Peer-to-Peer Insult Detection in Online Communities

Priya Goyal (10535)
Department of Mathematics & Statistics
prigoyal@iitk.ac.in
IIT Kanpur

Gaganpreet Singh Kalra (10258)
Department of Computer Science & Engineering
gpskalra@iitk.ac.in
IIT Kanpur

Mentor: Dr. Amitabha Mukerjee
Department of Computer Science and Engineering
amit@cse.iitk.ac.in
IIT Kanpur

April 17, 2013

Abstract

The aim of this project is to detect the comments that are considered as insulting to other participants of blog/forum conversation. For feature selection, we use n-grams, skip grams and special words like ‘you’ to build vector model. We present two new hypotheses to construct more feature and use tools in supervised learning like logistic regression and SVM for training and testing. Evaluation on the insult dataset gives considerable improvement in accuracy to 86% over the best 80-82% earlier due to the inclusion of two new features proposed.

1. Introduction

People today around the globe are using various online forums, blogs, social networking sites, newsgroups as source of their networking, sharing and receiving knowledge. Sometimes, some users may make comments (like reference to handicap) that are considered as inappropriate by other users and may hurt their feelings. Besides, it is a barrier to user participant and prevents new users from participating. Also, sometimes you are looking for some information on some site and find insults then it leads to frustration. Though, the *Terms of Service* of social networking sites like *Facebook, Yahoo* etc. prohibits from posting content that is unlawful, abusive and harassing, users’ posts are only partially filtered for some particular collection of offensive words. Also, while some sites like *YouTube,* some newsgroups etc. provide flag facility to mark content as insulting/ inappropriate, they are prone to collusion and are highly misused (marking a non- insulting comment because it wasn’t liked). Also, it is not possible to have a human moderator to review the comments before posting because of the increasing amount of online data. Hence, we need an automatic classifier that will detect the insulting comments.

Our aim is to focus on comments that are insulting to other participant of the blog/ forum conversation. However, the comments containing insults but are targeted to a non-participant of conversation (like a celebrity etc.) are not marked as insults. Insults are of many types like: Taunts, reference to handicaps, squalid language, slurs and racism which are aimed at attacking the other person. However some insults are aimed at abuse or embarrassing the reader (not an attack) like crude language, disguise provocative words, disguise, sarcasm, innuendo (indirect reference). We are aimed at detecting comments that are intended to be obviously insulting to other participants.

We obtained our annotated dataset from the Kaggle website. The training dataset consists of 7000 comments in total of which nearly 3000 are marked as insulting This problem is treated as binary classification problem and various supervised learning tools like Logistic regression, SVM have been used. The implementations have been done using Scikit learn Toolkit which is a natural language toolkit in Python.

This report has been organized as: in Section 2, we present the related background work on this problem. In Section 3, we present the methodology we have used followed by explanations of each intermediate process following sub-sections. In section 4, we state our implementation results with analysis. In section 5, we summarize the work done in this project and finally in Section 6, we present the future scope of this project.

2. Related Work

Various attempts have been made to classify the insulting comments in on line forums. The work by Ellen Spertus [1] used static dictionary approach and defined some patterns based on socio-linguistic observation to build a feature vector for training. This work suffers from high false positive rates and low coverage. Another work by Altaf Mahmud et. al [2] tries to differentiate between insult and factual statements by parsing the sentences and using the semantic rules. This works doesn't distinguish between the insults directed to non-participant and participant of conversation. Also the rules and seed words approaches are rigid and lack generality because of flexibility of conversation. The work by Razavi et. al [3] makes use of Insulting and Abusing language dictionary on top of bag-of words features in their proposed three-level classification machine learning model. This works rely heavily on dictionary which is not easily available. Another recent effort by Carolyn P. Rose et. al [4] in classifying offensive tweets builds topical features and lexicon features using some dictionary and uses various machine learning approach.

However, besides the limited approach of using seed words and pattern matching, these works also do not distinguish between the insult directed towards the people participating in blog/forum conversation and non-participants such as celebrities, public figures etc. Comments which contain profanity or racial slurs may not necessarily be insulting to other person. We aim at building an efficient classifier that detects peer-to-peer insults and we adopt the machine learning approach entirely.

