TRACKING MENTAL HEALTH USING SOCIAL MEDIA
by

MANISH RANJAN

(Under the guidance of Shannon Quinn)

Recently, “Big Data” techniques have been successfully used to solve challenging problems in healthcare. Such techniques have given rise to the development of “biosurveillance” frameworks. These frameworks are an application of big data processing paradigms which addresses the problem of identifying and predicting threats to public health. However, existing biosurveillance platforms are limited in their applicability to task such as detecting seasonal outbreaks of flu or specific mental disorder conditions like schizophrenia. We present a biosurveillance framework that not only anticipates public health threats, but can identify at risk individuals from social media for non contagious diseases in the realm of mental health disorders. In our proposed framework, we combine topic modeling with sentiment analysis to provide an estimate of toxic or abusive behavior, identifying a pool of potentially at-risk users via their content on social media. This framework can be tuned on incoming data incrementally over a period of time, which ensures better results over time on unobserved data. In a more mature phase our framework could be used by medical professionals to monitor and study users for their mental health disorders more closely and accurately.
INDEX WORDS: Mental Health, Behavioral health, health-surveillance, Twitter, Data Mining, Scalable Machine Learning, wellness, Topic modelling, NLP techniques.

# TRACKING MENTAL HEALTH USING SOCIAL MEDIA

by

MANISH RANJAN

B.TECH., SASTRA UNIVERSITY, INDIA, 2008

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2016

© 2016

Manish Ranjan

All Rights Reserved

TRACKING MENTAL HEALTH USING SOCIAL MEDIA

by

MANISH RANJAN

| | |
|---|---|
| Major Professor: | Shannon Quinn |
| Committee: | Lakshmish Ramaswamy |
| | Timothy G. Heckman |

Electronic Version Approved:

Suzanne Barbour
Dean of the Graduate School
The University of Georgia
May 2016

DEDICATION

This one for you, Arpita.

ACKNOWLEDGEMENTS

I would like to thank Dr. Shannon Quinn for his continued guidance throughout my graduate academic career. I would also like to thank Dr. Lakshmish Ramaswamy from whom I learned a lot about distributed systems and his guidance. I would like to thank Dr. Timothy Heckman for accepting to provide me guidance. I would also like to thank Al Aila, BahaaEddin, a MS in Artificial Intelligence student at the University of Georgia for all those white board discussions. Lastly, I would like to thank all my professors from whom I have learnt a lot throughout my academic career and University of Georgia for providing a conducive learning environment.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

In this chapter, we have briefly discussed the quantitative aspect of problem we are trying to address and how social media can be helpful in solving this. We have also given overview of generic challenges associated with consuming social media for data science. In the last subsection we give a concise idea on our contribution toward solving this challenge.

## 1.1 Introduction

Being mentally healthy is defined by World Health Organization (WHO) as state of well being in which (1) every individual realizes her or his own potential, (2) can cope with the normal stresses of life, (3) can work productively and fruitfully, and (4) is able to make a contribution to her or his community. However, having mental illness is a rather more serious problem. National Institute of Mental Illness (NAMI's) report suggest that there are 43.8 million adults experiencing mental illness per year. NAMI's latest reports also indicate that,

- 1 in every 5 adults in America experiences a mental illness.
- Nearly one in 25 adults in America live with a serious mental illness.
- One half of all the chronic illnesses begin by the age of 14 and three quarter by age of 30.

Above data suggests mental disorders are common in the United States. However, mental disorder can be categorized as a disease only if a person experiences disability due to

serious mental illness (SMI). The criterion to define SMI is as follows as per National Survey on Drug Use and Health (NSDUH).

- A mental, behavioral, or emotional disorder (excluding developmental and substance use disorders)
- Diagnosable currently or within the past year;
- Of sufficient duration to meet diagnostic criteria specified within the 4th edition of the Diagnostic and Statistical Manual of Mental Disorders (DSM-IV); and,
- Resulting in serious functional impairment, which substantially interferes with or limits one or more major life activities.

Given the widespread presence of variations of mental disorder, tracking it from its early onset with a generic approach is very important.

### 1.2 Why Social Media

NAMI's statistics suggests that mental illness is a problem which affects younger demographics more than adult ones. However, mental illness does not manifest the same way as more traditional disease like influenza; often there are no physical symptoms until it is too late.

The spectrum of disorder consists of: anxiety, mood, psychotic, eating, addiction, personality, obsessive compulsive disorders (OCD), Post-traumatic Stress disorders (PTSD) etc. [1]. Here, anxiety disorder is chronic mental condition characterized by an excessive and persistent sense of apprehension. Mood disorder is another psychological disorder characterized by the elevation or lowering of a person's mood, such as depression or bipolar disorder, to define a few.

Many states have already started the process to set up “Bio Surveillance” systems to detect epidemics, finding new health topics and trends using social media as input to harvest upon the social data derived knowledge power. US Department of Health & Human Services (http://nowtrending.hhs.gov/) is one such example.

However, being able to set up a broader criterion like “negativity over all” or “sharing toxic content” is not yet solved in a meaningful way. There has been some work like “Tracking Mental Disorders Across Twitter Users” [2]. It is a quality work but it restricts itself to very specific mental health issues. Further, validation of build rules, relies on crowdsourcing. A potential problem with crowdsourcing with respect to tagging mental illness is, it is a very subtle problem in early phases. Detection by just looking at text is difficult even for very experienced psychiatrist. Crowdsourcing such detection hence is definitely not an accurate as well as scalable way to go forward. “Economic Costs of Alcohol and Drug Abuse and Mental Illness” 1985 is another sound theoretical work from an economist’s perspective, but suffers from the lack of readily available implementation strategy.

### 1.3 Who Else is Trying to Solve the Same Problem

Figure 1: Twitter’s related ongoing research

On the same line, Apple has released an app named "HealthKit" for tracking mental health of individuals in last quarter (Quarter 4 - 2015) and has shared plans on roadmap of this app as well.

### 1.4 Challenges in Social Media

Tracking mental health behavior in an online age could be attempted by tracking social media posts. Online social data has in general has two famous outlets, blog posts and Twitter/Facebook. Blog posts are very difficult to sample content from a large number of unique users. One possible way to use blog posts is to connect with 4k sampled unique user's blog individually and download their content. This is a non scalable approach since it involves detection of user of interest, finding their blog posts and then downloading content one by one. Twitter on the other hand makes it very easy with its open interface architecture. We can plugin the API to get access to samples from a public stream.

Despite twitter's ease of access to data, it has few unique challenges. Firstly, Twitter provides a very conducive platform for the rapid development of online rhetoric. These rhetorics neither appear in dictionary nor are formally recognized as a part of language. But it carries semantic meaning to large number of users and it is therefore critical to identify user's intent from the rhetorics. Secondly, 140 characters' limitation on tweets forces users to place many words together with incomplete and incorrect spellings. This makes data processing and building models in NLP domain a challenge. This is partially unsupervised problem in some sense, as we do not have a priori knowledge of

all possible terms that are indicative of mental well-being. Furthermore, these terms are changing constantly, as new terms appear on the fly in response to changing circumstances.

### 1.5 Major Contribution

We wanted to study users who were filtered under broader criterion like "negativity over all" or "sharing toxic content". Further we wanted to study in detail if sharing of negative and toxic content on social media could be a potential proxy for underlying mental illness.

We focus on such filtered users based on not only toxic or negative content they share, whom they follow, who follows them and what signals they choose to amplify. Along with our own sentiment analysis algorithm, by adding network features, we seek to improve detection accuracy.

For detection of such user at high scale, we use generative statistical model Latent Dirichlet Allocation (LDA) from Natural Language Processing (NLP) domain, which allows set of observations to be explained by unobserved groups. Our goal is to design a scalable pipeline which could, use twitter data in holistic way, using scalable machine learning technique. Our pipeline should also tune its learning by training on incoming data incrementally over a period of time. Incremental learning would ensure better results with time on unseen test data.

We wanted to avoid using crowdsourcing as validation mechanism. Mental illness in early stages is really subtle and has little or no physical symptoms. Hence, even experienced psychiatrists could find it difficult to diagnose it accurately. Hence we could not have left it to internet to decide how good the model was. Another issue with crowdsourcing is, it's not a scalable approach. We consciously wanted to pick only scalable components for this pipeline, because we wanted the pipeline to work at scale of Twitter [13].

### 1.6 Broader Impact

One such pipeline could not only filter negative users at a very early stage but also unearth hidden pattern associated with negative content on social media. Pipeline uses NLP's generative statistical model technique to filter and create unsupervised cluster of users and is also capable of classifying unseen users based on content of tweets. This pipeline could self learn with time to improve accuracy by updating its model on incoming data, and falls backs on an improved sentiment analysis algorithm for validation.

The rest of the thesis is organized as follows: CHAPTER 2 describes the characteristics of twitter, use of sentiment analysis and topic modelling, clustering techniques attempted and use of smart and efficient visualization technique t-Distributed Stochastic Neighbor Embedding (t-SNE). CHAPTER 3 discusses architecture of pipeline, data preprocessing, and approaches toward exploring useful features using Twitter's traditional data and topic modelling. CHAPTER 4, takes us through the results and their

analysis. CHAPTER 5 is about evaluating the pipeline on an unobserved data, to avoid data bias and measure the accuracy of pipeline in total. CHAPTER 6 is about conclusion of the analysis we have performed, discusses limitation of presented framework and proposed future work to overcome those limitations.

# CHAPTER 2

# BACKGROUND

## 2.1 Twitter and its characteristics

Tracking mental health behavior in an online age could be attempted by tracking social media posts. Online social data has in general has two famous outlets, blog posts and Twitter/Facebook. Blog posts are very difficult to sample content from a large number of unique users. One possible way to use blog posts is to connect with 4k (our user's database size) sampled unique user's blog individually and download their content. This is a non scalable approach since it involves detection of user of interest, finding their blog posts and then downloading content one by one. Twitter on the other hand makes it very easy with its open interface architecture. We can plugin the API to get access to samples from a public stream.

Last 4-5 years we have seen a huge surge in effort to design good distributed systems [11], [14] which can handle data at large scale. These systems are also designed to consume huge volume of data to train machine learning algorithm which are specifically designed to work in distributed setup. This has created opportunity for researchers and scientist to create bio-surveillance systems which are predictive in nature and consume huge social media generated data as input. Some noticeable works are predicting PTSD [10], Postpartum changes [22] using social media as input to name a few. While there has been lot of quality work using twitter as a medium to detect and predict specific mental health disorders, we will use this section to mention few of the most correlated with our problem definition.

As a part of literature survey, we investigated different research papers related to mental disorders and use of social media like Twitter and Facebook as users put an increasing amount of personal information on these platforms. Research at Johns Hopkins to use Twitter to track the flu [1] and tweets analysis to provide insight into mental illness [2] have established methods that can link the content of tweets to disease outbreak as well as specific mental disorders. Another notable work is around discovering co-occurrence Patterns of Asthma and Influenza [3]. There has been lot of work on finding accurate sentence sentiment of given short text using multiple techniques. However, work by Sara Rosentha on combined study of many popular algorithms to find most accurate sentiment [4] was most comprehensive. The impact of celebrities' influence on their followers was also studied in detail [5]. Looking at related area research papers, we have concluded to the best of our knowledge, that no direct work has been done to detect toxic content sharing among people. Further, most of the early attempts are very specific to one mental conditions. [8][9] [10]

**2.2 Sentiment Analysis**

Sentiment analysis has been handled as NLP task at many levels. Starting from document level classification task to (Turney, 2002; Pang and Lee, 2004), to at the sentence level (Hu and Liu, 2004; Kim and Hovy, 2004) and most recently at the phrase level Wilson et al., 2005; Agarwal et al., 2009).

However, for a more detailed and summarized study of role of social media in mental health research, we would like to refer readers to De Choudhury 2013. De

Choudhury has identified many ways in which NLP can be used to identify and predict mental health issues both at individual and population level.

For population-level analysis, surveys such as the Behavioral Risk Factor Surveillance System (BRFSS) are conducted via telephone (Centers for Disease Control and Prevention (CDC), 2010). Some of these surveys cover relatively few participants (often in the thousands), have significant cost, and have long delays between data collection and dissemination of the findings.

We were interested in usage of negative and positive words used by users. At the same time, we also wanted to know the distribution of positive and negative score for each individual. This was to get an overall sense of mental health by combining it with user's network feature like who follows user, user follows whom, what is the impact of celebrity count.

We zeroed down on two approaches to handle sentiment analysis. (1) To use TextBlob package. [26] (2) python piece of code we wrote, which works on bag of word model to assign sentiment to a word based on a published sentiment scores file [27]. We will get in to detail of our implementation later in this section. We wanted two different approaches one taking sentence context in to consideration and another just using bag of word model so we could cross verify results.

TextBlob's [26] python package's sentiment property returns a namedtuple of the form Sentiment (polarity, subjectivity). The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. It could take a sentence and return the sentence score.

e.g.

```
>>> testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")
>>> testimonial.sentiment
Sentiment(polarity=0.39166666666666666, subjectivity=0.4357142857142857)
>>> testimonial.sentiment.polarity
0.39166666666666666
```

Figure 2: Sentiment score computed by TextBlob package

TextBlob's sentiment package had really consistent and rich set of Application Programming Interfaces (APIs). Additionally, this decision was also influenced by TextBlob's API's capability to correct word spelling and spellcheck.

e.g.

```
>>> b = TextBlob("I havv goood speling!")
>>> print(b.correct())
I have good spelling!

>>> from textblob import Word
>>> w = Word('falibility')
>>> w.spellcheck()
[('fallibility', 1.0)]
```

Figure 3: Sentence correctness and spell check example

Accuracy of TextBlob to convert misspelled words into correct one was really impressive. However, it had a major challenge associated with its speed. We believe that our data quality was the reason for such a low speed. For instance, if we had to look into a formatted text doc and correct the spellings, the no of occurrences API will have to invoke corrector method will be very less. However, tweets are known to be misspelled, shortened more frequently than any other doc. Hence, it was just proving to be overhead

and was reducing the framework speed overall. Hence we moved on with our own dictionary based scorer, which we will be discussing in detail in section 3.3.

### 2.3 Latent Dirichlet Allocation (LDA)

In this section we will briefly discuss variation of topic modelling approach LDA [28] we used. In later section 3.5 we will introduce the adaptation of this model for twitter data.

LDA is an unsupervised machine learning technique whose primary purpose is to identify latent topics and word probability distribution for those topics from a large document collection. In LDA each document is represented as probability distribution of topics while each topic itself is probability distribution of words. LDA as a model allows us to play around with number of topic we expect the document to produce and also words in each topic, for a given text corpus. LDA is designed on bag-of-word concept, i.e. it does not try to take the context of word usage in account while deriving topics and word distribution for those topics. In context of this problem, each document here is created by preprocessing a given user's last accessible tweets. This kind of unsupervised topic modelling was especially suitable for us as we were looking at topics which gets derived out of given tweet's text content and find the distance of each user's personal topic distribution from the overall topic distribution of corpus.

### 2.4 Clustering Techniques

Two unsupervised clustering used in this experiments to cluster users are:

1. K-Mean [30]

2. Expectation Maximization (EM) [31]

The basic rationale behind selection these two algorithms was to find if the users have clear boundary when they are clustered together (K–Mean would work well) or they have fuzzier distribution. (EM should work well). We used “sklearn. cluster” [32] module of python to perform these clustering operation.

K means

1. Hard assign a data point to one particular cluster on convergence.
2. It makes use of the L2 norm when optimizing (Min {Theta} L2 norm point and its centroid coordinates).

In contrast, EM

1. Soft assigns a point to clusters (so it gives a probability of any point belonging to any centroid).
2. It doesn't depend on the L2 norm, but is based on the Expectation, i.e., the probability of the point belonging to a particular cluster. This makes K-means biased towards spherical clusters.

### 2.5 t-SNE (t-Distributed Stochastic Neighbor Embedding)

t-SNE is a machine learning algorithm for dimensionality reduction developed by Laurens van der Maaten and Geoffrey Hinton. It is a nonlinear dimensionality reduction technique that is particularly well suited for embedding high-dimensional data into a space of two or three dimensions, which can then be visualized in a scatter plot. Specifically, it models each high-dimensional object into a two- or three-dimensional

point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points [33].

It was well suited for visualization of our user's cluster as they were expected to have high dimensional data as each topic is one of the dimension of user's feature vector. We will get into finer details of our user feature vector in section 3.3.

# CHAPTER 3

# PIPELINE ARCHITECTURE

Here is an Architecture diagram of our LDA pipeline at a very high level.

