Grenze International Journal of Engineering and Technology, July Issue

Tweets Sentiment Analysis using Naive Bayes Classifier

Saroj Kumar[1], Santosh Kumar[2] and Bhawana Chaudhary[3]
[1-3]Maharishi University of Information Technology - Lucknow, Uttar Pradesh, India
Email: saroj.kumar999@gmail.com, sant7783@hotmail.com and rcbhawana844@gmail.com

***Abstract*—Twitter is a celebrated miniaturized scale running on the web administration to compose a blog or remark that place client's impact status messages (called "tweets"). These tweets now and again explicit feelings contacting one of kind points. The thought process in regards to that request is in similarity with construct a calculation so that execute definitely. Twitter messages might be in positive or negative, along regard as per an inquiry term or according to disposition of client. Our hypothesis is to that sum we perform accomplishing unnecessary respectability over characterizing feeling of Twitter messages by the utilization of AI systems. Gullible Bayes Classifier in particular that is a typical classifier with capacities in Natural Language Processing (NLP). Regardless of its straightforwardness, that is capable as per get over normal execution of unmistakable obligations like sense investigation. We expound thoughts in regards to it model or beneath put into impact it inside Python. By and large, this sort over slant investigation is useful in light of the fact that purchasers whosoever are endeavoring in congruity with inquire about an item or administration, yet advertisers picking up information on masses decision for theirs organization.**

***Index Terms*— Naive Bayes, Tweets, Natural Language Processing, Entropy, Bigrams, Semantics, OpenNLP.**

I. INTRODUCTION

Assessments are contemplations which are come in anyone's psyche without a moment's delay. So these are most alterable marvels of any ones days. That why assessments investigation is critical piece of any system or techniques to catch the example of notion. Assumption are sticks according to enormous time factor risking, presently someone's slant are gotten basic like in the event that he search 10 melodies in YOUTUBE, at that point at prepared 20 to 30 % information will coordinate with his last hunt and his ordinary pursuit.
Here I am utilizing Naives Bayes to investigation suppositions. As suppositions are consistently produced antiques. Credulous Bayes given a class c, the nearness of an individual component of our record is autonomous on the others.

A. Defining Sentiment

For the reasons for our examination, we characterize assessment to be "an individual positive or negative inclination." Here are a few models:
On the off chance that we have a few tweets and not satisfactory cut about their tendency, at that point we

Grenze ID: 01.GIJET.6.2.507

TABLE I. EXAMPLE SENTIMENT AND TWEET

Sentiment	Tweet
Positive	Ramesh: Rani is my new best friend.
Neutral	Rani: I know Ramesh
Negative	Rahul: Friendship with Rani is Very Difficult.

can utilize the accompanying litmus test: If the tweet would ever show up as a paper title text or as a sentence in Wikipedia, at that point it has a place in the unbiased class. For instance, the accompanying tweet would be set apart as impartial in light of the fact that it is reality from a paper title text, despite the fact that it anticipates a general negative inclination about GM.

For examining estimation we use here Naive Bayes Classifier. Fundamental math documentations which are utilizing here for effectively arrange a survey as positive or negative. These are the two classes to which each archive has a place. In increasingly scientific terms, we need to locate the most likely class given a record, which is actually what the recipe passes on.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

C is the set of all possible classes, **c** one of these classes and **d** the document that we are currently classifying. We read **P(c|d)** as the probability of class c, given document d. We can rewrite this equation using the well known Bayes' Rule, one of the most fundamental rules in machine learning. Since we want to maximize the equation we can drop the denominator, which doesn't depend on class c.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

The changed type of our classifier's objective normally parts it into two sections, the probability and the earlier. You can think about the last as "the likelihood that given a class c, record d has a place with it" and the previous as "the likelihood of having an archive from class c". To go above and beyond we have to present the supposition that gives this model its name.

Naive Bayes sup-position: given a class c, the nearness of an individual element of our archive is free on the others.

We think about every individual expression of our report to be a component. On the off chance that we compose this officially we acquire:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

The Naive Bayes assumption lets us substitute P(d|c) by the product of the probability of each feature conditioned on the class because it assumes their independence.

