TWITTER SENITIMENT ANALYSIS

Ajay Chakraborty, Akash Kumar Jai, Praksha Jain, Rajneek Singh Toor, Ritika Saha, Siddharth Singh
Prof. Poojarni Mitra(Mentor)
Department of Computer Science & Engineering
University of Engineering and Management, Kolkata

Abstract

With the advancement of web technology and its growth, there is a huge volume of data present on the web for internet users and a lot of data is generated too. The Internet has become a platform for online learning, exchanging ideas and sharing opinions. Social networking sites like Twitter, Facebook, Google+ are rapidly gaining popularity as they allow people to share and express their views about topics, have a discussion with different communities, or post messages across the world. There has been a lot of work in the field of sentiment analysis of twitter data. In this project, we have chosen to classify tweets from Twitter into "positive" or "negative" sentiment by building a model based on probabilities. Twitter is a micro blogging website where people can share their feelings quickly and spontaneously by sending tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign "@" or participate in a topic by adding a hashtag "#" to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

I. INTRODUCTION

In the past few years, there has been a huge growth in the use of micro blogging platforms such as Twitter. Spurred by that growth, companies and media organizations are increasingly seeking ways to mine Twitter for information about what people think and feel about their products and services. Companies such as Twitter (twitter.com), tweetfeel (www.tweetfeel.com), and Social Mention (www.socialmention.com) are just a few who advertise Twitter sentiment analysis as one of their services. While there has been a fair amount of research on how sentiments are expressed in genres such as online reviews and news articles, how sentiments are expressed given the informal language and message-length constraints of micro blogging has been much less studied. Features such as automatic part-of-speech tags and resources such as sentiment lexicons have proved useful for sentiment analysis in other domains, but will they also prove useful for sentiment analysis in Twitter? In this paper, we begin to investigate this question. Another challenge of micro blogging is the incredible breadth of topic that is covered. It is not an exaggeration to say that people tweet about anything and everything. Therefore, to be able to build systems to mine Twitter sentiment about any given topic, we need a method for quickly identifying data that can be used for training. In this paper, we explore one method for building such data: using Twitter hashtags (e.g., #bestfeeling, #epicfail, #news) to identify positive, negative, and neutral tweets to use for training three way sentiment classifiers. The online medium has become a significant way for people to express their opinions and with social media, there is an abundance of opinion information available. Using sentiment analysis, the polarity of opinions can be found, such as positive, negative, or neutral by analyzing the text of the opinion. Sentiment analysis has been useful for companies to get their customer's opinions on their products predicting outcomes of elections , and getting opinions from movie reviews. The information gained from sentiment analysis is useful for companies making future decisions. Many traditional approaches in sentiment analysis uses the bag of words method. The bag of words technique does not consider language morphology, and it could incorrectly classify two phrases of having the same meaning because it could have the same bag of words. The relationship between the collection of words is considered instead of the relationship between individual words. When determining the overall sentiment, the sentiment of each word is determined and combined using a function. Bag of words also ignores word order, which leads to phrases with negation in them to be incorrectly classified. Other techniques discussed in sentiment analysis include Naive Bayes, Maximum Entropy, and Support Vector Machines. In the Literature Survey section, approaches used for sentiment analysis and text classification are summarized. Sentiment analysis refers to the broad area of natural language processing which deals with the computational study of opinions, sentiments and emotions expressed in text. Sentiment Analysis (SA) or Opinion Mining (OM) aims at learning people's opinions, attitudes and emotions towards an entity. The entity can represent individuals, events or topics. An immense amount of research has been performed in the area of sentiment analysis. But most of them focused on classifying formal and larger pieces of text data like reviews. With the wide popularity of social networking and micro blogging websites and an immense amount of data available from these resources, research projects on sentiment analysis have witnessed a gradual domain shift. The past few years have witnessed a huge growth in the use of micro blogging platforms. Popular micro blogging websites like Twitter have evolved to become a source of varied information. This diversity in the information owes to such micro blogs being elevated as platforms where people post real time messages

about their opinions on a wide variety of topics, discuss current affairs and share their experience on products and services they use in daily life. Stimulated by the growth of micro blogging platforms, organizations are exploring ways to mine Twitter for information about how people are responding to their products and services. A fair amount of research has been carried out on how sentiments are expressed in formal text patterns such as product or movie reviews and news articles, but how sentiments are expressed given the informal language and message-length constraints of micro blogging has been less explored.

