Personality Analysis Based on Social Media Data

By

Biswadeep Roy (CSE/2014/016)
Swarup Banerjee (CSE/2014/023)
Sayan Pal (CSE/2014/030)
Arpan Roy (CSE/2014/053)

UNDER THE GUIDANCE OF

Dr. Anup Kumar Kolya

PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

Session 2015-2016

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY
[Affiliated to West Bengal University of Technology]
CANAL SOUTH ROAD, BELIAGHATA, KOLKATA 700015

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

TO WHOM IT MAY CONCERN

I hereby recommend that the Project **Personality Analysis Based on Social Media Data** prepared under my supervision by **Biswadeep Roy (University Roll No: 11700114024 | Class Roll No: CSE/2014/016), Swarup Banerjee (University Roll No: 11700114086 | Class Roll No: CSE/2014/023), Sayan Pal (University Roll No: 11700114086 | Class Roll No: CSE/2014/030), Arpan Roy (University Roll No: 11700114018 | Class Roll No: CSE/2014/053)** of B.Tech (8^{th} Semester), may be accepted in partial fulfilment for the degree of **Bachelor of Technology in Computer Science & Engineering** under West Bengal University of Technology (WBUT).

.

.....

Project Supervisor
Department of Computer Science and Engineering
RCC Institute of Information Technology

Countersigned:

.....

Head
Department of Computer Sc. & Engg,
RCC Institute of Information Technology
Kolkata – 700015.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RCC INSTITUTE OF INFORMATION TECHNOLOGY

CERTIFICATE OF APPROVAL

The foregoing Project is hereby accepted as a credible study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not necessarily endorse or approve any statement made, opinion expressed or conclusion drawn therein, but approve the project only for the purpose for which it is submitted.

FINAL EXAMINATION FOR

EVALUATION OF PROJECT

1. ———————————

2. ———————————

(Signature of Examiners)

ACKNOWLEDGEMENT

We would like to express special thanks & gratitude to our guide, Dr. Anup Kumar Kolya who gave us this golden opportunity to work on this scalable project on the topic of "Personality Analysis Based on Social Media Data", which led us into doing a lot of Research which diversified our knowledge to a huge extent for which we are thankful.
Also, we would like to thank our parents and friends who supported us a lot in finalising this project within the limited time frame.

Biswadeep Roy (CSE/2014/016)

Swarup Banerjee (CSE/2014/023)

Sayan Pal (CSE/2014/030)

Arpan Roy (CSE/2014/053)

Table of Contents

Abstract

Social media is a place where users present themselves to the world, revealing personal details and insights into their lives. We are beginning to understand how some of this information can be utilized to improve the users' experiences with interfaces and with one another. In this paper, we are interested in the personality of users. Personality has been shown to be relevant to many types of interactions; it has been shown to be useful in predicting job satisfaction, professional and romantic relationship success, and even preference for different interfaces. Until now, to accurately gauge users' personalities, they needed to take a personality test. This made it impractical to use personality analysis in many social media domains. In this paper, we present a method by which a user's personality can be accurately predicted through the publicly available information on their Twitter tweets. We will describe the type of data collected, our methods of analysis, and the results of predicting personality traits through machine learning. We then discuss the implications this has for social media design, interface design, and broader domains.

Introduction

From the day of start of social media, a prodigious amount of status updates, tweets, and comments have been posted online. The language people use to express themselves can provide clues about the kind of people they are, online and off. Current efforts to understand personality from writing samples rely on theories and survey data from the 1980s. New research from the New England Complex Systems Institute (NECSI) uses social media data to successfully identify differences and similarities between users without prior assumptions. This is a step toward building a better understanding of the psychology of human personality.

Personality psychologists study available social media data in addition to solicited surveys. However, they still start with predefined traits like extroversion, neuroticism or narcissism and correlate them with the writing. In other research, linguists have used algorithms to identify topics of conversation, but they do not have much to say about the personalities of the conversationalists. NECSI's approach uses an unguided process that identifies word usage patterns among individual users without prior assumptions.

The data as clusters of words frequently used by similar subsets of users. These word clusters include interests or hobbies like "hockey," "global politics," and "video games." Other topics have no obvious theme. Because these topics emerge from the data through an unguided process, they are not biased by prior assumptions.

Importantly, this analysis preserves the complex relationships between topics and pronoun use. For example, a cluster of users was identified with the topic "hockey." They frequently used words like NHL, puck, ice, Bruins, and Canucks. These users also typed a lot of third person male pronouns (he, his, him), reflecting the male-dominance of the sport. They also frequently used first person plural pronouns (we, us, our), suggesting a focus on teamwork. "Understanding the ways people can be different from each other is one of the most exciting topics in science," NECSI president and an author of the paper, Prof. Yaneer Bar-Yam said. "This paper shows how we can make progress."

These preliminary findings establish the potential for identifying differences between individuals from abundant social media data. The words people use online can tell us about their patterns of behaviour. This analysis can also lead to a better understanding of personality, informing existing psychological models.

The contribution of our work is listed as follows:

•We compile a Twitter dataset with around 80,000 users by extracting and filtering all random tweets on Twitter from 2017 to 2018. The dataset contains not only the personality types, but also the most recent tweets for all the 80,000 users, which creates enormous opportunities to study the relation- ship between tweets and personality.

•We design and implement three categories of linguistic features based on tweets, and further explore the correlations between each linguistic feature and each personality type. Several interesting findings can be observed here. For instance, extroverts tend to use hashtag and phrases like ”so proud”, ”so ex- cited”, and ”can’t wait”. People who like to use emoticon are more likely to be Sensing and Feeling personality type.

•We investigate the predictive powerful of the three categories of linguistic features we design by predicting each personality trait respectively. With the combination of all the three categories of features, we can predict- ing between introversion and extroversion with an AUC of 0.691 and an average AUC 0.661 of all personality traits.

Review of Literature

Personality prediction is a task where information about an individual's personality trait is identified, given a set of data. There have been several approaches on automated personality prediction based on different kinds of dataset, such as essays, social media posts, videos, and social media behaviour. This paper will only focus on studies of personality prediction from text based on social media posts. There are several tools and corpora which are widely used in personality prediction studies, including the ones mentioned in this research. The first tool is called LIWC (Linguistic Inquiry Word Count), which is a text analysis tool used to evaluate psychological properties from language. The LIWC tool supports several languages such as Arabic, Chinese, Dutch, English, German, Italian, Korean, Norwegian, Portuguese, and Spanish. Approaches using the LIWC tool are often referred to as closed-vocabulary approaches or category-based analysis. Some other tools used in the closed-vocabulary approach are MRC, NRC, SentiStrength and SPLICE. The second, while not exactly a tool, which is commonly used in the personality prediction task is the MyPersonality corpus. It contains records about psychometric scores and social media posts from Facebook users.

A. Early Research on Blogs

One of the earliest research regarding personality traits and social media text was done on Blogger [2]. The objective of this study was to find the correlations between personality traits and social media text. The dataset used contained 694 blogs from Google's Blogger service. The personality model used in this study are The Big Five and NEO-PI-R. Yarkoni used both closed and open vocabulary approaches for the personality prediction model. 66 LIWC categories were used for the closed vocabulary approach, while the open vocabulary consisted of dividing the text into individual words. His results showed that Openness correlated with 393 words, whereas the other traits correlated with fewer than 30 words. Lower order facets from the NEO-PI-R models were also found to correlate with categories from the LIWC tool.