We can roll out one greater improvement: augment the log of our capacity. The explanation behind this is absolutely computational, since the log space will in general be less inclined to sub-current and increasingly productive. We show up at the last detailing of the objective of the classifier.

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i|c)$$

So how exactly does this reformulation help us? Let's look at each term individually.

- **P(c)** is just the likelihood of experiencing a record of a specific class inside our corpus. This is effortlessly determined by simply partitioning the quantity of events of class c by the complete number of archives.
- **P(w_i|c)** is the likelihood of word w_i happening in an archive of class c. Again we can utilize the frequencies in our corpus to register this. This will basically be the occasions word w_i happens in records of class c, isolated by the whole of the checks of each word that shows up in archives of class c.

We can figure all the terms in our plan, implying that we can ascertain the most probable class of our test archive! There is just one issue that we have to manage: zero probabilities.

B. Smoothing

Envision that we are attempting to arrange a survey that contains the word 'dynamite' and that our classifier hasn't seen this word previously. Normally, the likelihood P(wi|c) will be 0, causing the second term of our condition to go to negative endlessness! This is a typical issue in NLP yet fortunately it has a simple fix with smoothing. This system comprises in adding a consistent to each include in the P(wi|c) equation, with the most fundamental sort of smoothing being called include one (Laplace) smoothing, where the steady is only 1.

$$\hat{P}(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w\in V}(count(w,c)+1)} = \frac{count(w_i,c)+1}{\left(\sum_{w\in V} count(w,c)\right)+|V|}$$

This solves the zero probabilities problem and we will see later just how much it impacts the accuracy of our model.

II. PROCEDURE -DATA COLLECTION

Information assortment for the examination isn't as straightforward as it might appear at extreme idea. There are suppositions and choices to be made. There are three diversely gathered datasets: test information, emotional preparing information, and goal (unbiased) preparing information.

There are no current informational collections of Twitter assumption messages. We gathered our own arrangement of information. For the preparation information, we gathered messages that contained the emojis :) and :(by means of the Twitter API.

The test information was physically. A lot of 75 negative tweets and 108 positive tweets were physically stamped. A web interface instrument was worked to help in the manual characterization task.

A. Twitter API

Twitter gives two APIs: REST and Streaming. REST API comprises of two APIs: one just called the REST API and another called SEARCH API. The contrast between Streaming API and REST APIs are: Streaming API underpins seemingly perpetual association and gives information in practically continuous.

B. Training Data

There are two datasets that are utilized for the preparation of a classifier: abstract information and nonpartisan information. Abstract information is information that includes positive and additionally negative conclusion while unbiased information is information that doesn't show estimation. The accompanying information was gathered to be utilized to prepare a classifier

C. Subjective Data Collection

Abstract information right now information that contains negative or potentially positive emojis. While it is conceivable to gather enough negative and positive information in a couple of continuous days, the abstract information was gathered in four non-back to back days to empower irregularity. Likewise I chose to gather negative and positive tweets simultaneously as opposed to gathering them in various days. Gathering them simultaneously was viewed as acceptable in light of the fact that it can catch some unobtrusive contrasts among negative and positive Twitter posts for a few.

D. Neutral tweets

For unbiased tweets, I gathered tweets from Twitter records of 20 significant papers, for example, Times of India, The Hindu, Dainik Jagrans and others that a news title text is an impartial post. The unbiased tweets were gathered from Twitter spilling API. They were gathered on days unique in relation to when the emotional information was gathered.

We will execute our classifier as a Naive Bayes Classifier class. We will part the calculation into two fundamental parts, the preparation and characterizing.