A careful investigation of tweets reveals that the 140 character length text restricts the vocabulary which imparts the sentiment. The hyperlinks often present in these tweets in turn restrict the vocabulary size. The varied domains discussed would surely impose hurdles for training. The frequency of misspellings and slang words in tweets (micro blogs in general) is much higher than in other language resources which is another hurdle that needs to be overcome. On the other way around the tremendous volume of data available from micro blogging websites on varied domains are incomparable with other data resources available. Micro blogging language is characterized by expressive punctuations which convey a lot of sentiments. Bold lettered phrases, exclamations, question marks, quoted text etc. leave scope for sentiment extraction. The proposed work attempts a novel approach on twitter data by aggregating an adapted polarity lexicon which has learnt from product reviews of the domains under consideration, the tweet specific features and unigrams to build a classifier model using machine learning techniques.

II. TWITTER SENTIMENT ANALYSIS

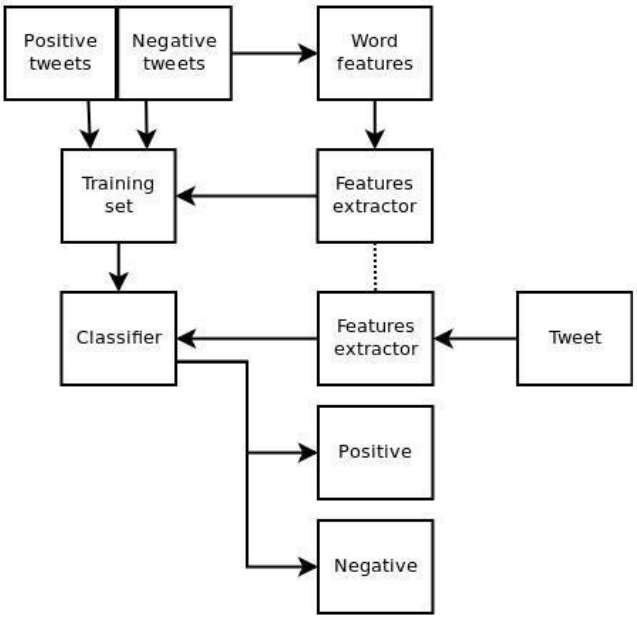
A. Project Work

Sentiment analysis also refers to opinion mining, is a submachine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to working on.

B. Project Definition

Sentiment analysis can be defined as a process that automates mining of attitudes, opinions, views and emotions from text, speech, tweets and database sources through Natural Language Processing (NLP). Sentiment analysis involves classifying opinions in text into categories like "positive" or "negative" or "neutral". It's also referred as subjectivity analysis, opinion mining, and appraisal extraction. The words opinion, sentiment, view and belief are used interchangeably but there are differences between them.

- Opinion: A conclusion open to dispute (because different experts have different opinions)
- View: subjective opinion
- Belief: deliberate acceptance and intellectual assent
- Sentiment: opinion representing one's feelings.



C. Pre-processing of the datasets

A tweet contains a lot of opinions about the data which are expressed in different ways by different users .The twitter dataset used in this survey work is already labeled into two classes viz. negative and positive polarity and thus the sentiment analysis of the data becomes easy to observe the effect of various features. We have exactly 790177 positive tweets and 788435 negative tweets which signify that the dataset is well balanced. There are also no duplicates. The raw data having polarity is highly susceptible to inconsistency and redundancy. Preprocessing of tweet include following points,

- Replace all emoticons by their sentiment polarity ||pos||/||neg|| using the emoticon dictionary.
- Replace all URLs with a tag ||url||.
- Remove Unicode characters.
- Decode HTML entities.
- Reduce all letters to lowercase (We should take care of proper nouns but for simplicity we will lower them as well) (After emoticons because they can use upper case letters)
- Replace all usernames/targets @ with ||target||.
- Replace all acronyms with their translation.

