentication

Tweepy tries to make OAuth as painless as possible for you. To begin the process we need to register our client as in (Twitter OAuth flow above) application with Twitter. Create a new application and once you are done you should have your consumer token and secret. The next step is creating an OAuthHandler instance. Into this we pass our consumer token and secret which was given to us in the previous paragraph:

auth = tweepy.OAuthHandler(consumer_token, consumer_secret)

If you have a web application and are using a callback URL that needs to be supplied dynamically you would pass it in like so:

```
auth = tweepy.OAuthHandler(consumer_token, consumer_secret, callback_url)
```

If the callback URL will not be changing, it is best to just configure it statically on twitter.com when setting up your application's profile.

Unlike basic auth, we must do the OAuth "dance" before we can start using the API. We must complete the following steps:

1. Get a request token from twitter
2. Redirect user to twitter.com to authorize our application
3. If using a callback, twitter will redirect the user to us. Otherwise, the user must manually supply us with the verifier code.
4. Exchange the authorized request token for an access token.

Appendix 2

Codes courtesy of *Python Programming Tutorials*. (2016). *Pythonprogramming.net*. Retrieved 3 December 2016, from https://pythonprogramming.net/search/?q=nltk.

```
In [10]: # %load Classifier_wordCloudtweeps.py
   ...: import time
   ...: import pymysql
   ...: import time
   ...: import urllib.request, urllib.parse, urllib.error
   ...: import zlib
   ...: import string
   ...:
   ...: conn = pymysql.connect("localhost","root","","sentitwit", charset = 'utf8mb4')
   ...: conn.text_factory = str
   ...: cur = conn.cursor()
   ...:
   ...: cur.execute('SELECT time, username FROM safaricom')
   ...: subjects = dict()
   ...: for message_row in cur :
   ...:     subjects[message_row[0]] = message_row[1]
   ...: # print (message_row)
   ...:
   ...: # cur.execute('SELECT time,username, tweet, FROM sentitwit')
   ...: cur.execute('SELECT time FROM safaricom')
   ...: counts = dict()
   ...: for message_row in cur :
   ...:     text = subjects[message_row[0]]
   ...:     text = text.translate(string.punctuation)
   ...:     text = text.translate('1234567890')
   ...:     text = text.strip()
   ...:     text = text.lower()
   ...:     words = text.split()
   ...:     for word in words:
   ...:         if len(word) < 4 : continue
   ...:         counts[word] = counts.get(word,0) + 1
   ...:         print(counts[word])
   ...:
   ...: x = sorted(counts, key=counts.get, reverse=True)
   ...: highest = None
   ...: lowest = None
   ...: for k in x[:100]:
   ...:     if highest is None or highest < counts[k] :
   ...:         highest = counts[k]
   ...:     if lowest is None or lowest > counts[k] :
   ...:         lowest = counts[k]
   ...: print('Range of counts:',highest,lowest)
   ...:
   ...: # Spread the font sizes across 20-100 based on the count
   ...: bigsize = 80
   ...: smallsize = 20
   ...:
   ...: fhand = open('gword.js','w')
   ...: fhand.write("gword = [")
   ...: first = True
   ...: for k in x[:100]:
   ...:     if not first : fhand.write( ",\n")
   ...:     first = False
   ...:     size = counts[k]
   ...:     size = (size - lowest) / float(highest - lowest)
   ...:     size = int((size * bigsize) + smallsize)
   ...:     fhand.write("{text: '"+k+"', size: "+str(size)+"}")
   ...: fhand.write( "\n];\n")
   ...:
   ...: print("Output written to gword.js")
   ...:
```