III. TRAINING

Right now, give our classifier an (ideally) enormous corpus of content, indicated as D, which processes all the tallies important to figure the two terms of the reformulated.

for each class $c \in C$ # Calculate $P(c)$ terms
  N_{doc} = number of documents in D
  N_c = number of documents from D in class c
  $logprior[c] \leftarrow \log \frac{N_c}{N_{doc}}$
  $V \leftarrow$ vocabulary of D
  $bigdoc[c] \leftarrow$ **append**(d) **for** d ∈ D **with** class c
  for each word w in V # Calculate $P(w|c)$ terms
    $count(w,c) \leftarrow$ # of occurrences of w in $bigdoc[c]$
    $loglikelihood[w,c] \leftarrow \log \frac{count(w,c) + 1}{\sum_{w' \text{ in } V} (count(w',c) + 1)}$
return $logprior$, $loglikelihood$, V

Program 1 - Pseudocode for Naive Bayes training

While actualizing, despite the fact that the pseudocode begins with a circle once again all classes, we will start by figuring everything that doesn't rely upon class c before the circle. This is the situation for N_doc, the jargon and the arrangement all things considered.

Within the loop we just follow the order as given in the pseudocode.

1. First, we check the quantity of archives from D in class c.

2. Then we figure the logprior for that specific class.

3. Next, we cause a circle over our jargon so we to can get an all out mean the quantity of words inside class c.

4. Finally, we figure the log-probabilities of each word for class c utilizing smoothing to evade division-by-zero mistakes

A. Classifiers

Several different classifiers were used. A Naive Bayes classifier was built from scratch. Third-party libraries were used for Maximum Entropy and Support Vector Machines. The following table summarizes the results.

TABLE VI. ACCURACY RESULTS FROM VARIOUS CLASSIFIERS

Accuracy	Keyword	Multinomial Naive Bayes Unigram	Multinomial Naive Bayes Unigram using MI	MaxEnt OpenNlp	MaxEnt Stanford Classifier (Sigma= 10)
Unigram	0.42622950819672	0.80874316939891	0.84153005464481	79.23	0.61
Unigram Limited with feature threshold = 100	-	0.80874316939891	-	-	0.63
Unigram + POS	-	0.74863387978142	0.78142076502732	-	0.683
Bigram	-	0.75956284153005	0.73770491803279	0.7322	0.667

Training size also has an effect on performance. Figure 1 shows the effect of training size on accuracy.

When the training is done we have all the necessary values to make a prediction. This will simply consist in taking a new (unseen) document and computing the probabilities for each class that has been observed during training.

for each class $c \in C$
  $sum[c] \leftarrow logprior[c]$
  for each position i in $testdoc$
    $word \leftarrow testdoc[i]$
    if $word \in V$
      $sum[c] \leftarrow sum[c] + loglikelihood[word,c]$
return $\text{argmax}_c\ sum[c]$

Pseudocode for the classification part

IV. NAIVE BAYES

Naive Bayes is a simple model for classification. It is simple and works well on text categoration. We adopt multinomial Naive Bayes in our project. It assumes each feature is conditional independent to other features

$$P(c \mid t) = \frac{P(c)P(t \mid c)}{p(t)}$$

given the class. That is,
where c is a particular class and t is content we need to arrange. P(c) and P(t) is the earlier probabilities of this class and this content. Also, P(t | c) is the likelihood the content seems given this class. For our situation, the estimation of class c may be POSITIVE or NEGATIVE, and t is only a sentence..
The objective is picking estimation of c to expand P(c | t):
Where P(wi | c) is the likelihood of the ith include in content t seems given class c. We have to prepare parameters of P(c) and P(wi | c). It is basic for getting these parameters in Naive Bayes model. They are simply greatest probability estimation (MLE) of every one. When making expectation to another sentence t, we figure the log probability log P(c) +
∑ilogP(wi|c) of various classes, and take the class with most noteworthy log probability as forecast.
By and by, it needs smoothing to keep away from zero probabilities. Something else, the probability will be 0 if there is a concealed word when it making expectation. We just use include 1 smoothing in our task and it functions admirably.

Feature selection
For unigram highlight, there are normally 260,000 distinct highlights. This is a huge number. It makes model higher fluctuation. (Since progressively convoluted model has higher difference). So it will require significantly more preparing information to abstain from over fitting. Our preparation set contains many thousands sentences. Be that as it may, it is as yet countless highlights for our preparation set. It is useful on the off chance that we dispose of some futile highlights. We attempt 3 distinctive component choice calculations.

