QUEENSLAND UNIVERSITY OF TECHNOLOGY

Discovering social spammers from multiple views

Project plan

IFN 701: Project 1

Student Name: Jianwei Tang – N10057862

Supervisor: Darshika Koggalahewa

EXECUTIVE SUMMARY

Social platforms are so popular in recent years and using them is now a part of people's daily life. It also provides access to social spammer to abuse the information distribution system and pollute the information that legitimate users receive, which raises a significant concern. In order to remove those social spammers, identifying them from legitimate users would be the first step. In this report, we will illustrate the process of extracting the features that social spammer might have, and the usage of these features on machine learning for identifying them.

This project extracts features from public datasets: Social Honeypot and The Fake Project. The features should be able to be analysed from different views including sentiment analysis and similarity analysis. Sentiment analysis identifies spammers from the content they like to use, however similarity analysis group unidentified spammers to identified spammers based on their similar characteristics. This is so called Multi-View learning methodology. The features we aim to extract are vital ingredients for this learning method and the process of extracting them also has great value for later implementation.

The method used in this project is by using Python to build a 4-dimensional data from original raw datasets. Then all featured are extracted from the 4-dimensional data, in a way of exploring meaningful relationships between different dimensions. The dimensions are USER, CONTENT, HASHTAG and MENTION. Our outcome includes the data frames of USER_TO_CONTENT, USER_TO_HASHTAG, USER_TO_MENTION, HASHTAG_TO_CONTENT, MUTUAL_MENTION, AVG_HASHTAG_PER_TWEET, DUPLICATE_HASHTAG, COMMON_TOPIC and CONTENT_SIMILARITY_TO_TOPIC. They are stored in .csv files in a comma-separated format. In addition, the process of extracting those features is also considered as a part of outcome as it is well-structured in Python programs. The reusability of the python program enables data scientists to extract similar features with much less effort.

The report will have a discussion of the future development and verification at the very end. More features can be extracted experimentally since this project has only discovered a few combinations of different views. The verification of the importance of each feature will be shown when they are put into machine learning as ingredients, which can be an appreciated complementary for this project. The importance can be indicated from the impact of the feature toward spammer identification by machine learning models.

1 CONTENTS

1 INTRODUCTION

Online Social networks have become more popular platforms for the spammers to perform malicious activities. Majority of the current systems used machine learning based approaches to detect spammers. Among them classification-based techniques are the most common in many efforts where they use single view of information to build the classifiers. The aim of this project is to develop a classification system for spam detection using multiple views. In this project we try to use multiple information to learn the classification model and test the system with real world datasets.

1.1 BACKGROUND

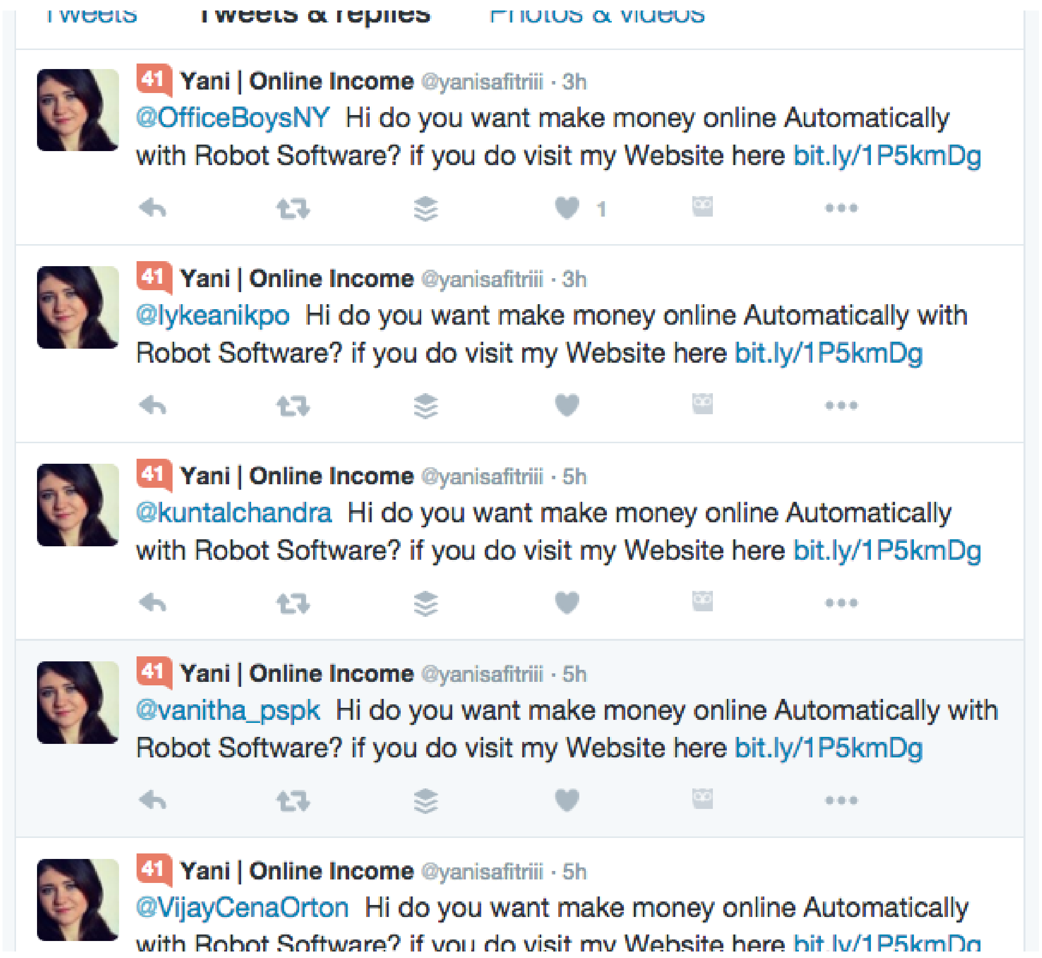
Online social networks, such as Twitter and Facebook, have been combating spammers for a long time. spammers are taking advantages from these platforms to spread their phishing, promoting and malicious information which usually disguise legitimate users. The strategies of spamming keep changing and evolving to make them undetected by filtering system. Therefore, an affective social spammer detecting system is in need for winning this anti-spam combat.

Conventional approaches for detecting social spam are mostly collecting user-generated information including content, hash tags and short URLs, and embed them with network structures to form a single view data for producing a model to identify spam. It may not able to characterise spammer completely. For example, some spammers may post legal text but adding unfriendly shorten URLs or tempting hashtags to achieve their malicious purposes(Shen et al., 2017).

Multiview learning can be a promising solution to tackle these problems as it processes data from multiple perspective. Multiview learning methodology interprets each view of data comprehensively and use algorithms to ensemble each view into a classifier to achieve a better precision on identification of spammer. It is the method we are going to focused on in this development project.

1.1.1 Characteristics on Social spammers

Social spammers have some observable behaviours. 1. They always abuse harsh tags. Adding trending hashtags to their posts to get seen by more people. 2. They follow accounts in specific rules as they are doing follower fraud to increase the value of target accounts. 3. They tend to use short URLs for phishing or promoting.


Tweets Tweets & replies Photos & videos

41 Yani | Online Income @yanisafitriii · 3h
@OfficeBoysNY Hi do you want make money online Automatically with Robot Software? if you do visit my Website here bit.ly/1P5kmDg
1

41 Yani | Online Income @yanisafitriii · 3h
@lykeanikpo Hi do you want make money online Automatically with Robot Software? if you do visit my Website here bit.ly/1P5kmDg

41 Yani | Online Income @yanisafitriii · 5h
@kuntalchandra Hi do you want make money online Automatically with Robot Software? if you do visit my Website here bit.ly/1P5kmDg

41 Yani | Online Income @yanisafitriii · 5h
@vanitha_pspk Hi do you want make money online Automatically with Robot Software? if you do visit my Website here bit.ly/1P5kmDg

41 Yani | Online Income @yanisafitriii · 5h
@VijayCenaOrton Hi do you want make money online Automatically with Robot Software? if you do visit my Website here bit.ly/1P5kmDg

Figure 1. A typical spammer on twitter

1.1.2 Multiview learning

For building classifier, we usually collect data from multiple resources. Conventional machine learning algorithms tend to concatenate all multiple views into one single view to adapt to the learning setting (Xu, Tao, & Xu, 2013). However, this single view learning algorithms might not able to achieve a decent performance as the data from different resources have different properties. Multiview learning is then being paid attention as a novel method for achieving a better classification performance.

The figure 2 below shows an example of Multiview learning applied on social spammer detection. Social network and users' generated content are trained independently as two views, and then Multiview learning algorithms will base on them to create a particular model for spammer classification.