The various challenges involved in this process are: Grammatical mistakes e.g. “What on earth a BIGGOT like you is doing walking on the face of earth?” , Typography: s h i t (shit) , People usually circumvent dictionary due to flexibility of conversation e.g. @$$hole (asshole) , Wordplays e.g. kucf oyu , Sarcasm e.g. “Sometimes I don’t know whether to laugh at you or pity you.” , Innuendo where the indirect mention to someone has been made e.g. “Only cowards, thieves, cheats and liars hide behind pseudonyms.”.

3. Our Methodology

Here is the basic philosophy of our work. The following sub-sections discuss each step in detail.

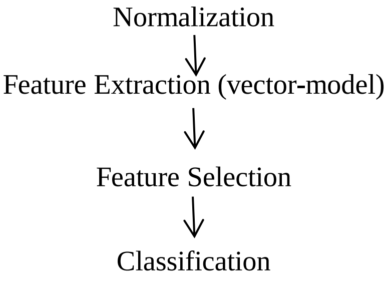

Figure1: General classification strategy

3.1 Normalization

The raw data obtained from internet can’t be used directly for learning models. Some pre-processing of data is necessary and we need to convert it to the input format of various machine learning algorithms. However, this pre-processing should not lead to loss of information so depending on the task; necessary pre-processing steps should be identified.

The first step of this process involves normalization. We use various techniques for this (identified by looking at the dataset):

3.1.1 Removal of unwanted strings:

We clean our data by removing some encoding parts like \\xa0, \\xc2, \\n, etc. and some unwanted html tags. These put bias on the results if not removed. For example: “\\xc2 \\xa0 If you take out the fags and booze, leaving say \\xc2, \\xa0, \\c2, \\xa3170 per week say and then recognize that they are feeding 8 people then that doesn't seem too extreme”. After cleaning the encoding parts, it looks like “If you take out the fags and booze, leaving say 3170 per week say and then recognize that they are feeding 8 people then that doesn’t seem too extreme.”

3.1.2 Stemming:

Stemming involves reducing the words to their root or stem form like “retarded” to “retard”, “embarrassing” to “embarrass” etc. This is necessary because otherwise it results in unnecessary increase in the number of features. There is a standard algorithm called Porter Stemming algorithm whose python is easily available on the internet. However, for our project, we don’t use this algorithm because this might result in the loss of information from dataset as it may wrongly reduce some words which effect insult detection to their root.

3.1.3 Correcting some common words:

Due to the flexibility of conversation on the internet people prefer to write short forms of words like “ur” for “your”, “nopes’ for “no” etc. If we can convert these types of words to their original correct form, then it reduces the size of feature set and also improves the accuracy of models. So we manually created a mapping for some commonly used words to their correct form and normalized the dataset using this mapping.

Also, while writing insult words, people manipulate the dictionary like “@$$hole” for “asshole”, “!d!ot” for “idiot” etc. These words are necessary for the insult detection and so these must be identified my models correctly. Hence we used a dictionary of approximately 500 words containing all the possible ways an insult word can be written and when these words are encountered we convert then to their true form using this dictionary. Though this dictionary is not exhaustive and it is also not possible to create an exhaustive dictionary, still using this help in improving the accuracy.

So, before normalizing, if a particular entry looks like “"\xa0you are truly an !d!ot.\xa0 the man pitched a no hitter.\xa0 just shut up." it changes to "you are truly an idiot. the man pitched a no hitter. just shut up." after normalization.

3.2 Feature Extraction

The strings should be converted to a numerical vector so that they can be used by machine learning algorithms. We use the various n-grams model and skip grams to construct the feature vector. The process occurs in following steps:

3.2.1. Tokenizing

Split the text into tokens. These tokens can be characters, 'words' or n-grams which are the sequence of n consecutive words. We take words as tokens. We build 2, 3, 4 and 5 n-grams for the feature vector construction.