Frequency-based feature selection
This is the easiest method to do highlight choice. We simply pick highlights (unigram words for our situation) for each class with high recurrence event right now. By and by, if the quantity of events of an element is bigger than some edge (3 or 100 in our analyses), this component is a decent one for that class. As we found in the outcome table, this basically calculation increments about 0.03 of exactness.

Mutual Information
The possibility of shared data is, for each class C and each component F, there is a score to quantify the amount F could add to settling on right choice on class C.

$$MI(C;F) = \sum_{ef \in \{1,0\}} \sum_{ec \in \{1,0\}} P(C = ec, F = ef) \log \frac{P(C = ec, F = ef)}{P(C = ec)P(F = ef)}$$

In practice, we also use add-1 smoothing for each Count(C = ec, F = ef) to avoid divided by zero. The code is below.

```
double n = polarityAndFeatureCount.totalCount() + 4;
for(String feature: featureCount.keySet())
{
for(int polarity : polarityCount.keySet())
{double n11 = polarityAndFeatureCount.getCount(polarity, feature) + 1; double n01 = polarityCount.getCount(polarity) -polarityAndFeatureCount.getCount(polarity, feature) + 1; double n10 = featureCount.getCount(feature) - polarityAndFeatureCount.getCount(polarity, feature) + 1; double n 00 = n - (n11 + n01 + n10); double n1dot = n11 + n10; double n0dot = n - n1dot; double ndot1 = n11 + n01; double ndot0 = n - ndot1;
double miScore =
(n11 / n) * Math.log((n * n11) / (n1dot * ndot1))+ (n01 / n) * Math.log((n * n01) / (n0dot * ndot1))+ (n10 / n) * Math.log((n * n10) / (n1dot * ndot0))+ (n00 / n) * Math.log((n * n00) / (n0dot * ndot0));
```

$$\chi^2(F,C) = \frac{N(N_{11}N_{00} - N_{10}N_{01})^2}{(N_{11} + N_{01})(N_{11} + N_{10})(N_{10} + N_{00})(N_{01} + N_{00})}$$

```
mi.setCount(polarity, feature, miScore);
}}
```

Subsequent to figuring MI score, just top k highlights with most elevated scores will be picked for include set to test. We can see that if k is little, the model is too basic that information is under fitting. In any case, if k is enormous, the model is too entangled that information is over fitting. The best number of highlights in our unigram case is around 40,000. As k grow up to 20,000, the exactness and F score are additionally grow up rapidly. This is on the grounds that right now, model is high inclination. So it is useful to add highlights to dodge under fitting information. At the point when the number is bigger than 100,000, the precision and F score decline step by step. Since the huge number of highlights makes model so confounded that there are insufficient preparing sentence to maintain a strategic distance from over fitting.

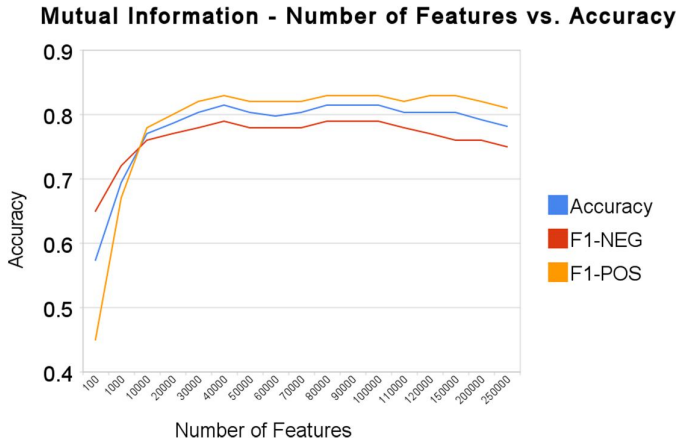

Figure 2 - Mutual Information - Number of Features vs. Accuracy [3]

X^2 Feature Selection

The possibility of X2 Feature determination is comparable as common data. For each component and class, there is likewise a score to gauge if the element and the class are autonomous to one another. It utilizes X2 test, which is a measurement strategy to check if two occasions are free. It expect the element and class are free and ascertains X2 esteem. The enormous score suggests they are not autonomous. For instance, the basic estimation of 0.001 is 10.83. This implies, on the off chance that they are free to one another, at that point the likelihood this score bigger than 10.83 is just 0.001. On the other hand, in the event that the score is bigger than 10.83, at that point it is far-fetched the component and the class free. The bigger the score is, the higher reliance they have. So we need save highlights for every class with most noteworthy X2 scores.