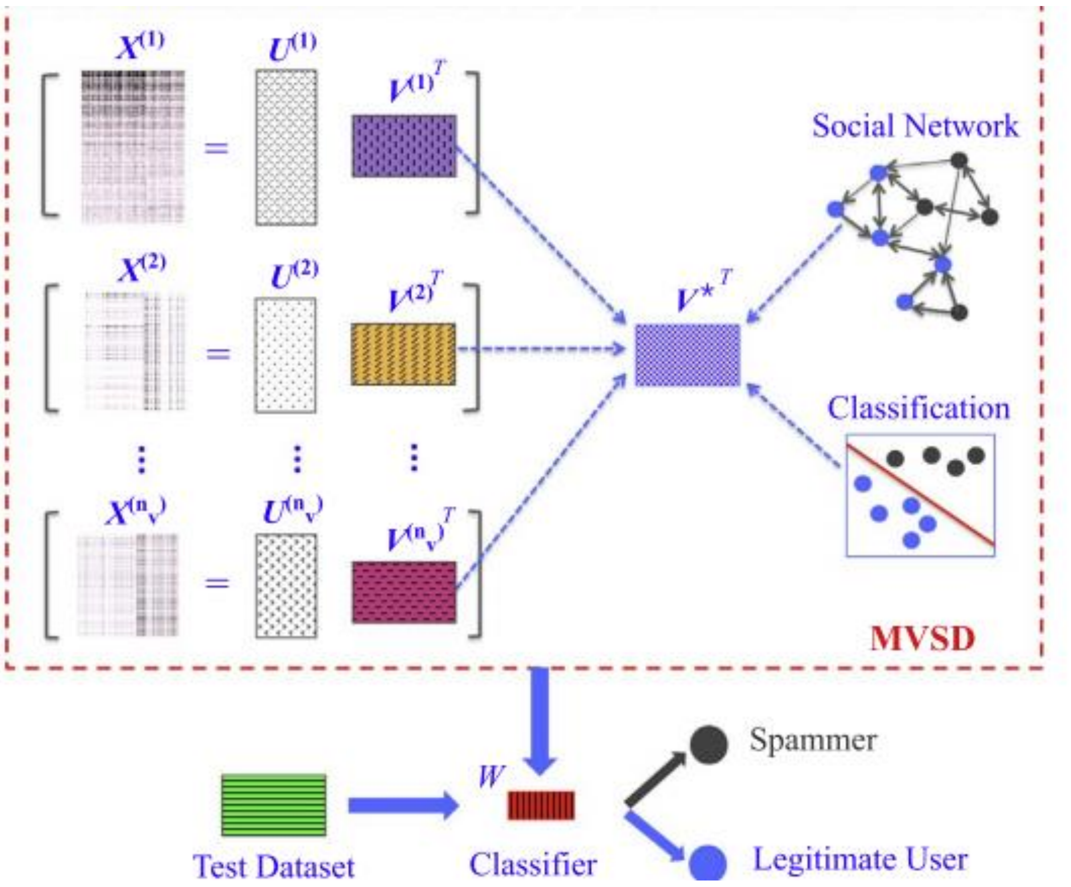


Figure 2. A Multiview learning framework can be used on Online Social Network (Shen et al., 2017)

1.2 INDUSTRY PROBLEM

Spammers bring a lot of burdens to Online Social Networks. The major parts are: 1. Loss of service effectiveness and credibility. When legitimate users getting information from Online Social Networks, they may receive abundant unacceptable information like fraud, phishing and other irrelevant information. 2. Waste of space. Online Social Networks need to store all the information. The increasing number of spammers brings a huge waste if they are not removed properly. 3. Waste of bandwidth. The storage and retrieval of those polluted information also take bandwidth of the system. 4. Loss of effectiveness of caches and content distribution networks. The way that spammers promote their posts is abusing the ranking system, which severely damages the content distribution network. (Reiter Langbehn et al., 2010)

1.3 AIMS AND OBJECTIVES

This project aimed to extract features from public datasets and suggest usage of these features on Multi-View learning. features should produce value to companies who operate online social network systems. The implementation of Multiview learning can help them provide users with better experience, including less polluted information, less risk of phishing/fraud, and more relevant search results. In addition, the cost of business could also be reduced. With the removal of spammers, the data storage can be free up; the servers get rid of those polluted content; and the employees received less reports about fraud/spammers.

1.4 PRIORITISED SCOPE

Prioritisation	Items
Must have (60%)	1. Features extracted for Sentiment analysis 2. Features extracted for Similarity analysis 3. Process of extracting features with reusability
Should have (20%)	4. Process applied on two datasets for verification 5. Additional datasets validating
Could have (20%)	6. Experimentally extract more features 7. Construct a classifier to test the importance of each feature
Out of scope	8. Algorithm or mathematical analysis

1.5 KEY DELIVERABLES

Deliverables can be introduced as two major parts: features and process.

1.5.1 Features stored in .csv format

This project extracts features from real world datasets and these features can be used as ingredients for Multi-View Learning to construct a model. The model helps identify spammer from legitimate users on Online Social Network systems. In addition, the importance of each feature can be further analysed by researchers. Researchers can gain deeper understanding on the behaviours of spammers.

1.5.2 Process of extracting features in python programs

The process of extracting has practical significance. Data scientists can use the code that this project produced to extract similar features from other datasets, since the python programs have reusability. The process will save a lot effort for future implementation.

2 ENVIRONMENTAL SCAN AND RELATED WORK

Hadian & Minaei (2013) have introduced Character-level-4-grams as a feature. It is extracted by a 4-character width window sliding over the content. It is a typical sentimental analysis method. They also pointed out a great advantage of multi-view learning is to apply algorithms to each feature set parallelly, which results in desired operation time.

In (Hendrickson R. Langbehn, 2010), the authors proposed a multi-view semi-supervised approach. It takes a small amount of manually labelled data as input, and then expands it from unlabeled data by similarity analysis. The expanded data can be used as training data for further spammer classification with multi-view learning process. The effort of labeling data manually can be saved significantly as it is the most time-consuming and money-consuming part.

Chao, Sun, & Bi (2017) defined multi-view clustering(MVC). "MVC is a machine learning paradigm to classify similar subjects into the same group and dissimilar subjects into different groups by combining the available multi-view feature information, and to search for consistent clusterings across different views." The similarity-based approaches are described as objective function optimization which minimize the average similarity within clusters and to maximize the average similarity between clusters.

Multi-view learning is closer to the nature of the data. Data can be collected from different sources with different views. Conventional machine learning algorithms tend to concatenate all multiple views into one single view, and then process machine learning. The model generated may not perform well as each view has specific statistical property.(Xu et al., 2013) there are three major types multi-view learning algorithms: co-training, multiple kernel learning, and subspace learning. Co-training is one of the earliest methods which maximize the mutual agreement on two separate views with unlabelled data. It has been developed a lot since 1998(Blum & Mitchell, 1998). In order to have a successful result from using this method, three requirements should be met: 1. Each view should be able to classify on its own. 2. Both views should have consistent prediction for overlapping features. 3. Views are conditionally independent given the label.

Feature engineering generates the components for Multiview learning. (Herzallah, 2018) has suggested a list of features including age of the account, reputation of the account, average time between posts, idles hours, ratio between retweets and tweets etc. The features have been tested using popular classifiers (Naive Bayes, support vector machines, multilayer perceptron neural networks, Decision Trees, Random forests and k-Nearest Neighbour) and their values have been proved.

3 PROJECT METHODOLOGY

This project is focused on the development of extracting social spammers' features for Multi-view learning methodology. In this section, we will be introducing the datasets and tools involved in this project. In addition, we will illustrate the project management methodology and project development methodology.

3.1 DATASETS CHOICES

The datasets in this project are Social Honeypot and The Fake Project. All the users in the datasets are well-labelled as spammers and legitimate users.

3.1.1 Social honeypot dataset

Social honeypot dataset was collected from December 30, 2009 to August 2, 2010 on Twitter. The dataset was used in the paper Seven Months with the Devils: A Long-Term Study of Content Polluters on Twitter in ICWSM 2011.(Caverlee, 2011)

The dataset contains 22,223 content polluters, their number of followings over time, 2,353,473 tweets, and 19,276 legitimate users, their number of followings over time and 3,259,693 tweets.

3.1.2 The fake project

The fake project contains 469 users with 563,693 tweets which collected in 2012. Users were asked to follow a particular account if they were not fake – they were subsequently verified by a CAPTCHA.

3.2 PROGRAMMING LANGUAGE AND PACKAGES

Python is the programming language in this project. It is the only general-purpose programming language with a solid ecosystem of scientific computing libraries. In addition, being an interpreted language with a very simple syntax, Python allows for rapid prototyping. It's also the undisputed king of deep learning. The libraries include Pandas, NumPy and Scikit-learn for our extracting process, and Tweepy to complete the datasets.

The package-Pandas should be emphasised here becasue all the data transformation are as Pandas Data Frame format. Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

3.3 Project Management Approach

DSDM is applied to make iterative changes to our project ideas and allow a unique flow of information between me and the supervisor throughout the implementation of the project.

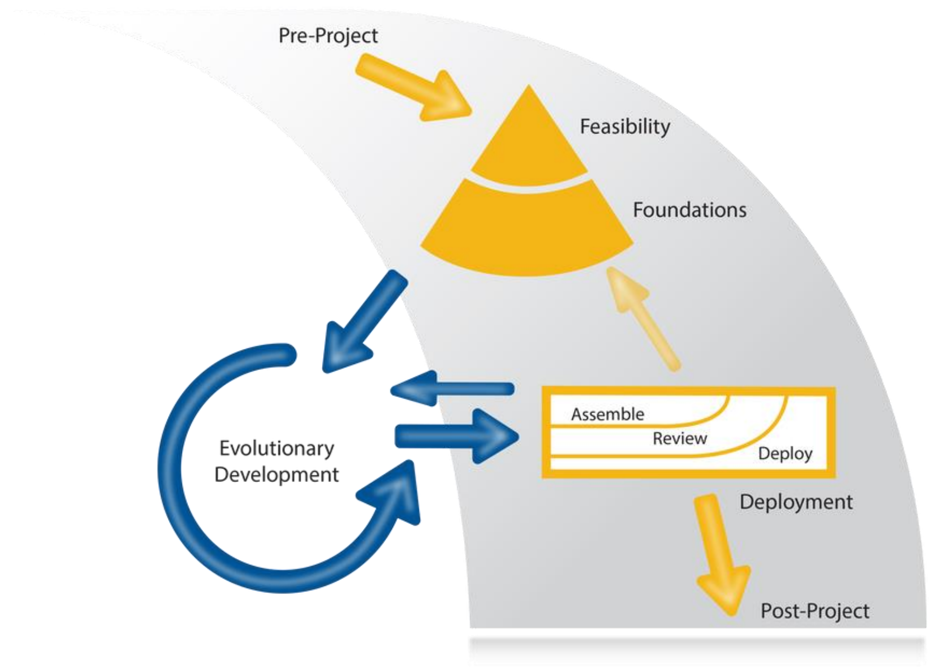


Figure 3.DSDM Process. ("The DSDM Agile Project Framework," 2014)

According to DSDM ("The DSDM Agile Project Framework," 2014), the project is made up of 6 phases, which are:

- Pre-Project Phase: Ensuring the right information is available at the right time
- Feasibility Phase: Establish if there is a feasible solution to the business problem
- Foundations Phase: Establish Firm foundations for the project/solution/management of the project.
- Evolutionary Development Phase: The iterative development begins, and the right solution should evolve over time.
- Deployment Phase: Bring a baseline of the Evolving Solution into operational use
- Post-Project Phase: Checks how well the expected business benefits have been met; Produces one or more Benefits Assessments for these realised benefits.