3.2.2. Counting

Count the number of times each of these tokens occurs in each of the text strings. This way we construct a (generally sparse) matrix of size N by V where N is the size of the training data (number of comments in our case) and V is the size of the vocabulary (the length of feature vector constructed over the whole training set using n-grams, skip grams and second-person rule) representing all the text strings where the number of occurrences of each token is a feature for that text string.

3.2.3 Normalizing:

The number of occurrences of each feature may not be a good feature to be used directly. For example a word like 'the' occurs in almost all the text strings. We might wrongly consider it as important due to the high frequency of occurrence. So, we need to reduce its importance. This is achieved by using vectorization for finding score which is a measure of the relevance of word.

3.2.4 Tf-idf Score:

Tf-idf stands for: “Term frequency x Inverse document Frequency” and it measures the ratio of the number of times the token occurs in a particular text string to the fraction of the documents in which the token occurs.

After constructing a key word set using n-grams, skip grams and special rule, using this list of features; we find the tf-idf score of each feature. The item in the tf-idf vector corresponds to the tf-idf weight of the feature (keyword) at the same index. A keyword’s term frequency is the number of times the word appears in the title. Its document frequency is the number of titles the word appears over the total number of titles. The keyword’s tf-idf weight is its term frequency divided by its document frequency. Then, the tf-idf vector is normalized. The tf-idf vector represents the occurrence of the important keywords a user is interested in. In this model, each document is first represented as a term-frequency vector in the term-space:

$$d_{jtf} = \left(tf_{1j}, tf_{2j}, \dots\dots\dots, tf_{nj}\right) \qquad j = 1,2,3,\dots\dots..D,$$

where tf_{ij} is the frequency of the ith term in documentd_j, n is the total number of the selected vocabulary, and D is the total number of documents in the collection.

Next, we weight each term based on its inverse document frequency (IDF) and obtain a tf-idf vector for each document:

$$d_j = \left(tf_{1j} * idf_1, tf_{2j} * idf_2, \dots\dots\dots, tf_{nj} * idf_n\right) \qquad j = 1,2,3,\dots\dots..D,$$

3.3 Additional Features

We use additional features in an effort to increase the efficiency of our algorithms. Based on our observation of the dataset, we found proposed the following two hypothesis which we believe will help tin increasing the accuracy. We call these two extra features 'skip grams' and 'second person rule'.

3.3.1 Skip grams:

Apart from adding the count of each word or n-gram in a document as a feature, we also consider the long distance words (at a particular distance) as a feature hence increasing the number of columns of our feature matrix. We do this because co-occurrence of some words like “you fools” etc. is a good indicator of insult. We try to capture these using skip grams to construct more features. This feature has as a parameter the number of words you skip between 2 words. For example: Given the sentence 'You are an idiot', using “2 skips” adds 2 new features: the count of 'you an' and the count of 'are idiot' to our set of features. We tried to use 1, 2, 3 and 4 skip grams and the results are shown in Section 4.

3.3.2 Second person feature:

We also add as a feature the count of 'the set of words' that occur just after words like “you are”, “you're” in a sentence. This is based on our observation that nearly 40% of the insults in our dataset had words like “you xxx” etc. This feature is a very strong hypothesis which greatly improves the accuracy as shown in the results section 4.

3.4 Feature Selection

The features generated are of the order of a hundred thousand features which is too big a number to be handled efficiently by algorithms like SVM and Logistic Regression. So we need to select a few best features out of our set of features. We apply a statistical test known as “**Chi Squared Test**” to our feature matrix to select best k number of features where k is our parameter roughly equal to 3000.