B. Twitter Dataset

Another research was attempted to identify personality traits of Twitter users based on The Big Five model [3]. The features used in their personality prediction model are a set of Twitter statistics (data that is already available from each Twitter user, such as number of followers, following, mentions, etc.) and language features. They utilized 79 text features from LIWC and 14 text features from MRC. Sentiment analysis was also done word by word to each user's tweets. The data is then run in Weka using Gaussian Process and ZeroR. Evaluation was done by calculating the Mean Absolute Error (MAE) for each personality trait. They managed to obtain the smallest MAE with a value of 0.11923333 for the Openness trait by using the ZeroR

algorithm. A similar task was conducted to find correlations between The Big Five personality traits and topics, posting platforms, and the tendency of a user to retweet [1]. Unlike Golbeck's research, Celli only made use of 12 cross-linguistic features based on a previous research [4]. Another research was conducted by using the Dark Triad personality model with 2,927 Twitter users [5]. 337 features were selected for the personality prediction task, consisting of Twitter statistic data and frequency of pre-defined words for each individual. The prediction task was then run with 4 algorithms from WEKA: Support Vector Machine, Random Forest, J48, and Naïve Bayes. Personality prediction has also been conducted in a semi-supervised way, with Brazilian TV shows as an additional label [6]. In their study, they used a list of meta-attribute features, which was then run using a Naïve Bayes classifier with a supervised and semi-supervised learning approach. Results showed that the semi-supervised learning outperforms supervised learning, with 0.8415 as their highest accuracy.

C. Blogger Dataset

Further research on personality prediction via social media was attempted on Blogger [7]. This study also utilizes texts from bloggers, but further improves Yarkoni's study by doing a personality prediction task based on the selected features. In addition, he also did a comparison of performances achieved between different approaches of the personality model. The different approaches used are: (a) different n-grams (n=1 or n=2), (b) utilization of stop words (using stop words or omitting stop words), and (c) term weighting (Boolean weighting or TF-IDF weighting. These approaches were also compared to the performance when using the LIWC tool. These features were then classified in Weka using Support Vector Machine (SVM). The experiment's results showed that the best accuracy, with a value of 84.36%, was achieved by using bigrams (n=2), utilizing stop words and implementing Boolean weighting. This proves that the open vocabulary approach (by extracting n-gram tokens) can be used to predict personality, since it outperforms the closed vocabulary approach (using LIWC). However, they also mention that the classification may have overfitted, due to the few amount of bigrams in each personality trait.

D. Facebook Dataset

Personality prediction was also attempted on Facebook. One of these studies utilized a Facebook dataset named MyPersonality corpus [8]. This study attempted to perform the personality prediction task using an open-vocabulary approach. Significant features were found between n-grams (n=1 to 3), extracted topics with Latent Dirichlet Allocation and personalities. The models created with these features outperformed the model created based on LIWC. Another open vocabulary approach on The Five Factor Model personality prediction using the MyPersonality dataset was conducted using Support Vector Regression and Latent Dirichlet Allocation models [9]. Their results show that the LDA models, sLDA (supervised Latent Dirchlet Allocation) and PT-LDA (Probabilistic Topic model-Latent Dirichlet Allocation) outperforms the Support Vector Regression model (topics and N-grams). Furthermore, they also proved that PTLDA is more robust and improves computational efficiency up to 64%. Meanwhile, another attempt at closed-vocabulary approach was done by using the LIWC tool [10]. This study used 81 LIWC features, 7 social network features, 6 time-related features, and 6 other features that can be extracted from

the posts' content. The learning algorithms tested on this study are Support Vector Machine, K-Nearest Neighbor, and Naïve Bayes. The highest precision achieved by combining all these features using K-Nearest Neighbour is 0.54, while highest precision was obtained by merely using social network features, reaching a precision of 0.71.

Possible Developments for Personality Prediction Task

This section provides an outline about further improvements that can be applied to the personality prediction task from text on social media. The first improvement that can be made to the personality prediction task is developing methods of said task for non-English language. This is in accordance with the second issue mentioned in the previous section, where not all languages are supported by LIWC. Secondly, improvements to the research can be done by exploring more methods to achieve higher accuracy than the current state-of-the-art research. The improvements may include more suitable machine learning algorithms, feature selection on more significant features on social media posts or methods to preprocess the dataset. The improvement on methods to preprocess the dataset is in accordance with the third issue mentioned in the previous section. As mentioned in [11], dealing with multilingual, noisy, short, and informal social media posts can result in a better personality prediction model. Lastly, while the mostly used model for personality prediction is the Five Factor model or The Big Five, further developments may include taking the Five Factor Model 30 facets into consideration, or conducting personality prediction of other personality models, such as the Dark Triad personality model which was implemented in [5].

[1] F. Celli, "Mining user personality in twitter," Lang. Interact. Comput.
CLIC, 2011.
[2] T. Yarkoni, "Personality in 100,000 words: A large-scale analysis of personality and word use among bloggers," J. Res. Pers., vol. 44, no. 3, pp. 363–373, 2010.
[3] J. Golbeck, C. Robles, M. Edmondson, and K. Turner, "Predicting personality from twitter," in Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on, 2011, pp. 149–156.
[4] F. Mairesse, M.A. Walker, M.R. Mehl, and R.K. Moore, "Using linguistic cues for the automatic recognition of personality in conversation and text," J. Artif. Intell. Res., vol. 30, no. 1, pp. 457–500, 2007.
[5] C. Sumner, A. Byers, R. Boochever, and G.J. Park, "Predicting dark triad personality traits from twitter usage and a linguistic analysis of tweets," in ICMLA '12 Proceedings of the 2012 11th International Conference on Machine Learning and Applications, 2012, vol. 2, pp. 386–393.
[6] A.C.E.S. Lima and L.N. de Castro, "Multi-label Semi-supervised Classification Applied to Personality Prediction in Tweets," in 2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence, 2013, pp. 195–203.

[7] F. Iacobelli, A.J. Gill, S. Nowson, and J. Oberlander, “Large Scale Personality Classification of Bloggers,” in Affective Computing and Intelligent Interaction: Fourth International Conference, ACII 2011, Memphis, TN, USA, October 9--12, 2011, Proceedings, Part II, S. D’Mello, A. Graesser, B. Schuller, and J.-C. Martin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 568–577
[8] H.A. Schwartz, J.C. Eichstaedt, M.L. Kern, L. Dziurzynski, S.M. Ramones, M. Agrawal, A. Shah, M. Kosinski, D. Stillwell, M.E.P. Seligman, and L.H. Ungar, “Personality, gender, and age in the language of social media: The open-vocabulary approach,” PLoS One, vol. 8, no. 9, p. e73791, 2013.
[9] Y. Liu, J. Wang, and Y. Jiang, “PT-LDA: A Latent Variable Model to Predict Personality Traits of Social Network Users,” Neurocomputing, vol. 210, pp. 155–163, 2016.
[10] G. Farnadi, S. Zoghbi, M.F. Moens, and M. De Cock, “Recognising personality traits using Facebook status updates,” in Proceedings of the workshop on computational personality recognition (WCPR13) at the 7th international AAAI conference on weblogs and social media (ICWSM13), 2013.
[11] M. Arroju, A. Hassan, and G. Farnadi, “Age, Gender and Personality Recognition using Tweets in a Multilingual Setting,” in 6th Conference and Labs of the Evaluation Forum (CLEF 2015): Experimental IR meets multilinguality, multimodality, and interaction, 2015

Objective of the Project

- Data mining of twitter live tweets and generation of a csv data set.
- Calculation of the MBTI index through POS tagging.
- Calculation of the MBTI index using the count of emoticon, links and hashtags.
- Calculation of the MBTI index from Bigram Tagging.
- Normalised value of the MBTI index calculated from the three above methods.
- Plotting of the graph based on the MBTI values.
- Verification of the algorithm with the test data set.

Data set details

Variable	Details
# of users	9,471
# of tweets	281945

System Design

Hardware Requirements:

- Core i5/i7 processor
- At least 4 GB RAM
- At least 30 GB of Usable Hard Disk Space

Software Requirements:

- Python 3.x
- Anaconda Distribution
- NLTK Toolkit
- UNIX/LINUX Operating System

Data Information:

Data used in this project was captured from live twitter data by data mining using python.

Methodology for Implementation (Formulation/Algorithm)

We will use a instance of status that is posted by users on social media on daily basis as a data set. We will take a exact number of status of each user. One of the major form of pre-processing of the data is to the filter out the useless data. Filtering data set includes removal of stop words, url and unicode. Before removing these useless words, we have to word tokenise the dataset.