3.4 Communication Plan

We have weekly meetings, email and screen-sharing for communication.

Weekly meetings are important for development project as the face to face kind of meeting is the most effective. The development approaches involved in this project will be discussed between me and supervisor, to ensure the project is on the right track, which is impossible via E-mail. For example, the structure of data can be orally explained with a pencil drawing graph, any questions from me can be answered immediately so the explanation goes smoothly. However, the Email cannot reach this effectiveness.

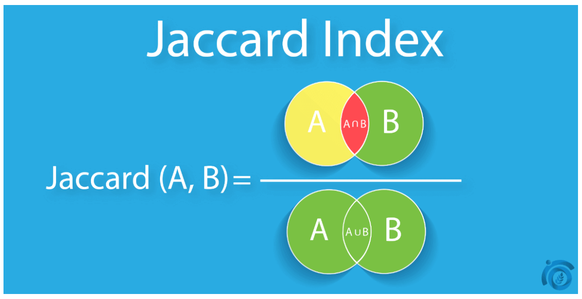
Email is an alternative tool for communication. Sometimes we come up with some problems which cannot be solved or answered straightaway and need to figure out after the meeting, then Email can be a complementary way. It has no time or location limits so we can send emails right after working out the solution.

Screen-sharing is specific for coding. This project required a lot programming and we cannot guarantee there will be no bug or obstacle during the development. By sharing the screen, supervisor is able to see the code while listening to my explanation. Thus, a quick understanding of my code can be achieved, which helps debugging and evaluating the code.

3.5 The approach of the project

We are going to use Supervised learning for building the classification model, as the datasets we are using are well labelled. It is also achievable for a medium-level programmer in a time restricted project.

Detailly, three major parts of model training are sentiment analysis, similarity analysis and Multiview learning. We build user profiles upon Sentiment analysis and Similarity analysis as essential components for Multiview learning. Specifically, Cosine similarity, Jaccard similarity, Pearson correlation are applied for similarity analysis.



Sentiment analysis is able to classify the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral. Advanced, "beyond polarity" sentiment classification looks, for instance, at emotional states such as "angry", "sad", and "happy". The usage of words, hashtags and mentions from users can indicate this polarity. On the other hand. Similarity analysis is to the measure how much alike two data objects are. In this report we will develop the process of measuring the distance of two users' contents for a given topic/hashtag.

The Multiview learning methodology we are going to use is Co-training. Co-training trains alternately to maximize the mutual agreement on two distinct views of the unlabelled data(Xu et al., 2013). It perfectly suits our project which has two user profiles.

There are also unknown challenges in this project, especially in programming phase. We apply developing-while-learning strategy, which is, when problems show up, we try to leverage from online question-and-answer sites like Stack Overflow, to find proper solutions related coding. The implementation of the algorithms can also be difficult and time-consuming. And then existed opensource solutions should be checked before working on the keyboard, so we do not "reinvent the wheel".

3.6 POTENTIAL PROJECT RISKS AND RISK MITIGATION STRATEGIES

	Risks	Impact	Mitigation plan
1.	Insufficient time	The number of features might be reduced	Keep the flexibility of the deliverables
2.	Insufficient programming skill	The development goes slow or halts	Read related work on GitHub beforehand as preparing
3.	Unavailability of the supervisor	Cannot get direct help or feedback	Contact with area coordinator to find solutions
4.	The model training process may require a long time to run	Low efficiency when working on PC at home	Contact with supervisor to find servers with high computing capability

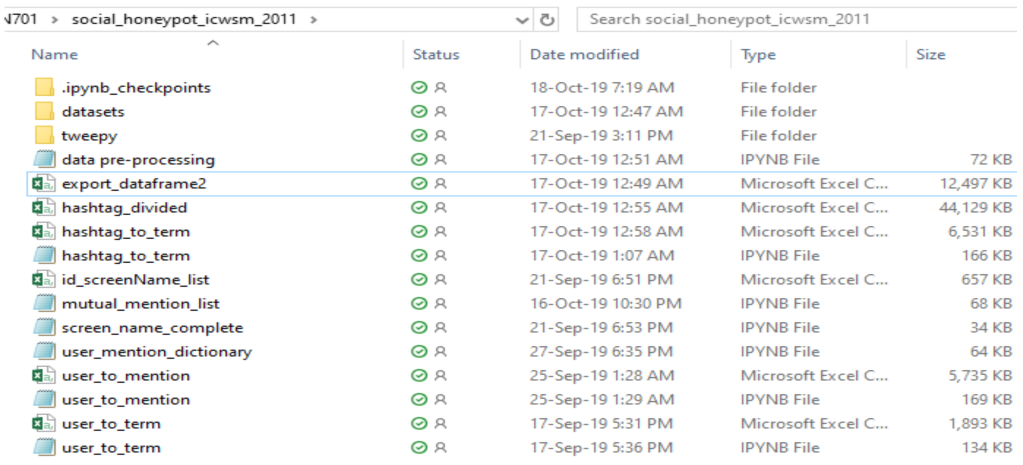
4 OUTCOMES

The output of this project is a list of features and the process of extracting them. Featrues are presented as Excel spread sheet, while processes are presented as IPYNB files. The screenshots between is an overview of the outputs for The Social Honeypot datasets and The Fake Project. We will discuss the detail of each IPYNB file in this section.

The list of features includes:

Feature	*Feature description*	*Feature criteria(optional)*
1	Term usage for each user	
2	Mention usage for each user	
3	Radio of Mention and Being mentioned	$\frac{No\ of\ Mentions\ for\ u_j}{No\ of\ Mentions\ by\ u_j}$
4	Mutual mention	
5	Term usage for each user on each hashtag	
6	Content similarity for each hashtag between two users	$\frac{No\ of\ common\ terms\ between\ u_i, u_j}{No\ of\ unique\ words\ between\ u_i, u_j}$
7	Common topics between two users	$\frac{No\ of\ common\ topics\ between\ u_i, u_j}{Total\ No\ of\ topics\ of\ u_i + Total\ No\ of\ topics\ of\ u_j}$
8	Average hashtags per tweet	$\frac{\sum Number\ of\ Hashtags\ in\ tweets}{Total\ number\ of\ tweet}$
9	Duplicate Hashtags	$\frac{\sum Number\ of\ duplicate\ Hashtags}{Total\ number\ of\ tweets \times Number\ of\ Hashtags}$

The structure of program files is presented below.


\701 > social_honeypot_icwsm_2011 >

Search social_honeypot_icwsm_2011

Name	Status	Date modified	Type	Size
.ipynb_checkpoints	✓	18-Oct-19 7:19 AM	File folder	
datasets	✓	17-Oct-19 12:47 AM	File folder	
tweepy	✓	21-Sep-19 3:11 PM	File folder	
data pre-processing	✓	17-Oct-19 12:51 AM	IPYNB File	72 KB
export_dataframe2	✓	17-Oct-19 12:49 AM	Microsoft Excel C...	12,497 KB
hashtag_divided	✓	17-Oct-19 12:55 AM	Microsoft Excel C...	44,129 KB
hashtag_to_term	✓	17-Oct-19 12:58 AM	Microsoft Excel C...	6,531 KB
hashtag_to_term	✓	17-Oct-19 1:07 AM	IPYNB File	166 KB
id_screenName_list	✓	21-Sep-19 6:51 PM	Microsoft Excel C...	657 KB
mutual_mention_list	✓	16-Oct-19 10:30 PM	IPYNB File	68 KB
screen_name_complete	✓	21-Sep-19 6:53 PM	IPYNB File	34 KB
user_mention_dictionary	✓	27-Sep-19 6:35 PM	IPYNB File	64 KB
user_to_mention	✓	25-Sep-19 1:28 AM	Microsoft Excel C...	5,735 KB
user_to_mention	✓	25-Sep-19 1:29 AM	IPYNB File	169 KB
user_to_term	✓	17-Sep-19 5:31 PM	Microsoft Excel C...	1,893 KB
user_to_term	✓	17-Sep-19 5:36 PM	IPYNB File	134 KB

4.1 EXTRACT HASHTAGS AND MENTIONS FROM RAW DATASETS

The first components we need are hashtags and mentions. We use the process for The Social Honeypot as an example here.

The trunk below shows how we filter the tweets with mention and get the hashtag(s) from the content.

```python
df_with_mention = df[df['content'].str.contains('@(\w+)')]
df_3d = df_with_mention
df_3d['mention'] = df_3d['content'].apply(lambda x: re.findall(r"@(\w+)", x)
)
df_3d['hashtag'] = df_3d['content'].apply(lambda x: re.findall(r"#(\w+)", x)
)
```

In this way, some tweets contain no hashtags or contain very rare hashtags which cannot provide much meaning here. Now we extract tweets the 500 most frequent hashtags.

```python
flat_list = []
for sublist in hashtag_to_list:
    for item in sublist:
        flat_list.append(item)

flat_list_np = np.array(flat_list)
unique,counts = np.unique(flat_list_np, return_counts=True)
hashtag_rank_np = np.asarray((unique,counts)).T
hashtag_rank_pd = pd.DataFrame(hashtag_rank_np)
hashtag_rank_pd[1] = pd.to_numeric(hashtag_rank_pd[1])
hashtag_rank_pd = hashtag_rank_pd.sort_values(by=[1],ascending = False)
hashtag_top500 = hashtag_rank_pd[:500]

df_3d['Used Frequent Tags'] = df_3d['hashtag'].apply(lambda x: set(x) & set(
hashtag_top500[0]) != set())
df_3d = df_3d[df_3d['Used Frequent Tags']==True]
df_3d = df_3d.drop(['Used Frequent Tags'],axis=1)
```

An output from this process will be formatted below.

text	user_id	spam	mention	hashtag
@frakintosh is also a Columbine survivor. Miles away from the big news coverage he is still	678033	FALSE	['frakintosh']	['fb']

struggling. http://t.co/SRbhQ3nweH#fb				
One reason my life has gotten better is the support of my dear friend @frakintosh. Now he needs help. Pls join me http://t.co/SRbhQ3nweH#fb	678033	FALSE	['frakintosh']	['fb']
RT @FINALLEVEL: Jon Stewart got it RIGHT again! Watch the wake up call he gave his #DailyShow viewers regarding the #BaltimoreRiots. http:/?€?	678033	FALSE	['FINALLEVEL']	['DailyShow', 'BaltimoreRiots']
@TWiBnation is grassroots media covering, on the ground, Baltimore, Ferguson and all the #BlackLivesMatter movement .	678033	FALSE	['TWiBnation']	['BlackLivesMatter']
MT@xeni All you need to know abt why there r there riots in Baltimore is in these charts. http://t.co/2xdobenYtf http://t.co/EeoXqLGJ2E #fb	678033	FALSE	['xeni']	['fb']

4.2 DATA COMPLETION FOR USERS' SCREENNAME

A list of users' screennames with matched id is a vital part for our project. Users' identifications are presented as id, but when they mention other people, they are using screenname.

