ECEN 602 - Project

Project Title - Analysis of Election Tweets to distinguish between campaigners and normal users

Vandana Bachani (UIN: 220009534)

Abstract

Networks are at the heart of every human phenomenon. One of the most recent discovery of network applications has been in the form of ones social and professional networks, which until recently were beyond the reach of computers and were discussed only in the realms of human psychology. With the onslaught of social networking websites like Facebook and Twitter, we have got the unique opportunity to view and analyze the characteristics of human networks. My project is about one particular aspect of human social networks, i.e. how different people utilize their social networks for their benefit or to promote their cause and what are the network characteristics which distinguish them from the regular users of the social networks.

Recently, the US presidential elections were held and obviously the online social networks were an important tool which influenced the results of the election, as they helped the candidates reach out to the masses beyond regular campaigning. There were several campaign accounts created on twitter to promote the candidates on the two sides (for President Obama and for Governor Romney). Besides the known campaign accounts there were implicit (non-funded) campaigners who tried to use the social networks to promote their respective candidates. In this project, I try to study what are distinguishing characteristics of these campaigners and their networks, which can help us identify these users from regular twitter users.

Introduction

All these years, human communication and networks, have inspired and have formed the basis for many theories of computer networks world. From protocols to flow of information, computer networks and human networks have lots of things in common. With the onslaught of social networking websites like Facebook and Twitter, we have got the unique opportunity to view and analyze the characteristics of human networks more closely and efficiently using the power of computer technology.

Though online social networks are a new phenomenon, they already have a huge set of associated problems such as spam, selfish promotion of personal motives, malicious users, etc. Many researchers [1][2] of the computer networks domain have been successful in applying the principles of network security and characteristics of flow of information in computer networks to detect such

users and prevent them from polluting the online social networks environment.

With better spam protection and increased safety measures, spammers and malicious users are trying to find new ways to get into the social circles of people to achieve their motives. A more sophisticated exploitation of social networks for the benefit of some groups of people in the society is the new kind of threat to social networks. A group of users who are trying to promote some cause for their benefit, proliferate in the social network, with the agenda to promote their cause by launching campaigns for the cause. Their activity is primarily focussed on their cause but they also demonstrate the characteristics of regular users which makes them hard to distinguish from the regular users. Campaigns such as promotion of celebrities by spreading rumors, promotion of candidates in elections, etc. are examples of campaigns for commercial benefit of a section of people in the society.

Recently, the US presidential elections were held and online campaigns were launched to promote the candidates. There were several campaign accounts created on twitter. Besides the known campaign accounts there were implicit (non-funded) campaigners who tried to use the social networks to promote their respective candidates. Several companies and researchers tried to predict the result of the elections using sentiment analysis and the tweeting behaviour of influential political and media people on twitter [3][4]. In this project, we try to analyze the characteristics which distinguish campaigners from regular users on Twitter.

Related Work

Twitter being the influential social network, is believed to have contributed significantly to the outcome of the US presidential elections. Several researchers and companies have tried the understand the impact of social networking and campaigning on the election outcome and tried to predict results based on the sentiment analysis of conversations of people on Twitter [3]. People have also created tools which could tell which side the person supports, thus predicting the strength of the social campaigning on the two sides [4].

The election campaigns are an expected phenomenon, but other campaigns such as spam campaigns are also prevalent phenomenon in social networks. We hypothesize that the campaigners share characteristics and spread information in the network using similar ways. Much research has been done to detect spam campaigns on twitter [5].

In this project, we are not trying to understand the impact of social campaigning or the influential social campaigners who impacted the election results. Instead, we are trying to analyze the characteristics of users who campaigners and try to distinguish them from normal users. We are starting with the fair assumption that during peak days of election campaigning a major part of it was being carried out on social networks where people explicitly and implicitly tried to promote their

own candidates. Using the baseline of an existing campaign, we try to distinguish campaigners from regular twitter users.

Objectives/Goals

1. Analyze the tweets from few months before the election and try to classify campaigners from regular/neutral users.
2. Arrive at a set of social network campaigner characteristics and try to use it to detect other kinds of campaigns in social networks.

Implementation and Experiments

The implementation can be divided into 3 parts:

1. Data collection and filtering
2. Extracting features and building the classifier
3. Testing/Validation

Tools Used: Python, MongoDB, scikit-learn, alchemyapi

Here is a description about the various aspects of the implementation:

Data Collection

Infolab crawlers collect tweets using the free twitter streaming api at all times. I took the data collected for the period (10/1/2012 - 11/06/2012) and filtered tweets related to US presidential election based on a set of filtering keywords (appendix B). From those tweets I extracted the user_ids who posted who posted those tweets. I saved user_ids and tweets for users with at least 3 or more tweets. This initial filtering gave me approx. 75000 user profiles.

I used the twitter search api then to crawl the user profile information of these ~75k users.
For getting the sentiment values I used the alchemyapi to query sentiment data for tweets of the ground truth and labelled users only due to time constraints.