Figure 4: Clasiifier_wordCloud

```
n [7]: # %load Classifier_twitanalysis.py
  ...: from tweepy import Stream
  ...: from tweepy import OAuthHandler
  ...: from tweepy.streaming import StreamListener
  ...: import json
  ...: import sentiment_mod as s
  ...: import time
  ...: from urllib.error import HTTPError
  ...: from requests.exceptions import Timeout, ConnectionError
  ...: from requests.packages.urllib3.exceptions import ReadTimeoutError
  ...:
  ...: #consumer key, consumer secret, access token, access secret.
  ...: ckey = ""
  ...: csecret = ""
  ...: atoken = ""
  ...: asecret = ""
  ...:
  ...: #from twitterapistuff import *
  ...:
  ...: class listener(StreamListener):
  ...:
  ...:     def on_data(self, data):
  ...:         try:
  ...:             all_data = json.loads(data)
  ...:
  ...:             tweet = all_data["text"]
  ...:             sentiment_value, confidence = s.sentiment(tweet)
  ...:             print(tweet, sentiment_value, confidence)
  ...:
  ...:             if confidence*100 >= 80:
  ...:                 output = open("twitter-out.txt","a")
  ...:                 output.write(sentiment_value)
  ...:                 output.write('\n')
  ...:                 output.close()
  ...:
  ...:             return True
  ...:         except (Timeout, ReadTimeoutError, ConnectionError) as exc:
  ...:             time.sleep(10)
  ...:             return True
  ...:
  ...:     def on_error(self, status):
  ...:         print(status)
  ...:
  ...: auth = OAuthHandler(ckey, csecret)
  ...: auth.set_access_token(atoken, asecret)
  ...:
  ...: twitterStream = Stream(auth, listener())
  ...: twitterStream.filter(track=["safaricom"])
  ...:
```

Figure 5: Classifier_twitanalysis

Figure 6: safaricom_tweets

```
In [10]: # %Load Classifier_wordCloudtweeps.py
   ...: import time
   ...: import pymysql
   ...: import time
   ...: import urllib.request, urllib.parse, urllib.error
   ...: import zlib
   ...: import string
   ...:
   ...: conn = pymysql.connect("localhost","root","","sentitwit", charset = 'utf8mb4')
   ...: conn.text_factory = str
   ...: cur = conn.cursor()
   ...:
   ...: cur.execute('SELECT time, username FROM safaricom')
   ...: subjects = dict()
   ...: for message_row in cur :
   ...:     subjects[message_row[0]] = message_row[1]
   ...: # print (message_row)
   ...:
   ...: # cur.execute('SELECT time,username, tweet, FROM sentitwit')
   ...: cur.execute('SELECT time FROM safaricom')
   ...: counts = dict()
   ...: for message_row in cur :
   ...:     text = subjects[message_row[0]]
   ...:     text = text.translate(string.punctuation)
   ...:     text = text.translate('1234567890')
   ...:     text = text.strip()
   ...:     text = text.lower()
   ...:     words = text.split()
   ...:     for word in words:
   ...:         if len(word) < 4 : continue
   ...:         counts[word] = counts.get(word,0) + 1
   ...:         print(counts[word])
   ...:
   ...: x = sorted(counts, key=counts.get, reverse=True)
   ...: highest = None
   ...: lowest = None
   ...: for k in x[:100]:
   ...:     if highest is None or highest < counts[k] :
   ...:         highest = counts[k]
   ...:     if lowest is None or lowest > counts[k] :
   ...:         lowest = counts[k]
   ...: print('Range of counts:',highest,lowest)
   ...:
   ...: # Spread the font sizes across 20-100 based on the count
   ...: bigsize = 80
   ...: smallsize = 20
   ...:
   ...: fhand = open('gword.js','w')
   ...: fhand.write("gword = [")
   ...: first = True
   ...: for k in x[:100]:
   ...:     if not first : fhand.write( ",\n")
   ...:     first = False
   ...:     size = counts[k]
   ...:     size = (size - lowest) / float(highest - lowest)
   ...:     size = int((size * bigsize) + smallsize)
   ...:     fhand.write("{text: '"+k+"', size: "+str(size)+"}")
   ...: fhand.write( "\n];\n")
   ...:
   ...: print("Output written to gword.js")
   ...:
```

