Toxic Comment Classifier
Progress Report

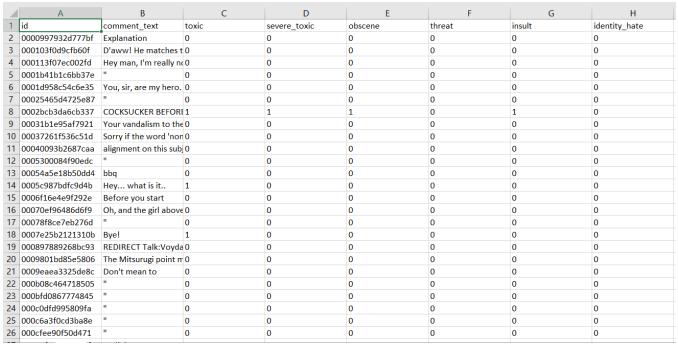
Aditya Sridhar, Chandan Shankarappa
College of Computer and Information Science, Northeastern University
sridhar.ad@husky.neu.edu, shankarappa.c@husky.neu.edu

Changes

No changes to report.

Pre-processing

Human-labeled data for comments is generally not so common-place, and the corpii are usually not large. But considering that the inspiration for this project was the competition held by Kaggle, we were presented with a labeled and relatively clean dataset. The training data is provided as csv file, and the image below represents the format of the instances.



	A	B	C	D	E	F	G	H
1	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
2	0000997932d777bf	Explanation	0	0	0	0	0	0
3	000103f0d9cfb60f	D'aww! He matches t	0	0	0	0	0	0
4	000113f07ec002fd	Hey man, I'm really n	0	0	0	0	0	0
5	0001b41b1c6bb37e	"	0	0	0	0	0	0
6	0001d958c54c6e35	You, sir, are my hero.	0	0	0	0	0	0
7	00025465d4725e87	"	0	0	0	0	0	0
8	0002bcb3da6cb337	COCKSUCKER BEFORI	1	1	1	0	1	0
9	00031b1e95af7921	Your vandalism to the	0	0	0	0	0	0
10	00037261f536c51d	Sorry if the word 'non	0	0	0	0	0	0
11	00040093b2687caa	alignment on this sub	0	0	0	0	0	0
12	0005300084f90edc	"	0	0	0	0	0	0
13	00054a5e18b50dd4	bbq	0	0	0	0	0	0
14	0005c987bdfc9d4b	Hey... what is it..	1	0	0	0	0	0
15	0006f16e4e9f292e	Before you start	0	0	0	0	0	0
16	00070ef96486d6f9	Oh, and the girl above	0	0	0	0	0	0
17	00078f8ce7eb276d	"	0	0	0	0	0	0
18	0007e25b2121310b	Bye!	1	0	0	0	0	0
19	000897889268bc93	REDIRECT Talk:Voyda	0	0	0	0	0	0
20	0009801bd85e5806	The Mitsurugi point m	0	0	0	0	0	0
21	0009eaea3325de8c	Don't mean to	0	0	0	0	0	0
22	000b08c464718505	"	0	0	0	0	0	0
23	000bfd0867774845	"	0	0	0	0	0	0
24	000c0dfd995809fa	"	0	0	0	0	0	0
25	000c6a3f0cd3ba8e	"	0	0	0	0	0	0
26	000cfee90f50d471	"	0	0	0	0	0	0

The first column represents the ID of the sentence, the second column is the sentence itself, the subsequent columns represent the various classes/types of toxicity of the sentence. The value 1 indicates that the sentence is of that type and 0 indicates the sentence isn't of that type. For the purpose of our project, we are only interested in the $obscene$, $threat$, $insult$ and $indentity\ hate$ columns. There are about 16000 such sentences in both the training data and test data.

We have a huge chunk of test data without any labels on them to evaluate. So currently, we are splitting the training data into training, cross-validation, and test sets. For most of the models we ran/plan to run, the data processing will remain the same. We clean the data, sentence by sentence. For each sentence, we removed punctuations, converted all the upper case letters to lower case, converted contractions to its actual forms (ex - you're $\rightarrow$ you are) and stored the sentences in a list and it's corresponding column types in a dictionary. This was however a potentially bad approach of cleaning the data as sometimes, toxic comments could appear in all capital letters with a lot of punctuations such as '!!!' and '...'. So we decided to filter the punctuations instead and keep the cases as they are instead of turning it into all small letters. For the perceptron model, where we just classify if the comment is toxic or not, we create another list of booleans corresponding to each sentence. If even one of the 4 columns has the value 1, we set it to true, implying the comment is toxic.