Features Used

I used the following features for building the classifier to distinguish campaigners from neutral users in the twitter data from election days.

polarity: Its a measure of how polar a user is in terms of the variance in his polarity scores for the two candidates (Obama and Romney)
For a given user the polarity score of the user for each of the candidates is calculated as the number

of tweets mentioning the words related* to a particular candidate. divided by the total number of election tweets. (Some tweets may belong to scores of both the candidates).
I wanted to quantify a users favoritism towards a particular candidate using this metric but because we don't have sophisticated NLP classifiers which can tell whether a given piece of text favors one thing over the other or it actually promotes one class by demoting the other class, the above score is the closest thing I could come up with.

num_election_tweets: This feature is considered on the heuristic that campaigners tend to tweet more than regular users during the days of the election. It is possible that some active neutral users might be tweeting a lot too, but that might not be a pattern.

friends_count and **followers_count**: This feature is based on the heuristic that a regular user should have more friends than followers. (This heuristic doesn't hold well for journalists though)

avg_retweet_count: This feature captures how popular the user's retweets are. Every tweet in twitter has a retweet found associated with it. If a tweet by user (retweet or original) has a retweet count we account for that in calculate this feature. We assume that avid followers of politics and campaigners would tend to retweet less tweet count messages (not famous but important) but normal users/neutral users tend to retweet the more famous tweets.

num_retweets: Number of retweets tweets the user has posted during the election campaign. We assume that there is a difference in the retweeting behavior of neutral users and campaigners.

listed_count: the number of times the user has been listed in other people's tweets. The heuristic behind this feature is that campaigners may be listed more than neutral users.

avg_sentiment_value: what is the average sentiment value of the tweets posted by the user during the election days (if it very positive or very negative for one side only) then our heuristic is that they favor one candidate strongly over the other which could be indicative of campaigner behavior. But if the sentiment classifier classifies as mixed sentiments then it is difficult to capture them (this situation arises when user praises one side and at the same time says negative things about the other side candidate, which confuses the sentiment analyzer as to whether the overall tweet is positive of negative).

description, current_status and tweets_during_election themselves are very important features but running them along with above mentioned numeric features was suppressing the value of numeric

features, hence not used for the time being.

Ground Truth

For collecting some ground truth regarding campaigners and neutral users, we decided to pick user profiles of well known/publicized campaign accounts from both the sides of election campaigns such as @BarackObama, @RomneyResponse, etc. For collecting twitter handles of neutral users we looked at the accounts of journalists covering the election campaigns of both the sides. Journalists and news sites are very good examples of neutral users as they are supposed to present both sides of the presidential elections without any bias. We collected this information from random online sources.The website [4] gives some very important information. For other accounts I also confirmed their affiliation to the respective sides or neutrality by actually looking at the twitter description and other information online available about the handles.
We collected 30 accounts, 16 campaigners and 14 journalists which also featured in the dataset that I collected.

Labeling Users for the classifier: I randomly selected 135 accounts (including the above mentioned ground truth accounts) from my dataset and based on the tweets posted by those accounts assigned classes to those user profiles (campaigner/neutral).

Classification

I used decision tree based classification for classifying between campaigners and neutral users as my feature vectors contains features which cannot be normalized.

Experiments and Results

With the available dataset I tried various features which could produce a good classifier. I indeed dropped some features because of the indirection they create in the learning process. After a few runs and some manual feature selection, for the final results I settled with the features mentioned in the above section.

Average Precision and Recall values after 5-fold cross-validation:

classification report:

	precision	recall	f1-score	support
campaigners	0.77	0.62	0.69	16
neutral	0.57	0.73	0.64	11

avg / total	0.69	0.67	0.67	27

classification report:

	precision	recall	f1-score	support
campaigners	0.64	0.69	0.67	13
neutral	0.69	0.64	0.67	14
avg / total	0.67	0.67	0.67	27

classification report:

	precision	recall	f1-score	support
campaigners	0.65	0.73	0.69	15
neutral	0.60	0.50	0.55	12
avg / total	0.63	0.63	0.62	27

classification report:

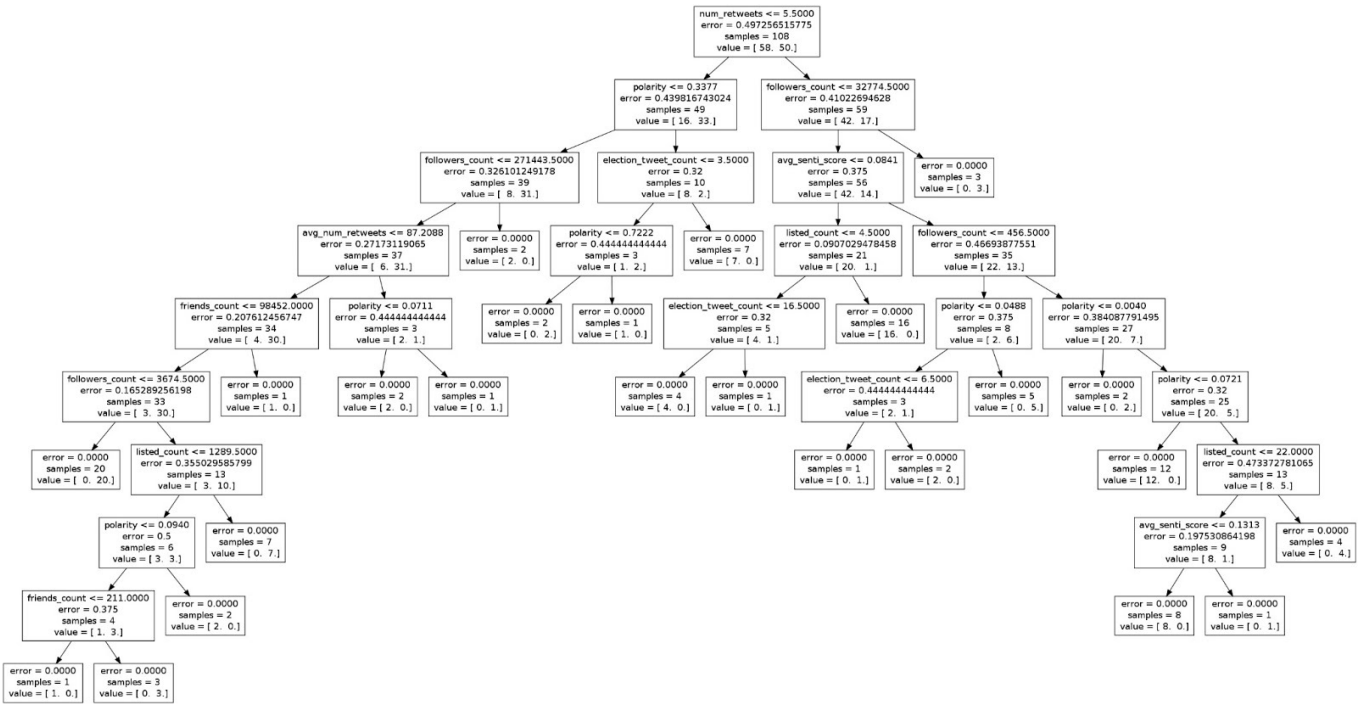
	precision	recall	f1-score	support
campaigners	0.72	0.93	0.81	14
neutral	0.89	0.62	0.73	13
avg / total	0.80	0.78	0.77	27

classification report:

	precision	recall	f1-score	support
campaigners	0.83	0.59	0.69	17
neutral	0.53	0.80	0.64	10
avg / total	0.72	0.67	0.67	27

Average	precision	recall	f1-score	support
campaigners	0.722	0.712	0.717	15
neutral	0.656	0.658	0.657	12

The decision tree obtained for the above run:



Appendix II discusses some interesting results for random ids from the dataset, along with the tweets to analyze whether the results obtained from the classifier are really good.

The above results indicate that there is need to experiment more before I can come up with a solid set of features for this problem. ‘polarity’, ‘num_retweets’, ‘num_election_count’ and ‘avg_senti_score’ are some of the important features that should be taken in all later versions of this problem.

Learnings and Challenges

The main learnings from this project are:

1. Applications of network theory concepts (especially the security and spam domain related concepts), to the world of social networks.
2. Challenges of identifying features for supervised learning based approach for the problem.

The main challenges that I faced when working on the project were:

1. Data collection for this project was a challenge in itself because of the twitter rate_limits for free usage. My crawler logic had to take care of that and patiently collect as much data as it

could. It took approximately 1-2 days to get all user profiles.

2. Its very hard to label user profiles based on the tweets and the user profile of a user on twitter.
3. Choice of the classification technique which is justified and works well for the dataset. I realized I could not use classifiers like SVM, k-nearestneighbors, etc as they depend on a distance metric and my feature vector is not normalized. So I chose Decision trees but I am not sure if thats the perfect choice to make.
4. Getting access to alchemyapi (student account) was difficult and took 5-6 days, which wasted a lot of time to collect the sentiment related data.

Improvements and Future Work

1. Label and collect more ground truth data and run the experiments with different features.
2. Try out other features derived from user tweet text, urls posted by user and user network information (friends and followers). Due to time constraints and complexity (involves additional crawling), I could not try some of the features such as url affiliation, num_democrat_followings, num_republican_followings, etc.
3. I was not able to accomplish the 2nd goal which is to identify the significant features. This could be done after I am comfortable with the results of my classification algorithm using metrics such as Chi-Square or other feature selection mechanisms.
4. It would be better to work with a 3-class classifier which tries to distinguish between campaigners, supporters and regular users. Difference between campaigners and supporters being that supporters happen to like one side more than the other and express their solidarity from time to time but do not want to change opinions of other people.
5. We need to find a good way to combine text features and numeric features (like counts etc). We could use ensemble classifiers or I am not aware if there exist any classifiers which can combine the features together (not being overwhelmed by the tf-idf based huge text features in the vector).
6. Need to find/implement NLP based classifiers or sentiment analyzers which can tell whether a tweet is for a particular class and against another class. An intelligent system which looks at the word associations and given the classes +obama, -obama, +romney, -romney should give us the combinations of two of these which can be a very important feature for distinguishing campaigners from neutral users. (example: if analyzer output is (+obama, -romney) => obama supporter, on the other hand combinations like (+obama, +romney), (-obama, -romney) and other mixed kind of pairs are more inclined to be characteristic of neutral users). This kind of analyzer can have very important applications in other scenarios too.