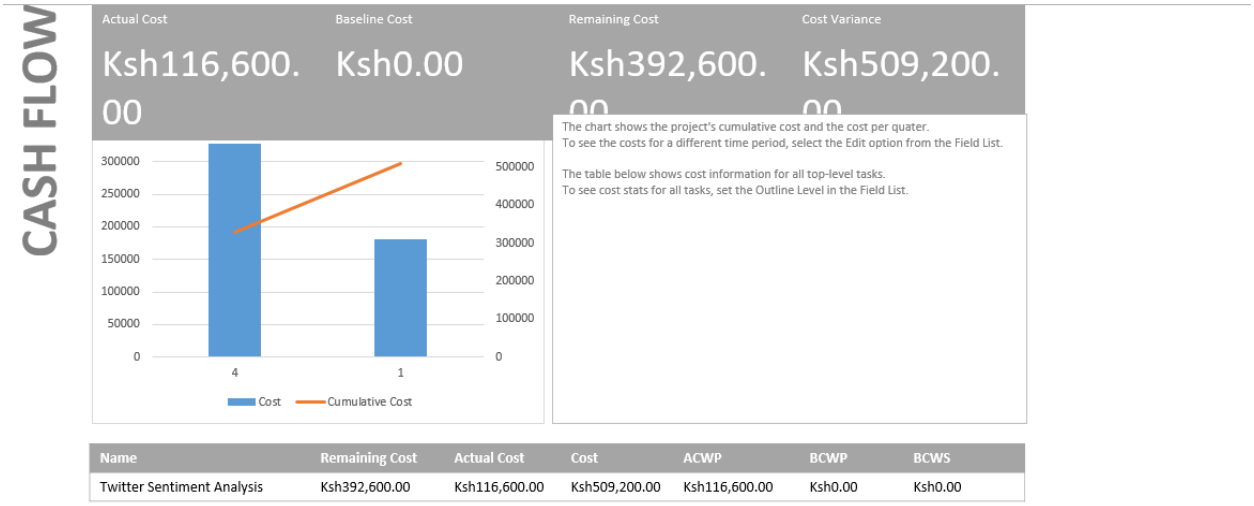
Figure 7:

```
In [10]: load safaricomdb.py

In [11]: # %load safaricomdb.py
    ...: # %load savingToDB.py
    ...: from tweepy import Stream
    ...: from tweepy import OAuthHandler
    ...: from tweepy.streaming import StreamListener
    ...: import pymysql
    ...: import time
    ...: import json
    ...: import sentiment_mod as s
    ...: import time
    ...: from urllib.error import HTTPError
    ...: from requests.exceptions import Timeout, ConnectionError
    ...: from requests.packages.urllib3.exceptions import ReadTimeoutError
    ...:
    ...:
    ...:
    ...: #        replace mysql.server with "localhost" if you are running via your own server!
    ...: #                        server       MySQL username   MySQL pass  Database name.
    ...: conn = pymysql.connect("localhost","root","","smm_sentiproject", charset = 'utf8mb4')
    ...:
    ...: c = conn.cursor()
    ...:
    ...:
    ...: #consumer key, consumer secret, access token, access secret.
    ...: ckey="P6kMnErRqCVqKpcYbOkLo5xLm"
    ...: csecret="km5IAnW06Nj2nxLseW1MX1B57faVOddkWLodC7TcvDviDv1v5c"
    ...: atoken="736849652-q4trxnc3TBXQcnySO9vSCVDV2DYYrwVeR7HHgxVy"
    ...: asecret="2yPbePhIKm09rwOiAlsNsiJeA6bR1xwQT7JnZNcJghH0y"
    ...:
    ...: class listener(StreamListener):
    ...:
    ...:     def on_data(self, data):
    ...:         all_data = json.loads(data)
    ...:
    ...:         tweet = all_data["text"]
    ...:
    ...:         username = all_data["user"]["screen_name"]
    ...:
    ...:         sentimentvalue, confidence = s.sentiment(tweet)
    ...:
    ...:         #"INSERT INTO mediciones_historial(column1,column2,column3) VALUES({0},{1},{2})".format(sensor, data, data_type)
    ...:
    ...:         c.execute("INSERT INTO safaricom (time,username,tweet,sentimentvalue,confidence) VALUES (%s,\"%s\",\"%s\",\"%s\",%s)",
    ...:             (time.time(),username,tweet,sentimentvalue,confidence))
    ...:
    ...:         conn.commit()
    ...:
    ...:         print((username,tweet,sentimentvalue,confidence))
    ...:         time.sleep(15)
    ...:
    ...:         return True
    ...:
    ...:     def on_error(self, status):
    ...:         print (status)
    ...:
    ...: auth = OAuthHandler(ckey, csecret)
    ...: auth.set_access_token(atoken, asecret)
    ...:
    ...: twitterStream = Stream(auth, listener())
    ...: twitterStream.filter(track=["safaricom"])
    ...:
    ...: |
```