Method

The multi-label classification problem differs significantly from the multi-class classification problem. This stems from the fact that a particular training instance could belong to multiple classes. This necessitates a different approach to modeling the classifier. Some of the common strategies include - building multiple classifiers (one vs all), chaining classifiers, label powersets, adaptive algorithms, and ensemble models.

In order to establish a benchmark against which we could evaluate other models, we started with the Naive Bayes classifier. We used a TF-IDF vectorizer available in scikit-learn, to parse the training data and transform it into features for training the classifier. Considering the fact that this is a multi-label classification problem

and not a multi-class classification problem, we used a One Vs Rest based approach coupled with a multinomial naive bayes model. The latter was chosen because it is better suited for text classification than Gaussian NB models.

Considering that multi-layer perceptron models naturally lend themselves to multi-label classification problems, we decided to use it as our first experimental model. For the multilayer perceptron model, the sentences were first converted into vectors using scikit's vectorizer. Each of the toxic sentences were given a binary representation of 1 and non toxic sentences, 0. We then utilized a module called 'GridSearchCV', which we can take in different parameters in the form of a dictionary of lists and run the training data across each of the combinations of the parameters and outputs the best ones. We used a variety of parameters and let the machine do its work for a day. We ran this model to get an understanding of the data set.

Results and Evaluation

The Multinomial Naive Bayes classifier yielded an accuracy score of 91.28% and forms the baseline against which we will benchmark the other models to be considered.

The perceptron model was implemented with not too many problems and the GridSearch approach performed well, optimizing the hyper-parameters, and yielded a precision of 97.86%. We used two hidden layers of size 10×10, the $relu$ activation function, and the $adam$ solver.

For evaluation, we already had the sentences marked into various types and since we used a part of the data for testing, we wrote a script to do the evaluation for us. Now that we have established a baseline and have a better understanding of the data set, we plan to improve this using more sophisticated models such as SVM, Random Forests, and LSTM. Lastly, it is worth noting that since this is a multi-label classification problem, we plan to use other metrics to evaluate the predictions, such as hamming loss, group precision-recall, in addition to looking for exact matches.

What is working and What is wrong

The perceptron model worked well and did not pose any particular difficulties. It enabled us to get a good gauge on the data set. Instead of using a built-in vectorizer, we could have implemented one ourselves, where each word was carefully examined using features such as bigram models and given certain weights. This would have given an improved performance in classification.

Naive Bayes was our go to model to establish a standard benchmark to compare our future work. This decision was further bolstered from the discussion we had with the professor and from our previous experience with machine learning. We ran into a few implementation issues, however. Initially, when we began, we did not use the One vs $Rest$ approach. We instead, tried to have it classify as one of 4 types. This chose the one type that closely matched the sentence. This worked very well but it was not what we intended to achieve. We wanted our classifier to be able to detect all types of classes and not just one. After some due reading, we decided to implement the One vs $Rest$. This baseline model worked better than we expected (91.28%), however, there still is a lot of tuning left to be done.

Future work and Work in progress

We are examining ways to improve the models that we have currently implemented. There is a lot of scope for tuning parameters used in the vectorizer, and the models. The TF-IDF vectorizer currently uses 5000 features, but this could be tuned using cross validation. We noticed that given the large amount of text data is cumbersome with regards to the number of features that it generates. So we are considering the possibility of investigating the HashingVectorizer that is available in scikit-learn. Another thing that could be optimized using cross-validation is the n-grams considered in the vectorization process.

Apart from tuning the existing classifiers, we are working on building a SVM model using the One vs $Rest$ method. The other classifiers that we plan to implement before the final presentation are the following - random forests classifier, and LSTM. Lastly, We also

plan to add visualizations from our exploratory data analysis to augment our descriptions of the data.

While there are plenty of binary/sentiment classifiers, there are very few that can establish the levels of toxicity, which explains the basis for the Kaggle challenge.

References

1. Ex Machina: Personal Attacks Seen at Scale - Ellery Wulczyn, Nithum Thain, Lucas Dixon - link
2. Toxic Comment Classification Challenge - Kaggle - link
3. Wikipedia Talk Labels: Personal Attacks - link
4. Wikipedia Talk Corpus - link
5. Wikipedia Talk Labels: Toxicity - link
6. Unintended Bias Analysis - Perspective - link
7. Introduction to multi-label classification - link
8. Document classification - link
9. Multiclass and multilabel algorithms - scikit-learn link
10. Multi-Label Classification: An Overview - Grigorios Tsoumakas, Ioannis Katakis link
11. Baselines and Bigrams: Simple, Good Sentiment and Topic Classification - Sida Wang, Christopher D. Manning link