- Replace all negations (e.g: not, no, never) by tag ||not||.
- Replace a sequence of repeated characters by two characters (e.g: "helloooo" = "helloo") to keep the emphasized usage of the word.

D. Training

Supervised learning is an important technique for solving classification problems. Training the classifier makes it easier for future predictions for unknown data.

III. MACHINE LEARNING APPROACHES

Once we have applied the different steps of the pre-processing part, we can now focus on the machine learning part. There are three major models used in sentiment analysis to classify a sentence into positive or negative: SVM, Naive Bayes and Language Models(N-Gram).SVM is known to be the model giving the best results but in this project we focus only on probabilistic model that are Naive Bayes and Language Models that have been widely used in this field. Let's first introduce the Naive Bayes model which is well-known for its simplicity and efficiency for text classification.

A. Naïve Bayes

In machine learning, Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum likelihood training can be done by evaluating a closed form expression (mathematical expression that can be evaluated in a finite number of operations), which takes linear time. It is based on the application of the Baye's rule given by the following formula:

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

Formula: Bayes' rule

where denotes the document and the category (label), and are *D C d c* instances of *D* and *C.* We can simplify this expression by,

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

Formula: Bayes' rule simplified

In our case, a tweet d is represented by a vector of K attributes such as d = (w1, w2, ..., wk) .Computing P(d|c) is not trivial and that's why the Naive Bayes introduces the assumption that all of the feature values are independent given the category label c . That is, for i =/ j , are conditionally independent given the category label c . So the Baye's rule can be re written as,

$$P(c|d) = P(c) \times \frac{\prod_{j=1}^{K} P(w_j|c)}{P(d)}$$

Formula: Baye's rule rewritten

Based on this equation, maximum a posterior (MAP) classifier can be constructing by seekingthe optimal category which maximizes the posterior P(c|d) :

$$c* = arg\max_{c \in C} P(c|d)$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \frac{\prod_{j=1}^{K} P(w_j|c)}{P(d)} \right\}$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{K} P(w_j|c) \right\}$$

Formula: Classifier maximizing the posterior probability P(c|d)

Note that $P(d)$ is removed since it is a constant for every category *c* .

There are several variants of Naive Bayes classifiers that are:

● The Multivariate Bernoulli Model: Also called binomial model, useful if our feature vectors are binary (e.g 0s and 1s). An application can be text classification with bag of words model where the 0s 1s are "word does not occur in the document" and "word occurs in the document" respectively.

● The Multinomial Model : Typically used for discrete counts. In text classification, we extend the Bernoulli model further by counting the number of times a word w_i appears over the number of words rather than saying 0 or 1 if word occurs or not.

● The Gaussian Model : We assume that features follow a normal distribution. Instead of discrete counts, we have continuous features. For text classification, the most used considered as the best choice is the Multinomial Naïve Bayes.

The prior distribution $P(c)$ can be used to incorporate additional assumptions about the relative frequencies of classes. It is computed by:

$$P(c) = \frac{N_i}{N}$$

Formula : Prior distribution P(c)

where *N* is the total number of training tweets and *Ni* is the number of training tweets in class *c*.
The likelihood is usually computed P(wj|c) using the formula:

$$P(w_i|c) = \frac{1 + count(w_j, c)}{|V| + N_i}$$

Formula: Likelihood P(wj|c)
where count(wj, c) is the number of times that word wj occurs within the training tweets of class c. This estimation uses the simplest smoothing method to solve the zero probability problem , that arises when our model encounters a word seen in the test set but not in the training set, Laplace or add onesince we use 1 as constant. We will see that Laplace smoothing method is not really effective compared to other smoothing methods used in language models.