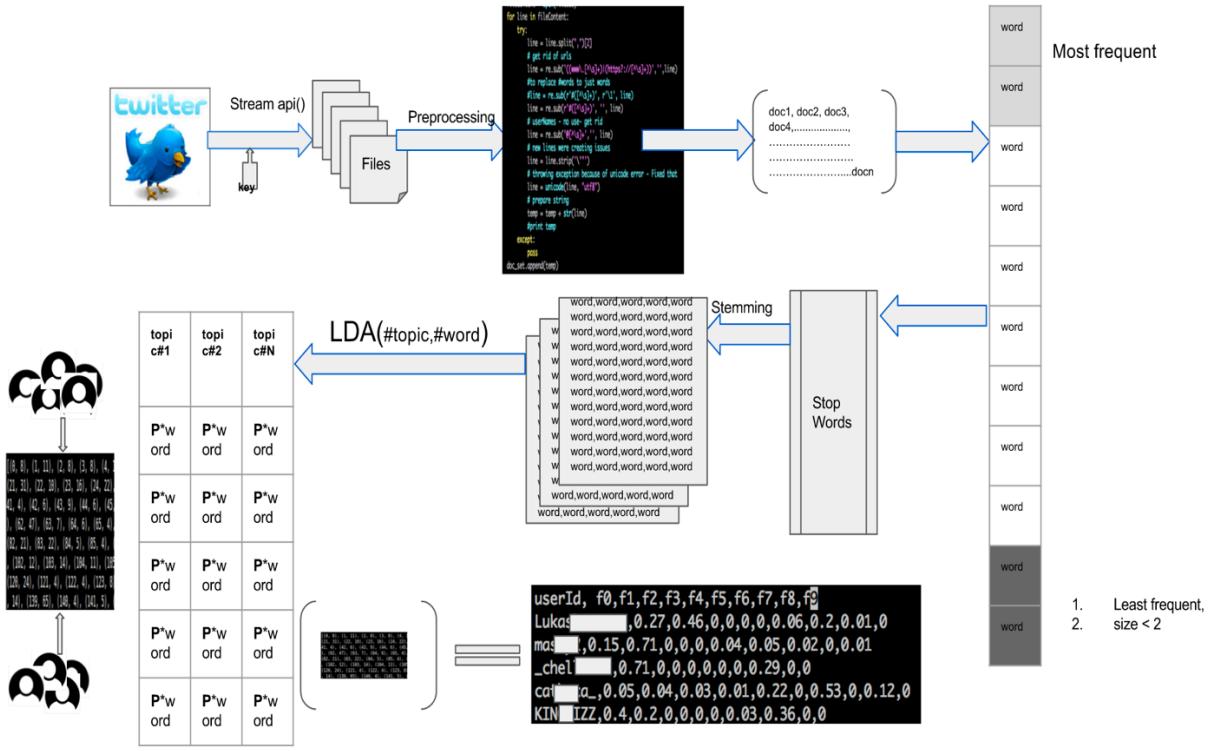
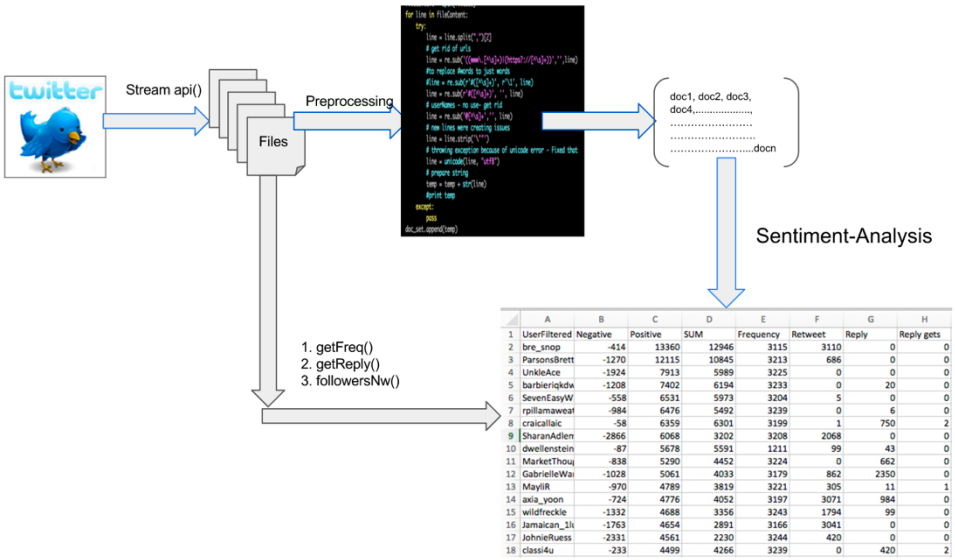



Figure 4: High level system architecture for generating user feature vector

This pipeline illustrates how do we calculate user vector based on correlation with topic generated by LDA model. Find below another diagram of how the pipeline was modified to study Twitter's traditional data properties.




| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | UserFiltered | Negative | Positive | SUM | Frequency | Retweet | Reply | Reply gets |
| 2 | bre_snop | -414 | 13360 | 12946 | 3115 | 3110 | 0 | 0 |
| 3 | ParsonsBrett | -1270 | 12115 | 10845 | 3213 | 686 | 0 | 0 |
| 4 | UnkleAce | -1924 | 7913 | 5989 | 3225 | 0 | 0 | 0 |
| 5 | barbieriqkdw | -1208 | 7402 | 6194 | 3233 | 0 | 20 | 0 |
| 6 | SevenEasyW | -558 | 6531 | 5973 | 3204 | 5 | 0 | 0 |
| 7 | rpillamaweat | -984 | 6476 | 5492 | 3239 | 0 | 6 | 0 |
| 8 | craicallaic | -58 | 6359 | 6301 | 3199 | 1 | 750 | 2 |
| 9 | SharanAdlen | -2866 | 6068 | 3202 | 3208 | 2068 | 0 | 0 |
| 10 | dwellenstein | -87 | 5678 | 5591 | 1211 | 99 | 43 | 0 |
| 11 | MarketThou | -838 | 5290 | 4452 | 3224 | 0 | 662 | 0 |
| 12 | GabrielleWa | -1028 | 5061 | 4033 | 3179 | 862 | 2350 | 0 |
| 13 | MayliR | -970 | 4789 | 3819 | 3221 | 305 | 11 | 1 |
| 14 | axia_yoon | -724 | 4776 | 4052 | 3197 | 3071 | 984 | 0 |
| 15 | wildfreckle | -1332 | 4688 | 3356 | 3243 | 1794 | 99 | 0 |
| 16 | Jamaican_1l | -1763 | 4654 | 2891 | 3166 | 3041 | 0 | 0 |
| 17 | JohnieRuess | -2331 | 4561 | 2230 | 3244 | 420 | 0 | 0 |
| 18 | classi4u | -233 | 4499 | 4266 | 3239 | 0 | 420 | 2 |

Figure5: High level system architecture for generating a feature vector in terms of user's twitter data properties

### 3.1 Data Collection

All the data we obtain is public, posted between 2014 and 2015 and was made available from Twitter via their APIs. At any point in time we did **not** made any attempt to obtain data which was either marked private or shared via direct message.

We used cloud virtual machine, Droplets by Digital Ocean [34] to download most of the twitter data. Some time when we needed more machines in parallel to work, we used Amazon's EC2 [35] as well.

Twitter provides two ways to access data. Because we wanted the data not to be biased, we went with stream option where we get access to 1% of public stream data without any filter.

Data downloading was mostly divided in 4 phases. Step 1 was achieved using stream API access where as 2,3 and 4 was achieved using REST APIs.

1. Connect to Twitter stream to download tweets from 1% sample access provided by twitter for free. We did this between Nov-Dec 2015. We then filtered user based on their language preference set as English (en).
2. At this stage we had more than 400,000 users. We just selected 12,000 of them randomly and downloaded their last available tweets. Twitter allows developers to have access to last 3200 tweets for a given userid (userid is identity of user on twitter and it has nothing of do with user's name or any demographic info), using their APIs. We had then data for almost 11,000 users as for some of the filtered users had private data policy set.
3. Out of 11,000 users, we ran a script to remove users who had less than 100 tweets. The number 100 was selected based on purely experimental basis. We were left with close to 10,000 users after applying this filter.
4. For these 10,000 users, we also wanted to look at their friend network. Hence we used 6 EC2 machines to download their friends. This was very time consuming as users tend to have many friends. We later restricted the number to 5000 maximum friends for a given userid.

### 3.2 Pre-processing

Extracted tweets were in Java Script Object Notation (JSON) format. The JSON provided by Twitter is in key value format, as shown below in Figure 6.

```
{
    "contributors": null,
    "coordinates": null,
    "created_at": "Sun Jan 24 23:20:16 +0000 2016",
    "entities": {
        "hashtags": [ ],
        "symbols": [ ],
        "urls": [ ],
        "user_mentions": [ ]
    },
    "favorite_count": 0,
    "favorited": false,
    "filter_level": "low",
    "geo": null,
    "id": 691400150640566300,
    "id_str": "691400150640566272",
    "in_reply_to_screen_name": null,
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
    "in_reply_to_user_id": null,
    "in_reply_to_user_id_str": null,
    "is_quote_status": false,
    "lang": "en",
    "place": null,
    "retweet_count": 0,
    "retweeted": false,
    "source": "<a href=\"http://twitter.com/download/iphone\" rel=\"nofollow\">Twitter for iPhone</a>",
    "text": "I be with the killers talkin dealer smoke",
    "timestamp_ms": "1453677616163",
    "truncated": false,
    "user": {
        "contributors_enabled": false,
        "created_at": "Wed May 18 02:04:00 +0000 2011",
        "default_profile": true,
        "default_profile_image": false,
        "description": "Paradise",
        "favourites_count": 5749,
        "follow_request_sent": null,
        "followers_count": 985,
        "following": null,
        "friends_count": 567,
        "geo_enabled": true,
        "id": 300612751,
        "id_str": "300612751",
        "is_translator": false,
        "lang": "en",
        "listed_count": 4,
        "location": "Hennessy Beach",
        "name": "Jar",
        "notifications": null,
        "profile_background_color": "C0DEED",
        "profile_background_image_url": "http://abs.twimg.com/images/themes/theme1/bg.png",
        "profile_background_image_url_https": "https://abs.twimg.com/images/themes/theme1/bg.png",
        "profile_background_tile": false,
        "profile_banner_url": "https://pbs.twimg.com/profile_banners/300612751/1453269827",
        "profile_image_url": "http://pbs.twimg.com/profile_images/678496042334208000/XEq22I1l_normal.jpg",
        "profile_image_url_https": "https://pbs.twimg.com/profile_images/678496042334208000/XEq22I1l_normal.jpg",
        "profile_link_color": "0084B4",
        "profile_sidebar_border_color": "C0DEED",
        "profile_sidebar_fill_color": "DDEEF6",
        "profile_text_color": "333333",
        "profile_use_background_image": true,
        "protected": false,
        "screen_name": "blaccjared",
        "statuses_count": 71070,
        "time_zone": "Eastern Time (US & Canada)",
        "url": "http://blaccorder.tumblr.com",
        "utc_offset": -18000,
        "verified": false
    }
}
```


Figure 6 Typical JSON format  tweet metadata downloaded from twitter

From Figure-6 its obvious that a typical tweet has lot of metadata associated with it. However, we were mostly interested in few very basic ones. At first just to get the user who had set English (en) as their language.

```
1.  def extractTweet(inputdata):
2.      #count = 0
3.      for line in inputdata:
```

```
        resultDict = json.loads(line)
        try:
            lan = resultDict["user”]["lang"]
            if lan == "en":
                var = resultDict["user”]["screen_name"]
                var = make_unicode(var)
                print (var)
        except KeyError:
            pass
        except ValueError:
            pass
        except:
            pass

def make_unicode(input):
    if type(input) != unicode:
        input =  input.decode('utf-8')
        return input
    else:
        return input
```

Once we had list of these users, we started downloading their last available tweets. For that we used tweepy [36]. We just downloaded in format {id, created_at, text} format e.g.

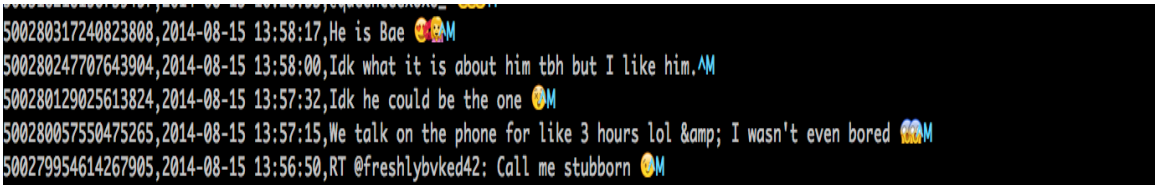



Figure 7: Sample of downloaded tweet

once we had a file for each user with their tweet text content, we were all set to perform sentiment analysis. These were the basic data preprocessing steps we ran on these tweet texts.

1. Remove the part of tweet's text which had universal resource locators (urls) i.e. https or https or www
2. Emoticons were removed as well.
3. Replaced hashtags (#) with just the text

4. Remove userids (@string), RT, RT”
5. Remove Unicode characters
6. Remove the stopwords (NLTK module and few more added based on experiment results)
7. Convert to lowercase for stemming purpose (Discussed in detail in section 3.6)

### 3.3 Exploring Twitter’s traditional data properties for features

After preprocessing step, we had a relatively cleaner twitter text corpus. we used TextBlob’s API to get sentiment score. This API call produces one sentiment score for a given text. We Modified the source code, for it to work on each string(word) as considering every word as text was huge overhead for this computation. One of the major challenge with sentiment API was, it uses inbuilt dictionary for assigning scores. As a result, it could not recognize and assign any weightage to slangs, shortcuts, hashtags (which are not traditional dictionary words) and misspelled words. Such words are very common in context of Twitter data.

We tried to get around this problem by using another TextBlob’s API to correct the misspelled words so we could use sentiment scores provided by TextBlob (Figure-3). However, experiment on sample data showed that, TextBlob.correct module which was responsible for correcting misspelled word was extremely slow. It took on an average two and half minute to process one document (one document is one user’s last accessible tweet text corpus, which typically has her/his last 3200 tweets). Keeping these number in mind we would have needed 2.30 minute each for 4000 users, which is approximately

10000 minutes, which is approximately 150 hours. Even with 3 machine working in parallel it was taking 2 days for one run. As a pert of experiment, we wanted to run this exercise, for all the possible topic counts like, 10, 20, 30, 40, and 50. Hence giving speed a higher priority, we went ahead with our own sentiment analysis code.

We used a simple algorithm which iterates over each tweets of a given user. After doing the preprocessing step, it tokenizes the text and then looks into to dictionary item for the score associated with the found word. Next, based on sign of number (positive integer for positive word, negative integers for negative words) add the numbers to positive score or negative score. This is a bag of word based model to compute the positive and negative score for each user. For creating a rich set of word and sentiment score we merged sentiment scores from two different sources to create a single word to score file which had words and their sentiment score in key value format. The files we used were:

1. AFINN-111.txt [27]
2. Negative-word.txt [36]

When a word was not found in dictionary, we gave it a zero (neutral) score. To get confidence on our sentiment module, we cross validated it with TextBlob's. We were getting 15 out of 20 users same in the top negative list. We plotted these top 20% users to get a sense of trend.

### 3.3.1 Users who fell in top 100 category of sharing negative content

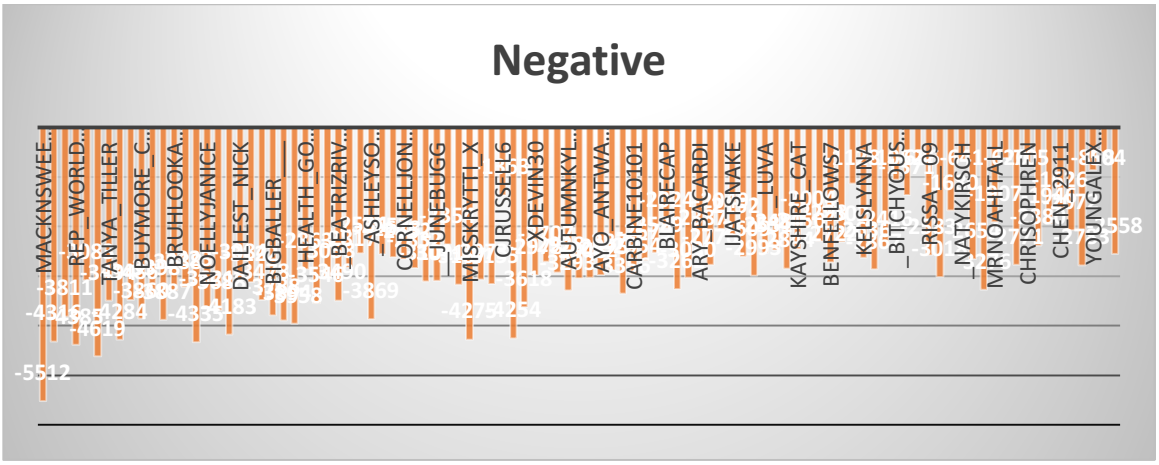



Figure 8: Users sorted on negative score

We also plotted users to whose negative contribution to social media was more than positive ones, i.e. the graph of user sorted on total score (positive + negative).

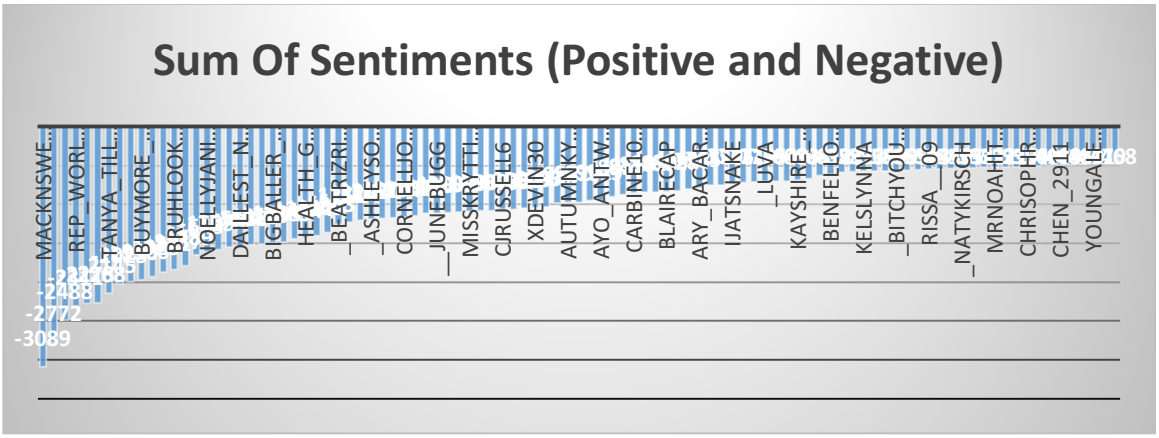



Figure 9: User sorted on total score( negative + positive)

### 3.3.2 Positive to Negative sentiment correlation

We also looked at the distribution of positive to negative score across users to figure out how the distribution.

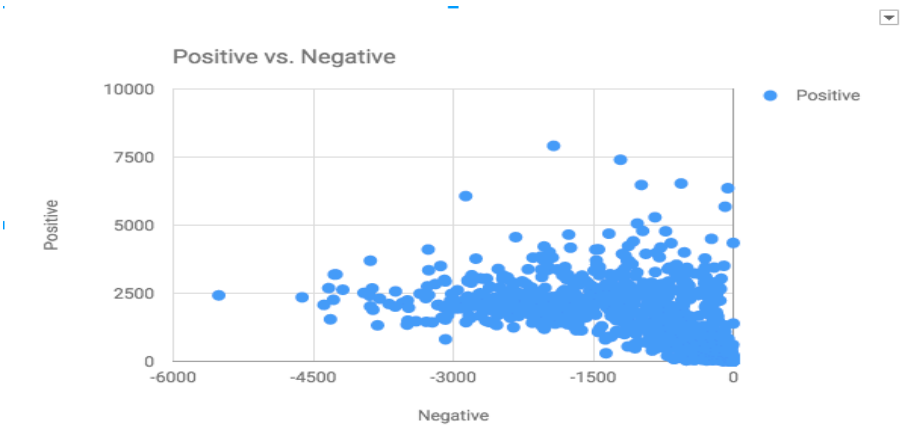



Figure 10: Positive and negative  distribution of scores across 1k sampled users

Distribution was showing a clear trend that some users were polarized to share negative content [between -4500 to -3000] predominantly, but also had positive sentiment score of 1000-2500. However, users with high positive sentiment (greater than 5000) were sharing very less negative content.