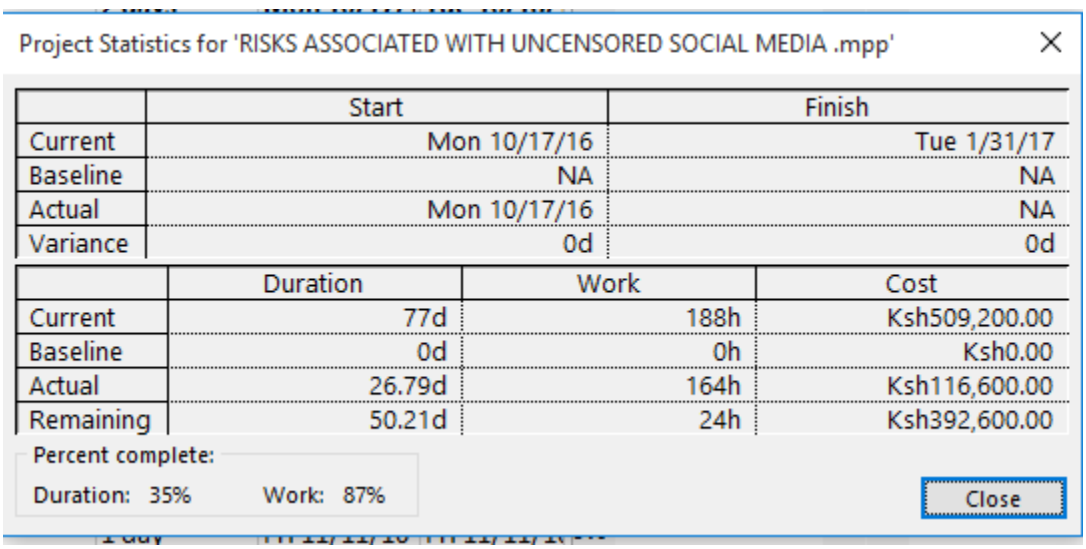
Classifier_Twitanalysis

Appendix 3



Name	Remaining Cost	Actual Cost	Cost	ACWP	BCWP	BCWS
Twitter Sentiment Analysis	Ksh392,600.00	Ksh116,600.00	Ksh509,200.00	Ksh116,600.00	Ksh0.00	Ksh0.00

Figure 8: Project cashflow diagram



Project Statistics for 'RISKS ASSOCIATED WITH UNCENSORED SOCIAL MEDIA .mpp'

	Start	Finish
Current	Mon 10/17/16	Tue 1/31/17
Baseline	NA	NA
Actual	Mon 10/17/16	NA
Variance	0d	0d

	Duration	Work	Cost
Current	77d	188h	Ksh509,200.00
Baseline	0d	0h	Ksh0.00
Actual	26.79d	164h	Ksh116,600.00
Remaining	50.21d	24h	Ksh392,600.00

Percent complete:

Duration: 35% Work: 87%

Close

Figure 9project Statistics

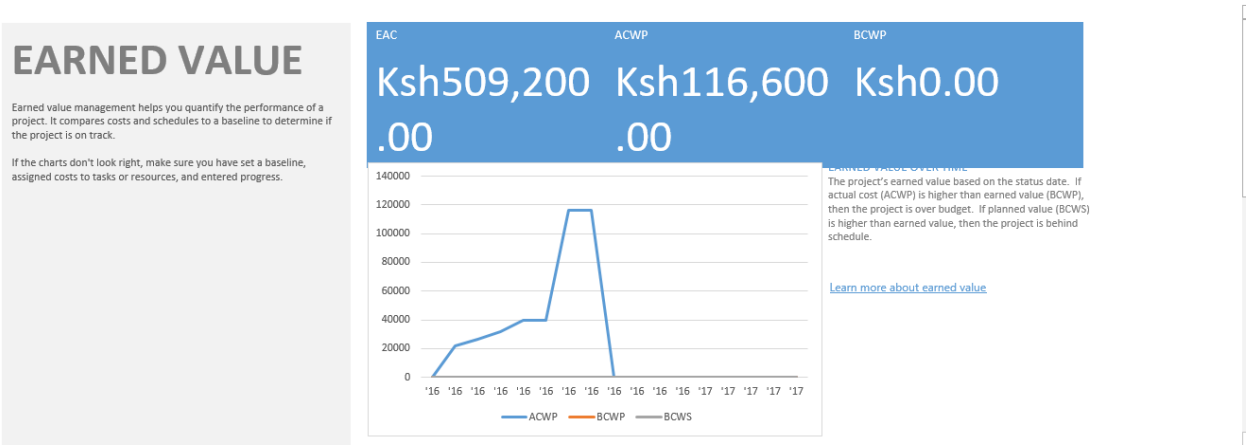


Figure 10: Project Burndown Chart