B. Baseline

In every machine learning task, it is always good to have what we called a baseline. It often a "quick and dirty" implementation of a basic model for doing the first classification and based on its accuracy, try to improve it. We use the Multinomial Naive Bayes as learning algorithm with the Laplace smoothing representing the classic way of doing text classification. Since we need to extract features from our data set of tweets, we use the bag of words model to represent it. The bag of words model is a simplifying representation of a document where it is represented as a bag of its words without taking consideration of the grammar or word order. In text classification, the count (number of time) of each word appears is a document is used as a feature for training the classifier. Firstly, we divide the data set into two parts, the training set and the test set. To do this, we first shuffle the data set to get rid of any order applied to the data, then we from the set of positive tweets and the set of negative tweets, we take 3/4 of tweets from each set and merge them together to make the training set. The rest is used to make the test set. Finally the size of the training set is 1183958 tweets and the test set is 394654 tweets. Notice that they are balanced and follow the same distribution of the initial data set. Once the training set and the test set are created we actually need a third set of data called the validation set. It is really useful because it will be used to validate our model against unseen data and tune the possible parameters of the learning algorithm to avoid under fitting and over fitting for example. We need this validation set because our test set should be used only to verify how well the model will generalize. If we use the test set rather than the validation set, our model could be overly optimistic and twist the results

To make the validation set, there are two main options:

● Split the training set into two parts (60%, 20%) with a ratio 2:8 where each part contains an equal distribution of example types. We train the classifier with the largest part, and make prediction with the smaller one to validate the model. This technique works well but has the disadvantage of our classifier not getting trained and validated on all examples in the data set (without counting the test set).

● The K-fold cross-validation. We split the data set into k parts, hold out one, combine the others and train on them, then validate against the held out portion. We repeat that process k times (each fold), holding out a different portion each time. Then we average the score measured for each fold to get a more accurate estimation of our model's performance.

We split the training data into 10 folds and cross validate on them using scikit learn As shown in the figure above. The number of K-folds is arbitrary and usually set to 10 it is not a rule. In fact, determine the best K is still an unsolved problem but with lower K:computationally cheaper, less variance, more bias. With large K: computationally expensive, higher variance, lower bias. We can now train the naive bayes classifier with the training set, validate it using the hold out part of data taken from the training set, the validation set, repeat this 10 times and average the results to get the final accuracy which is about 0.77 as shown in the screen results below,

```
Total tweets classified: 1183958
Score: 0.77653600187
Confusion matrix:
[[465021 126305]
 [136321 456311]]
```

Figure: Result of the naive baye's classifier with the score representing the average of the results of each 10-fold cross-validation, and the overall confusion matrix.

Notice that to evaluate our classifier we two methods, the F1 score and a confusion matrix.
The F1 Score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. It a measure of a classifier's accuracy. The F1 score is given by the following formula,

$$F1 = \frac{2 \times (precision \times recall)}{(precision + recall)}$$

Formula: F1 score
where the precision is the number of true positives (the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class,

$$Precision = \frac{TP}{TP + FP}$$

Formula: Precision

and the recall is the number of true positives divided by the total number of elements that actually belong to the positive class,

$$Recall = \frac{TP}{TP + FN}$$

Formula: Recall

A precision score of 1.0 means that every result retrieved was relevant (but says nothing about whether all relevant elements were retrieved) whereas a recall score of 1.0 means that all relevant documents were retrieved (but says nothing about how many irrelevant documents were also retrieved).

There is trade-off-between precision and recall where increasing one decrease the other and we usually use measures that combine precision and recall such as F-measurer or MCC. A confusion matrix helps to visualize how the model did during the classification and evaluate its accuracy. In our case we get 156715 false positive tweets and 139132 false negatives tweets.