Tokenising: Word Tokenising is splitting the text into words. NLTK provides a number of tokenisers in the tokeniser module. Word_tokenize tokeniser has been used to tokenise the text. For example,
Sample=" I am a good boy"
Tokenize=['I', 'am', 'a', 'good' , 'boy']

Cleaning Data Set: A stop word is a commonly used word (such as "the", "a", "an", "in") that can be ignored as it has no meaning after tokenising. Also, the dataset may contain unicode (i.e- by which each letter, digit, or symbol is assigned a unique numeric value that applies across different platforms and programs) and URL. As the further process can't be applied on this, it can be ignored.
Parts of Speech Tagging: The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. Parts of speech are also known as word classes. The collection of tags used for a particular task is known as a tag set.

There is a list of POS tag that is available in NLTK:
CC coordinating conjunction
CD cardinal digit
DT determiner
EX existential there
FW foreign word
IN preposition/subordinating conjunction
JJ adjective 'big'
JJR adjective, comparative
JJS adjective, superlative
LS list marker
MD modal could
NN noun, singular

NNS noun plural
NNP proper noun, singular
NNPS proper noun, plural
PDT predeterminer
POS possessive
PRP personal pronoun
PRP$ possessive pronoun
RB adverb
RP particle
UH interjection
VB verb, base form
VBD verb, past tense
VBG verb, gerund/present participle
VBN verb, past participle
VBP verb, present
VBZ verb, 3rd person sing. present
RBR adverb, comparative
For example,
text=nltk.word_tokenize("We are going to park") print nltk.pos_tag(text)
[('We', 'PRP'), ('are', 'VBP'), ('going', 'VBG'), (‘to’,’IN’),(‘park’,’NN’)]

Evaluating Personality Index: The usage of words of a user i.e- repetitions of words, kind of word using, using emoticons reflects one’s personality.
There are POS tags and their correlations with each personality trait ($p < 0.001$). Negative values mean that POS tags are positively correlated with I, S, T, or J, while positive values mean that POS tags are positively correlated with E, N, F or P. The top 3 positively correlated and negatively correlated POS tags.

The description of POS tags and their correlations with each personality trait (*** $p < 0.001$). Negative values mean that POS tags are positively correlated with I, S, T, or J, while positive values mean that POS tags are positively correlated with E, N, F or P. The top 3 positively correlated and negatively correlated POS tags are bold for each personality trait.

Tag	Description	I/E	S/N	T/F	J/P
	Nominal, Nominal+Verbal				
N	common noun (NN, NNS)	0.0474***	0.0535***	-0.0509***	-0.0586***
O	pronoun (personal/WH; not possessive)	-0.0715***	-0.0191***	0.1014***	0.0350***

^	proper noun (NNP, NNPS)	-0.0014	-0.0193***	-0.0788***	0.0468***
S	nominal + possessive	0.0179***	0.0205***	0.0064	-0.0406***
Z	proper noun + possessive	0.0260***	0.0202***	-0.0156***	-0.0357***
	Other open-class words				
V	verb incl. copula, auxiliaries (V*, MD)	-0.0299***	0.0081	0.0534***	-0.0142***
A	adjective (J*)	0.0037	0.0440***	0.0340***	-0.0388***
R	adverb (R*, WRB)	-0.0767***	-0.0121***	0.0702***	0.0145***
!	interjection (UH)	-0.0458***	-0.0592***	0.0274***	0.0639***
	Other closed-class words				
D	determiner (WDT, DT, WP$, PRP$)	0.0204***	0.0416***	0.0401***	-0.0519***
P	pre- or postposition, or subordinating conjunction (IN, TO)	0.0541***	0.0492***	0.0128	-0.0761***
&	coordinating conjunction (CC)	-0.0381***	0.0146***	0.0763***	-0.0090
T	verb particle (RP)	0.0197***	-0.0168***	0.0306***	-0.0204***
X	existential there, predeterminers (EX, PDT)	-0.0208***	0.0228***	0.0243***	-0.0108
	Twitter/online-specific				
#	hashtag (indicates topic/category for tweet)	0.0912***	-0.0305***	0.0225***	-0.0252***
@	at-mention (indicates another user as a recipient of a tweet)	0.0082	-0.0080	-0.0143***	0.0092
~	discourse marker, indications of continuation of a message across multiple tweets	-0.0324***	0.0034	0.0130***	0.0276***
U	URL or email address	0.0432***	0.0005	-0.0038	0.0036

E	emoticon	-0.0139***	-0.0546***	0.0744***	0.0025
	Miscellaneous				
$	numeral (CD)	0.0283***	-0.0502***	-0.0461***	-0.0031
,	punctuation	0.0455***	0.0365***	-0.0377***	-0.0741***
G	other abbreviations, foreign words, possessive endings, symbols, garbage (FW, POS, SYM, LS)	-0.0423***	-0.0197***	-0.0297***	0.0518***
	Other Compounds				
L	nominal + verbal; verbal + nominal	-0.0808***	-0.0268***	0.0632***	0.0417***
M	proper noun + verbal	0.0042	-0.0064	0.0028	0.0035
Y	X + verbal	-0.0256***	0.0002	0.0046	0.0172***

Implementation Details

1. Mining Twitter Data with Python

In order to have access to Twitter data programmatically, we need to create an app that interacts with the Twitter API.
The first step is the registration of your app. In particular, you need to point your browser to http://apps.twitter.com, log-in to Twitter (if you're not already logged in) and register a new application. You can now choose a name and a description for your app (for example "Mining Demo" or similar). You will receive a consumer key and a consumer secret: these are application settings that should always be kept private. From the configuration page of your app, you can also require an access token and an access token secret. Similarly to the consumer keys, these strings must also be kept private: they provide the application access to Twitter on behalf of your account. The default permissions are read-only, which is all we need in our case, but if you decide to change your permission to provide writing features in your app, you must negotiate a new access token.

```
import tweepy
import csv
consumer_key = "PQZB2GMwSjlcaIzGP1brJiIER"
consumer_secret = "WFSmWsg7PjRtyf73qLvgOM8mT3znvlAjg6LzhRS94scATPrIES"
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
api = tweepy.API(auth)
csvFile = open('testr.csv', 'a', newline='',encoding='utf-8')
csvWriter = csv.writer(csvFile)

for tweet in tweepy.Cursor(api.search,
                           q = "google",
                           lang = "en").items(10):
    u_id = tweet.user.id
    #print (tweet.user.id)

    for tweet in tweepy.Cursor(api.user_timeline,id=u_id).items(20):
        csvWriter.writerow([tweet.user.name, tweet.user.id, tweet.text.encode('utf8')])
```

2. Steps for data cleaning

1. Escaping HTML characters: Data obtained from web usually contains a lot of html entities like b'RT, @, RT, \\n; which gets embedded in the original data. It is thus necessary to get rid of these entities. One approach is to directly remove them by the use of specific regular expressions.

Another approach is to use appropriate packages and modules (for example htmlparser of Python), which can convert these entities to standard html tags.

2. Decoding data: This is the process of transforming information from complex symbols to simple and easier to understand characters. Text data may be subject to different forms of decoding like “Latin”, “UTF8” etc. Therefore, for better analysis, it is necessary to keep the complete data in standard encoding format. UTF-8 encoding is widely accepted and is recommended to use.

3. Apostrophe Lookup: To avoid any word sense disambiguation in text, it is recommended to maintain proper structure in it and to abide by the rules of context free grammar. When apostrophes are used, chances of disambiguation increases.

4. Identification of emoticons: Emoticons were identified from the raw data set and were replaced by `-emote-` in the specified location.

5. Identification of urls: URLs were identified from the raw data set and were replaced by `_link_` in the specified location.

6. Identification of tags and hashtags: Tags and hashtags were identified from the raw data set and were replaced by `-garbage-` in the specified location.