Acknowledgements

I would like to thank infolab@tamu for running their crawlers consistently and collecting twitter data on a daily basis, with the help of which I was able to get the data for doing this project.
I would also like to thank Prof. Reddy for the support and guidance he provided through the course of the project, especially the prompt emails and discussions on how to deal with the problems I faced through the course of the project.

References

1. Analyzing Spammer's Social Networks for Fun and Profit - http://faculty.cs.tamu.edu/guofei/paper/CyberEco_WWW12.pdf
2. Prof. Reddy's Paper
3. Deeper Insights on Electorate Perceptions: Connecting Candidates-Events-Topics Using Sentiments and Emotions - http://twitris.knoesis.org/election/
4. A tool built to predict affiliation of any user with the election candidates http://cs.uc.edu/twipolitico/
5. Content-Driven Detection of Campaigns in Social Media - http://students.cse.tamu.edu/kyumin../papers/lee11cikm.pdf
6. For accounts of journalists and campaigners http://www.geeksugar.com/Who-Follow-Twitter-Politics-25763281

Appendix I

Code

```
'''
Created on Nov 16, 2012

@author: vandana
Gets user_profiles from twitter user lookup api.
'''
import cjson
import httplib2
import sys
import time
import urllib
from utilities.twitter_request_session import RequestSession
from pymongo import Connection
```

```
class UserProfile:
  USER_LOOKUP_URL = 'http://api.twitter.com/1/users/lookup.json?' + \
      'screen_name={0}&include_entities=true'
  HTTP = httplib2.Http()
  @staticmethod
  def crawl(ids_file, output_file):
    f = open(ids_file, 'r')
    user_ids = [x.strip() for x in f.readlines()]
    f.close()
    f = open(output_file, 'w')
    c = 0
    num_userids = len(user_ids)
    request_ids = []
    rs = RequestSession()
    rs.start_session()
    for i in user_ids:
      request_ids.append(i)
      c += 1
      num_userids -= 1
      if c == 100 or num_userids == 0:
        rs.monitor_requests()
        response = UserProfile.lookup_user(request_ids)
        if response == None:
          print "error occurred. request cannot be completed."
          return
        print "processing response: ", rs.num_requests
        for profile in response:
          f.write(cjson.encode(profile)+'\n')
        request_ids = []
        c = 0

  @staticmethod
  def lookup_user(request_ids_list):
    str_ids_list = ','.join(request_ids_list)
                                          request_url           =
UserProfile.USER_LOOKUP_URL.format(urllib.quote_plus(str_ids_list))
    while True:
          response,  content  =  UserProfile.HTTP.request(request_url,
'GET')
      try:
        if response['status'] == '200':
          #print content
          return cjson.decode(content)
        elif response['status'] == '400':
          print "request monitoring not working..."
          print "response: ", response
          time.sleep(60*60) #sleep for 60 mins before continuing
        elif response['status'] == '502':
          time.sleep(120) #sleep for 2 mins server is busy
```

```
        else:
          print "error occurred. no response"
          print "response: ", response
          print "response content: ", content
          return
      except:
        return

def reduced_user_profile(input_file, output_file):
     f = open(input_file, 'r')
     f1 = open(output_file, 'w')
     for l in f:
          data = cjson.decode(l)
          towrite = {
               'followers_count': data['followers_count'],
               'location': data['location'],
               'statuses_count': data['statuses_count'],
               'description': data['description'],
               'friends_count': data['friends_count'],
               'screen_name': data['screen_name'],
               'favourites_count': data['favourites_count'],
               'name': data['name'],
               'url': data['url'],
               'created_at': data['created_at'],
               'listed_count': data['listed_count'],
               'id': data['id']
          }
          if 'status' in data:
               towrite['status'] = {'text':data['status']['text'],
                    'created_at': data['status']['created_at'],
                    'in_reply_to_screen_name':
data['status']['in_reply_to_screen_name'],
                    'entities': {'urls': [x['expanded_url'] for x in
data['status']['entities']['urls']],                    'hashtags':
data['status']['entities']['hashtags'],              'user_mentions':
[x['screen_name']               for               x               in
data['status']['entities']['user_mentions']]},
                    'retweet_count': data['status']['retweet_count']
               }
          f1.write(cjson.encode(towrite)+'\n')
     f1.close()
     f.close()

def write_profiles_to_db(filename):
     f = open(filename, 'r')
     conn = Connection('localhost', 27017)
     db = conn['election_analysis']
     for l in f:
          data = cjson.decode(l)
          data['_id'] = data['screen_name']
```

```
            db['user_profiles'].insert(data)
      f.close()

def main():
      """
  if len(sys.argv) < 2:
    UserProfile.crawl('user_ids.txt', 'user_profiles.txt')
  elif len(sys.argv) == 2:
    UserProfile.crawl(sys.argv[1], 'user_profiles.txt')
  else:
    UserProfile.crawl(sys.argv[1], sys.argv[2])
      """
      """
      reduced_user_profile('user_profiles.txt',
'user_profiles_shortened.txt')
      """
      #write profiles to db
      write_profiles_to_db('user_profiles_shortened.txt')

if __name__ == "__main__":
  sys.exit(main())


'''