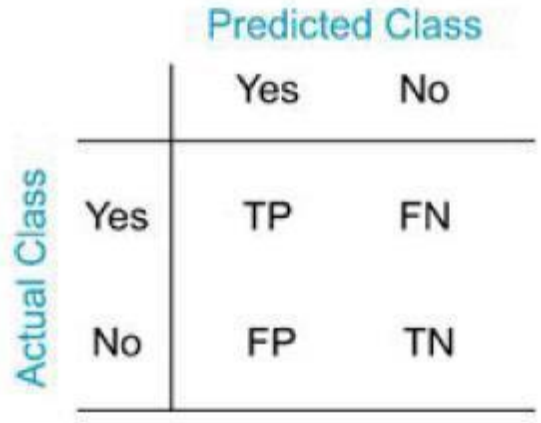


Figure: Example of confusion matrix

Notice that we still didn't use our test set, since we are going to tune our classifier for improving its results. Hopefully we can distinguish that the number of true positive and true negative classified tweets is higher than the number of false and positive and negative tweets. However from this result we try to improve the accuracy of the classifier by experimenting different techniques and we repeat the same process using the k-fold cross validation to evaluate its averaged accuracy.

From the baseline, the goal is to improve the accuracy of the classifier, which is 0.77, in order to determine better which tweet is positive or negative. There are several ways of doing this and we present only few possible improvements (or not).First we could try to removed what we called, stop words. Stop words usually refer to the most common words in the English language (in our case) such as: "the", "of", “to” and so on. They do not indicate any valuable information about the sentiment of a sentence and it can be necessary to remove them from the tweets in order to keep only words for which we are interested. To do this we use the list of 635 stopwords that we found. In the table below, you can see the most frequent words in the data set with their counts,

```
[('||target||', 780664),
 ('i', 778070),
 ('to', 614954),
 ('the', 538566),
 ('a', 383910),
 ('you', 341545),
 ('my', 336980),
 ('and', 316853),
 ('is', 236393),
 ('for', 236018),
 ('it', 235435),
 ('in', 217350),
 ('of', 192621),
 ('on', 169466),
 ('me', 163900),
 ('so', 158457),
 ('have', 150041),
 ('that', 146260),
 ('out', 143567),
 ('but', 132969)]
```

Table: Most frequent words in the data set with their corresponding count.

Recall that ||url|| corresponds to the URLs, ||target|| the twitter usernames with the symbol “@”before, ||not|| replaces the negation words, ||pos|| and ||neg|| replace the positive and negative smiley respectively. After removing the stop words we get the results below,

```
Total tweets classified: 1183958
Score: 0.758623708326
Confusion matrix:
[[437311 154015]
 [136343 456289]]
```

Figure: Result of the naive bayes classifier with stopwords removed.

Compared to the previous result, we lose in accuracy and the number of 0.02 false positive goes from 126305 to 154015. We conclude that stop words seem to be useful for our classification task and remove them do not represent an improvement. We could also try to stem the words in the data set. Stemming is the process by which endings are removed from words in order to remove things like tense or plurality. The stem form of a word could not exist in a dictionary (different from Lemmatization). This technique allows to unify words and reduce the dimensionality of the dataset. It's not appropriate for all cases but can make it easier to connect together tenses to see if you're covering the same subject matter. It is faster than Lemmatization (remove inflectional endings only and return the base or dictionary form of a word, which is known as the lemma). Using the library NLTK which is a library in Python specialized in natural language

processing, we get the following results after stemming the words in the data set,

```
Total tweets classified: 1183958
Score: 0.773106857186
Confusion matrix:
[[462537 128789]
 [138039 454593]]
```

Figure: Result of the naive baye's classifier after stemming.

We actually lose 0.002 in accuracy score compared to the results of the baseline. We conclude that stemming words does not improve the classifier's accuracy and actually do not make any sensible changes.

