Mukund Seethamraju
mukund.seethamaraju@gmail.com
SR NO:11-01-00-10-91-12-1-09969

# Automatic identification of personally insulting comments in conversations

E0270 2015 Machine Learning Course Project

This is an implementation oriented 'individual' project.


## Abstract

In this project I aim to detect the comments that are intended to be insulting to other participants of blog/forum or any other different conversation. I have used n-grams, skip grams and special words like 'you' for feature selection to build a vector model. I have implemented SVM classifiers and then used neural networks and logistic regression. During implementation, I have used python(version 2.7.8), scikit-learn library (for the SVM implementation) and other libraries like numpy, scipy, nltk and few others.


## Introduction:

Now a days all of us use various online forums, social networking sites or blogs etc,. as a platform for conversations and exchange of knowledge. Sometimes, the participants in the conversation may make statements or say something that are considered as inappropriate by other users participating in the conversation and may hurt their feelings. These events might prevent new users from participating or using those platforms. This may also lead to frustration while looking for some information on some site and finding insults. Although,almost all the social networking sites and other platforms like Facebook, Yahoo etc. prohibit from posting inappropriate content which is considered abusive and harassing, these posts(comments) are only partially filtered for some particular collection of offensive words which in practical is very small compared to list of abusive words generally people use. And it is certainly impossible to have a human moderator to review the comments before posting because of the enormously increasing amount of online data. Thus, an effective approach to detecting inappropriate online content is of great practical importance. I address this question through a machine learning approach to Automatic identification of personally insulting comments in conversations by building an automatic classifier.

The main aim of this project is to focus on detecting comments that are intended to be obviously insulting to another participant in the blog/forum conversation. We are not looking for insults directed to non-participants. Insults could contain profanity, racial slurs, or other offensive language. Comments which contain profanity or racial slurs, but are not necessarily insulting to another person will not be labeled as such. This is a one class classification problem. The predictions will be a real number in the range [0,1] where 0 indicates 0% probability of comment being an insult, and 1 represents 100% insult.

## Related Work

Many attempts have been made to detect the abusive comments in on-line conversations. In 1997, the approach presented in *Smokey: Automatic recognition of hostile messages* [7] used static dictionary approach and some patterns were defined based on socio-linguistic observation to build a feature vector for training. But this suffers from high false positive rates and low coverage. In *Detecting flames and insults in text*[4] they tried to differentiate between insult and factual statements by parsing sentences and using semantic rules. But this work doesn't distinguish between the insults directed to non-participant and participant of conversation. In the work presented by Razavi et. al [6] they made use of bad words dictionary on top of bag-of words features in three-level classification machine learning model they proposed. The limitation with this is model is that the dictionaries available are small. Another work by Carolyn P. Rose et. al [8] in classifying offensive tweets builds topical features and lexicon features using some dictionary. Even this work does not distinguish between the insult directed towards the participants and non participants of the conversation.

## Methodology

Here is my general approach. The below subsections discuss each step in detail. Initially all texts are preprocessed and tokenized. The feature extraction is done. Since a huge number of features will be extracted, all of them cannot be used due to high computational cost. So few best features will be selected through feature selection. Finally classification is done.

## Text Normalization

We cannot use the raw data obtained from the Internet to learn models. We need to input the data into machine learning algorithms and hence pre-processing of the data needs to be done to convert into input format. During pre-processing care needs to be taken so that data is not lost during the process. Initially normalization of the text needs to be done.

### Removing unnecessary strings

Encoding parts like \\xa0, \\xc2, \\n, etc. and links, html entities and codes are removed.

### Stemming

During conversations we use different forms of a word such as *organize*, *organizes* and *organizing* or like *democratic*, *demomcracy* and *democratization*. The goal of stemming is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Hence if stemming isn't done, the number of features unnecessarily increase.

### Correcting the common words

For easiness of conversations we tend to write short forms of words like *s* for *yes* and *f9* for *fine* etc,. If we can convert these types of words to their original correct form then we can reduce the size of feature set and can also improve the accuracy of models. Hence a map from commonly used words to their normal form is created manually and the data set is normalized. Sometimes we change the words like "@$$hole" for "asshole", "!d!ot" for "idiot" etc. But these words are as important as the words in normal form to detect the abusive comments. A dictionary of approximately 500 words containing most of the possible ways an insult word can be written has been used and when these words are encountered they are converted to their normal form. So, before normalizing, if a particular statement looks like ""\xa0you are truly a $tup!d.\xa0 just shut the f u c k up." it changes to "you are truly a stupid. just shut the fuck up." after normalizing.