user_features.py
@author: Vandana Bachani
'''
from __future__ import division
from pymongo import Connection
import cjson

class UserFeatures:
      DB_SERVER = "localhost"
      DB_PORT = 27017
      DB_NAME = "election_analysis"
      COLLECTION_UV = "user_vector"
      COLLECTION_UP = "user_profiles"
      MAJOR_ACCOUNTS = ['@TeamObama']
      def __init__(self, tweets_count_file):
            #<user_name,tweets_count> key value pairs
            self.tweet_counts = {}
            self.user_ids = []
            f = open(tweets_count_file, 'r')
            lines = f.readlines()
            for i in lines:
                  data = i.strip().split()
                  self.tweet_counts[data[0]] = int(data[1])
                  self.user_ids.append(data[0])
            f.close()
            self.connection = Connection(self.DB_SERVER, self.DB_PORT)
```

```
            self.db = self.connection[self.DB_NAME]

      def create_n_store_user_vectors(self):
            for i in self.user_ids:
                  user_obj = self.create_user_object(i)
                  if user_obj != None:
                        self.db[self.COLLECTION_UV].insert(user_obj)

      def create_user_object(self, user_id):
            user, user_profile_object = {}, None
            user['_id'] = user_id
            it = self.get_user_profile_object(user_id)
            for i in it:
                  user_profile_object = i
            if user_profile_object == None:
                  return None
            user['election_tweet_count'] = self.tweet_counts[user_id]
            tweets_info = self.get_tweets_info(user_id)
            user['tweets'] = tweets_info['tweets']
            user['favouring_tweet_count']                                  =
tweets_info['favouring_tweet_count']
            user['polarity'] = tweets_info['polarity']
            user['num_retweets'] = tweets_info['retweets']
            user['avg_num_retweets'] = tweets_info['avg_num_retweets']
            #user['avg_sentiment_value']                                   =
tweets_info['avg_sentiment_value']

            user['description'] = user_profile_object['description']
            #might not be useful
            user['statuses_count']                                         =
user_profile_object['statuses_count']
            user['friends_count']                                          =
user_profile_object['friends_count']
            user['followers_count']                                        =
user_profile_object['followers_count']
            user['favourites_count']                                       =
user_profile_object['favourites_count']
            user['listed_count'] = user_profile_object['listed_count']
            if 'status' in user_profile_object:
                  user['status']                                           =
user_profile_object['status']['text']
                  if is_election_related(user['status']):
                        user['status_variance'] = 0
                  else:
                        user['status_variance'] = 1
            #time of status or account creation might not be helpful

            #fill in fields from user profile features
            #urls, url_affiliation, latest_tweet_variation
```

```
            #lists campaigners as friends?
            #user['relation_with_major_players']  =  []  #0/1  relation
whether follows or not
            return user

      def get_tweets_info(self, user_id):
            tweets_file   =   '../data/results/user_tweets/%s.json'   %
user_id
            total, rt_from, avg_n_rts = 0, 0, 0
            obama_keywords = ['obama', 'democrat', 'biden']
            romney_keywords  =  ['romney',  'paul  ryan',  'republican',
'mitt']
            tweets_info = {}
            count_map = {'obama': 0.0, 'romney': 0.0}
            tweets = []

            f = open(tweets_file, 'r')
            lines = f.readlines()
            for l in lines:
                  data = cjson.decode(l)
                  tweets.append(data['tx'])
                  for ow in obama_keywords:
                        if ow in data['tx'].lower():
                              count_map['obama'] += 1
                              break
                  for rw in romney_keywords:
                        if rw in data['tx'].lower():
                              count_map['romney'] += 1
                              break
                  if 'rt_from' in data:
                        rt_from += 1
                  avg_n_rts += data['n_rts']
                  total += 1
            total *= 1.0
            tweets_info['tweets'] = tweets
            tweets_info['avg_num_retweets'] = avg_n_rts/total
            tweets_info['retweets'] = rt_from
            tweets_info['favouring_tweet_count'] = count_map
            tweets_info['polarity']              =             {'obama':
(count_map['obama']/total), 'romney': count_map['romney']/total}
            #tweets_info['avg_sentiment_value']
            return tweets_info

      def get_tweet_sentiment_value(self, user_id, user_tweets):
            #alchemy api call
            pass

      def get_user_profile_object(self, user_id):
            p = self.db[self.COLLECTION_UP].find({'_id': user_id})
            return p
```

```
def is_election_related(text):
     f = open('filter_list.txt', 'r')
     words = [x.strip() for x in f.readlines()]
     for w in words:
          if w in text.lower():
               return True
     return False

if __name__ == "__main__":
 uv = UserFeatures('high_vol_tweeters')
 uv.create_n_store_user_vectors()
```