C. Language Models

Let's introduce language models to see if we can have better results than those for our baseline. Language models are models assigning probabilities to sequence of words .
Initially, they are extensively used in speech recognition and spelling correction but it turns out that they give good results in text classification. The quality of a language model can be measured by the empirical perplexity (or entropy) using:

$$Perplexity = T\sqrt{\frac{1}{P(w_1,\ldots,w_T)}}$$

$$Entropy = log_2 Perplexity$$

Formula: Perplexity and Entropy to evaluate language models. The goal is to minimize the perplexity which is the same as maximizing probability. An N-Gram model is a type of probabilistic language model for predicting the next item in such a sequence in the form of (n 1) order Markov Model.
The Markov assumption is the probability of a word depends only on the probability of a limited history (previous words).

$$P(w_i|w_1,\ldots,w_{i-1}) = P(w_i|w_{i-n+1},\ldots,w_{i-1})$$

Formula: General form of N-grams.
A straightforward maximum likelihood estimate of n-gram 28 probabilities from a corpus is given by the observed frequency,

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = \frac{count(w_{i-n+1},\ldots,w_i)}{count(w_{i-n+1},\ldots,w_{i-1})}$$

Formula: MLE of N-grams.
There are several kind of n-grams but the most common are the unigram, bigram and trigram. The unigram model make the assumption that every word is independent and so we compute the probability of a sequence using the following formula,

$$P(w_1,w_2,\ldots,w_n) = \prod_i P(w_i)$$

Formula: Unigram.
In the case of the bigram model we make the assumption that a word is dependent of its previous word ,

$$P(w_i|w_1,w_2,\ldots,w_{i-1}) \approx P(w_i|w_{i-1})$$

Formula: Bigram.
To estimate the n-gram probabilities, we need to compute the Maximum Likelihood
Estimates .For Unigram:

$$P(w_i) = \frac{C(w_i)}{N}$$

Formula: MLE for unigram.
For Bigram,

$$P(w_i,w_j) = \frac{count(w_i,w_j)}{N}$$

$$P(w_j|w_i) = \frac{P(w_i,w_j)}{P(w_i)} = \frac{count(w_i,w_j)}{\sum_w count(w_i,w)} = \frac{count(w_i,w_j)}{count(w_i)}$$

Formula: MLE for bigram.
Where is the number of words, means count, N C wiand wjare words.
There are two main practical issues:
● We compute everything in log space (log probabilities) to avoid underflow (multiplying so
many probabilities can lead to too small number) and because adding is faster than
multiplying (p1 × p2 × p3) = (logp1 log) + p2 + logp3
● We use smoothing techniques such as Laplace, Witten-Bell Discounting, Good-Turing Discounting to deal with unseen words in the training occurring in the test set.
An N-gram language model can be applied to text classification like Naive Bayes model does. A tweet is categorized according to,

$$c* = arg\max_{c\in C} P(c|d)$$

Formula: Objective function of n-gram.
and using Baye's rule, this can be rewritten as,

$$c* = arg\max_{c\in C}\{P(c)P(d|c)\}$$

$$c* = arg\max_{c\in C}\left\{P(c)\times\prod_{i=1}^{T}P(w_i|w_{i-n+1},\ldots,w_{i-1},c)\right\}$$

$$c* = arg\max_{c\in C}\left\{P(c)\times\prod_{i=1}^{T}P_c(w_i|w_{i-n+1},\ldots,w_{i-1})\right\}$$

Formula: Objective function rewritten using baye's rule of n-gram.

P(d|c)is the likelihood of under category which can be computed by P(d|c) d c an n-gram language model. An important note is that n-gram classifiers are in fact a

generalization of Naive Bayes. A unigram classifier with Laplace smoothing corresponds exactly to the traditional naïve Bayes classifier. Since we use bag of words model, meaning we translate this sentence: "I don't like chocolate "into "I", "don't", "like", "chocolate", we could try to use bigram model to take care of negation with "don't like" for this example. Using bigrams as feature in the classifier we get the following results,

```
Total tweets classified: 1183958
Score: 0.784149223247
Confusion matrix:
[[480120 111206]
 [138700 453932]]
```

Formula: Results of the naive bayes classifier with bigram features

Using only bigram features we have slightly improved our accuracy score about 0.01. Based on that we can think of adding unigram and bigram could increase the accuracy score more.

```
Total tweets classified: 1183958
Score: 0.795370054626
Confusion matrix:
[[486521 104805]
 [132142 460490]]
```

Formula: Results of the naive baye's classifier with unigram and bigram features

and indeed, we increased slightly the accuracy score about 0.02 compared to the baseline.

