SOCIAL MEDIA ANALYTICS

THESIS

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

K. K. BIRLA - GOA CAMPUS

November 2011

SOCIAL MEDIA ANALYTICS

THESIS

Submitted in partial fulfillment of the requirements of BITS C422T Thesis

By

Nazneen Fatema Rajani

ID No. 2007C6TS425G

Under the supervision of

Dr. Gaurav Dar

Associate Professor, Department of Physics

Birla Institute of Technology and Science, Pilani - Goa Campus, Goa

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (RAJASTHAN)

NOVEMBER 2011

CERTIFICATE

This is to certify that the Thesis entitled, Social Media Analytics and submitted by Rajani Nazneen Fatema Naushad, ID No. 2007C6TS425G in partial fulfilment of the requirement of BITS C421T/422T Thesis embodies the work done by her under my supervision.

Date: 30th November 2011

Signature of the supervisor

Name: Dr Gaurav Dar

Designation: Associate Professor

ACKNOWLEDGEMENTS

At the outset, I would like to thank my instructor and guide Dr.Onkar Dabeer, Tata Institute of Fundamental Research, Mumbai for guiding and mentoring me throughout my project and helping me understand the various math concepts involved. I would like to thank Dr.Gaurav Dar for co-guiding my thesis, and motivating me throughout my project

To my parents, I want to express my love and appreciation for their continued support for all my endeavours, including this, my latest one.

I apologize to and thank all those people whom I have forgotten to mention here.

ABSTRACT

The aim of this thesis is to implement and build a recommendation system based on opinion mining. Many a times, individuals tend to display their opinions or view on a particular product or brand through social media like Facebook and Twitter. If we try to make sense of this enormous data it could be very useful specifically for companies to take crucial marketing decisions.

Our aim is to explore the addition of a collaborative filter to filter noise in the sentiment estimates and also to extrapolate the sentiment to topics for which explicit tweets by the user are not available.

The main conclusion we derive is that although it is possible to predict and individual's taste based on his tweets we cannot do it very accurately. In this report, we describe in our procedure, related datasets, and performance results. While our procedure appears to be promising (and we plan further experimentation with real data), for the tweet data considered in this report, we do not find enough richness to see benefits from using the method.

TABLE OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

Introduction

Twitter is a popular social networking and microblogging service for sharing short messages called “tweets.” The tweets often reflect the opinions of individuals about different topics. Can we identify the opinion of an individual about a product or a company from his/her tweets? We explore this question here. The problem of assigning a positive or negative sentiment for a tweet has been investigated by others before - see for example TwitterSentiment [1]. The common approach is to apply language processing tools or to build a large training dataset and then use classification techniques from machine learning. Due to the weakness of language processing techniques and due to lack of large representative datasets, we can at best expect such sentiment classifiers to be noisy. Our aim is to explore the addition of a collaborative filter to filter noise in the sentiment estimates and also to extrapolate the sentiment to topics for which explicit tweets by the user are not available.

1.1 Sentiment Analysis

Sentiment analysis or opinion mining refers to the application of natural language processing, computational linguistics, and text analytics to identify and extract subjective information in source materials.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgement or evaluation, affective state (that is to say, the emotional state of the author when writing), or the intended emotional communication (that is to say, the emotional effect the author wishes to have on the reader).

A basic task in sentiment analysis is classifying the *polarity* of a given text at the document, sentence, or feature/aspect level — whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative, or neutral.

1.2 Collaborative Filtering

Collaborative filtering (CF) is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including sensing and monitoring data.

Collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the CF approach is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering or recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.[6]

CHAPTER 2

Tweets Collection

The first step was to have a tweet corpus large enough to make a good approximation of predicting people's opinions on a product/company and accurate enough to make a recommendation system. The tweets were collected using the twitter API and the twitter server was pinged for a maximum of 150 times in an hour and each request to the server collected a maximum of 20 tweets. We shortlisted 2475 users for analysis that were provided by GM(General Motors). For collecting, past six months tweets for large number of users using the twitter API constraint required the process to be done in parallel by 95 different servers on our side.

After finishing the tedious work of tweet collection, we realized that many users did not tweet often. Considering opinion of such users would only make the dataset spurious and would lead to false conclusions. Thus we decided to consider only those users who had at least 20 tweets in past 6 months and this turned out to be 1172 users.

The tweets were collected in a particular format so as to be useful in later analysis stage.

2.1 Tools

A python wrapper around the Twitter API was used for collection of tweets from users. The tweets were collected as text files with each for containing tweets for individual users over past six months. The tweets format was as follows:

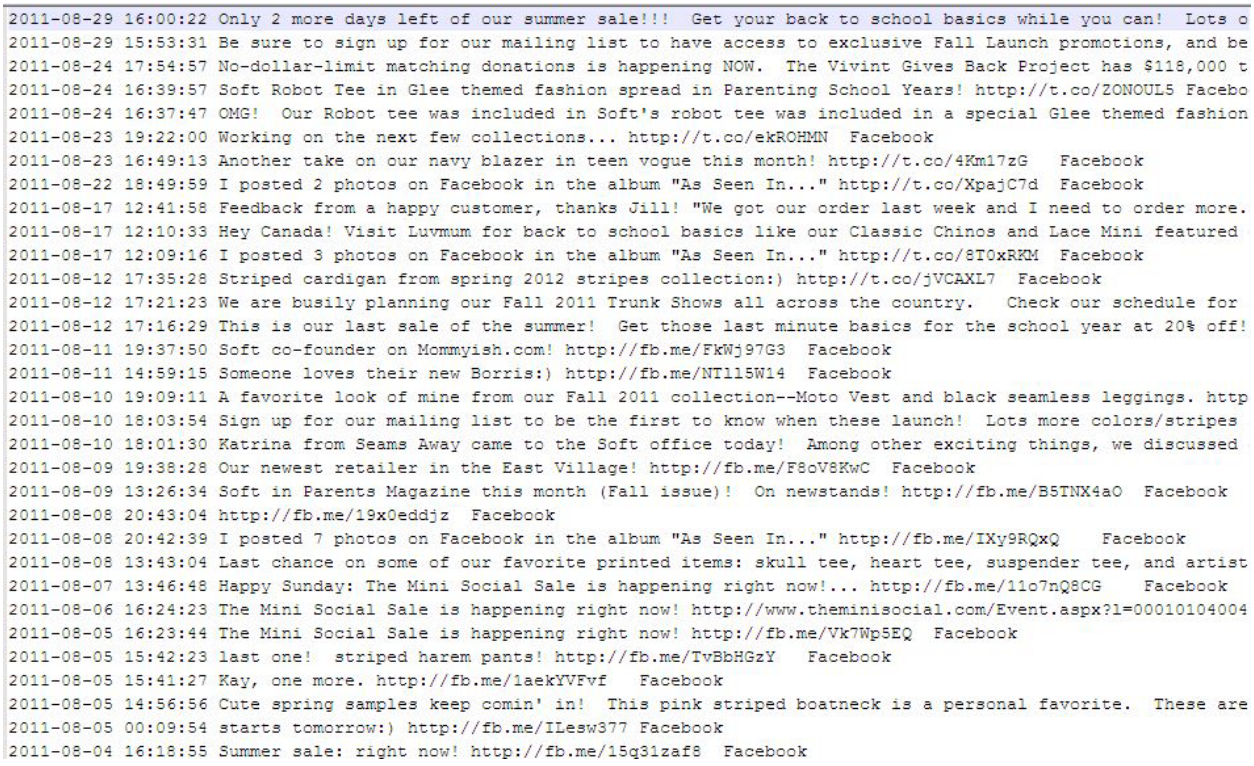

Fig. 2.1-1 Tweets Format

2.2 Short Listing Users:

Once the tweets for 1172 users were consolidated with each dedicated text file for a user containing all the individual's tweets, we plotted a cumulative density function graph of fraction of users v/s number of tweets as follows:

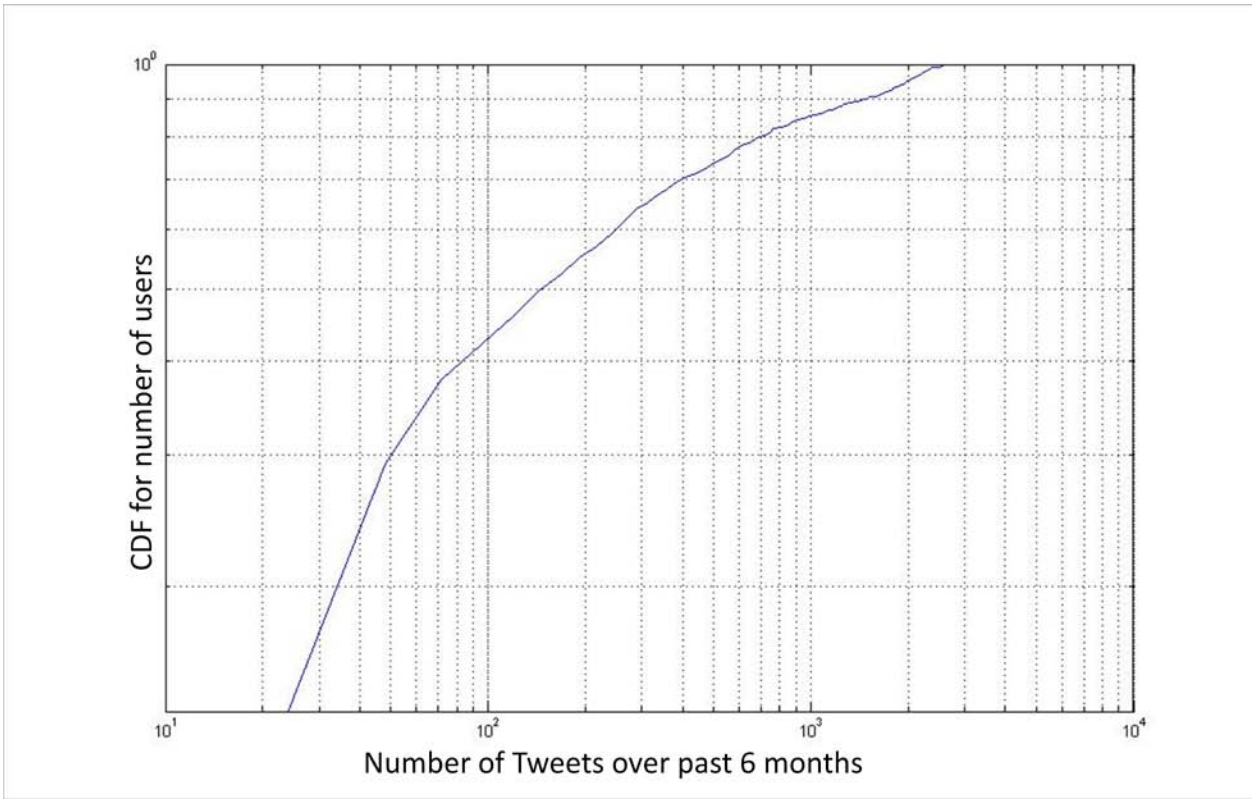

Fig. 2.2-1 Cumulative Density Function of number of Users

Looking at the graph, we can conclude that even lesser than 50 percent of the users have at least 180 tweets in the past six months, which is at least 1 tweet/day. This is necessary since we do not want the dataset to be spurious taking into account people's opinions who do not tweet much. Thus leading us to still cut down on the number of users. Finally, we decided on 572 users for our recommendation system and a tweet corpus of nearly 0.7 million tweets for these users.

CHAPTER 3

Tagging and Finding Important Nouns

3.1 Tagging Nouns

As part of our next step, we were supposed to find the most frequently occurring nouns that is company/ products, in the tweets. To do this, we needed to first tag parts of speech for each tweet. This was done using the Stanford Parts of Speech Tagger, which is a NLP tool. Once the nouns were tagged, we calculated the frequency of occurrence of each noun and finally, made a list of top thousand highly occurring nouns.

3.2 Finding Important Nouns

A plot of nouns that were tagged in the tweets and their frequencies is as follows:

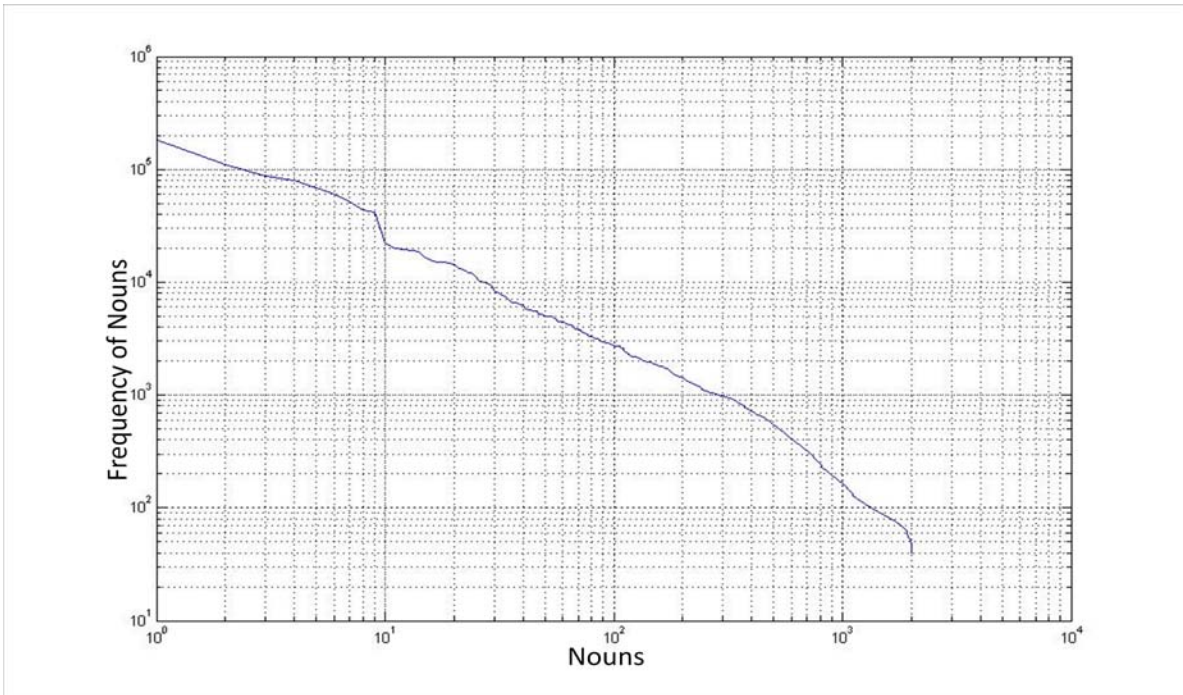

Fig. 3.2-1 Plot of nouns v/s their frequencies

From the above graph, we can conclude that only 300 nouns have a frequency of more than thousand which is significant enough for a noun in a 7 lac corpus. Once, the shortlisted nouns were consolidated, we faced another issue. A lot of these nouns were common nouns like cars, man, woman, child. These nouns are anything but useful in the recommendation system. They are too vague to understand a person's opinion on these nouns since knowing a individual's opinion on a car doesn't help us gauge his tastes on any product/company. Thus, we decided to ignore the common nouns and were eventually left with only 166 nouns of interest. Some of the nouns along with their frequencies are as follows:

Twitter	79778
iPhone	41335
Facebook	22340
Tweet	18734
BlackBerry	16531
foursquare	15476
Google	14617
Android	12183
iPad	9934
UberSocial	9665
Mac	8387
Silver	6313
Ford	5503
Pluggio	4934
GM	4707
Twidroyd	4507
Seesmic	4333
Detroit	4311
Instagram	4081
TrashCars	3190
Trend	3190
TweetGrid.com	2914
Tweetbot	2870
Obama	2787
TweetMeme	2770

Fig. 3.2-2 Sample nouns and corresponding frequencies

CHAPTER 4

Sentiment Classification

4.1 Training Dataset:

We train our classifiers using two datasets taken from different sources so as to cross check and validate that our tweets have indeed been classified correctly.

4.1.1 **First Dataset:** We used tweet data collected from followers of GM Blogs node. In particular, from all of its followers, we identified 86 followers who were most responsive to GM blogs in the recent past. (The responsiveness is measured in terms of the retweets in a fixed time duration; see [2, 3] for details.) We focus on the tweets written by these 86 followers in the time period of six months. For testing the sentiment classification methods, test data is constructed from these collected tweets. From these 86 followers, top 10 followers are selected based on the number of tweets written. For the test data, 400 tweets are randomly selected from all the tweets written by top 10 followers in a time period of six months. The ratio of the number of the tweets written by a follower and total number of tweets (all tweets written by top 10 followers) is computed. This ratio is almost same for all these top 10 followers. So, an equal number of tweets are randomly selected from the tweets of each follower. These 400 tweets are manually tagged as being positive, negative or neutral (depending on the sentiment expressed by the tweet).

4.1.2 **Second dataset:** For the training and testing the classifier, the tweet corpus from[1] is downloaded. It contains two datasets, one for training and one for testing. One needs to have the training dataset to be labeled (as positive or negative). Since the size of the data is very large, it is not feasible to label the data manually. For labeling the tweets, the creators of this dataset have used ‘emoticons’. Using the twitter API, the tweets with emoticons can be extracted. These emoticons can be used as labels (positive or negative). The training data contains tweets which are classified based on the emoticons. On the other hand, the test dataset contains manually labeled tweets.

We have used the following classifiers on the tweet data to classify it into two classes positive and negative.

- Naive-Bayes classifier
- Back-Propagation neural network.

4.2 Naïve-Bayes Classifier:

In our case, there are two classes, positive and negative. For a tweet d and a class c, we compute the conditional probability P(c|d). We assign the class with higher conditional probability to a tweet. The Bayes' Rule relates the conditional probabilities as follows:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

We have implemented the following algorithm to classify the tweets.

Algorithm:

C = The set of classes

D = The set of tweets

Training Phase:

V ← AllUnigramsInTraining(D)

N ← NumberOfTweets(D)

for each c € C do

N_c ← Number of tweets in Class c

prior (c) = $\frac{Nc}{N}$

$Tweets_c$ ← = All Tweets in class c

for each term t € V do

T_{ct} = No. of times term t appeared in $Tweets_c$

end for

for each term t € V do

prob[t][c] = $\frac{Tct}{\sum t' (Tct' + 1)}$

end for

end for

Test Phase:

Applying Naive-Bayes algorithm:

Consider a tweet from test data,

W← tokens from vocab V present in the tweet

for each c € C do

score[c] = log prior[c]

for each t € W do

score[c] = score[c] + log(prob[t][c])

end for

end for

return class c for which score is maximum

The training phase of the above algorithm was implemented using a initially tagged tweet corpus of 40K tweets from second dataset. The output of the training was the probabilities of nouns occurring in the positive class and negative class respectively.

After the training, our 0.7 million tweet corpus was given as an input and the algorithm was tested for classifying the tweets as positive or negative by calculating the overall score of all the nouns for each individual tweet.

4.3 Back-Propagation Neural Network

1. Definitions:

- the error signal for unit j
- the (negative) gradient for weight wij

- the set of nodes anterior to unit i
- the set of nodes posterior to unit j

2. The gradient. As we did for linear networks before, we expand the gradient into two factors by use of the chain rule:

$$\Delta w_{ij} = - \frac{\partial E}{\partial neti} \frac{\partial neti}{\partial wij}$$

The first factor is the error of unit i. The second is

$$\frac{\partial neti}{\partial wij} = \frac{\partial}{\partial wij} \sum w_{ik} y_k = y_j$$

Putting the two together, we get

$$\Delta w_{ij} = \partial_i y_j .$$

To compute this gradient, we thus need to know the activity and the error for all relevant nodes in the network.

3. Forward activation. The activity of the input units is determined by the network's external input x. For all other units, the activity is propagated forward:

$$y_i = f_i (\sum w_{ij} y_j)$$

Note that before the activity of unit i can be calculated, the activity of all its anterior nodes (forming the set Ai) must be known. Since feed forward networks do not contain cycles, there is an ordering of nodes from input to output that respects this condition.