For Social Honeypot Dataset, users' Screen Name were not provided. We completed this data by using twitter API. A python library Tweepy is also used. https://www.tweepy.org/

The start of the process is to divide the list of user_id into small groups with the size of 100, because each request to Twitter sever is restricted with this size. After that, each group of user_id goes to Twitter API for a json as return. We process this json into the list we want.

```python
def splitDataFrameIntoSmaller(df, chunkSize = 10000):
    listOfDf = []
    numberChunks = len(df) // chunkSize + 1
    for i in range(numberChunks):
        listOfDf.append(df[i*chunkSize:(i+1)*chunkSize])
    return listOfDf

user_list_split = splitDataFrameIntoSmaller(user_list, chunkSize = 100)

result = []
for each in user_list_split:
    id_list = each['user_id'].to_list()
    retrieved_json = api.lookup_users(user_ids=id_list)
    for item in retrieved_json:
        result.append(item)

df = pd.DataFrame(columns=['user_id', 'screen_name'])

for index,each in enumerate(result):
    df.at[index,'user_id'] = each.id
    df.at[index,'screen_name']=each.screen_name
```

Now we get the completed list of id_to_screenname as below.

user_id	screen_name
6301	msjacbowie
10836	rex_huang
10997	philcampbell
633293	iboughtamac
717883	djdick
763068	vuchuu
783705	davepeck

It is worth noting that many suspended accounts have been removed by twitter, so we cannot find their Screen Name from User ID. The sample size becomes smaller in this way.

4.3 USER TO TERM

The relationship between users and the content they are using is a major part of the sentimental analysis. Spammers prefer some terms over legitimate users. Here we want to discover this characteristic.

In the trunk below, we firstly remove all the punctuations and Stopwords which are words having non-significant meanings. And then the words are sorted according to their frequency in the entire document. The top 100 most frequent words are what we want for the next move.

```
top_N = 100

text = df['content'].str.lower().str.cat(sep=' ')
text = ''.join([word for word in text if word not in string.punctuation])
tokens = re.split('\W+', text)
tokens_stem = [ps.stem(token) for token in tokens]
word_dist = nltk.FreqDist(tokens_stem)

stopwords = nltk.corpus.stopwords.words('english')
words_except_stop_dist = nltk.FreqDist(w for w in tokens_stem if w not in st
opwords)

words_except_stop_dist.most_common(top_N)[:5]

top100_list = []
for a, b in words_except_stop_dist.most_common(top_N):
    top100_list.append(a)
```

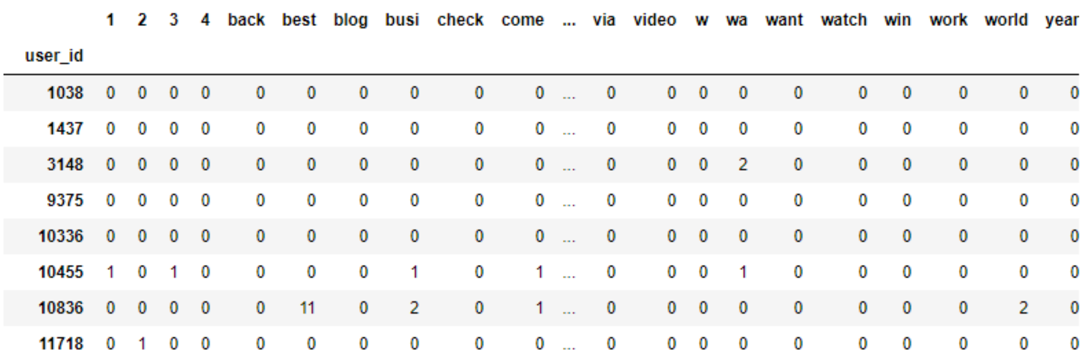
We get the top 100 most frequent terms for the datasets.

```
['rt', 'ff', 'follow', 'followfriday', 'thank', 'u', 'thi',
'via', 'new', 'love', 'get', '2', 'twitter', 'lol', 'de', 'im',
'e', 'quot', '1', 'make', 'free', 'go', 'like', 'great', 'fb',
'one', 'win', 'pleas', 'dont', 'good', 'tcot', 'day', 'wa',
'video', '4', 'que', 'ha', 'tweet', 'nowplay', 'time', 'know',
'see', 'us', 'live', 'da', 'shoutout', 'music', 'news', 'got',
'hi', 'peopl', 'want', 'fail', 'musicmonday', 'need', 'best',
'3', 'use', 'today', 'check', 'iranelect', 'look', 'say', 'um',
'think', 'work', 'show', 'back', 'eu', 'friday', 'ur', 'right',
'la', 'friend', 'year', 'money', 'take', 'man',
'tebakbandtransl', 'world', 'forex', 'come', 'watch', 'happi',
'market', 'iran', 'give', 'mm', 'blog', 'help', 'everi', 'por',
'tip', 'w', 'onli', 'life', 'busi', 'travel', 'let',
'justinbieb']
```

Once we have the most frequent terms, we can detect every user's usage on each of these terms. The CountVectorizer built-in in SK-Learn can help us achieve this goal.

```python
from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer(analyzer=clean_text)
X_count = count_vect.fit_transform(df['content'])

X_count_df = pd.DataFrame(X_count.toarray())
X_count_df.columns = count_vect.get_feature_names()

vectorized_df = pd.concat([df['user_id'],X_count_df],axis = 1)

user_to_term = vectorized_df.groupby(['user_id'])
user_to_term.sum()

user_to_term_sum=user_to_term.sum()
user_to_term_nonzeros =user_to_term_sum.loc[(user_to_term_sum!=0).any(axis=1
)]
```

The result of this feature is a Data Frame shown below



	1	2	3	4	back	best	blog	busi	check	come	...	via	video	w	wa	want	watch	win	work	world	year
user_id																					
1038	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1437	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3148	0	0	0	0	0	0	0	0	0	0	...	0	0	0	2	0	0	0	0	0	0
9375	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10336	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
10455	1	0	1	0	0	0	0	1	0	1	...	0	0	0	1	0	0	0	0	0	0
10836	0	0	0	0	0	11	0	2	0	1	...	0	0	0	0	0	0	0	0	2	0
11718	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

4.4 USER TO MENTION

We believe the mention relationship can indicate their possibility to be spammers. Firstly, we add screenname as a column into the dataset, so we can recognise who they are mentioning to.

The origin data is in a format as below.

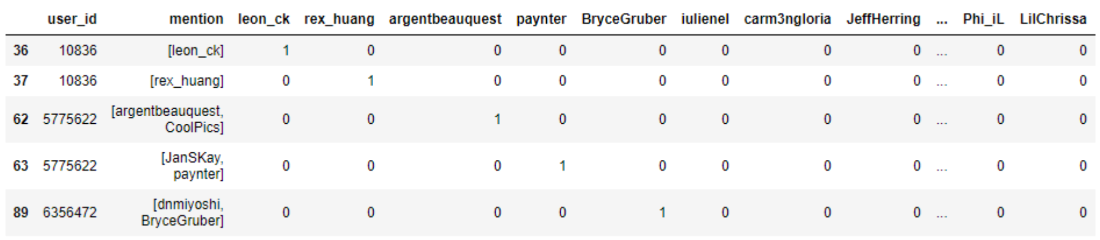
user_id	mention
678033	['frakintosh']
678033	['frakintosh']

678033	['FINALLEVEL']
678033	['TWiBnation']
91369005	[wayansjr, CRAIGWAYANS, AFFIONCROCKETT]

The list of id_screenname we have previously completed in the data completion. We will be building a vector-like data frame based on it. The users who are in the list should be removed as we are not able to recognise them.

```python
df_mention_inlist = df[df['mention'].apply(lambda a: set(a)&set(user_name_list) != set())]

df_mention_inlist_play = df_mention_inlist.copy()
for index,row in df_mention_inlist_play.iterrows():
    for mention in row['mention']:
        if mention in user_name_list:
            df_mention_inlist_play.at[index,mention] = 1

df_mention_inlist_play = df_mention_inlist_play.fillna(0)
To_int_list = list(df_mention_inlist_play.columns)
To_int_list.remove('mention')
```

Now we get a vectorized dataframe.