### 3.3.3 Sentiment correlation with “reply user does” and “gets”

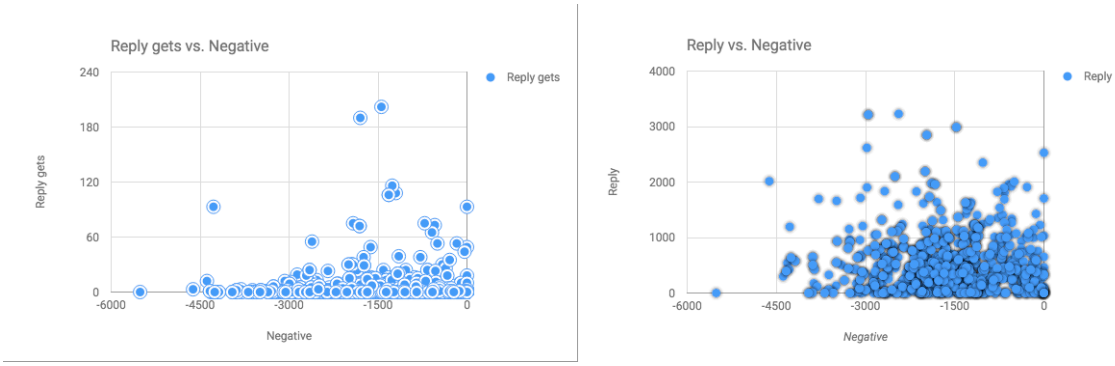



Figure 11: ‘Negative to reply’ and ‘Negative to reply gets’

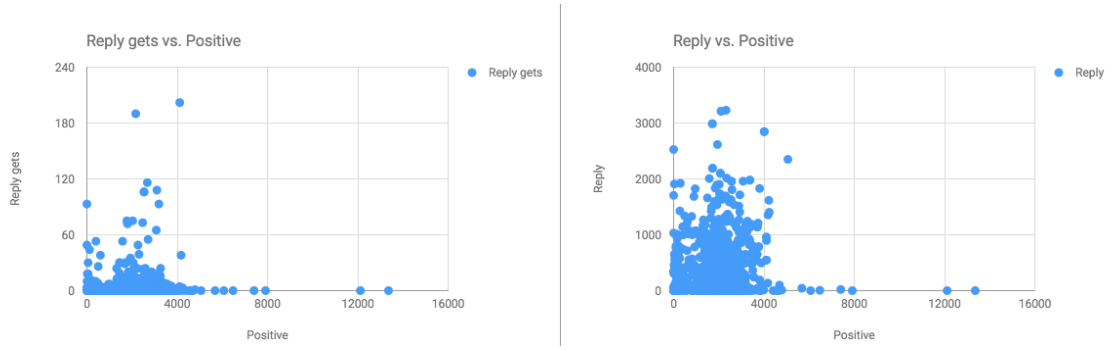



Figure 12: 'Positive sentiment to reply' and 'Positive sentiment to reply gets'

Above graphs suggest that top 1% of negative user tend to reply lot more than top 1% positive users. This is a good signal to utilize as it seems user who have high negative score, get involve in communication with network users.

### 3.3.4 Sentiment correlation with "Retweet Count"

Next we plotted Below average negative to retweet count and above average positive to retweet count as well to see if there was something obviously standing out as trend.

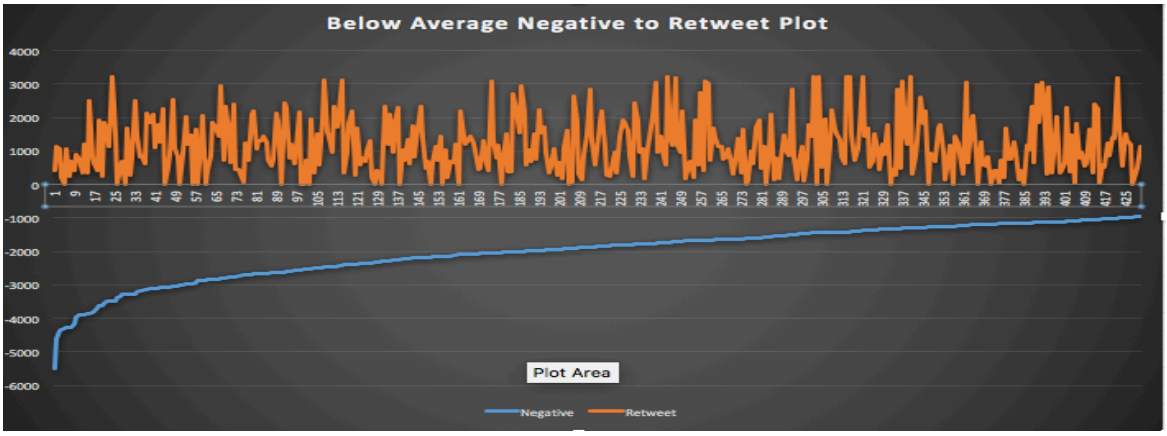



Figure 13 "Below average negative sentiment" to "Retweet count"

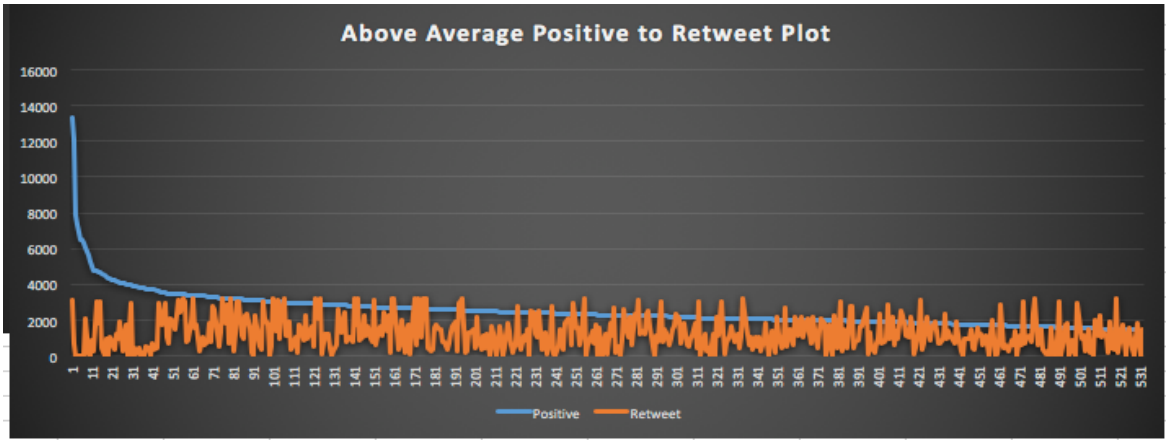



Figure 14 "Above average positive sentiment" to "Retweet count"

The plotted graph suggested that there wasn't any pattern in the data we had for experiment. So we stopped pursuing this for feature selection.

### 3.3.5 Sentiment correlation with "correctness (spelling) of tweets"

However, there were few interesting patterns about sentiment to sentence correctness factor. We found it quite fascinating that there was obvious link between sentence correctness to negative sentiment.

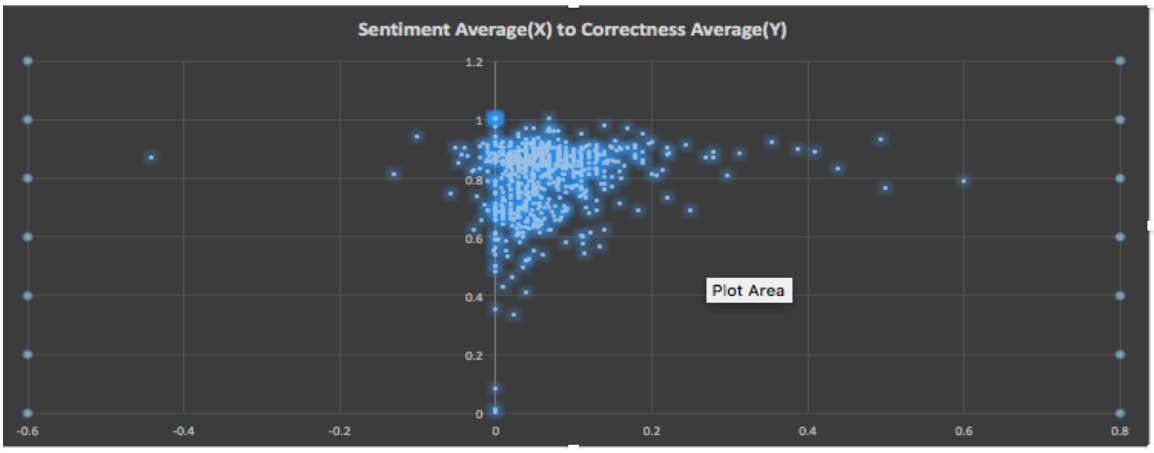



Figure 15Sentiment to correctness correlation

The graph seems to be suggesting that while low sentiments did not had a clear correlation with correctness, the most positive ones also had a very high correlation score. However, it could not use it as a strong feature.

### 3.3.6 Sentiment correlation with "Choice of words"

We also investigated the idea if "choice of word" was different for people who were mostly tweeting negative in comparison to random people. For this after doing preprocessing steps on tweets, we ran a simple wordcount program to look at the frequency of words.


Figure 16: Wordcloud of "choice-of-words" of 20 most negative users


Figure 17: Wordcloud of "choice-of-words" of 20 most positive users

### 3.3.7 User's network structure

We also looked at the user's network structure to catch some strong differentiating network substructure between negative and positive subcategory. We dominantly used Gephi for that purpose. However, we quickly realized that, most of the users whose negative sentiment was dominant, were mostly either "comedy host shows" or "sports club" ids. We downloaded all the friends for userids and created a adjacency matrix representation for it.

| | u1 | u2 | u3 | . | . | . | u11000 |
|---|---|---|---|---|---|---|---|
| u1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| u2 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| u3 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| . | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| . | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| . | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| u11000 | 1 | 1 | 0 | 1 | 0 | 1 | 0 |

Figure 18: Adjacency matrix representation of user and their friend network

To understand this matrix, a value of 1 at $u_{3,2}$ represents that $u_3$ has $u_2$ in her/his friend's list. The idea behind this graph was to find if there were "in-network" celebrities and if their content is affecting the network negativity or positivity over all.

When we plotted this matrix using tool Gephi [37], we got a graph like plotted below. Here, every user is represented by a node and size of the node is proportional to indegree of that node. Also, every edge represents a direct relationship between two nodes.

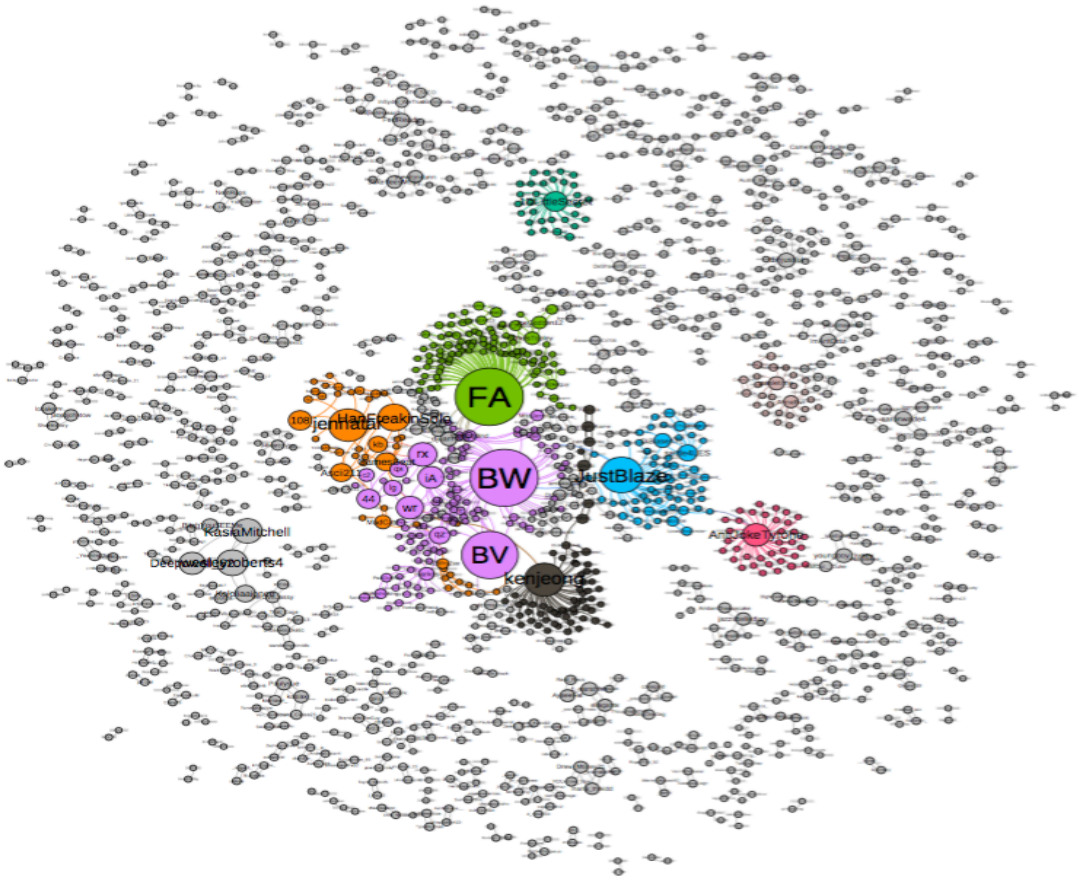



Figure 19 : Network graph of users and their friends inside network

| Rank | Label |
|---|---|
| 76 | FA |
| 53 | kenjeong |
| 44 | BW |
| 34 | AntiJokeTyrone |
| 34 | JustBlaze |
| 26 | 1DLittleSecret |
| 20 | qz |
| 15 | tbhjustlol |
| 13 | AnnaRvr |
| 11 | iA |
| 11 | pasletime |
| 11 | markos |
| 8 | FaZePryZee |
| 7 | jennatar |
| 7 | ImAntOrtiz |
| 7 | BV |
| 7 | 94MarcoVss |
| 7 | 11W |
| 7 | settle4LES |

| Rank | Label |
|---|---|
| 0.021120711763778725 | BW |
| 0.018887163716460858 | BV |
| 0.01099413921247216 | FA |
| 0.006786429255630636 | kenjeong |
| 0.005814836216341584 | JustBlaze |
| 0.004989170790934327 | jennatar |
| 0.004813619325944358 | wr |
| 0.004519226735955418 5 | AntiJokeTyrone |
| 0.004302694376331392 | ImAntOrtiz |
| 0.003810105965614487 5 | joselinewest |
| 0.00365166368160103 | tnewtondunn |
| 0.003645708475541441 | JakeReesMogg |
| 0.003498589932589939 6 | 1DLittleSecret |
| 0.003246397153490894 | rx |
| 0.003119858787410562 6 | HanFreakinSolo |
| 0.003097233566091294 | iA |
| 0.002944342387405354 7 | Leathercrocs |
| 0.002872271127000200 3 | itsSamCollins |
| 0.002680780267363654 4 | 3FE |

Figure 20: Users in network sorted based on their indegree and pageRank

We did basic manual analysis on users and figured that the users were mostly either comedy shows official page or “one line offensive jokes”. Because most of such ids tweet at a very high frequency, it was very natural for these userids to have high negative sentiment.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | UserFiltered | Negative | Positive | SUM | Frequency | Retweet | Reply | Reply gets |
| 2 | zvqrryobrien | 0 | 0 | 0 | 3 | $0.00 | 0 | 0 |
| 3 | yvk60 | 0 | 0 | 0 | 67 | 0 | 24 | 0 |
| 4 | welikeitloud | 0 | 0 | 0 | 4 | 0 | 0 | 0 |
| 5 | wana355849 | -3 | 0 | -3 | 137 | 78 | 58 | 0 |
| 6 | ukon993301 | 0 | $0.00 | 0 | 542 | 31 | 217 | 0 |
| 7 | tatunori513 | 0 | 0 | 0 | 153 | 46 | 29 | 0 |
| 8 | stwa29209 | 0 | 0 | 0 | 3201 | 0 | 0 | 0 |
| 9 | spe77pp | 0 | 0 | 0 | 3250 | 0 | 0 | 0 |
| 10 | sena_kishito | 0 | 0 | 0 | 219 | 13 | 133 | 0 |
| 11 | sarsha121 | 0 | 0 | 0 | 941 | 99 | 228 | 0 |
| 12 | SamiBhai22 | 0 | 0 | 0 | 7 | 0 | 4 | 0 |
| 13 | ribkashi | 0 | 0 | 0 | 3116 | 764 | 1705 | 49 |
| 14 | pem_cocoa | 0 | 0 | 0 | 1416 | 73 | 336 | 3 |
| 15 | olt160 | 0 | 0 | 0 | 3230 | 0 | 0 | 0 |
| 16 | nt9909 | 0 | 0 | 0 | 432 | 3 | 0 | 0 |
| 17 | nmbleke | 0 | 0 | 0 | 415 | 103 | 0 | 0 |

Figure 21: Feature vector of each user based on traditional data analysis

### 3.4 Exploring Topic Modeling for Features

Hence we decided to try a new pipeline altogether using "Topic modeling" [38] as a technique. Our overall architecture for this, was as described in the following figure.

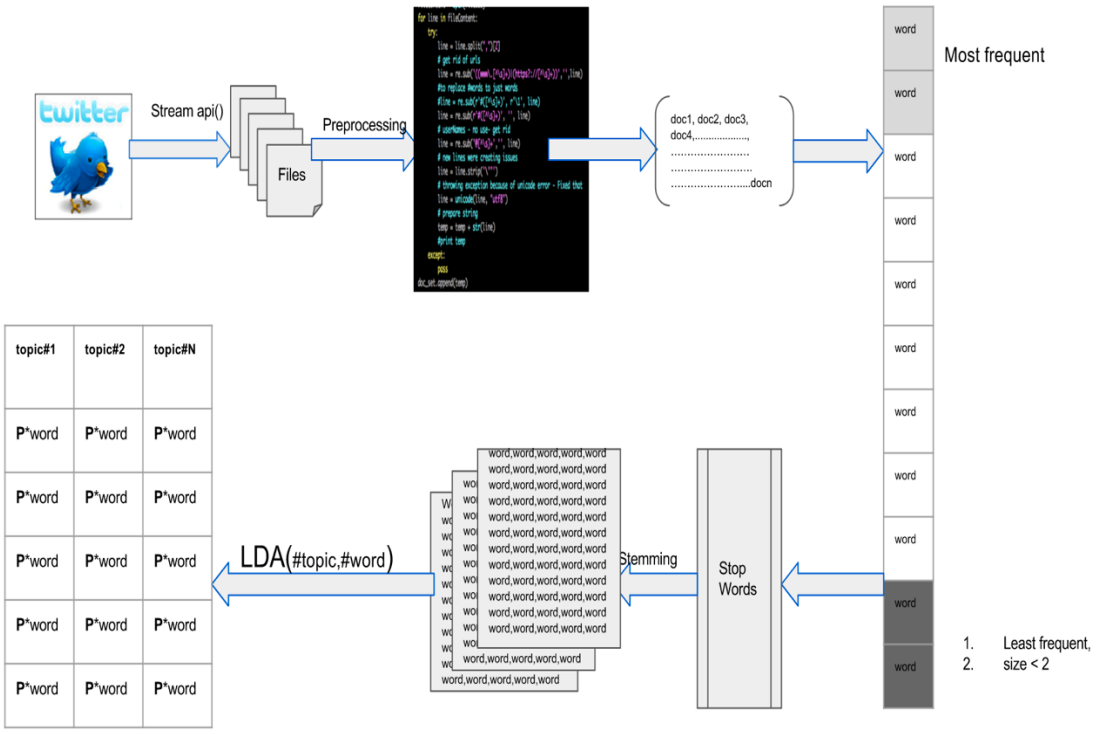



Figure 22: Topic modelling pipeline design to generate topics from tweet's text corpus

Steps involved in doing topic modelling over tweet corpus:

1. Data was downloaded as mentioned in section 3.1.
2. On downloaded data, preprocessing steps were applied as discussed in detail in section 3.2. The outcome of this result is to have a clean text corpus for each user, also knows as 'document' in this context.