3. Tokenising of raw data

```
import csv
from nltk.tokenize import word_tokenize
j=0
user_id=[0]
stop_words=["b'RT",'@','_link_','b',"'",'```',"'",'"','`-emote-``',';','//','RT','\\n','`-garbage-`','-emote-','`-emote-','-emote-`','-garbage-','-garbage-``','`-garbage-']
files=open("tokenized_Data.txt",'a+')
with open('CleanedData.csv','r') as f:
    reader = csv.reader(f)
    for row in reader:
        temp=row[0]
        if(temp != user_id[-1]):
            user_id.append(temp)
            j=j+1
    del user_id[0]
    del user_id[0]
    j=j-1
    f.seek(0)
    tweet=[]
    for row in reader:
        for i in range(0,j):
            if(user_id[i]==row[0]):
                tweet.append(row[1])
                files.write(str(user_id[i]))
                files.write("+++")
                word_tokens=word_tokenize(str(tweet[-1]))
                print(word_tokens)
                filtered_sentence = []
                for w in word_tokens:
                    if w not in stop_words:
                        filtered_sentence.append(w)
                files.write(str(filtered_sentence))
                print(filtered_sentence)
                files.write("\n")
```

4. Parts-of-speech (POS) tagging

It is one of the many tasks in NLP. It is defined as the process of assigning a particular parts-of-speech tag to individual words in a sentence. The parts-of-speech tag identifies whether a word is a noun, verb, adjective, and so on. There are numerous applications of parts-of-speech tagging, such as information retrieval, machine translation, NER, language analysis, and so on.

```
from nltk.tag import pos_tag
f=open("tokenized_Data.txt",'r')
files=open("POSTagged.txt",'a+')
list1=[]
for line in f:
    list1.append(line)
for line in list1:
    string=str(line)
    user_id,tweet = string.split('+++')
    files.write(user_id)
    files.write("+++")
    #print(tweet[0:-1])
    lst = eval(tweet[0:-1])
    #print(li)
    tweets_tagged = pos_tag(lst)
    #print(tweets_tagged)
    files.write(str(tweets_tagged))
    files.write("\n")
```

5. Counting of the POS tagged list of words:

```
f=open("tags.txt",'r')
tag=[]
dict={}
for line in f:
    string=line
tag=list(string.split(' '))
#print(tag)
for i in tag:
   dict[i]=0
f1=open("POSTagged.txt",'r')
files=open("POS_count.txt",'a+')
list1=[]
for line in f1:
    list1.append(line)
for line in list1:
    string=str(line)
    user_id,tweet = string.split('+++')
    files.write(user_id)
    files.write("+++")
    #li=eval(tweet[0:-1])
    #print(li)
    li=list(tweet[0:-1].split("'"))
    for i in li:
        for j in tag:
            if(i==j):
                dict[i]+=1
                #print(i)
    files.write(str(dict))
    print(dict)
    files.write("\n")
    for i in tag:
        dict[i]=0
```

6. MBTI Index of tweets

The MBTI Manual states that the indicator "is designed to implement a theory; therefore, the theory must be understood to understand the MBTI". Fundamental to the MBTI is the theory of psychological type as originally developed by Carl Jung. Jung proposed the existence of two dichotomous pairs of cognitive functions:

- The "rational" (judging) functions: thinking and feeling
- The "irrational" (perceiving) functions: sensation and intuition

Jung believed that for every person, each of the functions is expressed primarily in either an introverted or extraverted form. Based on Jung's original concepts, Briggs and Myers developed their own theory of psychological type, described below, on which the MBTI is based. However, although psychologist Hans Eysenck called the MBTI a moderately successful quantification of Jung's original principles as outlined in Psychological Types, he also said, "[The MBTI] creates 16 personality types which are said to be similar to Jung's theoretical concepts. I have always found difficulties with this identification, which omits one half of Jung's theory (he had 32 types, by asserting that for every conscious combination of traits there was an opposite unconscious one). Obviously, the latter half of his theory does not admit of questionnaire measurement, but to leave it out and pretend that the scales measure Jungian concepts is hardly fair to Jung." In any event, both models remain hypothetical, with no controlled scientific studies supporting either Jung's original concept of type or the Myers–Briggs variation.

```
import ast
import csv
u_id=[0]
i=-1
cnt=0
f=open("MBTI_Indiviual.txt",'r')
for line in f:
    user_id,index=line.split('+++')
    if(u_id[-1]!=user_id):
        u_id.append(user_id)
        cnt+=1
del u_id[0]
f.seek(0)
while(i<cnt-1):
    i+=1
    ie=0.0
    sn=0.0
    tf=0.0
    jp=0.0
    count=0
    st=""
    st+=u_id[i]
    for line in f:
        u,index1=line.split('+++')
        l1=ast.literal_eval(index1)
        if(u_id[i]==u):
            ie+=l1[0]
            sn+=l1[1]
            tf+=l1[2]
            jp+=l1[3]
            count+=1
    st+=" "+str(ie/count)
    st+=" "+str(sn/count)
    st+=" "+str(tf/count)
    st+=" "+str(jp/count)
    li=st.split()
    with open("FinalMBTI(POS).csv", "a",newline='') as fp:
        wr = csv.writer(fp, dialect='excel')
        wr.writerow(li)

    f.seek(0)
```

7. Normalised MBTI values derived from a users POS tagged tweet

```
import ast
import csv
u_id=[0]
i=-1
cnt=0
f=open("MBTI_Indiviual.txt",'r')
for line in f:
    user_id,index=line.split('+++')
    if(u_id[-1]!=user_id):
        u_id.append(user_id)
        cnt+=1
del u_id[0]
f.seek(0)
while(i<cnt-1):
    i+=1
    ie=0.0
    sn=0.0
    tf=0.0
    jp=0.0
    count=0
    st=""
    st+=u_id[i]
    for line in f:
        u,index1=line.split('+++')
        l1=ast.literal_eval(index1)
        if(u_id[i]==u):
            ie+=l1[0]
            sn+=l1[1]
            tf+=l1[2]
            jp+=l1[3]
            count+=1
    st+=" "+str(ie/count)
    st+=" "+str(sn/count)
    st+=" "+str(tf/count)
    st+=" "+str(jp/count)
    li=st.split()
    with open("FinalMBTI(POS).csv", "a",newline='') as fp:
        wr = csv.writer(fp, dialect='excel')
        wr.writerow(li)

    f.seek(0)
```

8. Counting emoji, link and tags

```
import re
import csv
files=open("Emoticon&Linkcount.txt",'a+')
with open('CleanedData.csv','r') as f:
    reader = csv.reader(f)
    for row in reader:
        user_id=row[0]
        if(str(user_id)=="user_id"):
            continue
        string=str(row[1])
        dict1={'EMT':0,'LNK':0,'GRB':0}
        li=re.findall(r'`-emote-`',string)
        dict1['EMT']=len(li)
        li2=re.findall(r'_link_',string)
        dict1['LNK']=len(li2)
        li3=re.findall(r'`-garbage-`',string)
        dict1['GRB']=len(li3)
        files.write(str(user_id))
        files.write("+++")
        files.write(str(dict1))
        files.write('\n')
```

9. Bigram tagging of the tokenised tweets

Bigram Models care about the order of the words, so it considers the context of each word by analyzing it by pairs. Whereas a unigram model will tag a word independent of the other words.

```
import nltk
stop_words=[':',',','.','?','#','%','@','!','&']
f=open("tokenized_Data.txt",'r')
files=open("BigramTagged.txt",'w')
for line in f:
    string=str(line)
    user_id,tweet = string.split('+++')
    #print(words[0:-1])
    files.write(user_id)
    files.write("+++")
    words=eval(tweet[0:-1])
    filtered_sentence=[]
    for w in words:
            if w not in stop_words:
                filtered_sentence.append(w)
    my_bigrams = nltk.bigrams(filtered_sentence)
    li=[]
#my_trigrams = nltk.trigrams(words)
    for i in my_bigrams:
        li.append(nltk.pos_tag(i))
    files.write(str(li))
    files.write("\n")
```