alchemy api call

```
def get_sentiment():
     APIKEY = 'c8231ecbb8f6dbe91199d3318a6a604ac24e5da9'
     http = httplib2.Http()
     DBSERVER = "localhost"
     DBPORT = 27017
     DBNAME = "election_analysis"
     conn = Connection(DBSERVER, DBPORT)
     db = conn[DBNAME]
     f = open("labelled_users.txt", 'r')
     users = [x.strip() for x in f.readlines()]
     for u in users:
          it = db['user_vector'].find({'_id': u})
          for i in it:
               tweets = i['tweets']
               i['sentiments'] = []
               i['senti_score'] = 0
               for t in tweets:
                    try:
                         query                                          =
'http://access.alchemyapi.com/calls/text/TextGetTextSentiment?apikey=
{0}&text={1}&outputMode=json'.format(urllib.quote_plus(APIKEY),
urllib.quote_plus(t))
                         (header,  content)  =  http.request(query,
'GET')
                         if header['status'] == '200':
                              s = cjson.decode(content)
                              print s
                              if s['status'] == 'OK':

i['sentiments'].append(s['docSentiment'])
                                   if  s['docSentiment']['type']  ==
'neutral': continue
                                   else:
                                        if
s['docSentiment']['type'] == 'negative': s['docSentiment']['score'] =
```

```
float(s['docSentiment']['score'])*(-1)
                                        i['senti_score']        +=
float(s['docSentiment']['score'])
                        else:
                              print header
                    except:
                        continue
               db['user_vector'].update({'_id':    i['_id']},    i,
upsert=False)
     f.close()



'''
classifier.py
@author: Vandana Bachani
Classification of campaigners, supporters and neutral users from
Election Tweets
'''
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import tree
from sklearn import metrics
from pymongo import Connection
import random

class UserClassifier:
     DB_SERVER = "localhost"
     DB_PORT = 27017
     DB_NAME = "election_analysis"
     COLLECTION_UV = "user_vector"
     def __init__(self, labelled_user_ids_file, train_percent):
          f = open(labelled_user_ids_file, 'r')
          self.data_ids = [x.strip() for x in f.readlines()]
          random.shuffle(self.data_ids)

          num_ids = len(self.data_ids) * 1
          num_train = (train_percent/100.0) * num_ids
          self.train_ids   =   [self.data_ids[i].strip()   for   i   in
range(int(num_train))]
          #print self.train_ids
          self.test_ids   =   [self.data_ids[int(num_train+i)].strip()
for i in range(int(num_ids - num_train))]
          #print self.test_ids
          #self.features     =     ['polarity',     'statuses_count',
'election_tweet_count',      'friends_count',      'followers_count',
'listed_count',        'favourites_count',        'avg_num_retweets',
'num_retweets', 'avg_senti_score', 'status_variance']
          self.features   =   ['polarity',   'election_tweet_count',
'friends_count',          'followers_count',          'listed_count',
'avg_num_retweets',         'num_retweets',         'avg_senti_score',
```

```
'status_variance']

          self.connection = Connection(self.DB_SERVER, self.DB_PORT)
          self.db = self.connection[self.DB_NAME]

     def get_X_y(self, test_only=False):
          if not test_only:
               #training data
               user_vectors_train = []
               for i in self.train_ids:
                    it   =   self.db[self.COLLECTION_UV].find({'_id':
i})
                    for j in it:
                         user_vectors_train.append(j)
               """
               texts = []
               for i in user_vectors_train:
                    texts.append(''.join(i['tweets'])   +   '   '   +
i['description'] + ' ' + i['status'])
               self.vectorizer      =      TfidfVectorizer(min_df=5,
max_features=10000, stop_words='english')
               X_train = self.vectorizer.fit_transform(texts)
               print       'num       text       features:       ',
len(self.vectorizer.get_feature_names())
               X_tr = X_train.toarray()
               """
               X_tr, y_tr = [], []
               for i in user_vectors_train:
                    X_tr.append(self.get_non_text_features(i))
                    y_tr.append(i['class'])
                    #print X_tr, y_tr

          #test data
          user_vectors_test = []
          for i in self.test_ids:
               it = self.db[self.COLLECTION_UV].find({'_id': i})
               for j in it:
                    user_vectors_test.append(j)

          """
          texts = []
          for i in user_vectors_test:
               texts.append(''.join(i['tweets'])    +    '    '    +
i['description'] + ' ' + i['status'])
          X_test = self.vectorizer.transform(texts)
          X_te = X_test.toarray()
          """
          X_te, y_te = [], []
          for i in user_vectors_test:
               X_te.append(self.get_non_text_features(i))
```

```
                if(not test_only):
                        y_te.append(i['class'])
          if test_only:
                return {'train': {'X': self.train_data['X'], 'y':
self.train_data['y']}, 'test': {'X': X_te, 'y': y_te}}
          else:
                return {'train': {'X': X_tr, 'y': y_tr}, 'test':
{'X': X_te, 'y': y_te}}


     def initialize_data(self, test_only=False):
          X_y_dict = self.get_X_y(test_only)
          if not test_only:
                self.train_data = X_y_dict['train']
          self.test_data = X_y_dict['test']

     def train(self):
          self.clf = tree.DecisionTreeClassifier()
          self.clf       =       self.clf.fit(self.train_data['X'],
self.train_data['y'])
          out            =            tree.export_graphviz(self.clf,
out_file='decision_tree.dot', feature_names=self.features)
          out.close()

     def prediction(self):
          y_predicted = self.clf.predict(self.test_data['X'])
          print "classification report:"
          print   metrics.classification_report(self.test_data['y'],
y_predicted,

                                        target_names=['campaigners',
'neutral'])

     def get_non_text_features(self, user_vector):
          fs      =      [(user_vector['polarity']['obama']      -
user_vector['polarity']['romney'])**2,
     #                          user_vector['statuses_count'],
                                user_vector['election_tweet_count'],
                                user_vector['friends_count'],
                                user_vector['followers_count'],
                                user_vector['listed_count'],
     #                          user_vector['favourites_count'],
                                user_vector['avg_num_retweets'],
                                user_vector['num_retweets']]
          if 'senti_score' in user_vector:

fs.append(user_vector['senti_score']/user_vector['election_tweet_coun
t'])
          else:
                fs.append(0)
```

```
            if 'status_variance' in user_vector:
                  fs.append(user_vector['status_variance'])
            else:
                  fs.append(1)
            return fs

      def cross_validate(self):
            data_size = len(self.data_ids)
            set_size = data_size/5;
            sets = []
            start_idx = 0
            for i in range(5):
                  sets.append([self.data_ids[x]      for      x      in
range(start_idx, start_idx + set_size)])
                  start_idx += set_size
            for j in range(5):
                  self.test_ids = sets[j]
                  self.train_ids = []
                  for i in range(5):
                        if i == j: continue
                        self.train_ids.extend(sets[i])
                  self.initialize_data()
                  self.train()
                  self.prediction()

      def test(self,ids_to_test):
            self.test_ids = ids_to_test
            self.initialize_data(True)
            y_predicted = self.clf.predict(self.test_data['X'])
            return y_predicted


def main():
      uc = UserClassifier('labelled_users.txt', 60)
      '''uc.initialize_data()
      uc.train()
      uc.prediction()'''
      uc.cross_validate()

      #test random
      ids_to_test = ["RyanfromJersey", "SASSIE163", "Streetglidin09",
"StudSlayer",      "Stupefix_",      "StupidManWaka",      "SuJuNevvs",
"SuJu_Wifey",      "SuSu4u12",      "SuburbanLucy",      "Succubus_aoi",
"SuckkOnMyTweets",      "Sudieyk3",      "SueHM",      "SugarCocaine_",
"SujuFor_ELFindo",   "SulledOut",   "SultanAlQassemi",   "SummerDepp",
"Summer_Faery"]
      p = uc.test(ids_to_test)
      for i in range(len(p)):
            print ids_to_test[i], ": ",
            if p[i] == 0: print "Campaigner"
```

```
            else: print "Neutral"
            it   =   uc.db['user_vector'].find({'_id':   ids_to_test[i]},
{'tweets': 'true'})
            for i in it:
                 print i

if __name__ == "__main__":
     main()
```