4. Calculating output error. Assuming that we are using the sum-squared loss

$$E = \frac{1}{2} \sum (t_o - y_o)^2$$

the error for output unit o is simply

$$\partial_o = t_o - y_o$$

5. Error back propagation. For hidden units, we must propagate the error back from the output nodes (hence the name of the algorithm). Again using the chain rule, we can expand the error of a hidden unit in terms of its posterior nodes:

$$\partial_j = -\sum \frac{\partial E}{\partial neti} \frac{\partial net\, i}{\partial yj} \frac{\partial yj}{\partial netj}$$

Of the three factors inside the sum, the first is just the error of node i. The second is

$$\frac{\partial neti}{\partial yj} = \frac{\partial}{\partial yj} \sum wik\, yk = wij$$

while the third is the derivative of node j's activation function:

$$\frac{\partial yj}{\partial netj} = \frac{\partial fj\,(netj)}{\partial netj} = f'_j\,(net_j)$$

$$f'_h\,(net_h) = 1 - y_h^2$$

Putting all the pieces together we get

$$\partial_j = f'_j\,(net_j) \sum \partial_i\, w_{ij}$$

Note that in order to calculate the error for unit j, we must first know the error of all its posterior nodes (forming the set Pj). Again, as long as there are no cycles in the network, there is an ordering of nodes from the output back to the input that respects this condition. For example, we can simply use the reverse of the order in which activity was propagated forward.

4.4 Aggregate Sentiment of a Noun

Once we have classified each tweet into positive and negative classes with respect to the noun occurring in that tweet, we determine the overall sentiment for each noun from all the tweets put together. The following algorithm is used for determining the overall sentiment:

- For this purpose, top 50 nouns are selected from the manually labeled data containing 400 tweets (selected in section 2.1). The tweets are classified as positive or negative using the two classifiers (mentioned above).
- For each noun from top 50 nouns, the number of times the noun appears in positive tweets and the number of times the noun appears in negative tweets is computed. Frequency of a noun in the positive tweets and frequency of a noun in the negative tweets are computed as follows: frequency of noun in positive tweets (in negative tweets) = no. of positive tweets containing the noun (no. of negative tweets containing the noun) / total no. of tweets containing the nouns.
- Algorithm for finding the aggregate sentiment for a noun is given by,

  $\partial = 0.1$

  if fraction positive tweets - fraction negative tweets < ∂ then

  declare sentiment of the noun as negative

  else if fraction positive tweets - fraction negative tweets > ∂ then

  declare sentiment of the noun as positive

  else

  declare as can’t decide

  end if

- Each classifier declares the overall sentiment for each noun as positive or negative.

CHAPTER 5

User-Noun Matrix

In order to build a recommendation system, we construct a user-noun matrix as follows:
The rows correspond to the 572 users chosen as in Section 2.1. We select 166 nouns which occur more frequently in the tweets. The columns correspond to these nouns.

Algorithm :

matrix M
no. of rows = no. of users
no. of columns = no. of nouns
Initialize all elements of matrix m to zero
for each user u do
    for each tweet t do
        for each noun n do
            if tweet t contains noun n then
                if polarity of t is positive then
                M[u][n] = M[u][n] + (1/total no. of nouns present in tweet t)
                else if polarity of t is negative then
                M[u][n] = M[u][n] − (1/total no. of nouns present in tweet t)
                end if
            end if
        end for
    end for
end for

Observations:

In the resultant user-noun ratings matrix, since the matrix is sparse, the fraction of non zero entries is observed to be 0.2151. We quantize this matrix into +1, 0 and -1.The fraction of 1's is observed to be 0.7259. If we trust our sentiment classifier, then this implies that if a noun is popularly tweeted about, then it is highly likely that it has a positive buzz.

CHAPTER 6

Analysis and Results

6.1 Analysis

Once, we have the user noun matrix, the next step is to hide some of the values and check if our algorithm can predict the hidden values and calculate the error between the original and predicted values.

6.1.1 Singular Value Decomposition (SVD)

The singular value decomposition can be used to solve the low-rank matrix approximation problem. This is done as follows:

1. We traverse the original sparse user noun matrix of size 572x166 and whenever we encounter a nonzero element, we flip a coin which has probability of head as 0.3, whenever it is a head we reset the value to zero and continue this till the end of the matrix. In this manner we have a matrix with approximately 30 % hidden values.
2. We take the Singular Value Decomposition (SVD) of the matrix and obtain vector U,D,V where D is a diagonal matrix with 166 non zero values that are singular.
3. We take the diagonal values and reset some of the 166 values to zero, in steps of 10.Let us say we call these steps M.
4. We then regenerate the matrix which is the predicted matrix using U*D*V' for each of the new D.
5. In this way, we get a few approximations of the original matrix, wherein we try to reduce the error and fir the values to the observed values.
6. We repeat the above steps for quantized matrix where in the original matrix is replaced with +1 for >+3 , -1 for <-3 and 0 for the remaining values.