10. Counting of Bigram tagged values of the tweets

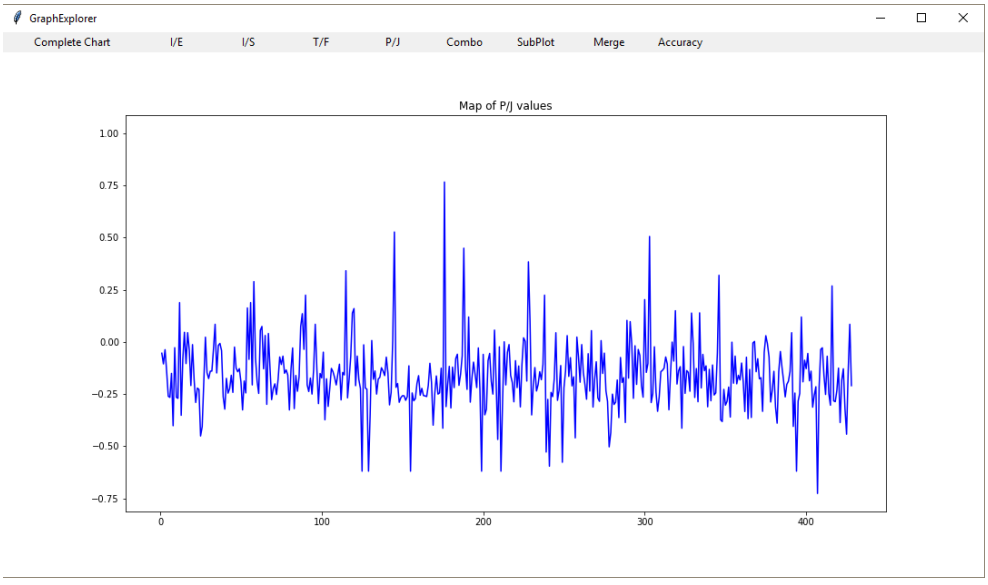
```
f=open("Bigram_tags.txt",'r')
dict1={}
tags=['NN','NNS','NNP','NNPS','VB','VBD','VBG','VBN','VBP','VBZ','EX','PDT']
for line in f:
    st=line.split()
for s in st:
    dict1[s]=0
#print(dict1)
f.close()
f2=open("BigramTagged.txt",'r')
files=open("BigramPOScount.txt",'a+')
for line in f2:
    string=str(line)
    user_id,tweet = string.split('+++')
    files.write(user_id)
    #print(user_id)
    #print("+++")
    files.write("+++")
    li=eval(tweet)
    for i in li:
        st1=str(i)
        st2=""
        l2=list(st1.split("'"))
        for j in l2:
            if j in tags:
                st2+=j
        for k in dict1.keys():
            if(k==st2):
                dict1[k]+=1
    files.write(str(dict1)+"\n")
    for k in dict1.keys():
        dict1[k]=0
```

11. Combining the MBTI values of the three results

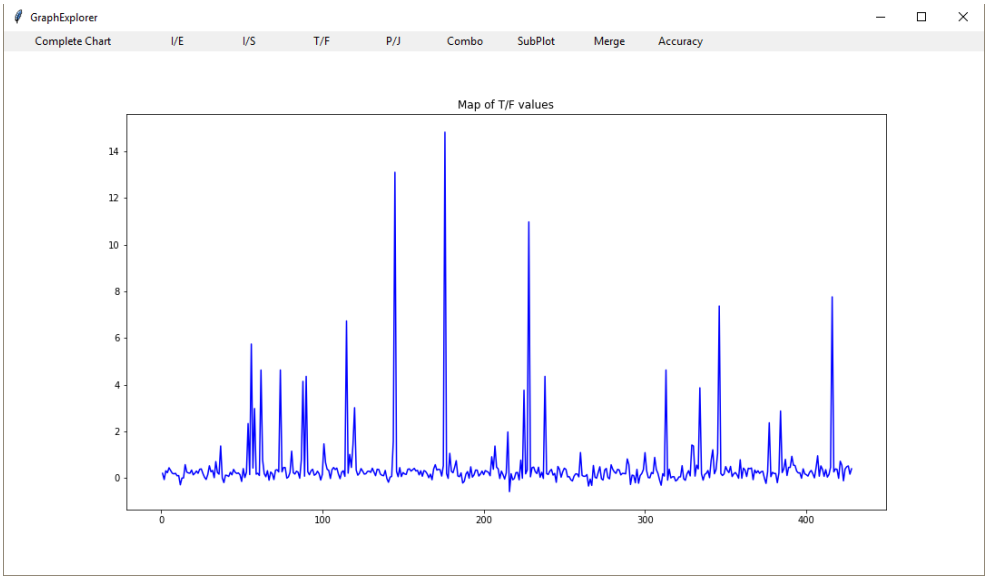
```
import csv
s=open('mbtiabs7.csv', 'a', newline='')
t=[]
reader2=csv.writer(s)
with open('FinalMBTIPOS.csv') as f:
    reader= csv.reader(f)
    with open('FinalMBTIemoji_by_3.csv') as c:
        reader1=csv.reader(c)
        for row in reader:
            for r in reader1:
                if row[0]==r[0]:
                    t.append(float(row[1])+float(r[1]))
                    t.append(float(row[2])+float(r[2]))
                    t.append(float(row[3])+float(r[3]))
                    t.append(float(row[4])+float(r[4]))
                    reader2.writerow([row[0],t[0],t[1],t[2],t[3]])

                    t=[]
                    w=1
                    break
                else:
                    continue
            if w==1:
                continue
            else:
                reader2.writerow([row[0],row[1],row[2],row[3],row[4]])

                w=0
            c.seek(0)
```

Results/Sample output

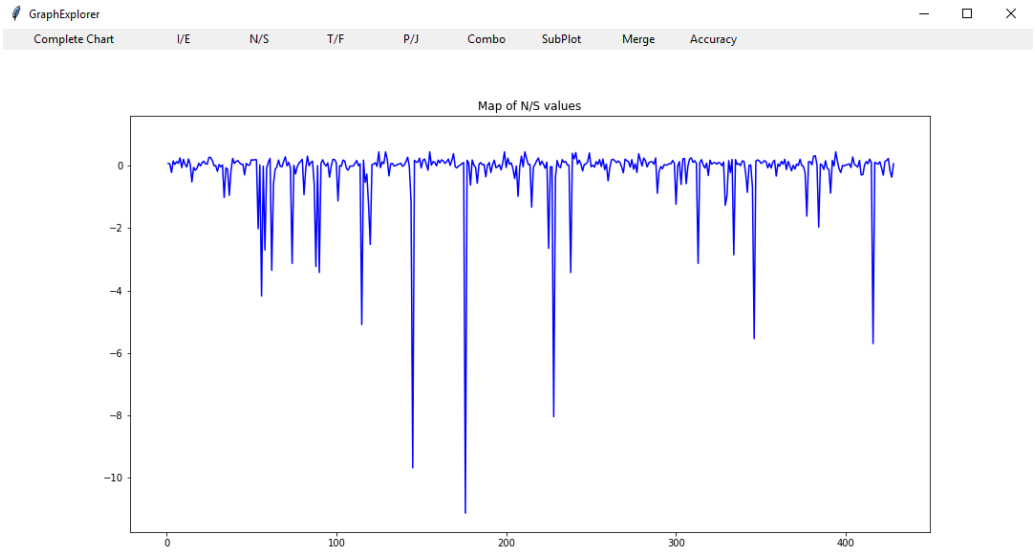
1. Map of P/J values:



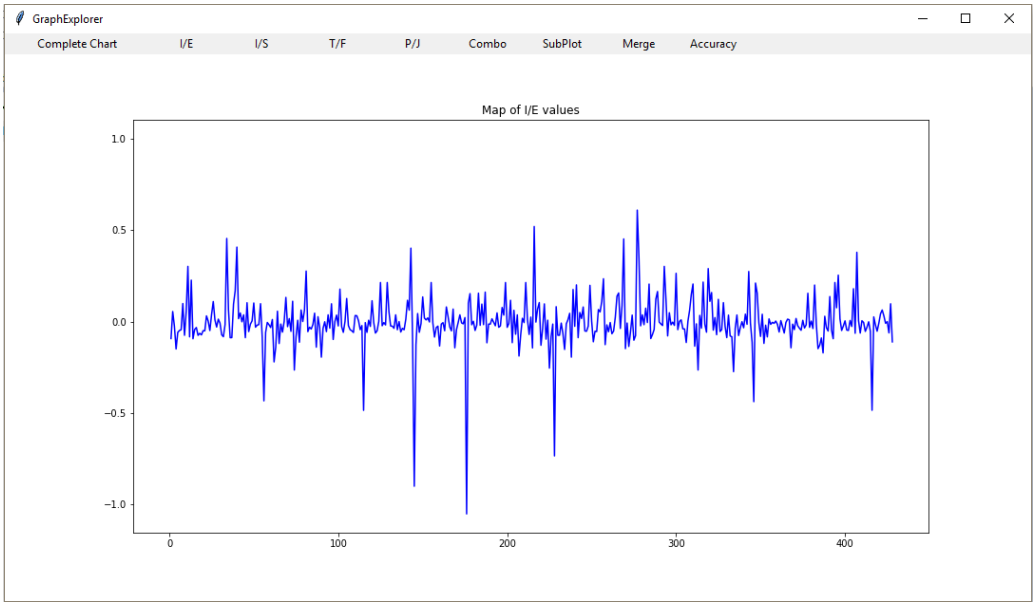
2. Map of T/F values:



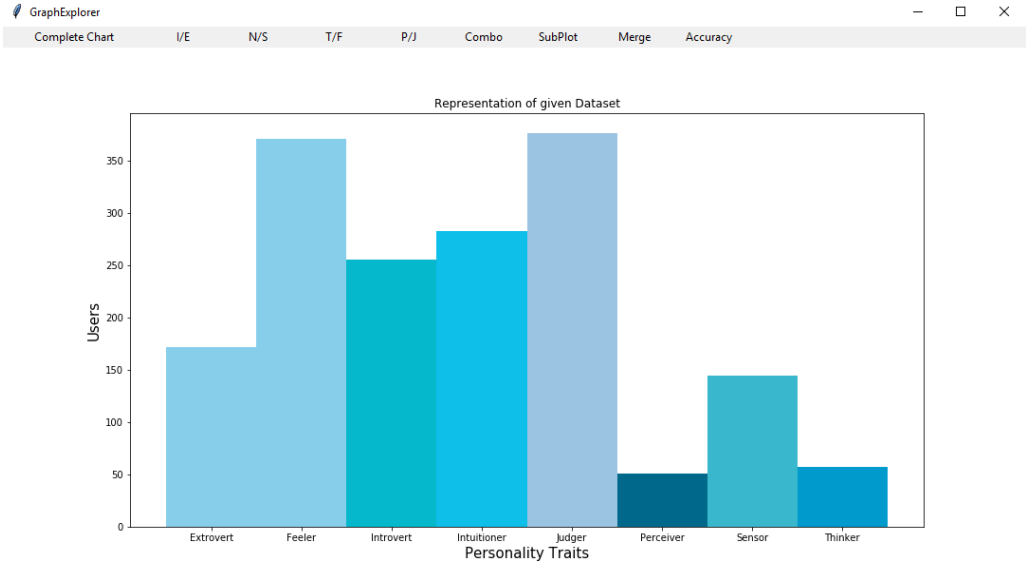
3. Map of N/S values:



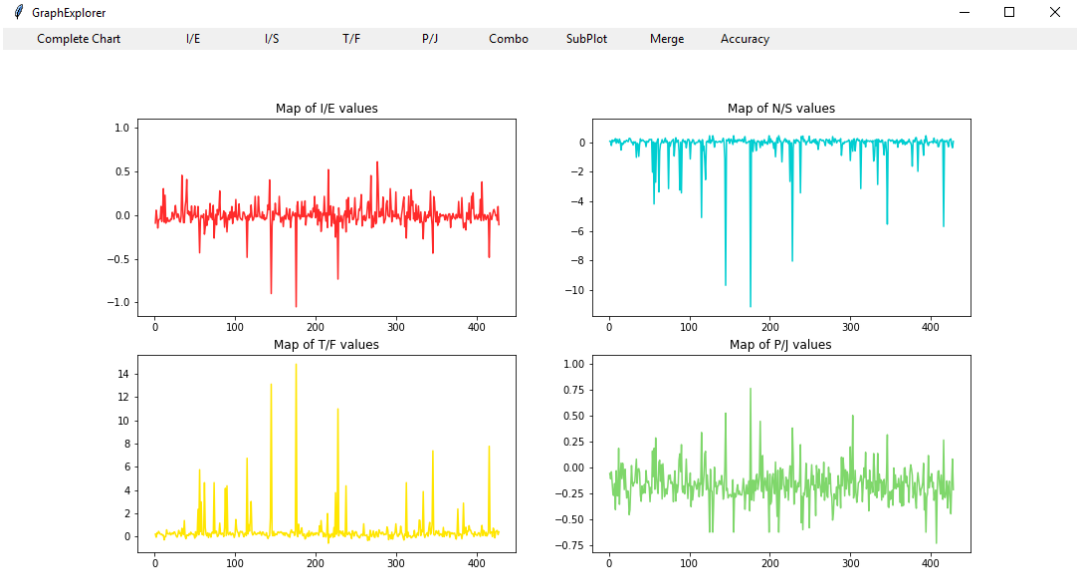
4. Map of I/E values:



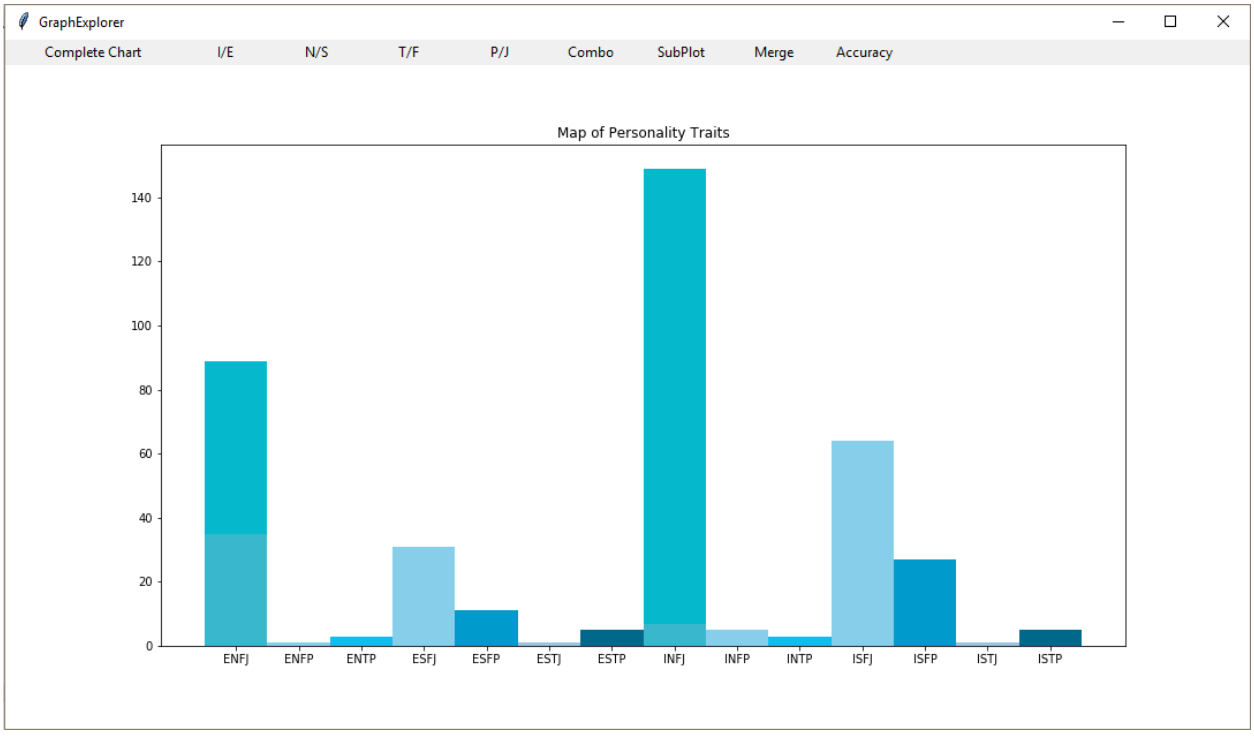
5. Representation of given Dataset:



6. Comparison between all types of personalities graph:



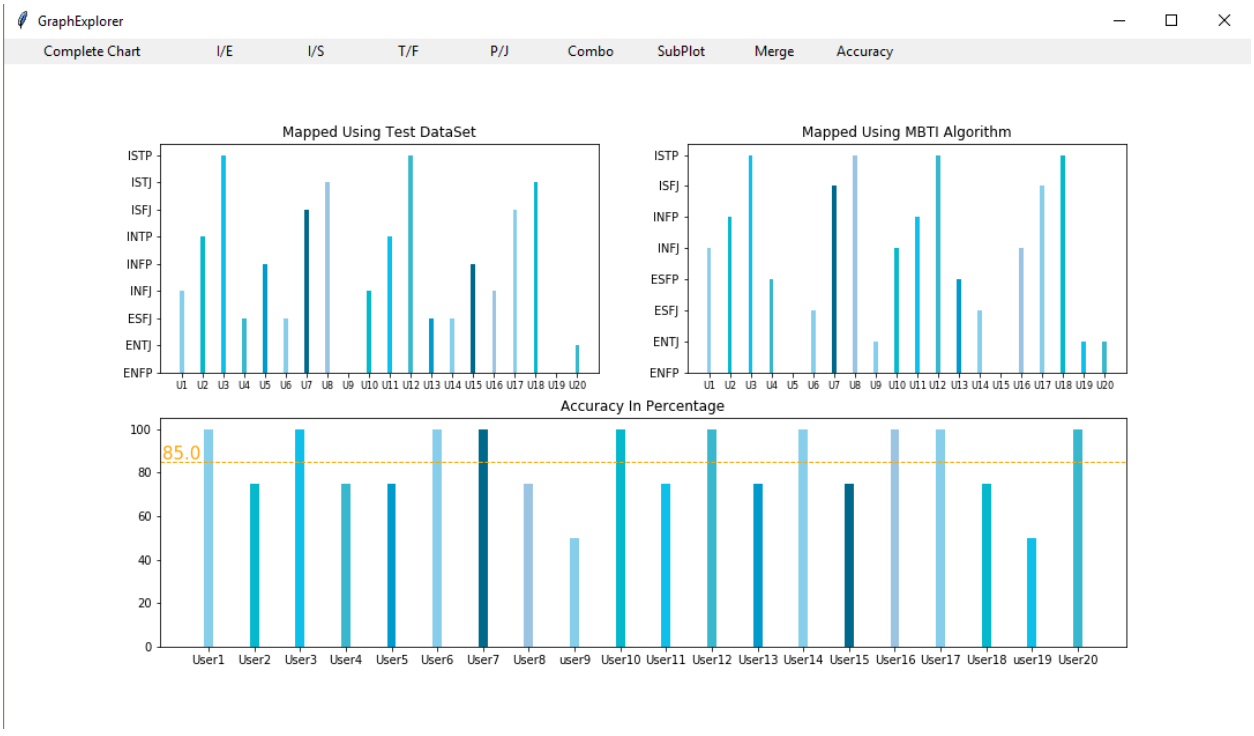
7. Map of personality traits:



8. Percentage Accuracy:

As we did not have any test data set (since all social media datas are restricted nowadays), with which we can find our algorithms accuracy we asked few of our friends and our group members performed an online test by which personality is calculated after giving a test on 100 questions. We then collected 15 to 20 tweets from their twitter accounts so that we can find our own result using our algorithm and then compare both the results.
www.16personalities.com the test on personality analysis is available on this website.

After comparing both the results we found out an accuracy percentage of around 85 percent. The graph of the accuracy percentage is shown below.



Error Analysis

- People has different personalities portrayed at different times, so it is difficult to say a persons personality exactly.
- Better cleaning of database will help in better calculation of personality.
- Unavailability of classifiers for identifying positive or negative minded people.
- Trigram bagging unavailability.

Conclusion

This study has produced two main insights. First, there are important personality similarities and differences among different types of Twitter users. All user types (listeners, popular, highly-read, and influential users) are emotionally stable (low in Neuroticism), and most of them are extrovert. These inferences have long been supported informally by intuition but have been difficult to make precise. Interestingly, popular users tend to be 'imaginative', while influential users tend to be 'organised'.

The second insight is that user personality can be easily and effectively predicted from public data, and that suggests future directions in a variety of areas, including : 1) Marketing: Since there is a relationship between marketing strategies and consumer personality, one could select ads to which a user is likely to be most receptive; 2) User Interface Design: One could match not just content but also the basic "look and feel" of a social media site to personality traits (this idea has been previously called "web site morphing".

Future developments of this study may utilise a larger training and testing dataset, which will allow the system to immerse itself in a wider variety of tweets. Improving n-gram normalisation functions may also increase the system's accuracy since it allows the system to recognise and assess more words.

Appendix

DATA MINING

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 24 18:09:35 2018

@author: The Hermit
"""

import tweepy
import csv
consumer_key = "#########################"
consumer_secret = "################################################"
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
api = tweepy.API(auth)
csvFile = open('testr.csv', 'a', newline='',encoding='utf-8')
csvWriter = csv.writer(csvFile)

for tweet in tweepy.Cursor(api.search,
                           q = "google",
                           lang = "en").items(10):
  u_id = tweet.user.id
  #print (tweet.user.id)

  for tweet in tweepy.Cursor(api.user_timeline,id=u_id).items(20):
     csvWriter.writerow([tweet.user.name, tweet.user.id, tweet.text.encode('utf8')])
```

TOKENIZING FOR UNIGRAM POS TAGGING

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 10 06:40:07 2018
@author: user
"""
import csv
from nltk.tokenize import word_tokenize
```

```
j=0
user_id=[0]
stop_words=["b'RT",'@','_link_','b',"'''",'``',"''",'"','`-emote-`',';','//','RT','\\n','\\n\\n','`-garbage-`','-
emote-','`-emote-','-emote-`','-garbage-','-garbage-`','`-garbage-']
files=open("tokenized_Data.txt",'a+')
with open('CleanedData.csv','r') as f:
    reader = csv.reader(f)
    for row in reader:
        temp=row[1]
        if(temp != user_id[-1]):
            user_id.append(temp)
            j=j+1
    del user_id[0]
    del user_id[0]
    j=j-2
    f.seek(0)
    tweet=[]
    for row in reader:
        for i in range(0,j):
            if(user_id[i]==row[1]):
                tweet.append(row[2])
                files.write(str(user_id[i]))
                files.write("+-+-+")
                word_tokens=word_tokenize(str(tweet[-1]))
                print(word_tokens)
                filtered_sentence = []
                for w in word_tokens:
                    if w not in stop_words:
                                filtered_sentence.append(w)
                files.write(str(filtered_sentence))
                files.write(“\n")
```

POS TAGGING (UNIGRAM)

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 17 12:21:25 2018

@author: administrator
"""
```

```
from nltk.tag import pos_tag
f=open("tokenized_Data.txt",'r')
files=open("POS_Tagged.txt",'a+')
list1=[]
for line in f:
    list1.append(line)
for line in list1:
    string=str(line)
    user_id,tweet = string.split('+-+-+')
    files.write(user_id)
    files.write("+-+-+")
    #print(tweet[0:-1])
    lst = eval(tweet[0:-1])
    #print(li)
    tweets_tagged = pos_tag(lst)
    #print(tweets_tagged)
    files.write(str(tweets_tagged))
    files.write("\n")
```

POS COUNT (UNIGRAM)

```
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 18 12:23:50 2018

@author: user
"""
f=open("tags.txt",'r')
tag=[]
dict={}
for line in f:
    string=line
tag=list(string.split(' '))
#print(tag)
for i in tag:
   dict[i]=0
f1=open("POSTagged2.txt",'r')
files=open("POS_count3.txt",'a+')
list1=[]
for line in f1:
```

```
    list1.append(line)
for line in list1:
    string=str(line)
    user_id,tweet = string.split('+-+-+')
    files.write(user_id)
    files.write("+-+-+")
    #li=eval(tweet[0:-1])
    #print(li)
    li=list(tweet[0:-1].split("'"))
    for i in li:
        for j in tag:
            if(i==j):
                dict[i]+=1
                #print(i)
    files.write(str(dict))
    print(dict)
    files.write("\n")
    for i in tag:
        dict[i]=0
```

EMOTICON,LINK COUNTER