Appendix II

Classification Result of running the classifier on random ids:

RyanfromJersey : Campaigner
sample tweet: u"@NatsGirl19 @TurtleZoot Our president can't even cure simple headaches and acid reflux. What a disgrace"

Streetglidin09 : Campaigner
Sample tweets: u'RT @edwrather: OBAMA LIES: The Health Care Package will pay for itself #tcot #ccot #teaparty #obamalies', u"RT @luvGodncountry: Never let a crises go 2 waste -Even self created ones that blow up in ur face #Libya Obama's use of Hillary 2 take t ...", u'RT @iowahawkblog: Obama using "gangbangers" reminds me of when Teddy Roosevelt went after "drugstore cowboys smoking Cubebs"', u"RT @Kimmi333: Let us not forget #Obama's Czar's

StudSlayer : Campaigner
Sample tweet: u'\u201c@HuffPostBiz: Wall Street average salary: $363,000 http:\\/\\/t.co\\/NVWpkIVj\u201d<~~middle class to Mitt Romney', u'BIDEN LOVES WOMEN!!!!!!!!!!!'

Stupefix_ : Neutral
Sample tweets: u'VOTEZ PLEASE : http:\\/\\/t.co\\/c8RIaPRK \u2665', u"@bintabarry3 Coucou :) \nTu peux voter s'il te plait ? http:\\\\\\/\\\\\\/t.co\\\\\\/c8RIaPRK :)",

StupidManWaka : Neutral
Sample tweets: u"RT @iAmNotA_Dyke: Seriously! If you don't like his politics, then just leave it there! RT @StupidManWaka: Romney is not the devil. Yall ...", u'RT @iSpeakYzairyen: #ParticKForPresident', u'RT @KBZO_: By the end of the week, we will either have a new president or a 2nd term for our current one. \\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8\\U0

001f1fa\\U0001f1f8\\U0001f1fa\\U0001f1f8',

SuJuNevvs : Neutral
Sample tweet: u'RT @SnowYeonKoh: Vote for Super Junior at Best Asia Act., EMA 2012. Come on ELF!', u'RT @lovemegege: RT @yeonjichi13: If you like Sorry, Sorry - Super Junior, VOTE here -\u25ba http:\\/\\/t.co\\/khfz4vEf currently #1 but still! RT pls'

SuJu_Wifey : Neutral
Sample tweet: u'@SuJu_wings RT vote Donghae use twitter \\/ FB \\/ mnet IDs, only few hrs left to vote: http:\\/\\/t.co\\/cEeCoIJs \u2026 \u2026 \u2026', u'@YESUNGFan_bot RT \ucd5c\uace0\uc758 \ubbf8\uc18c\ub97c \ubf51\uc544\ub77c!-\ub3d9\ud574. \uba87\uc2dc\uac04 \uc548\ub0a8\uc558\uad6c\uc694 \ud2b8\uc704\ud130,\ud398\uc774\uc2a4\ubd81,\uc5e0\ub137\uc544\uc774\ub514\ub97c \uc774\uc6a9\ud574\uc11c \ud22c\ud45c\ud560\uc218 \uc788\uc2b5\ub2c8\ub2e4! http:\\/\\/t.co\\/cEeCoIJs \u2026 \u2026 \u2026 \u2026 \u2026 \u2026 \u2026',

SuSu4u12 : Neutral
Sample tweet: u'RT @TrishMac777: @SuSu4u12 @News957 @TomAskForMore was great! I was listening! ;) #Halifax #hrmvotes', u'RT @LeanneMarriott: #kings students #ask4more accessible transit #hrmvotes #studentvotecounts #votehrm http:\\\\\\/\\\\\\/t.co\\\\\\/FtYT99SU'

Succubus_aoi : Neutral
Sample tweet: u"China's next President Xi Jinping is the leader of Chinese military that killed many Uighurs! http:\\\\\\/\\\\\\/t.co\\\\\\/dfNJtCCV @UN", u'Chinese government should not suppress free speech in China!@BarackObama', u'Chinese government should not suppress free speech in China! @BarackObama'

Sudieyk3 : Neutral
Sample tweet: u"Night Therapy 4'' Therapeutic Pressure Relief Memory Foam Topper - King Deals on... http:\\\\\\/\\\\\\/t.co\\\\\\/Iyyb9xDV", u'Better Home & Gardens Embroidered Ivory Throw Blanket to Enjoy Savings And... http:\\\\\\/\\\\\\/t.co\\\\\\/zoUIDRuI',

SueHM : Campaigner
Sample tweet: u'RT @LOLGOP: UPSIDE FOR THE GOP: 65% of voters said the Romney was much better at being rude to Candy Crowley.'

SujuFor_ELFindo : Neutral
Sample tweet: [u'[VOTE] 2012 MTV EMA VOTING INFO -- Bahasa Indonesian -- http:\\\\\\/\\\\\\/t.co\\\\\\/bukBgO1W (WorldwideELFs).hk.'

SulledOut : Campaigner

Sample tweet: 'RT @MOXXX: LA County Voters! Has your child asked about the @AIDSHealthcare & @YesonB erect rainbow condom billboards? What do you t ...'

SultanAlQassemi : Campaigner

{u'tweets': [u'Obama spent 20 hours in Africa in his four years as president - BBC World Service', u'RT @RawdaEg: About the US election. "How it looks from here: Gulf States", by @SultanAlQassemi | Open Democracy http:\\\\\\/\\\\\\/t.co\\\\\\/b7ulbMO5', u'RT @TheBoghdady: Yesterday, I had doubts that Obama will win and asked @SultanAlQassemi, and he said he most definitely is going to.'], u'_id': u'SultanAlQassemi'}

SummerDepp : Neutral

{u'tweets': [u'PRESIDENT Barack Obama', u'RT @ambriehlmonster: "electrical votes"', u'AYYOBAMA', u'Mitt has 1,720,105 followers on twitter. Barack has 21,860,146. lol'], u'_id': u'SummerDepp'}

Summer_Faery : Neutral

Sample tweet: u'RT @BrianHarnois: I can almost guarantee that if Obama told Romney that his tie was blue. Romney would rebuttal on how it is actually Aqua!!',