The graph of Relative Root Mean Square Error to the step M is as follows:

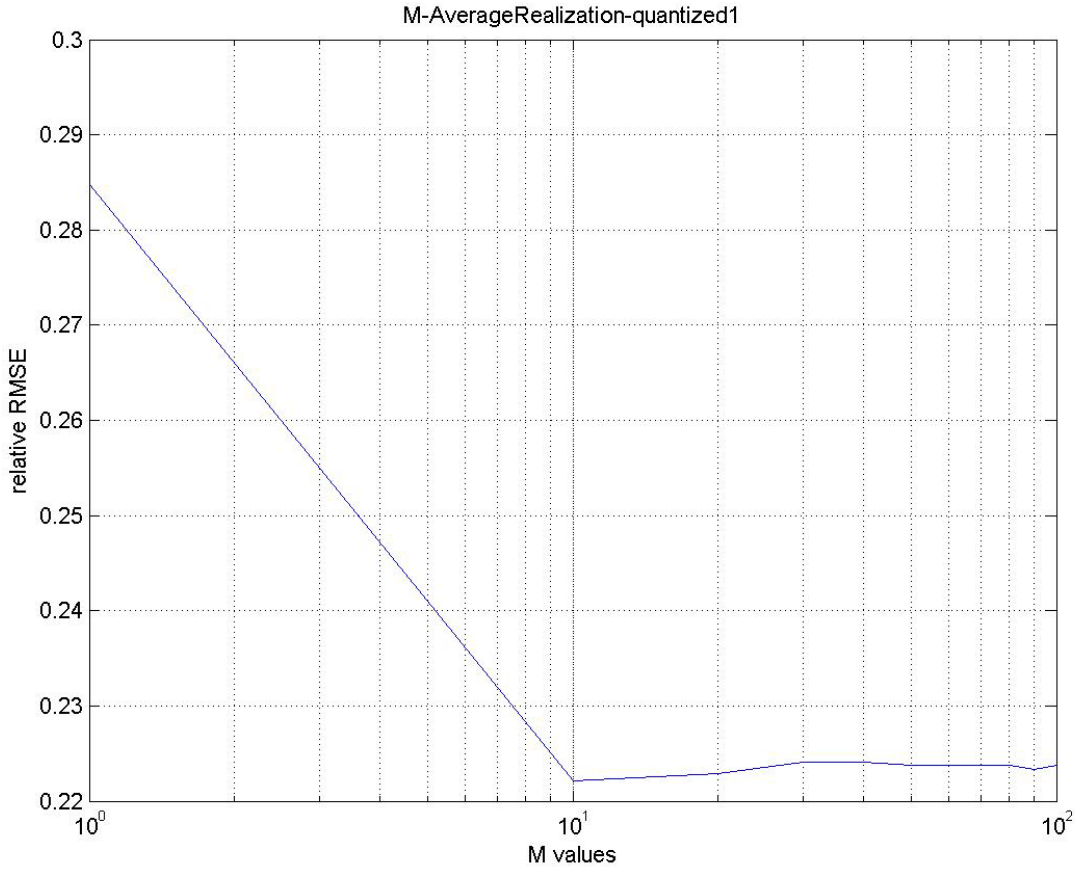

Fig. 6.1.1-1 RMSE v/s Diagonal steps M

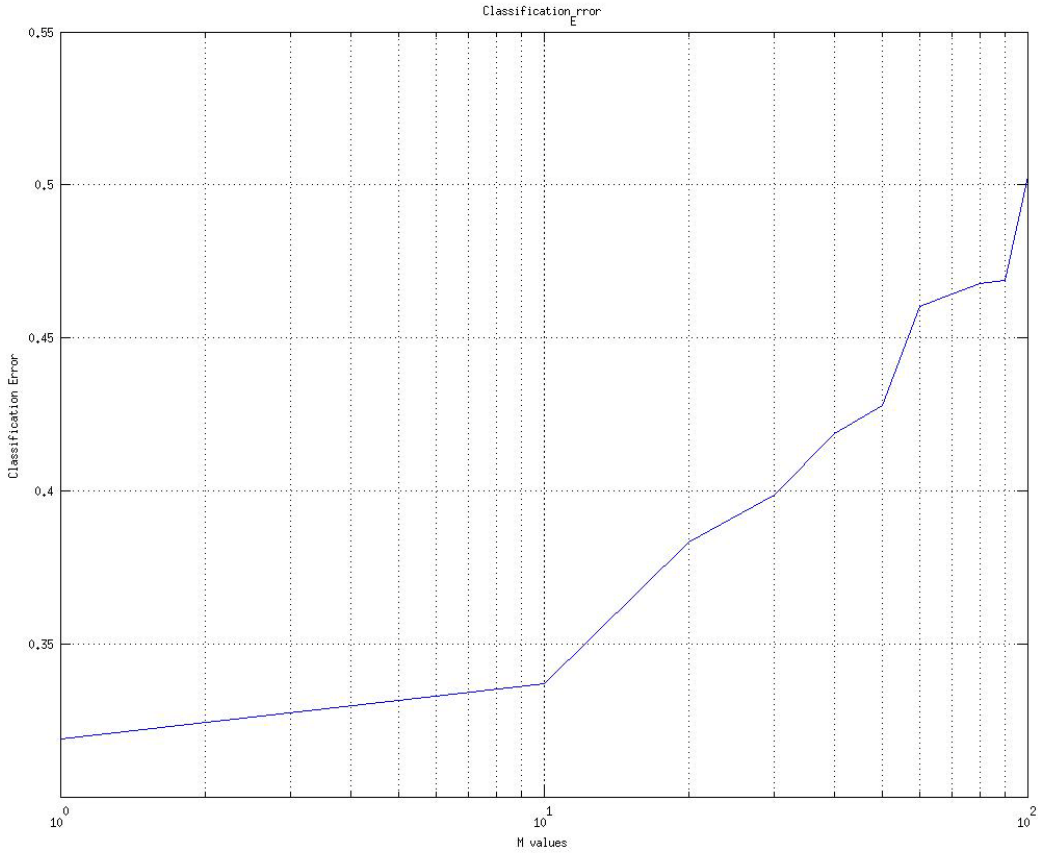

Fig. 6.1.1-2 Classification Error v/s Diagonal steps M

As we observe, the relative RMSE and classification error is very high even when we try to predict without altering the diagonal of matrix D. Thus, we decided to opt for another algorithm instead.

6.1.2 Singular Value Thresholding (SVT)

SVT is another way to obtain approximation of a Matrix and is more powerful than SVD and we used the algorithm developed by Stanford Statistics Department for SVT. However, although the algorithm gave a very good approximation for large matrices with normally distributed elements, it performed poorly for sparse matrices like the one we are using. Hence the algorithm failed to converge. Thus we had to use another algorithm.

6.1.3 OptSpace

This is an algorithm again developed by Stanford Statistics department. It is very powerful tool for approximating sparse matrices [5]. We obtained the following plots using this tool for approximating the user-noun matrix.