3. Once we had document for each user, tokenize the document set to words.
4. Removed most frequent words as well as least frequent words. This is important to get a good LDA model as the model inherently works on bag of word [40] concept. Hence in summary we can say that, the quality of topic generated by model is proportional to the quality of data.
5. Remove words which had size < 2 (This step was experiment driven)
6. All stop words had to be removed. We used stopwords from Natural Language Toolkit's (nltk) stopword's module.
7. We also added few words exclusively as they were meaningless words and must have been generated as a side product of some online rhetoric going on. Here is the list of words we extracted based on multiple experiment output.

```
from nltk.corpus import stopwords

# get all the words from nltl package
en_stop = get_stop_words('en')

# create English stop words list
en_stop = get_stop_words('en')

#extending the stop word list as the twitter data set is noisy
en_stop.extend(["demibestfans2016", "updat", "channel", "que", "eurekamag", "ang", "keo", "lang", "est", "fpjb", "fnfjb", "rbjb", "amp"
, "ini", "ada", "amant", "pushawardskathniel", "kathniel", "aku", "niond", "ich", "ero", "rtandfollow", "ich", "und", "ist", "ero", "co
m", "meu", "pra", "weather", "properti", "googl", "0mm", "thttps", "https", "http", "for", "i'll", "also", "via", "follow", "mali", "go
t", "nowplay", "periscop", "stat", "replay", "katch", "biztip", "via", "radio", "commerci", "ang", "aka", "kixmi", "capricorn", "tarnat
", "today", "sagittariu", "tauru", "votingdevonne23", "aku", "yang", "tak", "ada", "nak", "dutchschultz", "strictlybid", "lovat", "ihea
rtaward", "bestfanarmi", "que", "ain", "wit", "votingdevonne23", "giveyourheartdd", "amant", "ootd", "bandana", "updat", "get", "channe
l", "pushawardskathniel", "kathniel", "cbb", "uri", "santiago", "ain'", 'ain', "por", "para", "una", "der", "ein", "aja", "kamu", "sama
", "untuk", "lagi", "ako", "kau", "dah", "dia", "kalu", "lah", "bilai", "apa", "lagi", "pushawardskathniels", "hahaha", "haha", "hahaha
ha", "bestfanarmy", "can", "don"])
~
```

Figure 23: Stopwords used for filtering words

8. Next we perform stemming using PorterStemmer module of NLTK package. Stemming helps extensively in increasing the word quality. As it reduces the multiple variation of words to single one. This helps in keeping the words in dictionary to minimum. E.g. likes, like, liking get converted to same word "like"

after applying stemming. Which effectively increases the quality of document term matrix.

9. Once cleaning stage is completed, to generate an LDA model, we need to understand how frequently each term occurs within each document. Hence, we need to construct a **document –term matrix.** We achieved that using package gensim [12]. The output of this dictionary creation stage would be a dictionary made of unique tokens. The Dictionary () function assigns a unique integer id to each unique token and in parallel get the wordcount as well.
10. Next, the dictionary had to be converted in to bag-of-words. The resultant corpus, is a list of vectors which will be equal to the no of document (which is the user's count). gensim's doc2bow() function does that.
11. LDA model takes this corpus as input and generates LDA model. It also takes these user provided argument as input:
    a. num_topics: based on this input LDA model generates topic
    b. id2word: needs the dictionary, we created in step-9
    c. passes: this is an optional parameter. Typically, this value has to be chosen based on experiments. High value is good, but a very high value can make the algorithm go slow.

```
# convert tokenized documents into a document-term matrix
corpus = [dictionary.doc2bow(text) for text in texts]

# generate LDA model
ldamodel = gensim.models.ldamodel.LdaModel(corpus, num_topics=30, id2word = dictionary, passes=20)

print(ldamodel.print_topics(num_topics=30, num_words = 15))
```

Figure 24: Code Example of How LDA model is computed

12. Output of above step is list of topics generated by model along with probability value aligned with every word generated in each topic in "P*word" format. Which is also the last step of Figure-21. We run this for multiple options of num_topics. We experimented with 10, 20, 30, 40, 50 topics.
13. Once a model is generated, it can be saved for it to be used later with a simple command.

    **ldamodel.save**('./data/lda_twitter_6310user_30topics.model')
14. The result of above step 11 is (assuming num_topic = 10, num_word = 10)

**10 Topic resulted from running topic modelling algorithm**

[(0, u'0.016*like + 0.016*just + 0.011*love + 0.009*want + 0.009*peopl + 0.008*one + 0.008*know + 0.008*day + 0.007*time + 0.007*make'),

(1, u'0.011*just + 0.009*thank + 0.008*game + 0.007*will + 0.007*one + 0.007*day + 0.007*now + 0.007*good + 0.006*time + 0.006*win'),

(2, u'0.058*photo + 0.030*post + 0.017*facebook + 0.013*harri + 0.010*video + 0.008*love + 0.008*loui + 0.008*album + 0.007*day + 0.006*pour'),

(3, u'0.027*follow + 0.019*dan + 0.018*stat + 0.016*one + 0.015*music + 0.014*check + 0.011*automat + 0.010*peopl + 0.010*download + 0.009*play'),

(4, u'0.345*updat + 0.052*con + 0.032*como + 0.027*del + 0.014*hay + 0.011*son + 0.010*sin + 0.009*soy + 0.008*tan + 0.008*solo'),

(5, u'0.084*video + 0.077*like + 0.023*day + 0.017*feel + 0.017*rain + 0.016*high + 0.016*press + 0.016*wind + 0.016*low + 0.015*temp'),

(6, u'0.006*news + 0.004*will + 0.004*say + 0.004*car + 0.003*call + 0.003*need + 0.003*polic + 0.003*time + 0.003*year + 0.003*kill'),

(7, u'0.018*like + 0.016*fuck + 0.016*shit + 0.011*just + 0.009*bitch + 0.009*lmao + 0.008*free + 0.008*know + 0.008*ass + 0.007*girl'),

(8, u'0.006*obama + 0.004*use + 0.004*read + 0.004*great + 0.003*food + 0.003*chang + 0.003*will + 0.003*influenc + 0.003*human + 0.003*stori'),

(9, u'0.022*love + 0.022*tweet + 0.020*vote + 0.012*second + 0.012*summer + 0.011*collect + 0.011*happi + 0.011*thank + 0.010*may + 0.010*one')]

Figure 25: 10 - Topics generated by LDA

15. Once the model generates topic and saved as per step 13, now we can compute the individual document feature vector on the basis of its correlation with topic. For that we compute bag of word for each document and pass it trained LDA model. The output of this step is as follows.

```
for item  in corpus:
    print ldamodel[item]
```

Figure 26: Code Snippet to print user's individual feature vector

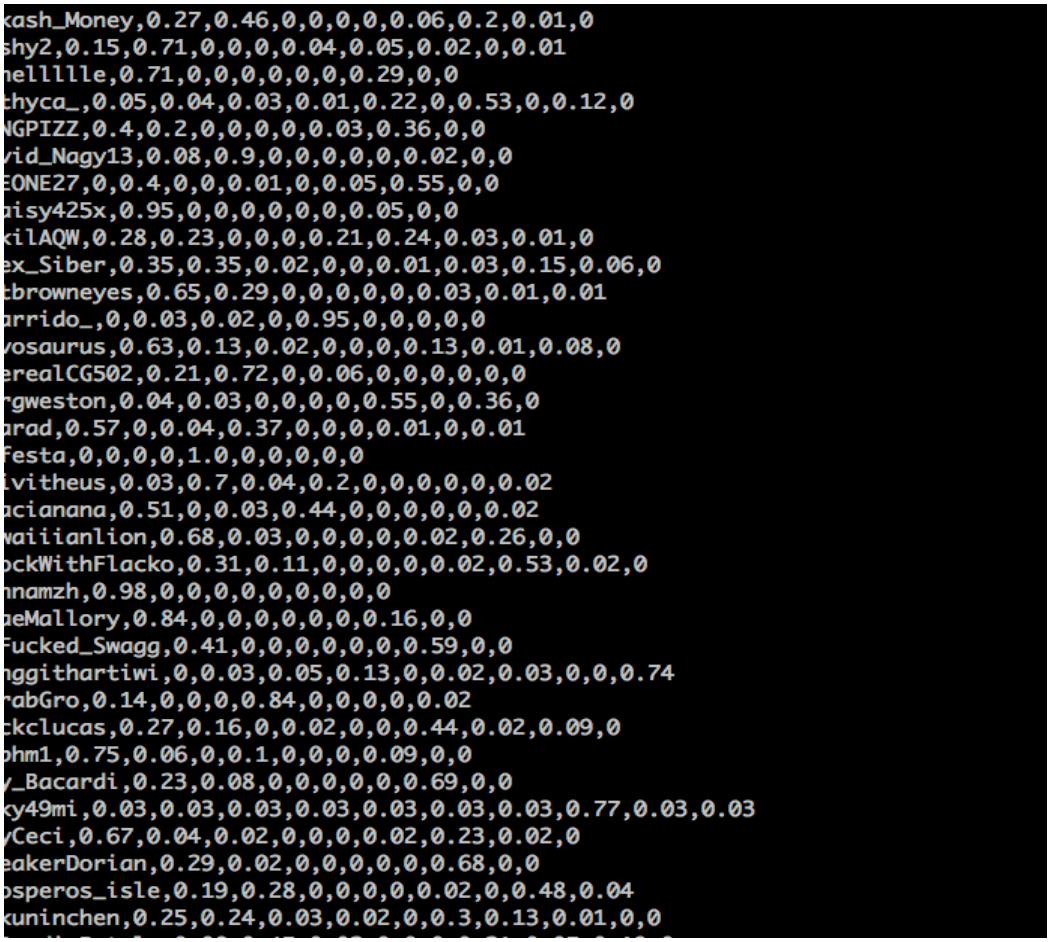



Figure 27: User feature vector in correlation with topics sample

CHAPTER 4

RESULTS AND ANALYSIS

## 4.1 Clustering on Twitter's Traditional Data

On this feature vector data, we tried to run EM, an unsupervised clustering algorithm. The expectation was that it should be able to cluster the most negative users in one group. EM run on this dataset gave us 7 possible clusters.

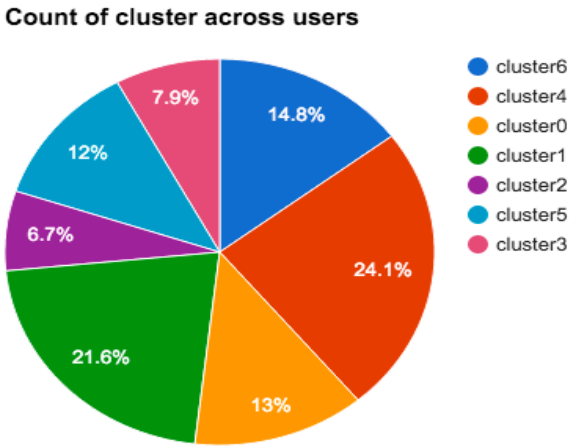



Figure 28: Distribution of clusters across dataset

However, most of the top 20 most negative users based on negative sentiment score were clustered in single group.

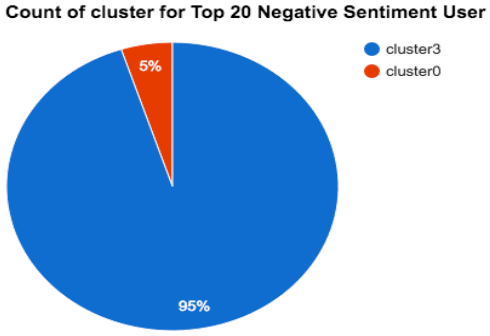



*Figure 29: Cluster distribution for top 20 users*

This was good result, as EM clustering algorithm was able to find correlation between all negative users. So we ran another experiment by tagging all users who had above average negative sentiment score as negative user and tried to see how the cluster distribution looks like.

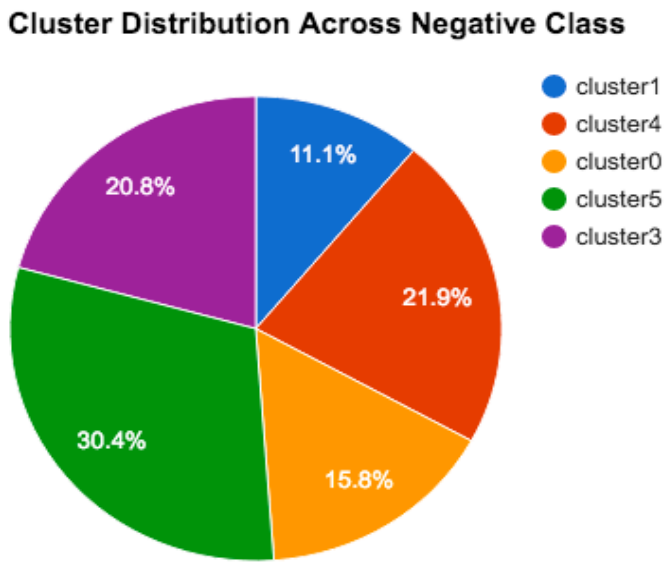



Figure 30: Cluster distribution of all users marked as negative

This result was not very encouraging as the negative users were all over the cluster. So, while the clustering did a really good job on top negative sentiment user, over all the cluster distribution was not okay keeping twitter traditional data properties as feature. We were also concerned about Sentiment analysis getting too much weightage in the analysis. In previous attempts [8][9] [10] sentiment analysis was already explored as a possibility.

For the validation of results, we had two obvious options in mind. Either manual tagging or crowdsourcing. Both of them highly impractical and non scalable.

### 4.2 Getting Sense of Data

To get a better sense of feature data obtained from section 3.5, we plotted the feature distribution across data set.

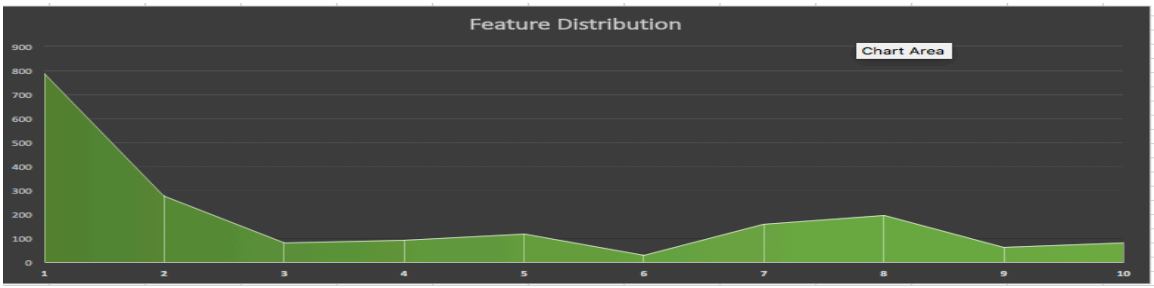



Figure 31:Feature distribution – 10 topics

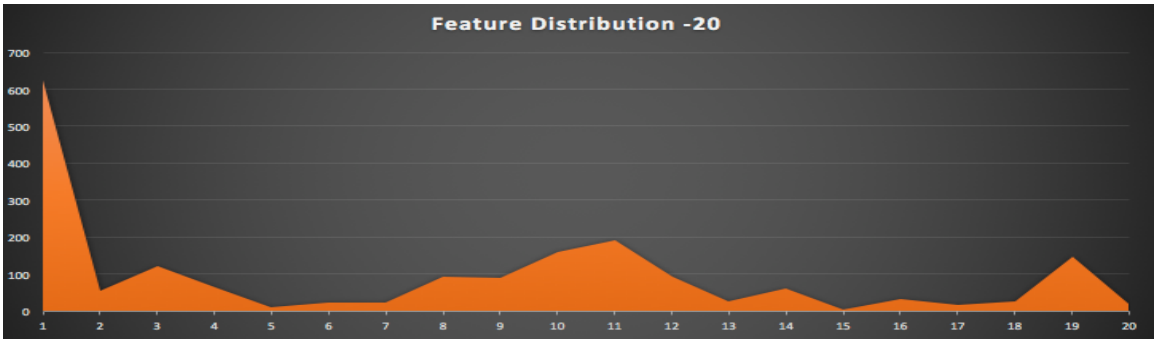



Figure 32:Feature distribution – 20 topics

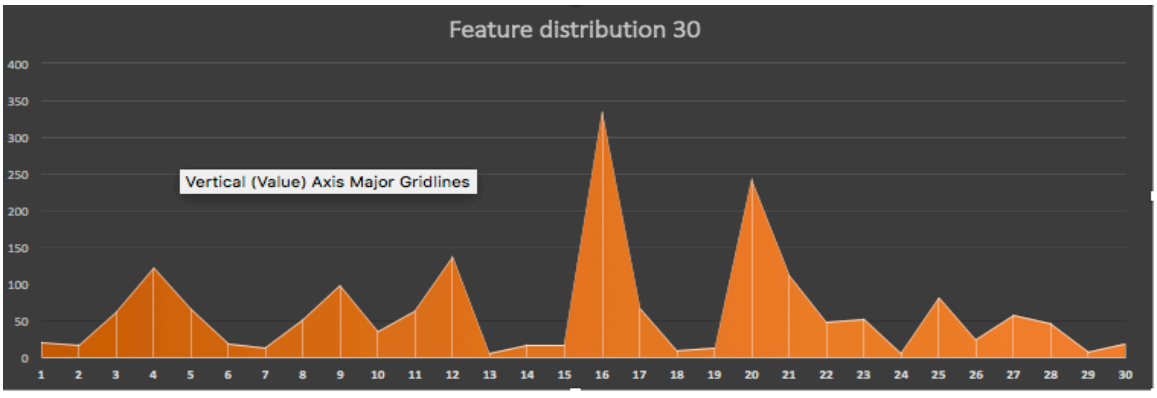



Figure 33: Feature distribution – 30 topics

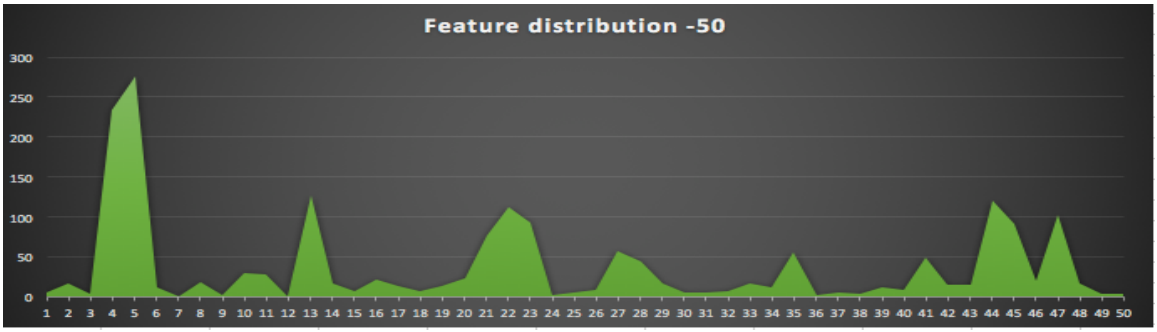



Figure 34:Feature distribution 50-topics

This study was done to get a sense of distribution of each topic.

### 4.3 Feature Distribution and Assigning class

As per Figure-26, we had user's feature vector available. This could be also seen as user's feature having 10 dimensional data where each dimension of feature represents one topic in the same order. Hence we plotted the feature vectors of top 20 negative and top 20 positive users to get a sense of feature vector distribution.

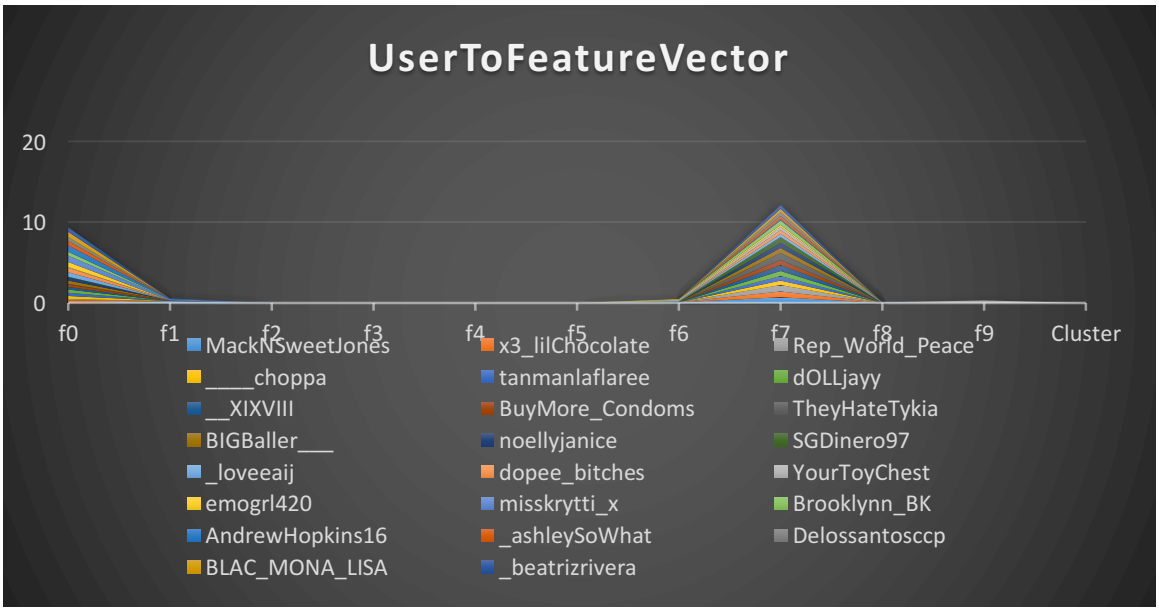



Figure 35: Top Negative 20 feature vector distribution

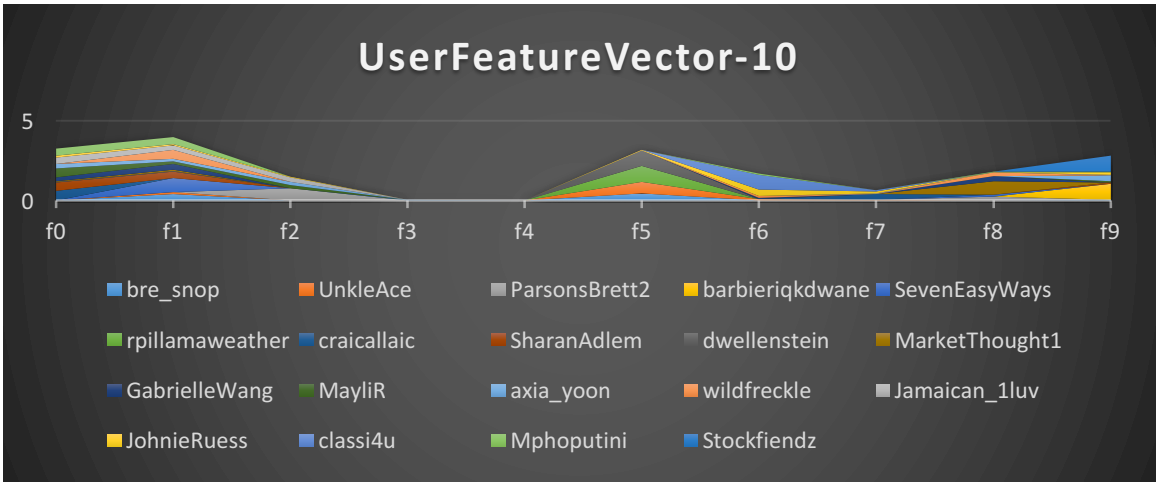



Figure 36: Positive 20 feature vector distribution

above graphs are very informative and these primary observations can be made.

1. Negative user tweets seem to be concentrated in two dimension, dimension 0 and dimension 7
2. However, positive sentiment feature vector spans across all dimensions and is not limited like Top Negative users.

Next we tried to assign {Negative, Nonnegative} class to each user. This decision of labeling a user was taken based on logic that "any user whose negative feature's value in the user feature vector was higher than median, then that user's class was marked as Negative".

This is how the distribution of each feature vector looked like for 10 topics.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 |
| mean | 0.084305 | 0.029858 | 0.012423 | 0.058831 | 0.094426 | 0.304761 | 0.089853 | 0.162417 | 0.085558 | 0.071437 |
| std | 0.164957 | 0.108231 | 0.100519 | 0.163413 | 0.159650 | 0.302060 | 0.205446 | 0.239549 | 0.207704 | 0.219131 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.010000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.010000 | 0.000000 | 0.000000 | 0.000000 | 0.020000 | 0.210000 | 0.000000 | 0.040000 | 0.000000 | 0.000000 |
| 75% | 0.080000 | 0.010000 | 0.000000 | 0.030000 | 0.110000 | 0.560000 | 0.050000 | 0.230000 | 0.040000 | 0.000000 |
| max | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.990000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

Figure 37: Statistics related with 10 features

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 |
| mean | 0.326848 | 0.028883 | 0.064588 | 0.033844 | 0.004777 | 0.012853 | 0.011940 | 0.048359 | 0.047703 | 0.083555 |
| std | 0.341902 | 0.094627 | 0.180307 | 0.103385 | 0.054537 | 0.090231 | 0.079115 | 0.144573 | 0.111066 | 0.178062 |
| min | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 50% | 0.210000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 75% | 0.630000 | 0.010000 | 0.010000 | 0.020000 | 0.000000 | 0.000000 | 0.000000 | 0.020000 | 0.050000 | 0.070000 |
| max | 1.000000 | 0.990000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.980000 | 0.990000 |

| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|
| 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 | 1907.000000 |
| 0.101353 | 0.049638 | 0.013456 | 0.032764 | 0.001961 | 0.017168 | 0.008673 | 0.013435 | 0.077132 | 0.010189 |
| 0.168699 | 0.142537 | 0.068886 | 0.119778 | 0.027836 | 0.091082 | 0.063288 | 0.076286 | 0.235345 | 0.096757 |
| 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 0.020000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 0.130000 | 0.030000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |

Figure 38: Statistics related with 20 features

Similarly, we computed it for for 30, 40, and 50 features. To find the feature value which represents negative one, we simply look in to the topic distribution as shown in Figure-24 and as from topic it seems obvious that Topic 8 (7 if 0 included) is the negative one. Look at the distribution of feature 8 in Figure-32. Standard deviation is 0.23. Now look at the feature vector of each user in Figure-26. So a simple logic, which takes the feature-8's value, compare it with the median value. If the the value is greater than the median we marked the user of class "N" (Negative) else "NN" (Non-negative).

```
class,id,f0,f1,f2,f3,f4,f5,f6,f7,f8,f9
NN,1,0.52,0,0,0.02,0.03,0.16,0.03,0.23,0,0
NN,2,0.81,0,0,0.02,0.12,0.04,0,0,0,0
N,3,0,0,0,0,0,0.3,0,0.69,0,0
NN,4,0.04,0,0,0.02,0,0.05,0.6,0,0.02,0.27
N,5,0.04,0,0,0.09,0.03,0.33,0.02,0.47,0.02,0
NN,6,0.82,0,0,0.01,0.11,0.02,0,0.03,0,0
N,7,0.36,0.04,0,0.01,0.11,0,0.02,0.45,0,0
NN,8,0,0,0,0,0.02,0.65,0,0.33,0,0
NN,9,0.53,0,0,0,0.21,0.07,0.16,0,0.03,0
NN,10,0.21,0.01,0,0,0.38,0.16,0.05,0.15,0.02,0
NN,11,0.08,0,0,0.01,0.5,0.17,0,0.23,0,0
NN,12,0,0,0,0,0,0,0,0,0,0.99
NN,13,0.05,0.02,0,0.03,0.07,0.64,0.16,0.02,0,0
NN,14,0.61,0,0,0,0.1,0.09,0,0.02,0.17,0
NN,15,0.02,0.02,0,0,0.05,0.04,0.87,0,0,0
NN,16,0,0,0,0,0.03,0.39,0,0.04,0.53,0
NN,17,0,0,0,0,0,0,0,0,0,1
NN,18,0,0.36,0,0.06,0.51,0,0,0,0.06,0
NN,19,0,0.01,0,0,0,0.15,0,0,0.84,0
```

Figure 39: Class included with feature vector

After doing this, to get a sense of class distribution we plotted different feature vectors.

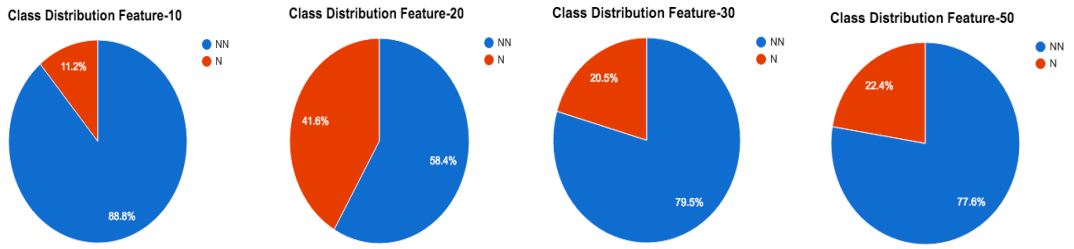



Figure 40 Class distribution across dataset

Looking at the class distribution, we can say that this data distribution looks near real time, as data percent of people who use abusive language hovers from 10-20. Hence, next we wanted to see what kind of clusters exists, like we did it for traditional twitter data in section 4.1.

### 4.4 Clustering

We attempted clustering on data represented in Figure 26, to see how many clusters are present.

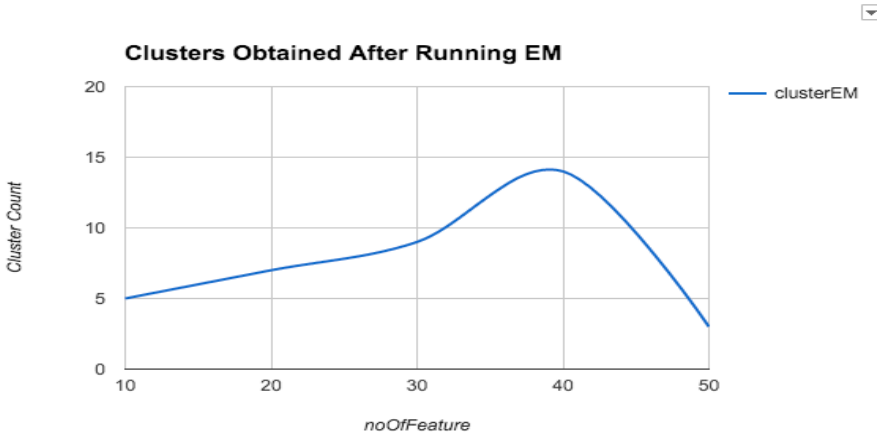



Figure 41: Clusters obtained after running EM on multiple topic size

We studied clustering performed on 10 – topic to understand it in detail.

### 4.4.1 Clustering on 10 Dimensional Feature vector – Explained

We ran the EM algorithm on Figure-34 data by excluding class feature. EM algorithm represented the entire data set in 5 clusters.

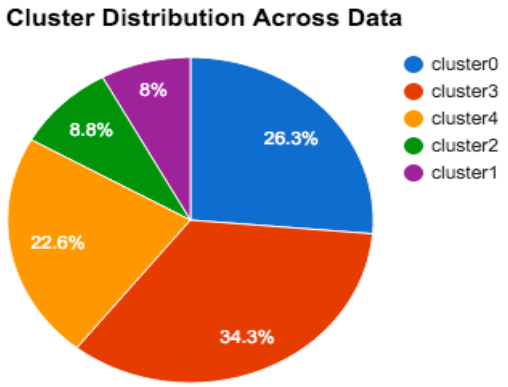



Figure 42: Cluster distribution across data

Next we picked top 20 negative users who were sorted on their negative sentiment score.

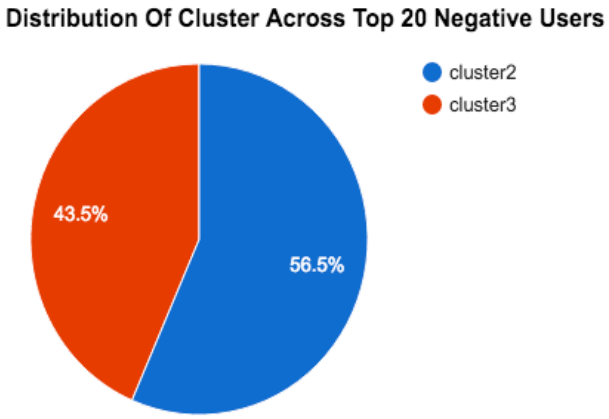



Figure 43: Distribution of cluster among top 20 negative users

Its obvious from graph that the users in top most negative sentiment belonged to either cluster 3 or cluster 2. Next, we wanted to see how the cluster distribution looked like for

all negative marked users. Later we also wanted to compare it with twitter cluster distribution, to see if we were doing better with EM on LDA feature or on Twitter's conventional data set.

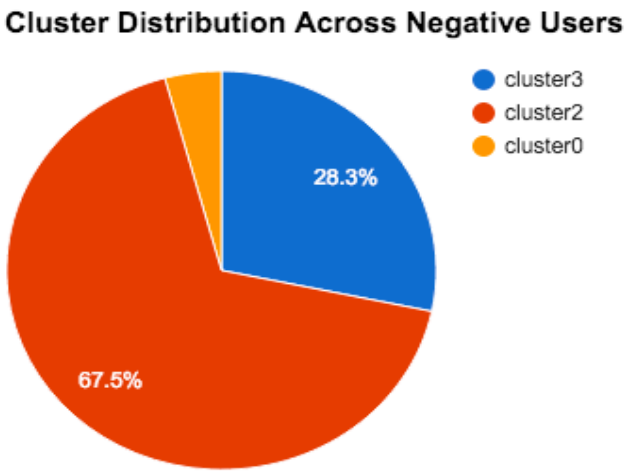



Figure 44: Cluster distribution across all negative users

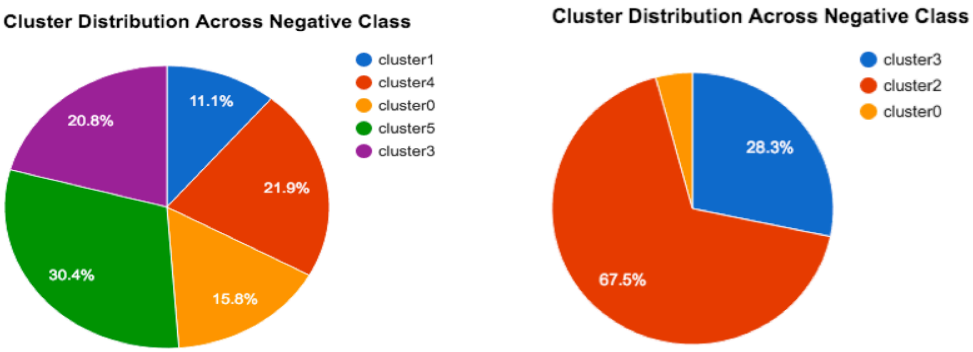



Figure 45: Comparing Twitter negative class clustering against LDA's feature vector's clustering

This is very encouraging, as the negative class is really well concentrated in the LDA feature vector analysis, that means EM could classify the user's based on their negative

feature dominance. As a next step, we were also interested in understanding the logic behind the clustering. Hence, we plotted the feature vector.

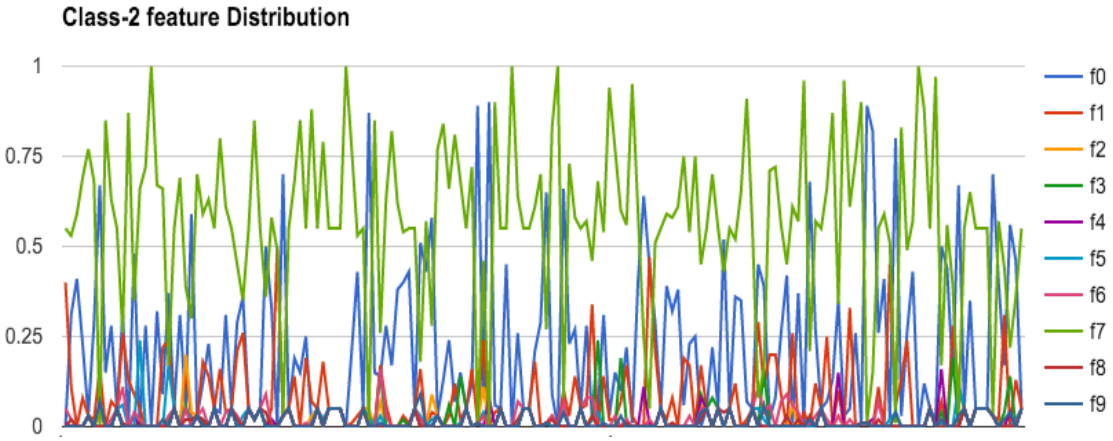



Figure 46: Feature distribution of users classified with class-2

The graph was suggesting clearly the dominance of f7 (feature-7) as a reason for classifying these users in class-2 category.

We also plotted users who were classified with class-2 with the labels (N/NN) we had created.

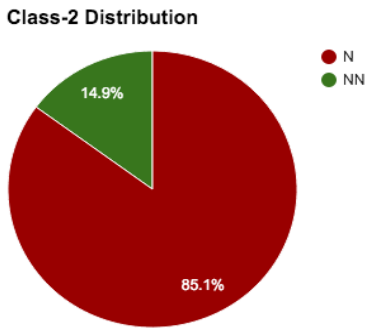



Figure 47: Class -2's distribution on labels Negative(N) / NonNegative(NN)

Which was suggesting very strongly that most of the negative users were marked as class-2. And because the top 20 negative users were marked either class-2 or class-3 we did the similar analysis of class-3


Figure 48: 36-Feature distribution of users classified with class-3

Which was suggesting that while these users had presence of feature 7, feature-0 was dominant. Hence, next we wanted to know, what percentage of total negative users, these class-2 and class-3 hold.

| | |
|---|---|
| Total N in entire set | 213 |
| Total N with combined classification of class-2 or class-3 | 204 |

Table 1: cluster distribution for user with Negative (N) class

i.e. 95.7 % of users with N tag were placed in these class with a logic that

- if user had a dominant negative feature: class 3
- if the user had negative feature but not dominant: clsss-2

We did a quick analysis on 30 dimensional feature vector to cross verify if clear trend seen in 10 dimensional data was not merely coincidence and on 50 dimensional, because

We wanted to investigate How well the trend seen with 10 and later 30 holds when the dimensionality of feature vector goes as high as 50.

### 4.4.2 Clustering on 30 Dimensional Feature vector – Explained

We first had the 30 topics generated from the same word corpus. Later we again picked the same top 20 with most negative sentiment users and plotted their feature vector. This is how the feature vector distribution looked like.

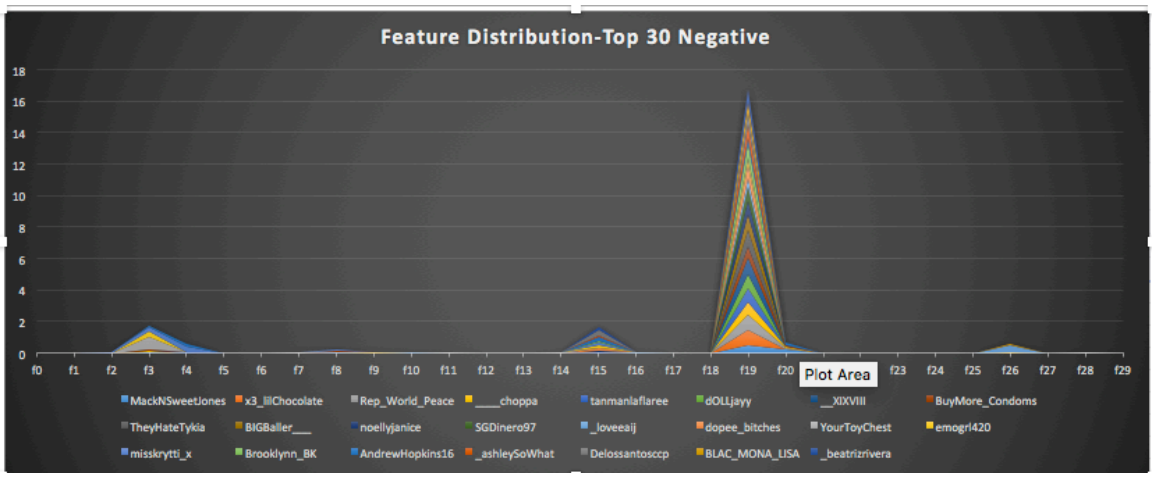



Figure 49: Top 20 negative feature vector distribution

From feature distribution its evident that, for top 20 most negative sentiment users have a strong f19, and f3 and f15 of similar strength. Here is what those features are.

**3, u'0.017*like + 0.015*just + 0.010*fuck + 0.009*one + 0.007*think + 0.007*know + 0.006*now + 0.006*good + 0.006*peopl + 0.006*look'**

**15, u'0.017*like + 0.016*just + 0.014*love + 0.012*want + 0.011*peopl + 0.009*someon + 0.009*life + 0.009*feel + 0.009*day + 0.008*know**

**(19, u'0.018*like + 0.015*just + 0.015*fuck + 0.013*shit + 0.009*know + 0.008*want + 0.008*need + 0.008*peopl + 0.008*bitch + 0.008*love')**

Figure 50: Topics which were showing dominance for most negative users

This is good, as this is inline with what the analysis suggested for 10 dimensional user feature vector. To cross check we also plotted the top 20 user's feature distribution, and this is how that looked like.

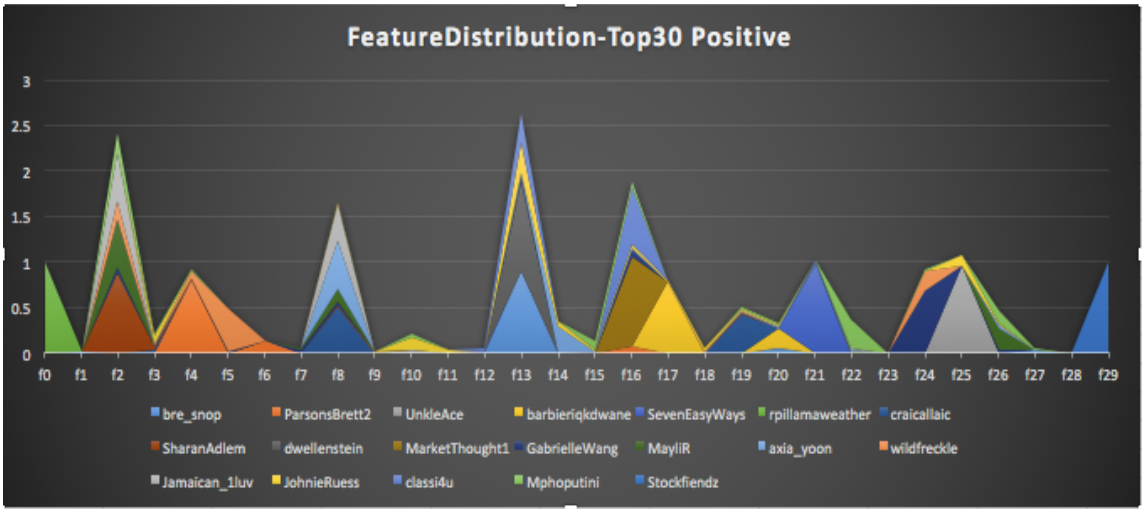


Figure 51: Top 20 Positive feature vector distribution

Which is also inline with our early observation that, while negative user's tweets are really focused, positive ones like to talk about almost everything. Next we wanted to see how the over all cluster distribution looked like.

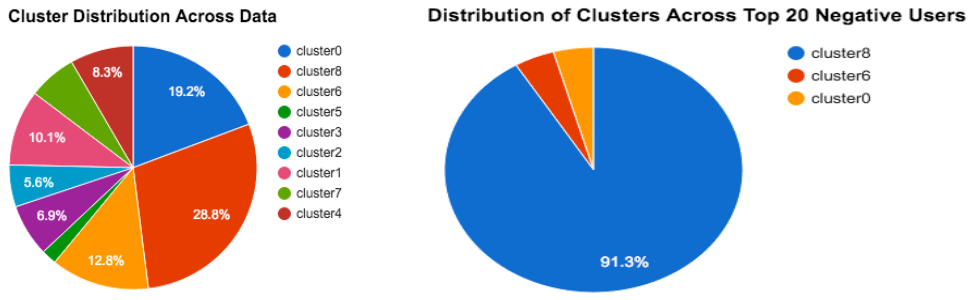


Figure 52: Cluster distribution across data and top 20 negative users

Figure-43 suggested that negative users were clusters in group cluster-8. Hence, we also wanted to have another look at the total negative user distribution across clusters.

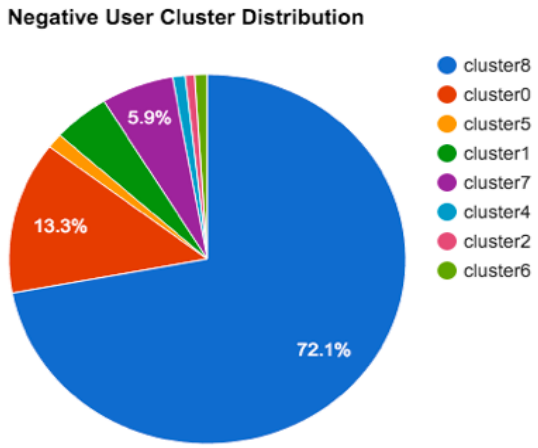



Figure 53: Negative user cluster distribution

Above infographics suggested that, 85 % of the total negative users were concentrated in two the clusters {cluster 8, cluster 0}. Which is always an indication of algorithm being able to group similar users in some space. Further analysis on cluster-8, gave a reasoning on clustering.

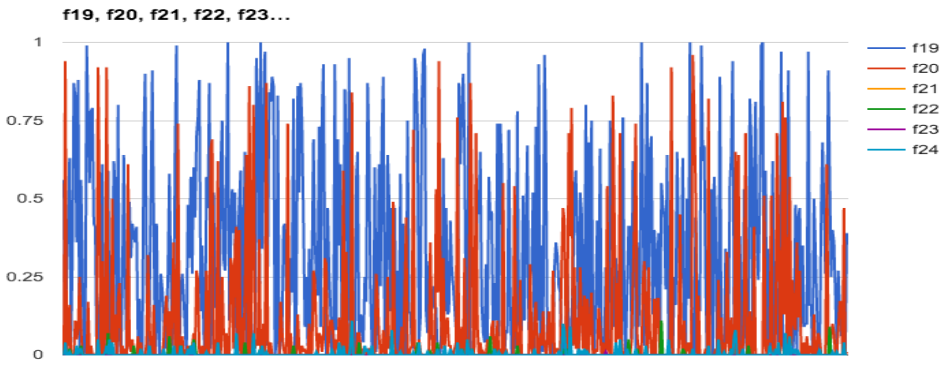



Figure 54: Cluser-8 was assigned based on f19 being the dominant feature

Upon experimenting further with feature vector sized 20, 40, and 50, we realized that the trends were just getting repeated. Hence we choose not to include that as a part of this document. Next challenge was to view these datasets in form of cluster.

### 4.5 Cluster Visualization using t-SNE approach.

t- Distributed Stochastic Neighbor Embedding (t-SNE) is a technique designed for dimensionality reduction, that is particularly well suited for the visualization of high dimensional dataset. This was perfect for us. Here is the visualization we got when we plotted them in unsupervised (by ignoring class N/NN) and later by including class definition. This is how trial run looked like on on multiple size dataset.

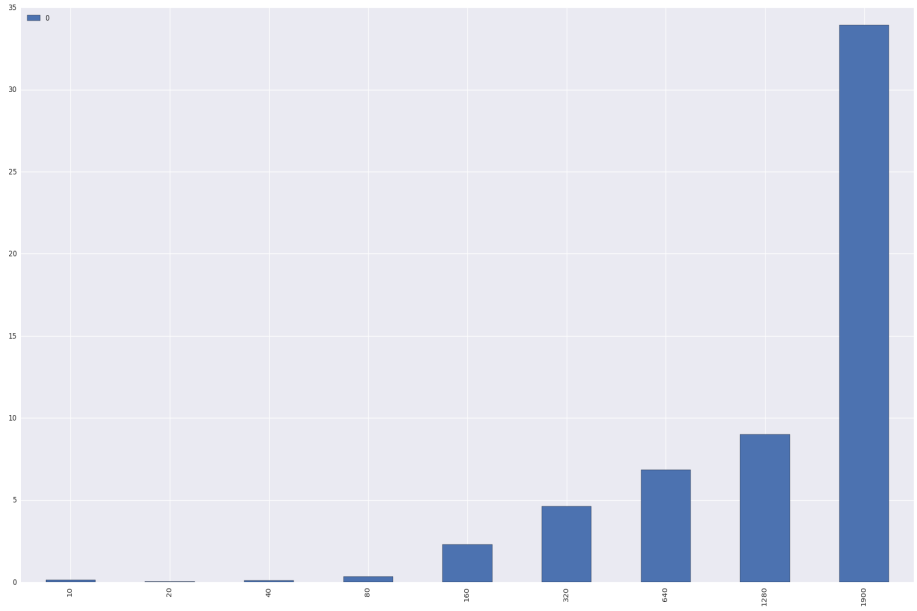



Figure 55: Data size to run time analysis – $O(n^2)$ Big-oh($n^2$)

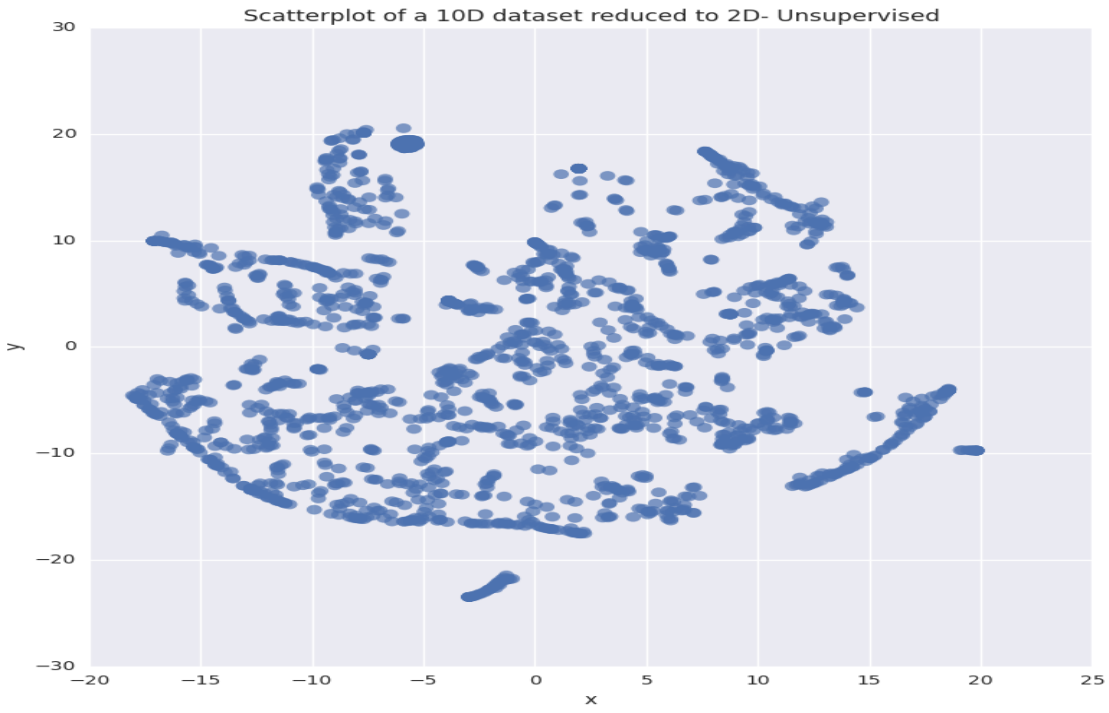



Figure 56: t-SNE 10 Dimensional projected on 2-D - Unsupervised

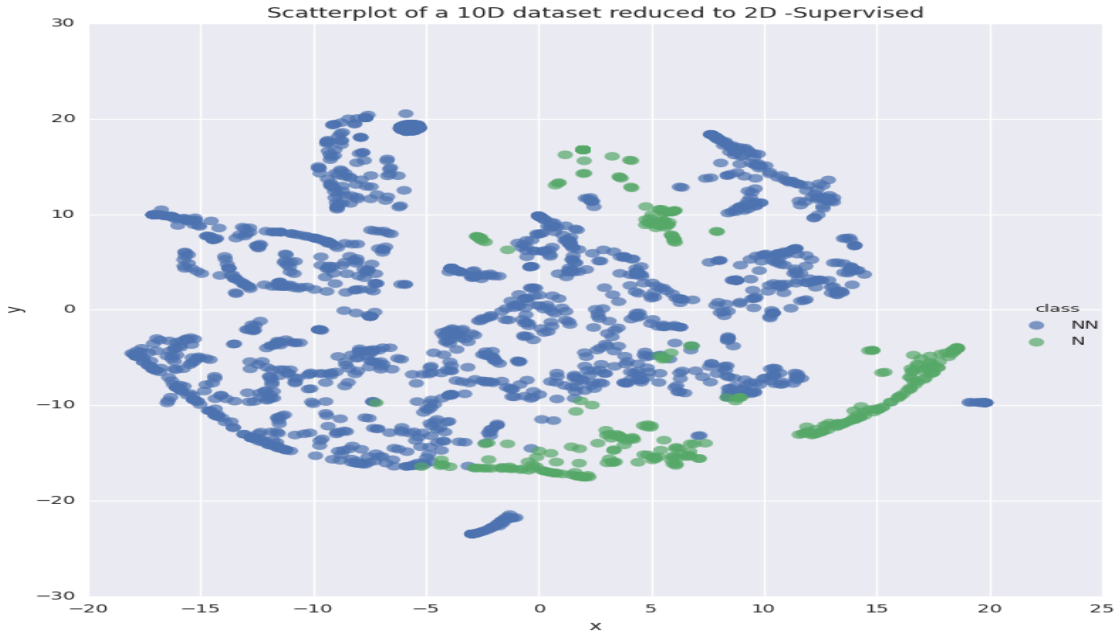



Figure 57: t-SNE 10 Dimensional projected on 2-D - Supervised

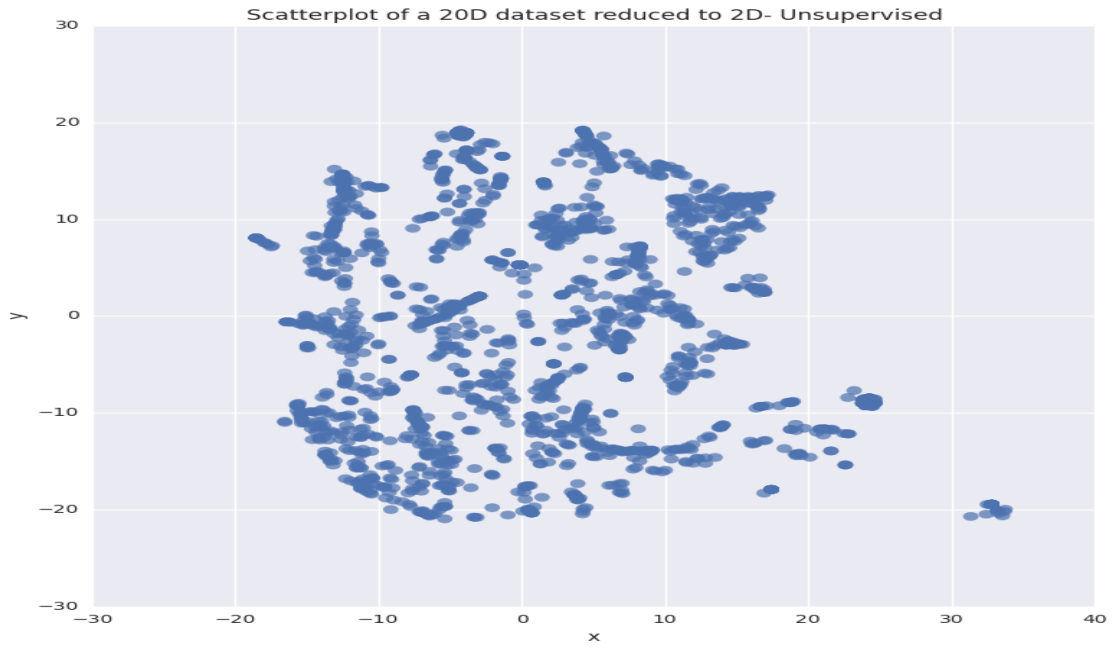



Figure 58: t-SNE 20 Dimensional projected on 2-D - Unsupervised

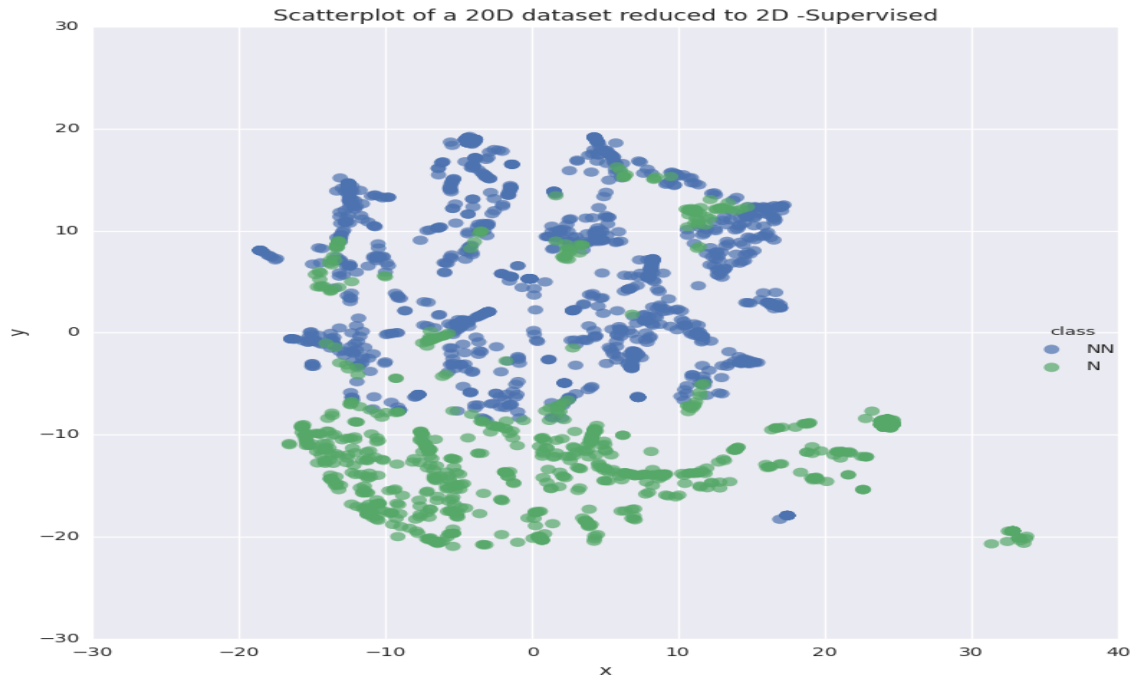



Figure 59: t-SNE 20 Dimensional projected on 2 D - Supervised

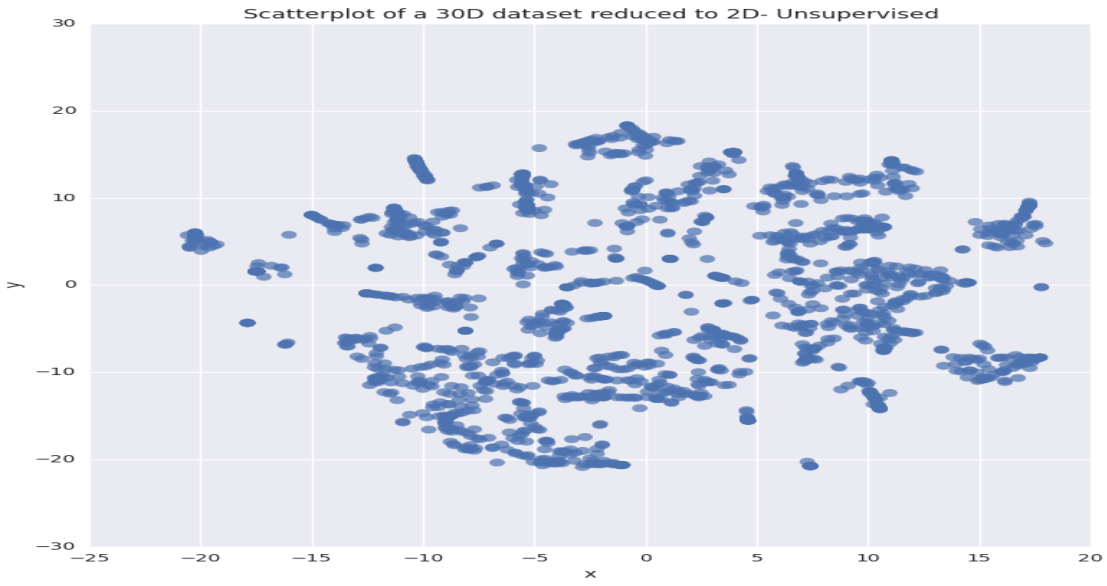



Figure 60: t-SNE 30 Dimensional projected on 2-D - Unsupervised

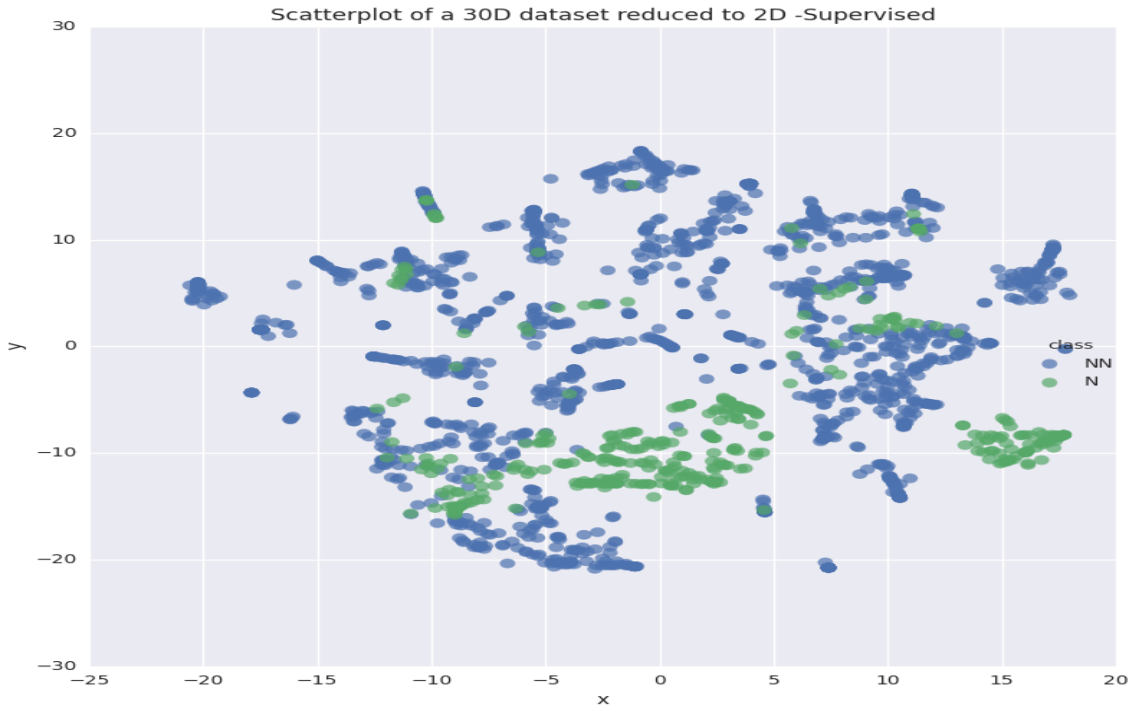



Figure 61: t-SNE 30 Dimensional projected on 2 D - Supervised

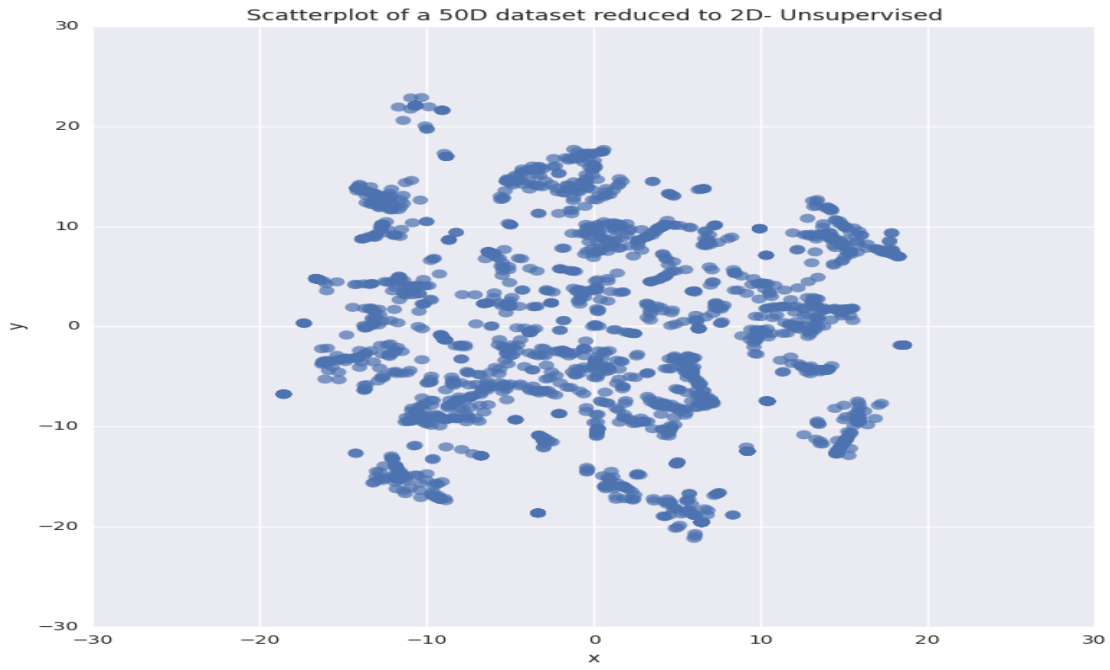



Figure 62 : t-SNE 50 Dimensional projected on 2 D – Unsupervised

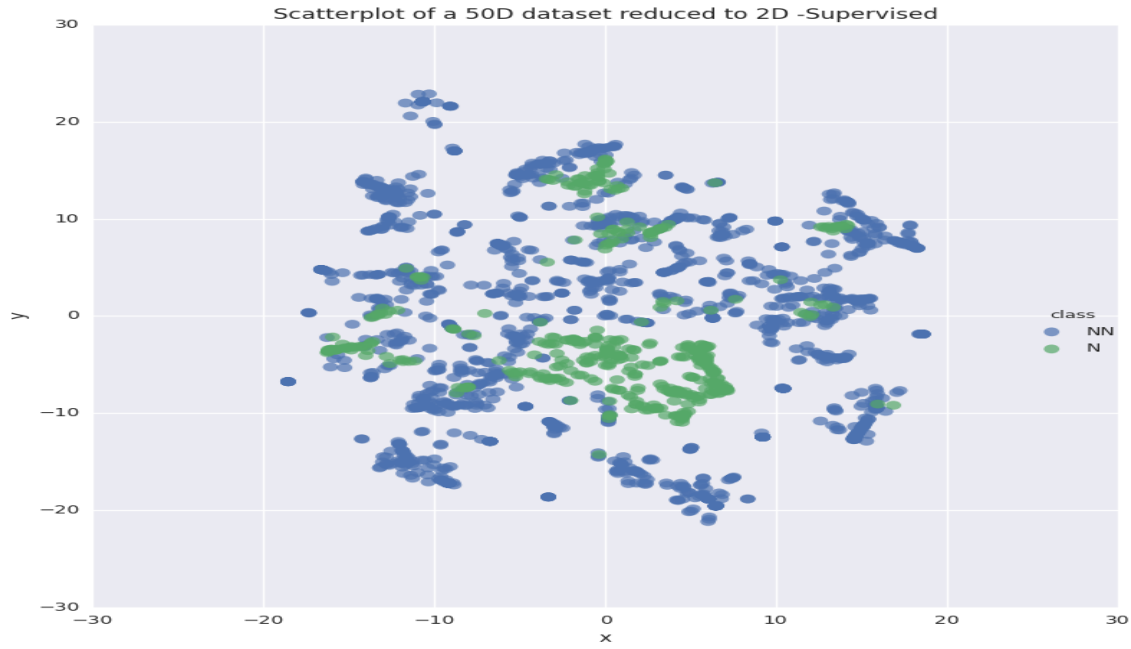



Figure 63: t-SNE 50 Dimensional projected on 2 D – Supervised

To understand the behavior of clusters even better, we experimented with one more approach, where we assigned the class to a feature vector based on its most dominant feature.

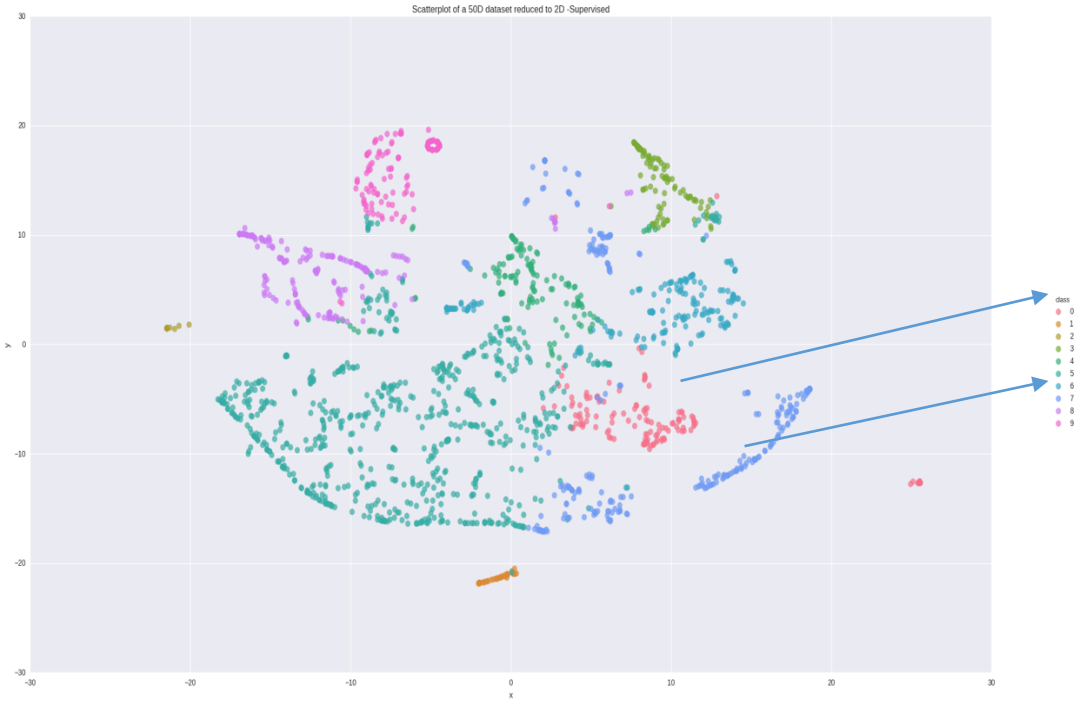



Figure 64: 10 Class distribution on t-SNE plot

This graph explains the distribution in even finer detail as negative topic 7 is placed very closely which has common words like, video, love etc. This is aligned with how EM clustered these instances. However, as expected food (class-9) and travel (class-8) is farthest from the negative topic. Which also verifies the claim of t-SNE that it could cluster a high dimension data in lower dimension while still keeping the clusters correlated by some distance measure. The class diagram for further 20, 30, 40, and 50 feature was getting clumsier. Hence we choose not to attach it. The same diagram is available upon request.

# CHAPTER 5

# EVALUATION

Results we got from EM Clustering on feature vectors, irrespective of its size, were very consistent with singling out extremely negative users from moderate one to the positive ones.

The combination of topic modeling and sentiment analysis was working great, as either one of these could be used as validation mechanism for another. However, in context of a scalable pipeline, LDA as primary method will make more sense. The reason being LDA has many distributed implementation available. Sentiment analysis could be used for validating in distributed set up as well using dipsy [40]. Together, they make a pipeline, which is less prone to false positive errors. Hence can successfully filter users with most abusive or negative content.

As an independent validation mechanism, we downloaded set of 6300 user's tweet independently and ran it through our pipeline. clusters made by feature detected out of LDA, could detect all the negative users, and sentiment analysis on the same set of users gave us 87 out of 100, same users. This study was done to establish that primary data on which analysis was done did not had any bias toward negative users and there was always at least one negative topic present in the topics generated. However, we must mention that we only validated this with topic count 20 and 30.