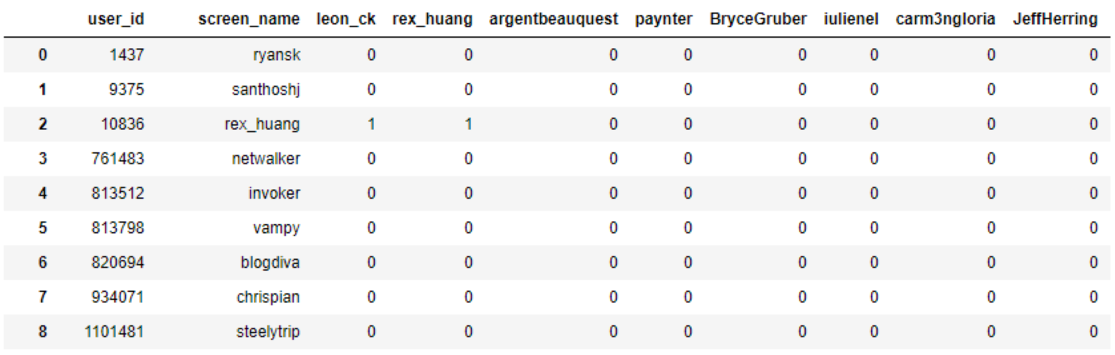


	user_id	mention	leon_ck	rex_huang	argentbeauquest	paynter	BryceGruber	iulienel	carm3ngloria	JeffHerring	...	Phi_iL	LilChrissa
36	10836	[leon_ck]	1	0	0	0	0	0	0	0	...	0	0
37	10836	[rex_huang]	0	1	0	0	0	0	0	0	...	0	0
62	5775622	[argentbeauquest, CoolPics]	0	0	1	0	0	0	0	0	...	0	0
63	5775622	[JanSKay, paynter]	0	0	0	1	0	0	0	0	...	0	0
89	6356472	[dnmiyoshi, BryceGruber]	0	0	0	0	1	0	0	0	...	0	0

They list of screen_name should be added into the dataframe, and the value should be grouped by the user_id.

```python
name_added_df = df_mention_inlist_play.join(user_list.set_index('user_id'), on='user_id',how='inner')

cols = name_added_df.columns.tolist()
cols = cols[-1:] + cols[:-1]
name_added_df = name_added_df[cols]

user_to_mention = name_added_df.groupby(['user_id','screen_name'])
user_to_mention.sum().reset_index()
```

We can finalised the user_to_mention data frame in this process.



	user_id	screen_name	leon_ck	rex_huang	argentbeauquest	paynter	BryceGruber	iulienel	carm3ngloria	JeffHerring
0	1437	ryansk	0	0	0	0	0	0	0	0
1	9375	santhoshj	0	0	0	0	0	0	0	0
2	10836	rex_huang	1	1	0	0	0	0	0	0
3	761483	netwalker	0	0	0	0	0	0	0	0
4	813512	invoker	0	0	0	0	0	0	0	0
5	813798	vampy	0	0	0	0	0	0	0	0
6	820694	blogdiva	0	0	0	0	0	0	0	0
7	934071	chrispian	0	0	0	0	0	0	0	0
8	1101481	steelytrip	0	0	0	0	0	0	0	0

4.5 RADIO OF IN-GOING AND OUT-GOING MENTION

From the data frame we have created, we can extract it further to create a meaningful feature. This feature is based on an assumption that spammers are more likely to mention others instead of being mentioned. We have the equation below.

$$\frac{No\ of\ Mentions\ for\ u_j}{No\ of\ Mentions\ by\ u_j}$$

What we need now is the number of outgoing mentions and the ingoing mentions. The number of out-going mentions is the summation of values in a row. However, the number of in-going mentions Is the summation of values in a column. Then the radio can be simply calculated by division.

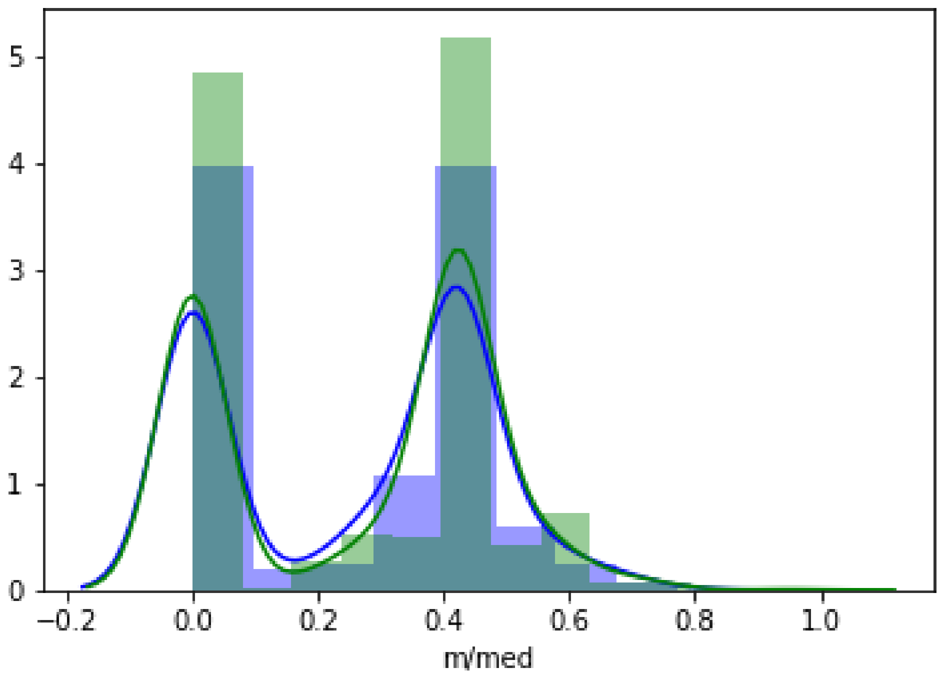
```
df['mention'] = df_new.sum(axis=1)

for x in df['screen_name']:
    if x in list(df):
        df.mentioned[df['screen_name']==x]=df_new[x].sum()

df['mentioned'] = df['mentioned'].fillna(0)
df['m/med'] = df['mentioned']/df['mention']
print(df[['user_id','screen_name','mention','mentioned','m/med']])
```

We get the data frame for this feature.

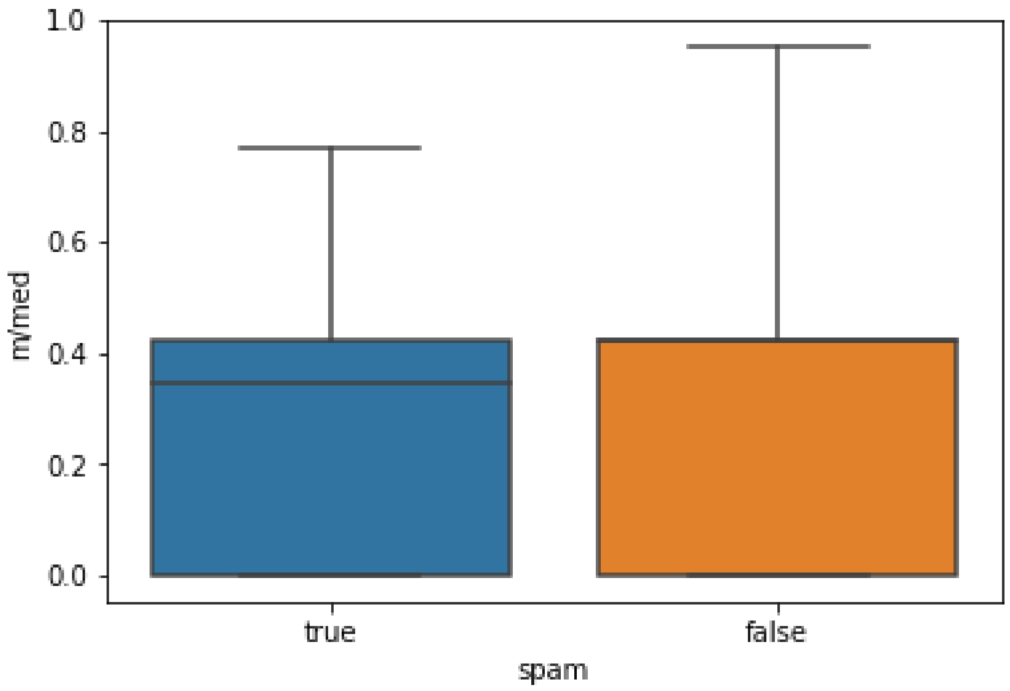
	user_id	screen_name	mention	mentioned	med/m
0	1437	ryansk	1	1.0	1.000000
1	9375	santhoshj	2	2.0	1.000000
2	10836	rex_huang	2	1.0	0.500000
3	761483	netwalker	1	0.0	0.000000
4	813512	invoker	1	3.0	3.000000
5	813798	vampy	1	2.0	2.000000
6	820694	blogdiva	4	2.0	0.500000

The distribution of the radio value from spammers and legitimate users can be compared. We normalise the data and plot them as histogram and boxplot.

```
if(True):
    col = 'm/med'
    df_with_spam[col] = df_with_spam[col].apply(lambda x: x+1)
    df_with_spam[col] = df_with_spam[col].apply(np.log)

    df_with_legit[col] = df_with_legit[col].apply(lambda x: x+1)
    df_with_legit[col] = df_with_legit[col].apply(np.log)

import matplotlib.pyplot as plt
import seaborn as sns


sns.distplot(df_with_spam['m/med'],norm_hist=True,color='blue')
sns.distplot(df_with_legit['m/med'],norm_hist=True,color='green')
plt.show()
```



```
 df_with_label['m/med']=df_with_label['m/med'].apply(lambda x: x+1)
df_with_label['m/med']=df_with_label['m/med'].apply(np.log)

sns.boxplot(x='spam', y='m/med',data=df_with_label)
plt.show()
```



4.6 MUTUAL MENTION

Another feature we can develop from the data frame of user_to_mention is the mutual mention list.