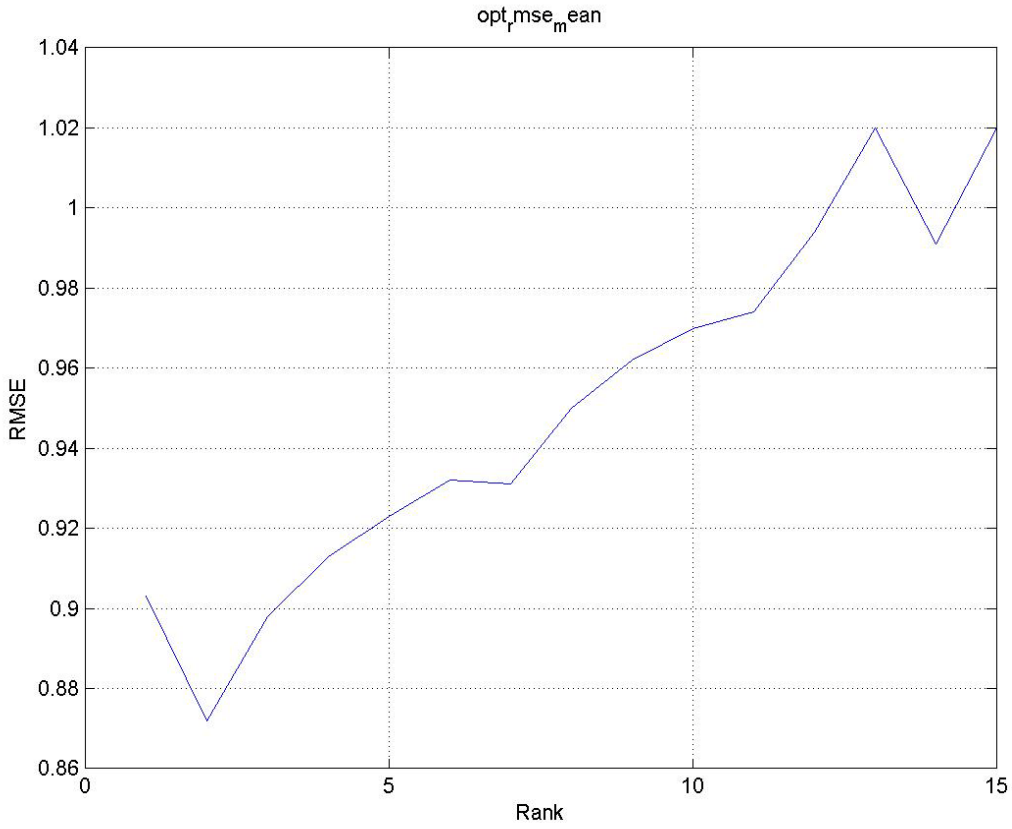

Fig. 6.1.3-1 RMSE v/s Rank of matrix

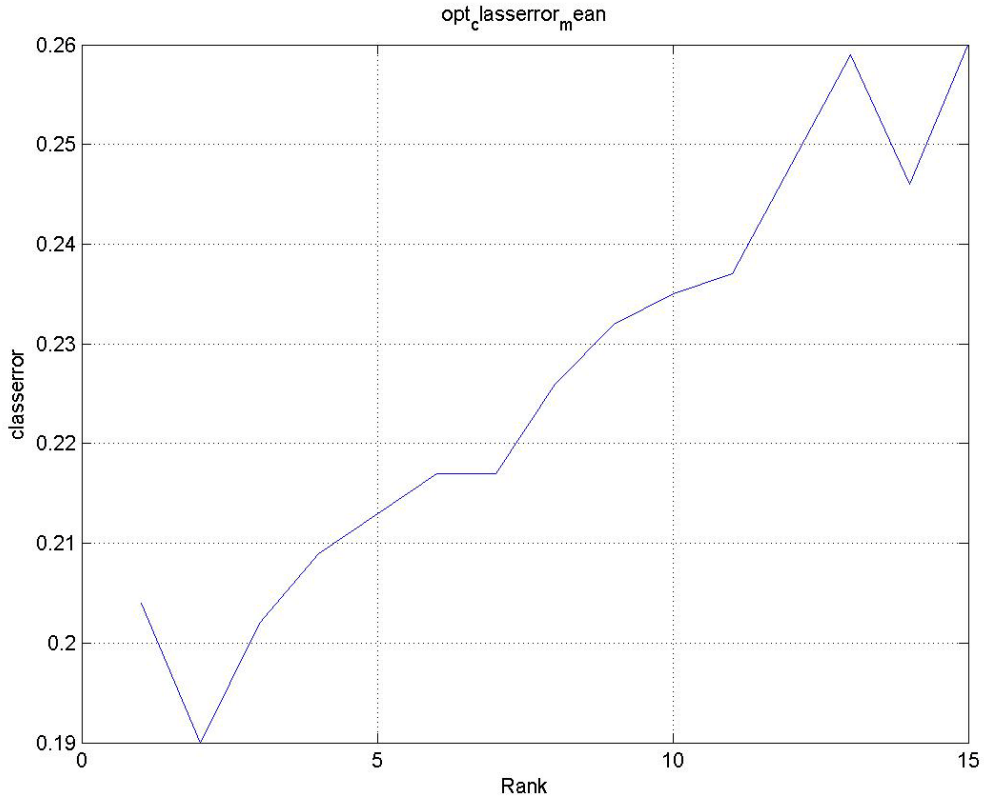

Fig. 6.1.3-2 Classification Error v/s Rank of Matrix

Thus we observe that predicting a lower rank approximation of the original matrix namely of rank 2 gives us minimum RMSE and Classification Error. We used this method to build the recommendation system which worked with high accuracy. The results below summarize the analysis.

6.2 Results

The results obtained using the OptSpace algorithm can be summarized as below. We have a quantized user-noun matrix (only 7 users and 12 nouns are shown). The upper part shows the original matrix and the latter shows the low rank approximation of the matrix.

Twitter	iPhone	Facebook	BlackBerry	foursquare	Google	Android	iPad	UberSocial	Mac	Silver	Ford	F
0	0	0	0	0	0	0	1	0	0	0	1	
1	-1	1	0	0	0	0	-1	0	-1	0	0	
1	1	1	0	1	0	0	1	0	1	0	0	
0	1	1	0	1	0	0	1	0	-1	0	0	
1	0	1	0	0	0	0	1	0	0	0	0	
1	-1	0	0	0	1	0	1	0	0	0	1	
0	-1	1	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	1	0	0	0	1	
1	-1	1	0	0	0	0	-1	0	-1	0	0	
1	1	1	0	1	0	0	1	0	1	0	0	
0	1	0	0	1	0	0	0	0	-1	0	0	
1	0	1	0	0	0	0	0	0	0	0	0	
0	-1	0	0	0	1	0	1	0	0	0	1	
0	-1	0	0	0	0	0	0	0	0	0	0	

Fig. 6.2-1 Results

From the results above it can be proven that there is about 19 % misclassification which justifies our analysis of the OptSpace algorithm.

7. Conclusions

This project has provided us the opportunity to understand the buzz that social media has created. The fact that what we tweet can provide a great deal about our individual opinion and our taste fascinates me.