### 5.1 Using Sentiment Analysis

Please find below the experiment results with similar analysis as done on test data.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | userName | Positive | Negative | Total |
| 2 | full_rips | 2624 | -50172 | -47548 |
| 3 | XXXBinge | 6566 | -21082 | -14516 |
| 4 | xhamsterpornos | 3376 | -16847 | -13471 |
| 5 | StefByn | 795 | -16396 | -15601 |
| 6 | Wendy_N85 | 1526 | -15475 | -13949 |
| 7 | totalpornsites | 4056 | -13993 | -9937 |
| 8 | alisoncams | 0 | -12972 | -12972 |
| 9 | naniariesgirl | 0 | -12948 | -12948 |
| 10 | fucksxxx | 3233 | -12932 | -9699 |
| 11 | marketastroblo | 0 | -12908 | -12908 |
| 12 | eyalfeed | 800 | -8802 | -8002 |
| 13 | Stephani_Phan | 5112 | -7780 | -2668 |
| 14 | finessinglikejj | 1722 | -7650 | -5928 |
| 15 | canadianpromo1 | 2686 | -7370 | -4684 |

Figure 65: Top 15 most negative user

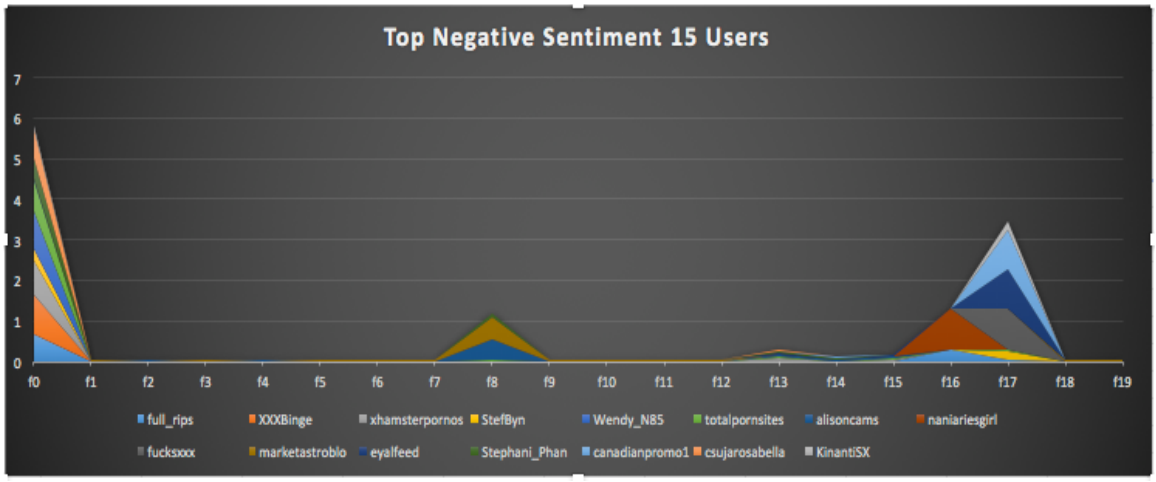



Figure 66: Feature distribution of top 15 users

### 5.2 Using LDA Feature Vector

We generated the feature vector of newly downloaded 6300 users and went through preprocessing steps as mentioned in 3.2 and later feature vector creation as mentioned in 3.4. We ran the experiment for 30 topics and 15 words under each topic.

This is the result we had:

[(**0**, u'0.028*hot + 0.020*fuck + 0.018*ass + 0.018*sexi + 0.014*porn + 0.014*girl + 0.012*blond + 0.011*amaz + 0.010*bdsm + 0.010*sweet + 0.009*wild + 0.009*kinki + 0.009*insan + 0.009*sizzl + 0.009*babe'),

(**1**, u'0.015*hahahahaha + 0.007*hehe + 0.007*hahahahahaha + 0.006*score + 0.006*anonym + 0.006*man + 0.005*nako + 0.005*gameid + 0.005*miss + 0.005*huhu + 0.005*wala + 0.004*incognito + 0.004*nalang + 0.004*goooaaal + 0.004*akong'),

(**2**, u'0.032*win + 0.017*enter + 0.011*deal + 0.010*chanc + 0.009*collect + 0.009*just + 0.009*black + 0.007*giveaway + 0.007*size + 0.007*set + 0.006*contest + 0.005*end + 0.005*men + 0.005*gift + 0.005*card'),

**(3**, u'0.028*may + 0.018*now + 0.017*follow + 0.015*might + 0.013*automat + 0.011*check + 0.011*person + 0.010*one + 0.008*feel + 0.008*peopl + 0.007*book + 0.007*just + 0.007*chang + 0.006*life + 0.006*cancer'),

**(4**, u'0.028*kabc + 0.017*trump + 0.006*say + 0.006*obama + 0.006*hillari + 0.005*berni + 0.005*donald + 0.005*clinton + 0.005*vote + 0.005*isi + 0.005*will + 0.005*kill + 0.004*sander + 0.004*peopl + 0.004*debat'),

(**5**, u'0.013*yung + 0.009*mga + 0.008*may + 0.008*naman + 0.008*love + 0.007*talaga + 0.006*hindi + 0.006*happi + 0.006*vote + 0.005*tweet + 0.005*yan + 0.005*nga + 0.005*main + 0.005*kaya + 0.005*pag'),

(**6**, u'0.016*guru + 0.014*bless + 0.014*msg + 0.010*fan + 0.009*human + 0.009*papa + 0.008*dhan + 0.007*god + 0.007*movi + 0.007*craze + 0.006*wow + 0.006*love + 0.006*great + 0.005*good + 0.005*work'),

(**7**, u'0.011*goal + 0.010*score + 0.010*game + 0.009*arsen + 0.009*leagu + 0.009*win + 0.008*player + 0.008*messi + 0.008*barcelona + 0.007*season + 0.007*team + 0.007*unit + 0.006*west + 0.006*fan + 0.005*chelsea'),

(**8**, u'0.019*lol + 0.017*like + 0.012*fuck + 0.012*just + 0.012*shit + 0.009*nigga + 0.008*know + 0.007*lmao + 0.006*bitch + 0.006*need + 0.006*ass + 0.006*man + 0.006*girl + 0.005*time + 0.005*good'),

(**9**, u'0.042*inc + 0.018*trade + 0.018*share + 0.014*manag + 0.013*corp + 0.013*sell + 0.013*rate + 0.012*price + 0.011*llc + 0.011*invest + 0.010*buy + 0.009*stake + 0.009*group + 0.008*capit + 0.008*een'),

(**10**, u'0.013*india + 0.007*hai + 0.007*delhi + 0.006*will + 0.006*ani + 0.004*indian + 0.004*pakistan + 0.004*govt + 0.004*nation + 0.004*modi + 0.004*court + 0.003*bjp + 0.003*jnu + 0.003*day + 0.003*case'),

(**11**, u'0.939*updat + 0.016*latest + 0.007*fin + 0.006*dolphin + 0.004*photonot + 0.000*text + 0.000*mod + 0.000*ejuic + 0.000*yoonaya + 0.000*twb + 0.000*line + 0.000*save + 0.000*tank + 0.000*code + 0.000*shop'),

**(12**, u'0.116*music + 0.106*trend + 0.098*search + 0.093*gay + 0.092*vid + 0.092*hottest + 0.090*militari + 0.020*zayn + 0.015*harri + 0.014*loui + 0.011*direct + 0.011*pleas + 0.011*bio + 0.010*ff + 0.010*give'),

**(13**, u'0.007*thank + 0.007*just + 0.007*good + 0.007*day + 0.007*one + 0.007*will + 0.006*look + 0.006*great + 0.006*like + 0.006*time + 0.005*now + 0.005*see + 0.004*think + 0.004*love + 0.004*make'),

**(14**, u'0.010*job + 0.010*market + 0.009*busi + 0.008*video + 0.008*2016 + 0.007*media + 0.007*find + 0.007*digit + 0.006*social + 0.006*make + 0.006*need + 0.006*manag + 0.006*design + 0.005*news + 0.005*tip'),

**(15**, u'0.019*bmh + 0.016*play + 0.014*now + 0.007*game + 0.006*video + 0.005*star + 0.004*2016 + 0.004*found + 0.004*top + 0.003*year + 0.003*season + 0.003*favorit + 0.003*hit + 0.003*basketbal + 0.003*week'),

**(16**, u'0.059*photo + 0.055*video + 0.043*post + 0.029*virgo + 0.027*facebook + 0.019*watch + 0.011*see + 0.007*man + 0.006*buhari + 0.006*nigerian + 0.006*download + 0.006*nigeria + 0.005*thing + 0.005*ladi + 0.005*tauru'),

**(17**, u'0.039*free + 0.031*download + 0.021*app + 0.015*onlin + 0.015*iphon + 0.015*use + 0.014*appl + 0.013*android + 0.012*now + 0.012*wifi + 0.011*googl + 0.009*code + 0.009*stock + 0.008*order + 0.007*avail'),

**(18**, u'0.044*gmt + 0.039*touch + 0.024*usd + 0.023*level + 0.021*figur + 0.015*posit + 0.015*open + 0.015*5so + 0.015*jpi + 0.014*sentiment + 0.014*swfx + 0.011*rank + 0.011*yesterday + 0.011*eur + 0.010*juli'),

**(19**, u'0.082*daili + 0.073*stori + 0.027*organ + 0.024*newspap + 0.020*news + 0.015*make + 0.015*energi + 0.014*food + 0.013*green + 0.013*product + 0.013*follow + 0.013*earth + 0.013*cat + 0.013*renew + 0.013*solut'),

**(20**, u'0.060*eat + 0.039*retweet + 0.037*follow + 0.028*thank + 0.028*stat + 0.014*unfollow + 0.014*back + 0.013*one + 0.013*rihanna + 0.012*vote + 0.012*love + 0.011*pleas + 0.011*now + 0.009*want + 0.009*happi'),

**(21**, u'0.019*manuel + 0.018*toma + 0.012*doc + 0.011*kay + 0.011*papa + 0.010*cardo + 0.010*tom + 0.009*team + 0.008*surf + 0.008*henri + 0.008*naman + 0.008*wave + 0.006*talaga + 0.006*ivan + 0.005*fair'),

**(22**, u'0.116*check + 0.073*buy + 0.068*thx + 0.065*want + 0.065*rule + 0.062*bi0 + 0.062*f0r + 0.062*with0ut + 0.016*hallelujah + 0.010*current + 0.009*humid + 0.009*wind + 0.007*fair + 0.007*amen + 0.006*email'),

**(23**, u'0.006*say + 0.006*will + 0.005*year + 0.005*bank + 0.004*peopl + 0.004*news + 0.004*report + 0.003*polic + 0.003*govern + 0.003*world + 0.003*attack + 0.003*kill + 0.003*state + 0.003*tax + 0.003*break'),

**(24**, u'0.016*like + 0.014*love + 0.014*just + 0.009*peopl + 0.009*want + 0.008*one + 0.007*feel + 0.007*make + 0.007*know + 0.007*day + 0.006*time + 0.006*someon + 0.006*life + 0.006*need + 0.006*think'),

**(25**, u'0.016*wkwk + 0.013*mau + 0.012*itu + 0.012*iya + 0.010*say + 0.010*udah + 0.010*nya + 0.010*gue + 0.009*gak + 0.008*kan + 0.008*juga + 0.007*jam + 0.007*kalo + 0.007*nih + 0.007*pakistan'),

**(26**, u'0.072*instagram + 0.050*eye + 0.049*ladi + 0.049*send + 0.048*featur + 0.048*candi + 0.048*becom + 0.047*section + 0.047*famou + 0.047*sexiest + 0.020*fonethedon1 + 0.018*mixtap + 0.016*congo + 0.016*nigga + 0.016*fiyahhh'),
**(27**, u'0.049*eurovis + 0.043*2015 + 0.023*video + 0.013*kimxi + 0.010*lyric + 0.009*tsou + 0.009*photosandbacon + 0.009*thestoryofu + 0.007*lyrik + 0.007*kavinoki + 0.006*kimdireurovis + 0.006*love + 0.006*kim + 0.005*wer + 0.005*notifica'),
**(28**, u'0.015*dan + 0.006*dari + 0.006*dengan + 0.005*bisa + 0.005*orang + 0.004*indonesia + 0.004*tidak + 0.004*akan + 0.004*hari + 0.004*kita + 0.004*foto + 0.003*naik + 0.003*dalam + 0.003*itu + 0.003*jadi'),
**(29**, u'0.009*bila + 0.008*kalau + 0.007*orang + 0.006*kita + 0.006*pun + 0.005*dengan + 0.005*buat + 0.005*boleh + 0.004*tapi + 0.004*kat + 0.004*macam + 0.004*nie + 0.004*hati + 0.004*laa + 0.004*dan')]

This output clearly states the fact that effectively these users are asking people to download porn. Manual validation aligns with observation. Method correctness in detecting user with abusive and negative content was 100 %. The best outcome with this result was it was not only clear that why a user was marked negative but also the context ion which the user was marked negative. Similarly, this pipeline can be easily used to download in context of specific mental health issue, or any disease.

CHAPTER 6

# CONCLUSION AND FUTURE WORK

## 5.1 Conclusion

To conclude, our proposed framework works in a scalable fashion and with very high accuracy. It can not only detect users who are posing threat because of their abusive and toxic threat but also the context in which they are posing threat.

Our experiments started with 1900 user's dataset, later we extended the same data set to 4300 users to find the generality in the behavior displayed. Once we were able to successfully not only anticipate public health threats, but also could identify at risk individuals from social media for non contagious disease in the realm of mental health, we also cross validated them with another algorithmic approach (sentiment analysis) and hence the pipeline could be fully automated. Based on our results, we have proven that our proposed framework, which combines topic modeling and sentiment analysis to provide an estimate of toxic, abusive behavior, can effectively identify and filter a pool of potentially at-risk users via their content on social media. Our framework can also be tuned on incoming data incrementally over a period of time, which ensures better results over time on unobserved data. Such a framework can work at scale, because of careful selection of components. Our final output support our claim that this framework can not only give accurate user detection but also the context in which user was marked negative. Also, as this framework works on generative model for deriving topics, it can also detect hidden signals which have suddenly started to appear in negative context.

## 5.2 Future Work

While we have done a detailed work on designing the pipeline in most efficient and informed way, here are few areas in which the pipeline performance can be improved further.

1. Sentiment analysis works as bag-of-word model. A more sophisticated algorithm can be used to gather sentiment where sentence context is also given importance.
2. Network properties of user, can be better utilized and sentiment of a user should be able to consume those properties as well. While we were able to point out how the network of negative users differ from positive one, we could not include them while building our model.
3. While building LDA model, correcting the word could help LDA model, outputting only correct words as twitter is known for misspelled and incomplete words.
4. We could not include online rhetorics generated while doing sentiment analysis. Such rhetorics have lot of contextual meaning. Hence these could be included to get a more accurate sentiment score.
5. Our framework can adjust its learning on incoming text by updating the LDA model with time. However, having a feedback component for the prediction could make learning more accurate as well as conclusive.

# REFERENCES


1. Mental Health Definition by WebMD, http://www.webmd.com/mental-health/
2. Sayali Kale UGA Graduate Student Computer Science.
3. Mark Dredge, 2014, 'Using Twitter to Track the Flu'
4. Mark Dredge, 2015, Analysis of Twitter posts could provide fresh insight into mental illness trends
5. Arvind Ramanathan, Laura L. Pullum, Tanner C. Hobson, Christopher G. Stahl, Chad A. Steed, Shannon P. Quinn, Chakra S. Chennubhotla and Silvia Valkova. 2015. Discovering Multi-Scale Co-occurrence Patterns of Asthma and Influenza with the Oak Ridge Bio-surveillance Toolkit."
6. Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, Veselin Stoyanov, 2015, "SemEval-2015 Sentiment analysis in twitter"
7. Evan T.R. Rosenman, 2012, "Retweets but not just Retweets - Quantifying and predicting influence on twitter"
8. Bridianne O'Dea, Stephen Wan, Philip J. Batterham, Alison L. Calear, Cecile Paris, Helen Christensen, 2015, "Detecting suicidality on Twitter"
9. Kimberly McManus, Emily K. Mallory, Rachel L. Goldfeder, Winston A. Abstract Haynes, Jonathan D. Tatum, 2015, "Mining Twitter Data to Improve Detection of Schizophrenia "

10. Glen A. Coppersmith Craig T. Harman Mark H. Dredze, 2014, “Measuring Post Traumatic Stress Disorder in Twitter”
11. “Lightning Fast Cluster Computing” http://spark.apache.org/
12. “Topic Modelling for Humans” https://radimrehurek.com/gensim/
13. “Twitter” https://about.twitter.com/company
14. “Hadoop” http://hadoop.apache.org/
15. Apoorv Agarwal, Fadi Biadsy, and Kathleen Mckeown. 2009. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 24–32, March.
16. Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. Proceedings of the 23rd International Conference on Computational Linguistics: Posters, pages 36–44.
17. Adam Bermingham and Alan Smeaton. 2010. Classifying sentiment in microblogs: is brevity an advantage is brevity an advantage? ACM, pages 1833–1836.
18. C. Fellbaum. 1998. Wordnet, an electronic lexical database. MIT Press.
19. Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. Proceedings of the 20th intern
20. Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013a. Major life changes and behavioral markers in social media: Case of childbirth. In

Proceedings of the ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW).

21. Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013b. Predicting postpartum changes in emotion and behavior via social media. In Proceedings of the ACM Annual Conference on Human Factors in Computing Systems (CHI), pages 3267–3276. ACM.
22. Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013c. Social media as a measurement tool of depression in populations. In Proceedings of the Annual ACM Web Science Conference.
23. Munmun De Choudhury, Michael Gamon, Scott Counts, and Eric Horvitz. 2013d. Predicting depression via social media. In Proceedings of the International AAAI Conference on Weblogs and Social Media (ICWSM).
24. Munmun De Choudhury, Andres Monroy-Hernandez, and Gloria Mark. 2014." narco" emotions: Affect and desensitization in social media during the Mexican drug war. Munmun De Choudhury. 2013. Role of social media in tackling challenges in mental health. In Proceedings of the 2nd International Workshop on Socially Aware Multimedia, pages 49–52.
25. TextBlob:http://textblob.readthedocs.org/en/dev/quickstart.html#sentiment-analysis
26. http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010
27. D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. The Journal of Machine Learning Research, 3:993–1022, 2003

28. Liangjie Hong and Brian D. Davison. Empirical study of topic modelling in twitter
29. K-Mean: https://en.wikipedia.org/wiki/K-means_clustering
30. https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
31. http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster
32. “Visualizing high dimensional data in small dimensions” https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
33. “Virtual Server hosting” https://www.digitalocean.com/
34. “Amazon EC2 Virtual server hosting” https://aws.amazon.com/ec2/
35. “Negative sentiment word collection” http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
36. “The Open Graph Visualization Platform” https://gephi.org/
37. “Topic Modelling ” from https://en.wikipedia.org/wiki/Topic_model
38. “Bag-of-Word Model” from https://en.wikipedia.org/wiki/Bag-of-words_model
39. “Python Distributed Module from ”http://dispy.sourceforge.net/