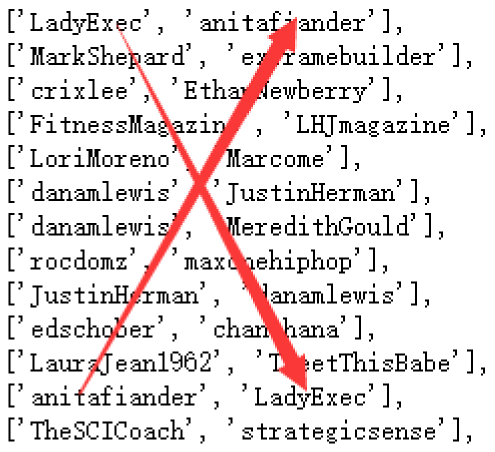
Firstly, we create a dictionary of mention relationship.

```python
mention_dic = {}
for index,row in df.iterrows():
    screen_name = row['screen_name']
    row[screen_name] = 0
    mention_dic[screen_name] = [item for item in columns_names if row[item]!= 0]

mention_dict = {k: v for k, v in mention_dic.items() if len(v)!=0}
```

A mutual mention list can then be created.

```python
mutual_list = []
for key in mention_dict:
    value = mention_dict[key]
    for each_value in value:
        if each_value in mention_dict:
            if key in mention_dict[each_value]:
                mutual_list.append([key,each_value])
```

We can see the mutual mention from the result below.



```
['LadyExec', 'anitafiander'],
['MarkShepard', 'extramebuilder'],
['crixlee', 'EthanNewberry'],
['FitnessMagazine', 'LHJmagazine'],
['LoriMoreno', 'Marcome'],
['danamlewis', 'JustinHerman'],
['danamlewis', 'MeredithGould'],
['rocdomz', 'maxthehiphop'],
['JustinHerman', 'danamlewis'],
['edschober', 'chanhana'],
['LauraJean1962', 'TweetThisBabe'],
['anitafiander', 'LadyExec'],
['TheSCICoach', 'strategicsense'],
```

4.7 Hashtag to Term

Hashtag always indicates the topic of the tweet content. However, the spammers only care the traffic that trending hashtags can bring. So the relationship between topic and tweet content is worth to be analysed.

The data frame we get from pre-processing have three columns below.

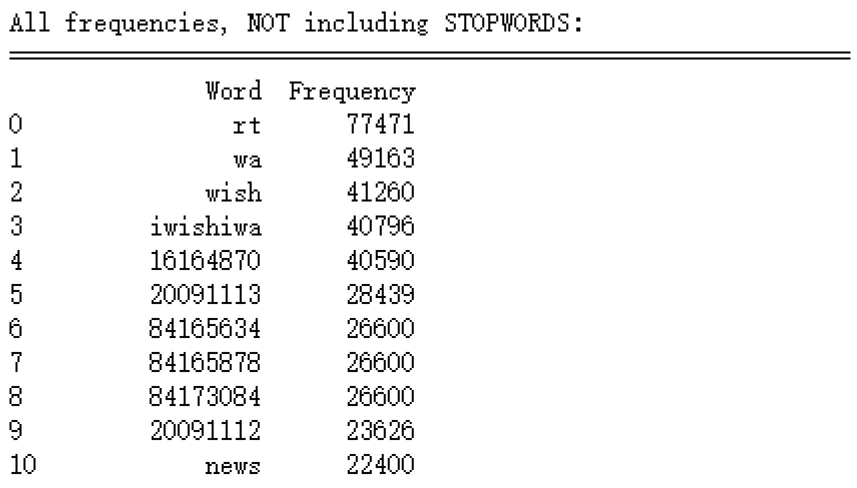
	user_id	content	hashtag
0	10836	RT @techsailorgroup ashley/techsailor at #tech...	[techsailor]
1	10836	RT@techsailorgroup Isaiah Pang / Entrepreneur...	[techsailor]
2	10836	RT@techsailorgroup Junji / turner at #techsail...	[techsailor]
3	10836	RT@tehcsailorgroup Lester Kok / Straits Times ...	[techsailor]
4	10836	RT@techsailorgroup Nor / Paddle Culture Inter...	[techsailor]

As we have done in user to term section, we aim to analyse the most frequent terms. We need a reasonable size of samples to fit the future model training.

```python
import nltk
import string
import re

stopwords = nltk.corpus.stopwords.words('english')
ps = nltk.PorterStemmer()

def clean_text(text):
    text = ''.join([word.lower() for word in text if word not in string.punc
tuation])
    tokens = re.split('\W+', text)
    text = [ps.stem(word) for word in tokens if ps.stem(word) in top100_list
]
    return text

```

```python
top_N = 100
text = df['content'].str.lower().str.cat(sep=' ')
text = ''.join([word for word in text if word not in string.punctuation])
tokens = re.split('\W+', text)
tokens_stem = [ps.stem(token) for token in tokens]
word_dist = nltk.FreqDist(tokens_stem)

stopwords = nltk.corpus.stopwords.words('english')
words_except_stop_dist = nltk.FreqDist(w for w in tokens_stem if w not in stopwords)

print('All frequencies, NOT including STOPWORDS:')
print('=' * 60)
rslt = pd.DataFrame(words_except_stop_dist.most_common(top_N),
                    columns=['Word', 'Frequency'])
print(rslt)
print('=' * 60)
```

the most frequent hashtags can be printed out.



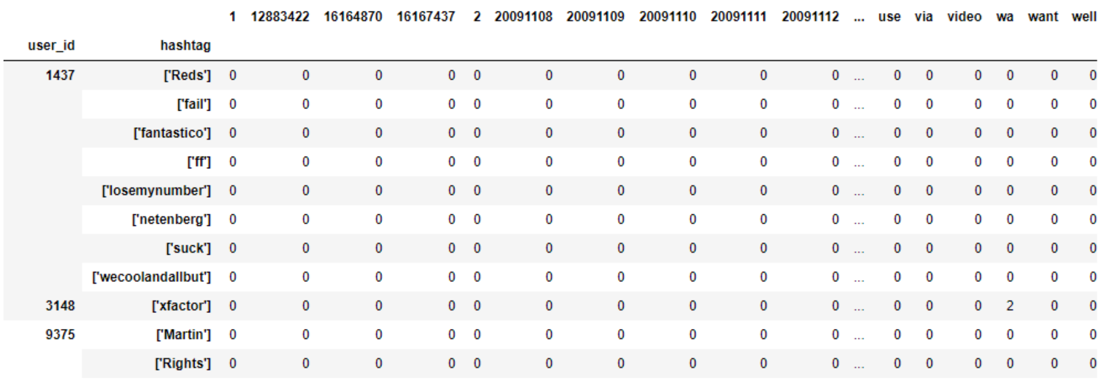
```
All frequencies, NOT including STOPWORDS:
============================================================
        Word  Frequency
0         rt      77471
1         wa      49163
2       wish      41260
3   iwishiwa      40796
4   16164870      40590
5   20091113      28439
6   84165634      26600
7   84165878      26600
8   84173084      26600
9   20091112      23626
10      news      22400
```

We will build a vectorized data frame with these frequent terms. Hashtags without frequent terms will be removed from data frame.

```python
 from sklearn.feature_extraction.text import CountVectorizer

count_vect = CountVectorizer(analyzer=clean_text)
X_count = count_vect.fit_transform(df['content'])

X_count_df = pd.DataFrame(X_count.toarray())
X_count_df.columns = count_vect.get_feature_names()

```

```
vectorized_df = pd.concat([df['user_id'],df['hashtag'],X_count_df],axis = 1)

vectorized_df = vectorized_df.astype({'hashtag':str})

user_to_term = vectorized_df.groupby(['user_id','hashtag'])
user_to_term.sum()
user_to_term_sum=user_to_term.sum()
user_to_term_nonzeros =user_to_term_sum.loc[(user_to_term_sum!=0).any(axis=1)]
```

The result is shown below.



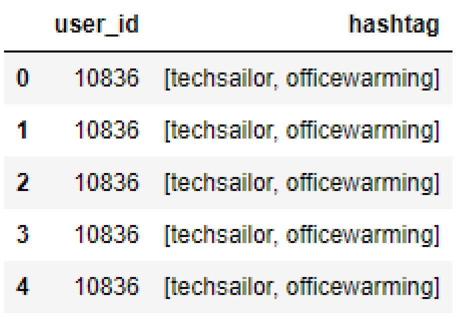
		1	12883422	16164870	16167437	2	20091108	20091109	20091110	20091111	20091112	...	use	via	video	wa	want	well
user_id	hashtag																	
1437	['Reds']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['fail']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['fantastico']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['ff']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['losemynumber']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['netenberg']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['suck']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['wecoolandallbut']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3148	['xfactor']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	2	0	0
9375	['Martin']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
	['Rights']	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0

4.8 Features for Similarity Analysis

Groups of users can be identified by their similarity. It can be calculated by their usage of common topics which show their interests. The equation shows below.

$$\frac{No\ of\ common\ topics\ between\ u_i, u_j}{Total\ No\ of\ topics\ of\ u_i\ + Total\ No\ of\ topics\ of\ u_j}$$

We have processed data like below



	user_id	hashtag
0	10836	[techsailor, officewarming]
1	10836	[techsailor, officewarming]
2	10836	[techsailor, officewarming]
3	10836	[techsailor, officewarming]
4	10836	[techsailor, officewarming]

In order to get the number of commons topics, we transform the data into a format which grouped by user_id. Redundant hashtags for each user are removed by using set() function.

```
from functools import reduce
```

```python

def Union(lst1, lst2):
    final_list = list(set(lst1) | set(lst2))
    return final_list

def common_topic(series):
    return reduce(lambda x, y: Union(x,y), series)
df = df.groupby('user_id').agg({'hashtag': common_topic})
df.head()
```

user_id	hashtag
1038	[paniekpeeuw, 2012, fail]
1437	[ff, Reds, netenberg, losemynumber, fantastico...
3148	[xfactor]
9375	[Wynne, celtic, of, man, Martin, music, Rights]
10336	[ff, vegan, blogher, hcr]

We define a function to get the intersection of hashtags and union of hashtags for computing the ratio.

```python
def radio_common_topic(df,user1,user2):
    x1 = df.loc[user1]['hashtag']
    x2 = df.loc[user2]['hashtag']
    intersection = len(set(x1)&set(x2))
    union = len(x1)+len(x2)
    print("Score: " + str(intersection/union))
    return intersection/union
```

For example, the two users with 1437 and 304713 as user_id have a similarity of 0.0625 toward their common topics.

```
radio_common_topic(df,1437,304713)
```

```
Score: 0.0625
```

Another feature can be Similarity of content used in two users' contents for a given topic.