A. Maximum Entropy

The thought behind MaxEnt classifiers is that we ought to lean toward the most uniform models that fulfill any given imperative. MaxEnt models are highlight based models. We utilize these highlights to discover a circulation over the various classes utilizing strategic relapse. The likelihood of a specific information direct having a place toward a specific class is determined as follows:

Where, c is the class, d is the data point we are looking at, and λ is a weight vector.

MaxEnt makes no independence assumptions for its features, unlike Naïve Bayes. This means we can add features like bigrams and phrases to MaxEnt without worrying about feature overlapping.

$$p(c \mid d, \vec{\lambda}) = \frac{\exp\left[\sum_i \lambda_i f_i(c, d)\right]}{\sum_{c'} \exp\left[\sum_i \lambda_i f_i(c', d)\right]}$$

We tried using two packages for the MaxEnt implementation: the Stanford Classifier and the OpenNLP package.

B. Performance

The Stanford Classifier bundle gave awful outcomes for the default parameter settings. Over various preparing sizes (Figure 1) it improved a piece, however was a ton more terrible than different classifiers. We changed the smoothing constants, however it never got near the NB classifier as far as exactness. As appeared in Figure 3, changed sigma (smoothing) values didn't contribute a lot to higher exactness.

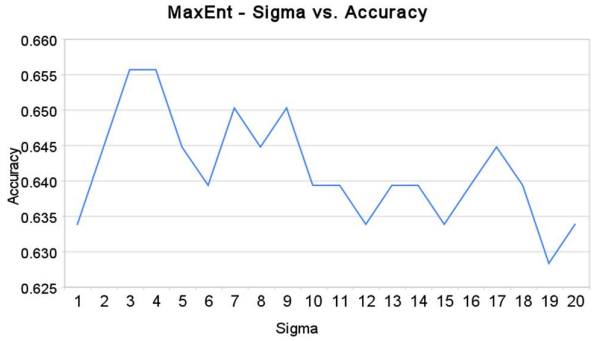

Figure 3. Sigma (the smoothing parameter) vs accuracy [7]

Basic unigram feature selection for NB	Presence	Frequency-based(3)	Frequency-base(Presence, 100)	MI (Presence, 40000)	Chi2 (Presence, 40000)
Accuracy	0.7650273224043	0.78142076502732	0.79781420765027	0.81420765027322	0.80327868852459
F1-NEG	0.73	0.75	0.79	0.79	0.78
F1-POS	0.79	0.81	0.81	0.83	0.83

After testing for different smoothing values and trying different functions in place of ConjugateDescent, we decided to try OpenNLP's MaxEnt classifier since time was running short.

MaxEnt from OpenNLP improved. As should be obvious from Figure 1, MaxEnt performs like how the NB performs. Since it doesn't altogether improve execution and takes long to prepare and test, we chose to seek after NB for some different trials.

V. Naive Bayes Error Analysis

Example 1

Naive Bayes's independence assumption sometimes causes havoc in classification. This is most notable for negative words like "not" that precede adjectives. Here is an example:

As u may have noticed, not too happy about the GM situation, nor AIG, Lehman, et al

The actual sentiment is negative, but Naive Bayes predicted positive. The unigram model has a probability on the word "happy" for the positive class, which doesn't take into account the negative word "not" before it.

Example 2

In some cases, our language model was simply not rich enough. For example, the Naive Bayes classifier failed on the following example:

Cheney and Bush are the real culprits - http://fwix.com/article/939496

The real feeling is negative, yet the Naive Bayes classifier anticipated positive. The explanation is that "guilty parties" just happened once in our preparation information, as a positive estimation. We thought stemming words may help in light of the fact that "guilty party" shows up in the preparation corpus: 1 time in the positive class and multiple times in the negative class. We attempted the Porter Stemmer in the unigram highlight extractor to help with this circumstance, however it wound up cutting down by and large precision by 3%.