## Feature Extraction

Since we can input only numerical values into a machine learning algorithm, the texts should be converted into numerical vectors. n-grams with n = 2,3,4,5 and 1,2,3,4 skip grams are used to construct the feature vector.

Since we are aiming at insults directed towards participants in the conversation, the insult should probably contain atleast a word directing the second person such as "you", "you're" etc,. This feature should greatly improve the accuracy.

### Tokenization

The text is split into tokens which can be characters, words or n-grams.

### Counting

Number of times of occurrence of tokens in text strings has been enumerated. N by V matrix has been constructed where N is the size of the training data and

V is the size of the vocabulary representing all the text strings where feature for a text string is the number of occurrences of each token in that text string.

### Normalization

Few words like *the* and *he* etc, occur in most of the strings. After counting, we may consider these as important features due to high occurrence frequency and hence their importance need to be reduced and this can be achieved by vectorization for finding score which measures the relevance of word.

### Tf-idf Score

Tf-idf stands for *term frequency* $\times$ *inverse document frequency*, and the tf-idf weight is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Tf-idf can be successfully used for stop-words filtering in various subject fields including text summarization and classification. [9]
**TF: Term Frequency**, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length (aka. the total number of terms in the document) as a way of normalization:
TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
**IDF: Inverse Document Frequency**, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
IDF(t) = $\log_e$(Total number of documents / Number of documents with term t in it).

The tf-idf score of each feature is found. The item in the tf-idf vector relates to the tf-idf weight of the feature at the same index. The keyword's tf-idf weight is its term frequency divided by its document frequency. Then, the tf-idf vector is normalized.

## Feature Selection

The generated features will be generally of the order 1000,000. And hence algorithms cannot be handled efficiently with these huge number of features. Hence a few best efficient features(relatively small number) need to be selected

to run the machine learning algorithms. "Chi-Squared Test" can be used to select required small number (of range 3000) of features.
**Chi-Squared Test**: The x2 test is used in statistics, among other things, to test the independence of two events. More specifically in feature selection we use it to test whether the occurrence of a specific term and the occurrence of a specific class are independent. Thus we estimate the following quantity for each term and we rank them by their score.[5] In this context,the variables are: The label of the sentence: "insult or not" and the presence or absence of a feature". Higher this value, more related is the two variables. Lesser the value, more independent are the variables. The features which score high in this test are efficient features and the other features are discarded.

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

where i = {present or absent} and j = {insult or not insult}

## Classification

After selecting the best features, SVM and logistic regression algorithms are applied on selected features. To achieve the required results both the algorithms are combined.

## Evaluation and Results

Accuracy of 86.5% has been achieved for the above model including skip grams feature.
Accuracy of 86.7% has been achieved when second-person feature has also been included. The Precision is .76 and Recall is 0.72
I have noticed that using skip grams hasn't improved the accuracy much. These results change with different data sets.

## Conclusion

I have tried modeling a machine learning approach to the problem proposed in the beginning. I have used SVM and Logistic Regression to train the models. Examining the results, I think a lot can be done to improve the accuracy, mainly by looking at the false positives and modeling robust methods in feature selection. And also looking at the data sets, insults made sarcastically need to be taken care of which hasn't been looked at in this project.

## Acknowledgement

The bad-words file used for this project is available online[1]. I would like to thank "kaggle.com" online forums for really posting useful discussions.[2]. The data sets are available at kaggle website.[3]. I have used open source "Scikit-Learn", "nltk" and few other packages in python (version 2.7.8).

## References

[1] List of badwords. http://urbanoalvarez.es/blog/2008/04/04/bad-words-list/.

[2] Previous discussions on the forum. www.kaggle.com/c/detecting-insults-in-social-commentary/forums.

[3] Used data set. www.kaggle.com/c/detecting-insults-in-social-commentary/data.

[4] Altaf Mahmud, Kazi Zubair Ahmed, and Mumit Khan. Detecting flames and insults in text. 2008.

[5] Online. . http://blog.datumbox.com/using-feature-selection-methods-in-text-classification//. [Online; accessed 20-March-2015].

[6] Amir H Razavi, Diana Inkpen, Sasha Uritsky, and Stan Matwin. Offensive language detection using multi-level classification. In *Advances in Artificial Intelligence*, pages 16–27. Springer, 2010.

[7] Sara Owsley Sood, Elizabeth F Churchill, and Judd Antin. Automatic identification of personal insults on social news sites. *Journal of the American Society for Information Science and Technology*, 63(2):270–285, 2012.

[8] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1980–1984. ACM, 2012.

[9] Bertrand Decoster; Fayadhoi Ibrahima ; Jiyan Yang. . http://www.tfidf.com/. [2012Online; accessed 20-March-2015].