3.4.1 Chi-Squared test:
This test finds if a pair of categorical variables on a data is statistically dependent or independent. In our case, this pair of variable is: The label of the sentence: “insult or not” and the occurrence or non-occurrence of a feature”. The features which score high in this test are the important features which we keep while we discard the remaining features. To understand this measure better, consider the following example.
Imagine a sample of our data that looks like this:

Text String	Feature 1 - 'you'	Feature 2 - 'the'	Feature 3 – 'you are'	Feature 4 - 'idiot'	Label
1	Present	Present	Not Present	Present	Insult
2	Present	Present	Not Present	Not Present	Not Insult
3	Not Present	Present	Not Present	Present	Insult
4	Present	Not Present	Present	Not Present	Insult

Had the presence of 'you' and 'being insult' been independent, the expected number of rows in which both of these happen would be:

$$\mathrm{E}(\text{'you present}^{\prime}, \text{insult}) = \frac{\mathrm{N}(\text{'you present'}) * \mathrm{N}(\text{'insult'})}{\mathrm{N}}$$

where N('you present', 'insult') is the number of rows which have the feature 'you' and are labeled as 'insult' and N is the total number of rows.

$$X^2 = \sum \frac{(Observerd(i,j) - Expected(i,j)))^2}{Expected(i,j)}$$

where i= {'yes present', 'yes not present'} and j= {'insult', 'not insult'} This is a measure of dependency in the two categorical variables. Higher this value, more related is the two variables. Lesser the value, more independent are the variables.

3.5 Classification

After we have constructed the features and extracted the top features, we apply machine learning algorithms to learn models. We use SVM and Logistic regression algorithms on our feature data. Finally, we combine the results of both the algorithms to get the final results. The results are obtained as probabilities which indicate the probability of a comment being an insult.

4 Evaluation and Results

- Accuracy without applying our hypothesis: 85.2381
- Accuracy with Skip Grams (2 words skipped) included: 86.5079
- Accuracy with Second-person rule included: 86.7725
- Accuracy with both Skip Grams and Second-person rule included in table:

Skips	Accuracy	Precision	Recall
2 skips	86.84	0.765	0.72
3 skips	86.84	0.764	0.71
2, 3 skips	86.92	0.769	0.71
2, 3 and 4 skips	86.66	0.759	0.71

As we observe, the skip grams didn’t really help much in improving the accuracy. However overall we gained 4% increase in accuracy over the previous models. We also found that the results vary over different test sets.

5 Conclusion and Future Work

We have made an attempt to detect intended to be insulting to a participant of conversation. For this we observed our dataset and proposed two new features: skip-grams and second person rule to test our hypothesis. We used machine learning algorithms SVM and Logistic Regression to train models. We tested it on a dataset and obtained a considerable gain in accuracy. However there are still false positives like: “you republican politicians will never get it in your head”- this is the insult to a non-participant but this was labeled as 0.92 probable insults by our model. Also, the false negative "I'm sure dumb-asses, like you, will be trolling full force under our next President...Romney" This is an insult to a participant but our algorithm marks it as having 23% probability of being an insult. So some more sophisticated techniques need to be built. Also we can investigate new features like user id, length of thread following a comment, time of comment etc. Also, some method to detect insult like sarcasm “you are the supreme boss” innuendo “Only cowards, thieves, cheats and liars hide behind pseudonyms.” need to be devised.

6 Acknowledgement

We thank Prof. Amitabha Mukerjee for his valuable support throughout the project, guiding us from time to time and helping us get through various roadblocks. We are thankful to "Kaggle.com" online forums as whose discussions proved really helpful to us and open source "scikit learn" package in python.

7 REFERENCES

[1] Ellen Spertus. 1997. Smokey: *Automatic recognition of hostile messages*. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence, pages 1058–1065.

[2] Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan, 2008. *Detecting flames and insults in text*. In Proceedings of the Sixth International Conference on Natural Language Processing.

[3] Amir H. Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. 2010 *Offensive language detection using multi-level classification*. In Proceedings of the 23rd Canadian Conference on Artificial Intelligence, pages 16–27.

[4] Xiang, G., Hong, J., & Rosé, C. P. (2012). *Detecting Offensive Tweets via Topical Feature Discovery over a Large Scale Twitter Corpus*, Proceedings of The 21st ACM Conference on Information and Knowledge Management, Sheraton, Maui Hawaii, Oct. 29- Nov. 2, 2012.

[5] For badwords file:

http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/

[6] For starter code:

www.kaggle.com/c/detecting-insults-in-social-commentary/forums

[7] For dataset:
www.kaggle.com/c/detecting-insults-in-social-commentary/data