VI. Result and Improvements

Now that is some accuracy! Smoothing makes our model good enough to correctly classify at least 4 out of 5 reviews, a very nice result. We also see that training and predicting both together take at most 1 second which is a relatively low runtime for a dataset with 2000 reviews.

Presently that is some exactness! Smoothing makes our model adequate to effectively arrange at any rate 4 out of 5 surveys, an exceptionally pleasant outcome. We additionally observe that preparation and foreseeing both together take all things considered 1 second which is a moderately low runtime for a dataset with 2000 audits.

Following Improvements can be made for future.

a. Our calculations order the general notion of a tweet. Contingent upon whose point of view you're seeing the tweet from the extremity may change.
b. Part of Speech (POS) tagger - The POS tagger took around 3 hours to prepare and henceforth we was unable to run such a large number of tests on it. It improved the precision if there should arise an occurrence of Maxent and could have been useful to NB with some more varieties yet we needed more time to lead these tests.
c. Domain-explicit tweets - Our classifiers produce around 85% precision for tweets over all spaces. This implies an incredibly huge jargon size. Whenever constrained to specific spaces, (for example, films) we feel our classifiers would perform surprisingly better.
d. Support Vector Machines - SVM played out the best while arranging film audits as positive or negative. A significant following stage is further investigate SVM parameters for ordering tweets.
e. Handling impartial tweets - In genuine applications, unbiased tweets can't just be disregarded. Appropriate consideration should be paid to impartial slant. There are a few methodologies that utilization a POS tagger to see descriptive words to decide whether a tweet contains an estimation.
f. Dealing with words like "not" fittingly Negative words like "not" have the mystical effect of turning around extremity. Our present classifier doesn't deal with this well overall.
g. Ensemble techniques A solitary classifier may not be the best methodology. It is intriguing to perceive what the outcomes are for joining various classifiers. For instance, we contemplated utilizing a blend model among unigrams and bigrams. Progressively refined troupe techniques, such as boosting, could be utilized.
h. Using cleaner preparing information. Our preparation information doesn't have the cleanest marks. The emojis fill in as a boisterous mark. There are a few cases wherein the emoji mark would regularly not sound good to a human evaluator. For instance client ayakyl tweeted, "agghhhh :) looosing my mind!!!!" If we expel the emoji from this expression, it becomes "agghhhh looosing my mind!!!!" in which a human evaluator would regularly survey as negative.

VII. Conclusion

As should have been obvious, even a fundamental execution of the Naive Bayes calculation can prompt shockingly great outcomes for the undertaking of supposition examination. Notice that this model is basically a twofold classifier, implying that it very well may be applied to any dataset in which we have two classifications. There are a wide range of uses for it, going from spam recognition to Bitcoin exchanging dependent on assessment. With a precision of 82%, there is actually a ton that you could do, all you need is a named dataset and obviously, the bigger it is, the better.

REFERENCES

[1] Saroj Kumar, Santosh Kumar, "World Wide Web - Cloud Boundaries", International Journal of Computer Sciences and Engineering, Vol.7, Issue.6, pp.483-490, 2019.

[2] Saroj Kumar,Ankit Kumar Singh,Priya Singh,Abdul Mutalib Khan,Vibhor Agrawal Mohd Saif Wajid, "Sentiment Analysis Based on A.I. Over Big Data", Publisher Name Springer, Singapore, Print ISBN978-981-10-1677-6, Online ISBN978-981-10-1678-3, https://link.springer.com/chapter/10.1007/978-981-10-1678-3_61.

[3] Saroj Kumar, Priya Singh, Shadab Siddiqui "Cloud security based on IaaS model prospective" **IEEE Xplore, Digital Library, INSPEC Accession Number:** 15110011, 04 May 2015, http://ieeexplore.ieee.org/document/7100623.

[4] Ankit Kumar Singh, Saroj Kumar and Abhishek Rai "Secure Cloud Architecture Based on YAK and ECC" International Journal of Computer Applications, ISBN 0975 – 8887 Volume 90, Number 19 (March 2014), pp. 29–33, © International Journal For Computer Application, http://www.ijca.org.