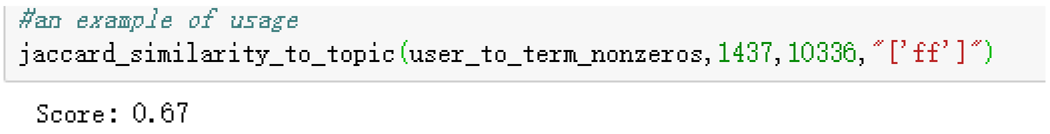
$$\text{For topic, } T_k \quad Sim(C_{u_i}, C_{u_j},)$$

Jaccard similarity is used : Jaccard Similarity = (Intersection of A and B) / (Union of A and B).

Based on the hashtag_to_term data frame, we define a function:

```python
import sklearn.metrics

def jaccard_similarity_to_topic(df,user1,user2,topic):
    columns_names = list(df.columns)
    x1 = [item for item in columns_names if df.loc[user1,topic][item]!= 0]
    x2 = [item for item in columns_names if df.loc[user2,topic][item]!= 0]
    intersection= len(set(x1)&set(x2))
    union = []
    union.extend(x1)
    union.extend(x2)
    union = len(set(union))
    score =intersection/union
    print('Score: %.2f' % (score))
```

for example, the two users with 1437 and 10336 as user_id have a similarity of 0.67 toward the topic 'ff'.



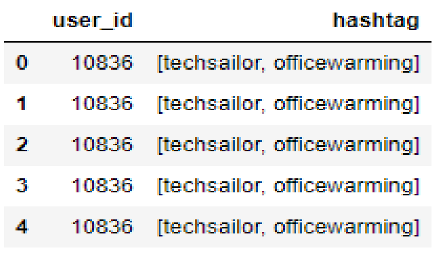
```python
#an example of usage
jaccard_similarity_to_topic(user_to_term_nonzeros,1437,10336,"['ff']")
```

```
Score: 0.67
```

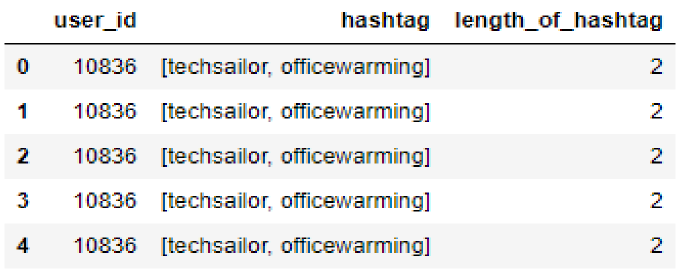
4.9 USER TO HASHTAG

According the equation, the total number of hashtags for each user is needed, and the total number of posts as well. The process of data frame transformation has three steps which are displayed below.

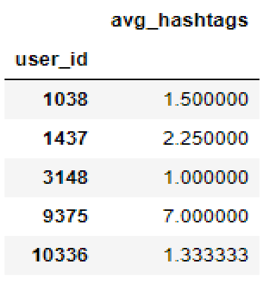
$$\frac{\sum Number\ of\ Hashtags\ in\ tweets}{Total\ number\ of\ tweet}$$



	user_id	hashtag
0	10836	[techsailor, officewarming]
1	10836	[techsailor, officewarming]
2	10836	[techsailor, officewarming]
3	10836	[techsailor, officewarming]
4	10836	[techsailor, officewarming]



	user_id	hashtag	length_of_hashtag
0	10836	[techsailor, officewarming]	2
1	10836	[techsailor, officewarming]	2
2	10836	[techsailor, officewarming]	2
3	10836	[techsailor, officewarming]	2
4	10836	[techsailor, officewarming]	2



	avg_hashtags
user_id	
1038	1.500000
1437	2.250000
3148	1.000000
9375	7.000000
10336	1.333333

The implementation on python is:

```python
df_avg_hashtag['length_of_hashtag']=df_avg_hashtag['hashtag'].apply(lambda x:len(x) if x!=len(x) else 0)

df_avg_hashtag=df_avg_hashtag.drop('hashtag',axis=1)
df_avg_hashtag=df_avg_hashtag.rename(columns={'length_of_hashtag':'avg_hashtags'})
df_avg_hashtag_avg = df_avg_hashtag.groupby('user_id').mean()
```

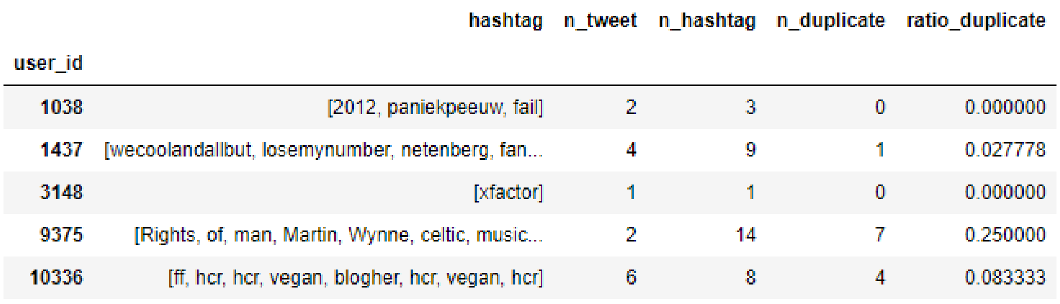
Similarly, the Duplicate Hashtag usage can be calculated, as we assume that spammers may post more duplicate hashtags than normal users.

According to the equation, the number of duplicate hashtags is needed. We show our code below:

$$\frac{\sum Number\ of\ duplicate\ Hashtags}{Total\ number\ of\ tweets\ \times\ Number\ of\ Hashtags}$$

```python
df_duplicate['n_tweet'] = '1'
df_duplicate = df_duplicate.groupby('user_id').sum()
df_duplicate['n_tweet'] = df_duplicate['n_tweet'].apply(lambda x:len(x))
df_duplicate['n_hashtag'] = df_duplicate['hashtag'].apply(lambda x:len(x))

def calculate_duplicate(hashtags):
    n_duplicate = len(hashtags) - len(set(hashtags))
    return n_duplicate

df_duplicate['n_duplicate'] = df_duplicate['hashtag'].apply(lambda x:calculate_duplicate(x))
df_duplicate['ratio_duplicate'] = df_duplicate['n_duplicate']/(df_duplicate['n_tweet']*df_duplicate['n_hashtag'])
```

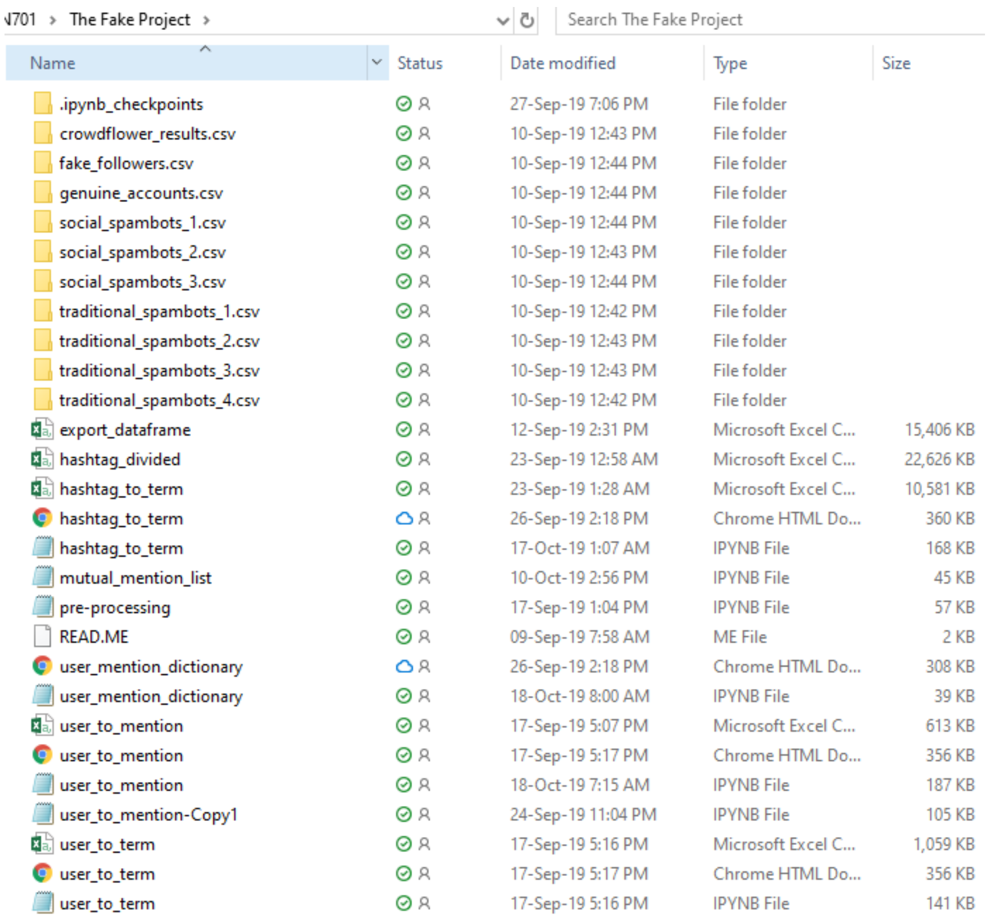
The output data frame shows the value for the usage of duplicate hashtags.



	hashtag	n_tweet	n_hashtag	n_duplicate	ratio_duplicate
user_id					
1038	[2012, paniekpeeuw, fail]	2	3	0	0.000000
1437	[wecoolandallbut, losemynumber, netenberg, fan...	4	9	1	0.027778
3148	[xfactor]	1	1	0	0.000000
9375	[Rights, of, man, Martin, Wynne, celtic, music...	2	14	7	0.250000
10336	[ff, hcr, hcr, vegan, blogher, hcr, vegan, hcr]	6	8	4	0.083333

4.10 THE FAKE PROJECT

In order to test if the processes we have applied on The Social Honeypot can be applied widely on other datasets as well. We select The Fake Project as the second dataset for testing. The result is satisfied as we extracted the same features successfully.