Building a recommendation purely on basis of an individual's tweets is a challenging task and we have been successful in approximating 80% of the times using OptSpace method for approximation as shown by the results. We wish to continue exploring and analyzing these results in further depth.

While our procedure appears to be promising (and we plan further experimentation with real data), for the tweet data considered in this report, we need to explore the feasibility of its implementation in deploying to building a real world recommendation system. The main conclusion we derive based on the data considered is that if a topic is popularly tweeted about, then it is very likely that the sentiment associated with it is positive. Another conclusion we can make is that sentiment classification of tweets is not very accurate particularly for negative tweets.

8. References

[1] http://twittersentiment.appspot.com/

[2] O. Dabeer, “Scheduling and Composing Tweets,” Report submitted to GM in Aug 2010

[3] O. Dabeer, P. Mehendale, A. Karnik, A. Saroop, “Timing Tweets to Increase Effectiveness of Information Campaigns,” Intl. Conf. on Weblogs and Social Media, July 2011

[4] http://nlp.stanford.edu/software/tagger.shtml

[5] R. H. Keshavan, S. Oh, and A. Montanari, “Matrix completion from a few entries,” CoRR, vol. abs/0901.3150, 2009; http://www.stanford.edu/ raghuram/optspace/

[6] Prem Melville and Vikas Sindhwani. “Recommender Systems.,” In Encyclopedia of Machine Learning, Claude Sammut and Geoffrey Webb (Eds), Springer, 2010.