[5] Ankit Kumar Singh, Saroj Kumar and Abhishek Rai " CLOUD SERVICE ARCHITECTURE FOR EDUCATION SYSTEM UNDER OBJECT ORIENTED METHODOLOGY" International Journal of Research in Engineering and Technology, eISSN: 2319-1163 | pISSN: 2321-7308 Volume: 03 Special Issue: 10 | NCCOTII 2014 | Jun-2014, Available @ http://www.ijret.org.

[6] Akhilesh Kumar, Saroj Kumar " ENERGY SAVING MODEL AND APPLICATION FOR SMART PHONES" International Journal of Research in Engineering and Technology, eISSN: 2319-1163 | pISSN: 2321-7308 Volume: 03 Special Issue: 10 | NCCOTII 2014 | Jun-2014, Available @ http://www.ijret.org .

[7] Ashutosh Gaur, Saroj Kumar " PROPOSED MAC PROTOCOL FOR REDUCE ENERGY CONSUMPTION OVER WSN NETWORK" International Journal of Research in Engineering and Technology, eISSN: 2319-1163 | pISSN: 2321-7308 Volume: 03 Special Issue: 10 | NCCOTII 2014 | Jun-2014, Available @ http://www.ijret.org

[8] Saroj Kumar, and Priya Singh "Cloud Computing Applications Architecture - ArchJava" International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2, Issue 6, June 2012, www.ijetae.com.

[9] B.Jansen, M. Zhang, K. Sobel, A. Chowdury. The Commerical Impact of Social Mediating Technologies: Micro-blogging as Online Word-of-Mouth Branding, 2009.

[10] B.Manning and H. Schuetze. Foundations of Statistical Natural Language Processing. 1999.

[11] B. Pang, L. Lee, S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques, 2002.

[12] B. Pang and L. Lee. "Opinion Mining and Sentiment Analysis" in Foundations and Trends in Information Retrieval, 2008.

[13] B. Pang and L. Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts" in Proceedings of ACL, 2004.

[14] J. Read. Using Emotions to Reduce Dependency in Machine Learning Techniques for Sentiment Classification, 2005.

[15] P. Turney. "Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews" in Proceedings of the 40th Annual Meeting of the Association for Computatoinal Linguistics (ACL), 2002.

[16] S. Li, Z. Wang, G. Zhou, and S. Y. M. Lee. Semi-Supervised Learning for Imbalanced Sentiment Classification. In Proc. IJCAI, pages 1826- 1831, 2011.

[17] Y. Saez, C. Navarro, A. Mochón, and P. Isasi. A System for Personality and Happiness Detection. IJIMAI, 2:7-15, 2014.

[18] H. Cordobés, A. Fernández-Anta, L. F. Chiroque, F. Pérez, T. Redondo, and A. Santos. Graph-based Techniques for Topic Classification of Tweets in Spanish. IJIMAI, 2:31-37, 2014.

[19] A. N. Jebaseeli and E. Kirubakaran. Genetic Optimized Neural Network Algorithm to Improve Classification Accuracy for Opinion Mining of M-Learning Reviews. IJETTCS, 2(3):345-349, 2013.

[20] L. N. de Castro and J. Timmis. Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002.

[21] Saroj Kumar, and Priya Singh "Cloud Computing Applications Architecture - ArchJava" International Journal of Emerging Technology and Advanced Engineering, ISSN 2250-2459, Volume 2, Issue 6, June 2012, www.ijetae.com.

[22] B.Jansen, M. Zhang, K. Sobel, A. Chowdury. The Commerical Impact of Social Mediating Technologies: Micro-blogging as Online Word-of-Mouth Branding, 2009.

[23] B.Manning and H. Schuetze. Foundations of Statistical Natural Language Processing. 1999.

[24] B. Pang, L. Lee, S. Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques, 2002.

[25] B. Pang and L. Lee. "Opinion Mining and Sentiment Analysis" in Foundations and Trends in Information Retrieval, 2008