701 › The Fake Project ›

Search The Fake Project

Name	Status	Date modified	Type	Size
.ipynb_checkpoints	✓	27-Sep-19 7:06 PM	File folder	
crowdflower_results.csv	✓	10-Sep-19 12:43 PM	File folder	
fake_followers.csv	✓	10-Sep-19 12:44 PM	File folder	
genuine_accounts.csv	✓	10-Sep-19 12:44 PM	File folder	
social_spambots_1.csv	✓	10-Sep-19 12:44 PM	File folder	
social_spambots_2.csv	✓	10-Sep-19 12:43 PM	File folder	
social_spambots_3.csv	✓	10-Sep-19 12:44 PM	File folder	
traditional_spambots_1.csv	✓	10-Sep-19 12:42 PM	File folder	
traditional_spambots_2.csv	✓	10-Sep-19 12:43 PM	File folder	
traditional_spambots_3.csv	✓	10-Sep-19 12:43 PM	File folder	
traditional_spambots_4.csv	✓	10-Sep-19 12:42 PM	File folder	
export_dataframe	✓	12-Sep-19 2:31 PM	Microsoft Excel C...	15,406 KB
hashtag_divided	✓	23-Sep-19 12:58 AM	Microsoft Excel C...	22,626 KB
hashtag_to_term	✓	23-Sep-19 1:28 AM	Microsoft Excel C...	10,581 KB
hashtag_to_term	☁	26-Sep-19 2:18 PM	Chrome HTML Do...	360 KB
hashtag_to_term	✓	17-Oct-19 1:07 AM	IPYNB File	168 KB
mutual_mention_list	✓	10-Oct-19 2:56 PM	IPYNB File	45 KB
pre-processing	✓	17-Sep-19 1:04 PM	IPYNB File	57 KB
READ.ME	✓	09-Sep-19 7:58 AM	ME File	2 KB
user_mention_dictionary	☁	26-Sep-19 2:18 PM	Chrome HTML Do...	308 KB
user_mention_dictionary	✓	18-Oct-19 8:00 AM	IPYNB File	39 KB
user_to_mention	✓	17-Sep-19 5:07 PM	Microsoft Excel C...	613 KB
user_to_mention	✓	17-Sep-19 5:17 PM	Chrome HTML Do...	356 KB
user_to_mention	✓	18-Oct-19 7:15 AM	IPYNB File	187 KB
user_to_mention-Copy1	✓	24-Sep-19 11:04 PM	IPYNB File	105 KB
user_to_term	✓	17-Sep-19 5:16 PM	Microsoft Excel C...	1,059 KB
user_to_term	✓	17-Sep-19 5:17 PM	Chrome HTML Do...	356 KB
user_to_term	✓	17-Sep-19 5:16 PM	IPYNB File	141 KB

5 DISCUSSION

Previous works put a lot of attention on the methodologies and algorithms of Multiview learning instead of the ingredients for Multiview learning, saying feature extraction. In this report, we focus on the extraction process as a complementary fragment to the previous works. In addition, this report is also an example of the implementation of the previous work. The effort of collecting these features can be clearly seen from the coding section. It can indicate the cost of Multiview learning.

The dataset choices in this project are real world datasets but they are outdated. We believe the spammers are evolving and adapting their strategies to abusing the ranking system on Online Social Networks. The characteristics of the modern spammers may not be seen from outdated datasets. However, the processes and concepts about these features can be applied when analysing a current business or research project.

The features for sentimental analysis in this project are very limited on the size of frequency terms. We only took the most 100 frequent terms into consideration while the top 200 or 300 terms remained. We would like to propose future comparation on the different size of terms for a deeper understanding.

The feature we extracted in this project are not fully verified. Some of them are based on the observation on the behaviour of the spammers, and some of them are just experimentally extracted. In addition, some features might not be useful for some datasets as the strategies that spammers use vary from time, location and platforms. In order to verify them, the further investigation is needed. The importance of each feature can be seen once they are taken as ingredients for machine learning. Multiview learning is suggested because that is all these features prepare for. The multiple view learning can produce a classifier and the impact of each feature on the classification indicates the value of it.

6 CONCLUSION

This project aimed to help identify spammers from legitimate users on the Online Social Network. In order to do this, we have built a four-dimensional data frame and then extracted features from it. The four-dimensional data frames were processed from the real word datasets, The Social Honeypot and The Fake Project. Four dimensions are User_ID, Hashtag, Mention and Content. They are the information we all can access in daily life and we investigate them further in this project.

Previous researchers have put a lot of effort on the algorithms development which is the most important part in the multiple learning. However, in this project, we focused on the feature extraction which outputs the ingredients as views for the algorithms of multiple view learning.

The features are the relationship between different dimensions. In this project, we have discovered User_to_hashtag, User_to_term, User_to_mention, Hashtag_to_terms, Mutual mention and Radio of in-going and outgoing mention. We classify them into two mains groups according their purposes, sentiment analysis and similarity analysis. Each feature was presented as Pandas data frame which is the most popular python library for data science.

Sentiment analysis is the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral. The usage of words, hashtags and mentions from users have been presented in a computational format for this analysis. On the other hand. Similarity analysis is to the measure how much alike two data objects are. In this report we have developed the process of measuring the distance of two users' contents for a given topic/hashtag. The distance is able to make clusters which help semi-supervised learning.

The outputs of this project are presented as python code and coma delimited files. We used python library like Panda, NumPy, NLTK and Scikit-learn to transform/process the tweet information into looked-for data frames. Missing data have been completed by Twitter API and Tweepy. All the contents have been vectorized into terms which can be interpreted by computers. The final outputs are different features based on the combination of different dimentions.

The processes applying on The Social Honeypot have been illustrated in this report, including coding and motivations. Moreover, the screenshots of output Data Frames were also allocated in each section. On the other hand, the processes applying on The Fake Project were for verification. The same format of results from The Fake Project dataset have proved the reusability which means the processes can be applies on other datasets as well.

7 REFERENCE

Blum, A., & Mitchell, T. (1998). *Combining labeled and unlabeled data with co-training*. Paper presented at the Proceedings of the eleventh annual conference on Computational learning theory, Madison, Wisconsin, USA.

Caverlee, K. L. a. B. D. E. a. J. (2011). Seven months with the devils: a long-term study of content polluters on Twitter. *In AAAI Int'l Conference on Weblogs and Social Media (ICWSM*.

Chao, G., Sun, S., & Bi, J. (2017). A Survey on Multi-View Clustering. *ArXiv, abs/1712.06246*.

The DSDM Agile Project Framework. (2014). Retrieved from https://www.agilebusiness.org/content/philosophy-and-fundamentals

Hadian, A., & Minaei, B. (2013). Multi-View Learning for Web Spam Detection. *Journal of Emerging Technologies in Web Intelligence, 5*. doi:10.4304/jetwi.5.4.395-400

Hendrickson R. Langbehn, S. R., Marcos A. Gonçalves, Jussara M. Almeida, Gisele L. Pappa, Fabrício Benevenuto. (2010). A Multi-view Approach for Detecting Non-Cooperative Users

Herzallah, W., Faris, H., & Adwan, O. (2018). Feature engineering for detecting spammers on Twitter: Modelling and analysis. Journal of Information Science, 44(2), 230–247. https://doi.org/10.1177/0165551516684296

in Online Video Sharing Systems. *Universidade Federal de Minas Gerais, Brazil*.

Reiter Langbehn, H., Ricci, S., Gonçalves, M., Almeida, J., Pappa, G., & Benevenuto, F. (2010). A Multi-view Approach for Detecting Non-Cooperative Users in Online Video Sharing Systems. *JIDM, 1*, 313-328.

Shen, H., Ma, F., Zhang, X., Zong, L., Liu, X., & Liang, W. (2017). Discovering social spammers from multiple views. *Neurocomputing, 225*, 49-57. doi:https://doi.org/10.1016/j.neucom.2016.11.013

Xu, C., Tao, D., & Xu, C. (2013). A Survey on Multi-view Learning. *ArXiv, abs/1304.5634*.

8 REFLECTION OF MY LEARNING

The project management is the part I feel the best. I have had frequent communication with my tutor. It makes sure our project is on the right track. We also designed sprints for each week based on the previous work with flexibility, so the workload throughout the semester was well balanced. Moreover, the instruction and feedback from my tutor every week ensured the quality of the work, and I really appreciate that.

A section I did least well might be the feature identifying. I did not research enough on the behaviour of spammers to identify valuable features. You may find there is not many innovative features in this project because of the lack of understanding toward spammers.

The most challenging part in this project is the paper reading. There were many terminologies which I had not seen before. A page full of strange words made me feel I could not even read them. Even the three main method of multiple view learning are still difficult to understand as they are quite abstract. In addition, I find it is very difficult to understand the algorithms for Multiview learning.

The mathematical skill is the one I need the most for future development. Once I get this skill, I will be able to fully understand the papers with a lot of equations and mathematical symbols. Then a project pipeline of multiple learning can be address by my own.

The most important thing I learnt from doing this project is the way of learning advanced technics. As a data science student, I was focused on programming skill a lot. I thought learning the documentation and tutorials from online are quite enough for being a good programmer. Now I realise the most advanced technics are not widely applied, which means there is not a lot information you can find online. Reading papers is an essential part to address new problems and create bigger values. It would be my daily life if I put my career on nature language processing which heavily relies on innovative methods.

This project is a good practice for my programming skill. I have always been learning by doing when writing these programs. I have read a lot of documentation of python packages and new methods of dealing with data on Stackoverflow. The skill will help me to do better in data science area.

I am looking forward to taking the next step, which is building a classifier from multiple learning algorithms. In this project features are not well verified in the datasets, but they could be if they are put into a machine learning algorithm. A model with capability to make more precise prediction is my desired result for putting into my resume. Moreover, there is not many Multiview learning library available online. I believe building a reusable library of Multiview learning is able to bring me a sense of achievement.