```
# -*- coding: utf-8 -*-
"""
Created on Fri May 11 20:24:31 2018

@author: sibasish
"""

import re
import csv
files=open("Emoticon&Linkcount.txt",'a+')
with open('CleanedData.csv','r') as f:
    reader = csv.reader(f)
    for row in reader:
        user_id=row[0]
        if(str(user_id)=="user_id"):
            continue
        string=str(row[1])
        dict1={'EMT':0,'LNK':0,'GRB':0}
        li=re.findall(r"-emote-`",string)
```

```
        dict1['EMT']=len(li)
        li2=re.findall(r'_link_',string)
        dict1['LNK']=len(li2)
        li3=re.findall(r`-garbage-`',string)
        dict1['GRB']=len(li3)
        files.write(str(user_id))
        files.write("+++")
        files.write(str(dict1))
        files.write('\n')
```

MBTI INDIVIDUAL TWEET CALCULATOR

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 20 14:58:55 2018

@author: user
"""
import yaml
import csv
f1=open("POS_count3.txt",'r')
files=open("MBTI_Indiviual3.txt",'a+')
with open('MBTI_values.csv','r') as f:
    reader = csv.reader(f)
    list1=[]
    for line in f1:
        list1.append(line)
    for line in list1:
        string=str(line)
        user_id,tweet = string.split('+-+-+')
        files.write(user_id)
        files.write("+-+-+")
        dict1=yaml.load(tweet)
        ie=0.0
        sn=0.0
        tf=0.0
        jp=0.0
        for row in reader:
            for i in dict1:
                if(i==row[0]):
```

```
                ie+=float(dict1[i]*float(row[1]))
                sn+=float(dict1[i]*float(row[2]))
                tf+=float(dict1[i]*float(row[3]))
                jp+=float(dict1[i]*float(row[4]))
        files.write("[ "+str(ie)+", "+str(sn)+", "+str(tf)+", "+str(jp)+"]")
        files.write("\n")
        f.seek(0)
```

MBTI AVERAGE CALCULATOR

```
# -*- coding: utf-8 -*-
"""
Created on Fri Apr 20 13:16:07 2018

@author: user
"""
import ast
import csv
u_id=[0]
i=-1
cnt=0
f=open("MBTI_Indiviual3.txt",'r')
for line in f:
    user_id,index=line.split('+-+-+')
    if(u_id[-1]!=user_id):
        u_id.append(user_id)
        cnt+=1
del u_id[0]
f.seek(0)
while(i<cnt-1):
    i+=1
    ie=0.0
    sn=0.0
    tf=0.0
    jp=0.0
    count=0
    st=""
    st+=u_id[i]
    for line in f:
        u,index1=line.split('+-+-+')
```

```
        l1=ast.literal_eval(index1)
        if(u_id[i]==u):
            ie+=l1[0]
            sn+=l1[1]
            tf+=l1[2]
            jp+=l1[3]
            count+=1
    st+=" "+str(ie/count)
    st+=" "+str(sn/count)
    st+=" "+str(tf/count)
    st+=" "+str(jp/count)
    li=st.split()
    with open("FinalMBTI(POS).csv", "a",newline='') as fp:
        wr = csv.writer(fp, dialect='excel')
        wr.writerow(li)
    '''
    if(ie<0):
        print(" Introvert -")
    else:
        print(" Extrovert -")
    if(sn<0):
        print(" Sensor -")
    else:
        print(" Intutioner -")
    if(tf<0):
        print(" Thinker -")
    else:
        print(" Feeler -")
    if(jp<0):
        print(" Judger ")
    else:
        print(" Perceiver ")'''
    #print(li)
    f.seek(0)
```

FINAL MBTI NORMALIZED VALUE CALCULATOR

```
# -*- coding: utf-8 -*-
"""
Created on Sat May 12 18:38:49 2018
```

```
@author: The Hermit
"""

import csv
s=open('mbtiabs7.csv', 'a', newline='')
t=[]
reader2=csv.writer(s)
with open('FinalMBTIPOS.csv') as f:
    reader= csv.reader(f)
    with open('FinalMBTIemoji_by_3.csv') as c:
        reader1=csv.reader(c)
        for row in reader:
            for r in reader1:
                if row[0]==r[0]:
                    t.append(float(row[1])+float(r[1]))
                    t.append(float(row[2])+float(r[2]))
                    t.append(float(row[3])+float(r[3]))
                    t.append(float(row[4])+float(r[4]))
                    reader2.writerow([row[0],t[0],t[1],t[2],t[3]])

                    t=[]
                    w=1
                    break
                else:
                    continue
            if w==1:
                continue
            else:
                reader2.writerow([row[0],row[1],row[2],row[3],row[4]])

                w=0
            c.seek(0)
```

BIGRAM POS TAGGER

```
# -*- coding: utf-8 -*-
"""
Created on Sat May 12 20:08:54 2018
```

```
@author: sibasish
"""
import nltk
stop_words=[':',',','.','?','#','%','@','!','&']
f=open("tokenized_Data.txt",'r')
files=open("BigramTagged.txt",'w')
for line in f:
    string=str(line)
    user_id,tweet = string.split('+++')
    #print(words[0:-1])
    files.write(user_id)
    files.write("+++")
    words=eval(tweet[0:-1])
    filtered_sentence=[]
    for w in words:
                    if w not in stop_words:
                    filtered_sentence.append(w)
    my_bigrams = nltk.bigrams(filtered_sentence)
    li=[]
#my_trigrams = nltk.trigrams(words)
    for i in my_bigrams:
        li.append(nltk.pos_tag(i))
    files.write(str(li))
    files.write("\n")
```

BIGRAM POS TAGGER

```
# -*- coding: utf-8 -*-
"""
Created on Sat May 12 21:50:07 2018
@author: sibasish
"""

f=open("Bigram_tags.txt",'r')
dict1={}
tags=['NN','NNS','NNP','NNPS','VB','VBD','VBG','VBN','VBP','VBZ','EX','PDT']
for line in f:
    st=line.split()
for s in st:
    dict1[s]=0
```

```
#print(dict1)
f.close()
f2=open("BigramTagged.txt",'r')
files=open("BigramPOScount.txt",'a+')
for line in f2:
    string=str(line)
    user_id,tweet = string.split('+++')
    files.write(user_id)
    #print(user_id)
    #print("+++")
    files.write("+++")
    li=eval(tweet)
    for i in li:
        st1=str(i)
        st2=""
        l2=list(st1.split("'"))
        for j in l2:
            if j in tags:
                st2+=j
        for k in dict1.keys():
            if(k==st2):
                dict1[k]+=1
    files.write(str(dict1)+"\n")
    for k in dict1.keys():
        dict1[k]=0
```

Reference

1. Yilun Wang - Department of Computer Science, Stanford University
yilunw@stanford.edu

2. Yoshua Bengio, Rejean´ Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic lan-guage model. The Journal of Machine Learning Re-search, 3:1137–1155.

3. Ryan L Boyd, Steven R Wilson, James W Pennebaker, Michal Kosinski, David J Stillwell, and Rada Mi-halcea. 2015. Values in words: Using language to evaluate and understand personal values. In Ninth International AAAI Conference on Web and Social Media.

4. Michal Kosinski, David Stillwell, and Thore Grae-pel. 2013. Private traits and attributes are pre-dictable from digital records of human behavior. Proceedings of the National Academy of Sciences, 110(15):5802–5805.

5. Tomas Mikolov, Kai Chen, Greg Corrado, and Jef-frey Dean. 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

6. Isabel Briggs Myers. 1962. The myers-briggs type indicator: Manual (1962).

7. Daniele Quercia, Michal Kosinski, David Stillwell, and Jon Crowcroft. 2011. Our twitter profiles, our selves: Predicting personality with twitter. In Pri-vacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Com-puting (SocialCom), 2011 IEEE Third International Conference on, pages 180–185. IEEE.

8. David Rawlings and Vera Ciancarelli. 1997. Music preference and the five-factor model of the neo per-sonality inventory. Psychology of Music, 25(2):120– 132.

9. 16personalities.com for test on personality analysis.

10. www.phys.org for further help.