IV. CHALLENGES IN SENTIMENT ANALYSIS

Sentiment Analysis is a very challenging task. Following are some of the challenges faced in Sentiment Analysis of Twitter.

1. Identifying subjective parts of text: Subjective parts represent sentiment-bearing content. The same word can be treated as subjective in one case, or an objective in some other. This makes it difficult to identify the subjective portions of text.
2. Domain dependence: The same sentence or phrase can have different meanings in different domains.
3. Sarcasm Detection: Sarcastic sentences express negative opinion about a target using positive words in unique way.
4. Thwarted expressions: There are some sentences in which only some part of text determines the overall polarity of the document.
5. Explicit Negation of sentiment: Sentiment can be negated in many ways as opposed to using simple no, not, never, etc. It is difficult to identify such negations.
6. Order dependence: Discourse Structure analysis is essential for Sentiment Analysis/Opinion Mining.
7. Entity Recognition: There is a need to separate out the text about a specific entity and then analyze sentiment towards it.
8. Building a classifier for subjective vs. objective tweets. Current research work focuses mostly on classifying positive vs. negative correctly. There is need to look at classifying tweets with sentiment vs. no sentiment closely.
9. Handling comparisons. Bag of words model doesn't handle comparisons very well.
10. Applying sentiment analysis to Facebook messages. There has been less work on sentiment analysis on Facebook data mainly due to various restrictions by Facebook graph api and security policies in accessing data.

V. APPLICATIONS OF SENTIMENT ANALYSIS

Sentiment Analysis has many applications in various Fields.

- Applications that use Reviews from Websites: Today Internet has a large collection of reviews and feedbacks on almost everything. This includes product reviews, feedbacks on political issues, comments about services, etc. Thus there is a need for a sentiment analysis system that can extract sentiments about a particular product or services. It will help us to automate in provision of feedback or rating for the given product, item, etc. This would serve the needs of both the users and the vendors.
- Applications as a Sub-component Technology A sentiment predictor system can be helpful in recommender systems as well. The recommender system will not recommend items that receive a lot of negative feedback or fewer ratings. In online communication, we come across abusive language and other negative elements. These can be detected simply by identifying a highly negative sentiment and correspondingly taking action against it.
- Applications in Business Intelligence It has been observed that people nowadays tend to look upon reviews of products which are available online before they buy them. And for many businesses, the online opinion decides the success or failure of their product. Thus, Sentiment Analysis plays an important role in businesses. Businesses also wish to extract sentiment from the online reviews in order to improve their products and in turn their reputation and help in customer satisfaction.
- Applications across Domains: Recent researches in sociology and other fields like medical, sports have also been benefitted by Sentiment Analysis that show trends in human emotions especially on social media.
- Applications In Smart Homes Smart homes are supposed to be the technology of the future. In future entire homes would be networked and people would be able to control any part of the home using a tablet device. Recently there has been lot of research going on Internet of Things(IoT).

Sentiment Analysis would also find its way in IoT. Like for example, based on the current sentiment or emotion of the user, the home could alter its ambiance to create a soothing and peaceful environment. Sentiment Analysis can also be used in trend prediction. By tracking public views, important data regarding sales trends and customer satisfaction can be extracted.

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on

the importance of the work or suggest applications and extensions.

VI. CONCLUSION

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes and NGram as baseline and how language are related NGram and Naive Bayes, and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together.

REFERENCES

.

1. Alexander Pak, Patrick Paroubek. 2010, Twitter as a Corpus for Sentiment Analysis and Opinion Mining.
2. V Kharde, 2016: Sentiment Analysis of Twitter Data
3. Funchun Peng. 2003, Augmenting Naive Bayes Classifiers with Statistical Language Models
4. Apporv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis of Twitter Data
5. Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision.