Toxic Comment Classification

Team: Dark Knights

Kotha Tejas
IMT2016112
International Institute of Information Technology
Bangalore
kotha.tejas@iiitb.org

Tanmay Jain
IMT20161023
International Institute of Information Technology
Bangalore
tanmay.jain@iiitb.org

***Abstract*—Online discussion about things one cares about can be difficult. The threat of abuse and harassment makes many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments.**
We are trying to understand negative online content, especially the comments (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). We are presenting a model using traditional machine learning methods to tackle toxic behaviour on the web. Our methodology allows us to explore some of the open questions about the nature of online personal attacks

***Index Terms*—Feature Engineering, TFIDF Vectorizer, One vs Rest classifiers, Grid Search, Cross Validation, Logistic Regression, XGBoost, AdaBoost, EasyEnsembleCLassifier, Naive Bayes, Random Forest Classifier, SGD Classifier, Voting, Stacking, Blending**

Introduction

In the ever growing world of social media especially Reddit, Twitter and Facebook, there is an evident rise in the sharing of the content which often leads to discussions and debates in the comments section and most of the time these can be vulgar or threatening which might make discussing issues or opinions difficult for people on these platforms. Hence, there is an urgent need to keep a check on the toxic (negative) content on the internet so as to facilitate users to just share things they care about or to spread a message across the masses.

The challenge of effective policy making to identify and appropriately respond to harassment is compounded by the difficulty of studying the phenomena at scale. Wikipedia for instance has 63M English talk page comments. Even using crowd-workers, getting human-annotations for a large corpus is very expensive and time consuming [1].

The report presents a methodology for quantitative, large-scale analysis of a large corpus of online comments with the help of traditional machine learning methods. The corpus consists of comments from Wikipedia which were labelled by crowd workers with high levels of inter-annotator agreement. We use this data to train a machine learning classifier and experiment with features. Our model validates and extends the findings of Nobata et al. [2]: character-level ngrams result in an impressively flexible and performant classifier for a variety of abusive language in English. With the help of our visualization and model we aim to help answer the obvious questions about the toxicity of online comments: What sort of toxicity is most common? When do attacks result in a moderator action? Who are the most vulnerable? etc.

The rest of the paper proceeds as follows: **Sec. 1** discusses related work on the prevalence, impact, and detection of personal attacks and closely related online behaviors. In **Sec. 2** we describe our dataset **Sec. 3** covers our visualizations, EDA and their inferences **Sec. 4** will discuss about Data preprocessing and feature extraction **Sec. 5** will cover model-building and evaluation approaches. **Sec. 6** will discuss training methods and comparison of our model with the various other approaches. We conclude in **Sec. 7** and outline challenges with our method and possible avenues of future work.

I. Related Work

The problem to comment classification is that there isn't a clear cut distinction between the classes of toxicity. The labels available are by the grace of crowd-workers which again might be dependent on their own individual perspective. The Google Jigsaw and Wikimedia released a paper in 2017 [1] which talked in detail about the data collection and labelling process which talks about the measuring scale of toxicity, understanding the demographics of users involved in such activities and model building with the help of Logistic Regression and Multi Layer Perceptrons.

While our work focuses on toxicity on online comments, there are projects that explore online harassment ([3], [4]), cyber-bullying [5] and various other antisocial behaviours on online discussion platforms [4] and their effects on victims offline like depression, quitting the forums and even suicides. In the recent years there have been number of papers involving the traditional sentiment analysis and spam detection techniques which are tackling the toxic behavior. These approaches include the use of Support Vector Machines by Yin et al.'s 2009 paper [3]. Dinakar et al.'s [6] approach on the approach of cyberbullying by training separate classifiers for training based on gender, race, location etc. on YouTube

comments. But most importantly, Nobata et al.'s [2] work on character n-grams, linguistic models on Yahoo! news comments to detect abusive language and Cheng et al.'s [4] approach of using Random Forest and Logistic Regression techniques to predict the users who are most likely to be banned for antisocial behaviour from several news sites have been the inspiration for our approach in the given problem statement.

II. Dataset

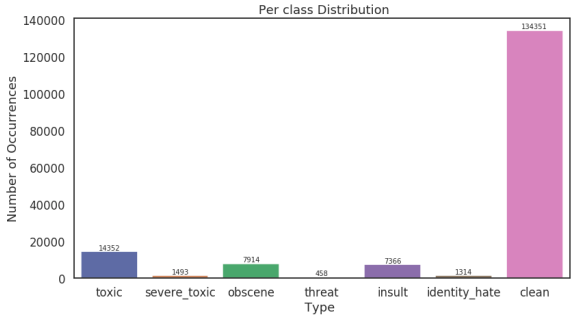
The dataset used in the project is a part of the Toxic Comment Classification's dataset of comments from Wikipedia's talk page edits by Jigsaw/Conversation AI (Alphabet) which was hosted on Kaggle.
Each data point in the training data consists of 8 columns, namely: id, comment_txt (content of the comment) and labels with 6 types of toxicity (toxic, severe_toxic, obscene, threat, insult and identity_hate) which were labelled by human raters. There are 1,49,571 datapoints in the training data and the primary test data consists of 10,000 datapoints for which the labels are not released.

Given the categories and the data, it's understandable that the problem is a multiclass and multilabel problem which means that a comment can be categorized as more than one class. We added an additional column in our dataframe with the label clean which will have 1 if it doesn't belong to any of the toxicity categories. This is a key step to understand the dynamics of our data.

III. Observations

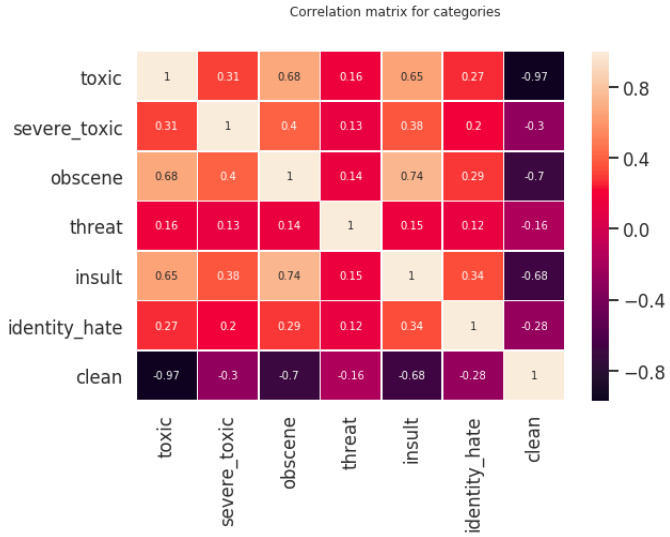
Before getting into preprocessing and feature extraction, it is very important to get to know the distribution of data in order to get better insights while feature selection. We are presenting a few of those here:

1) Distribution of data over the class labels in the dataset.



This shows that maximum data we have is actually clean, i.e., there is no toxicity in maximum comments. It is clear that the major labels are toxic, insult and obscene.

2) Correlation Between Categories

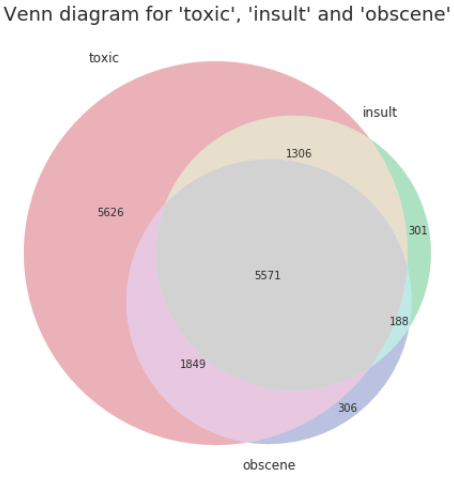


The categories which need to be worked on are:

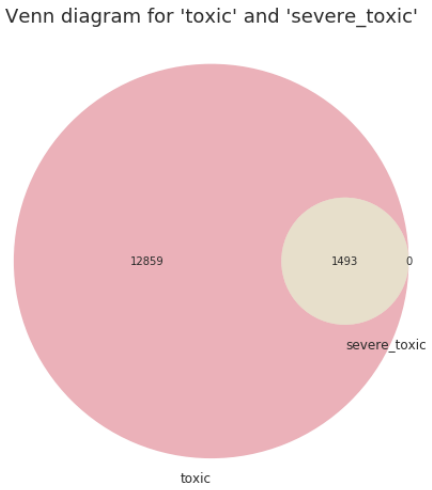
a) toxic and obscene
b) toxic and insult
c) insult and obscene
d) toxic and severe_toxic

3) Evaluation of overlap in classification of categories

The library used to plot the venn diagrams support a maximum of 3 objects at a time, so we had to plot 2 diagrams to visualize the categories. The best choice of combinations, based on the correlation matrix were 'toxic', 'insult' and 'obscene' in one diagram and 'toxic' and 'severe_toxic' in the other.



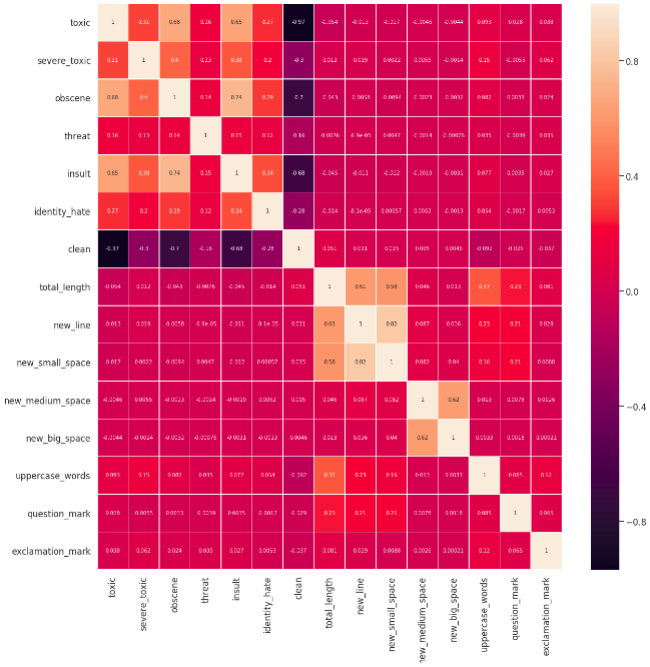
This venn diagram shows that almost every comment that is labelled 'insult' or 'obscene' is also labelled as 'toxic'. There are 3610 comments that are labelled with all 3 categories. There are only 183 comments that are just 'obscene' and only 191 that are 'insult'. It is clear from the venn diagram that the data is heavily skewed.



This venn diagram confirms that the 'severe_toxic' category is completely contained in the 'toxic' category, which was very evident by their names. The small correlation factor for these two categories is also explained by the fact that 'severe_toxic' category is only a small fraction of the 'toxic' category.

4) Correlation Between Categories and Features

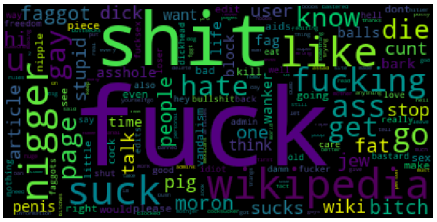
The added features show no strong correlations as such. But, one thing to note is that the 'uppercase_words', which can be safely assumed to be "yelling", are slightly more correlated. We can also note that 'uppercase_words' are correlated with 'exclamation_mark' up to 0.13 which means people generally use a lot of 'exclamation_marks' while "yelling".



5) Word Analysis - Word Cloud for all classes

We need to understand what kind of data we are playing with. That not only means knowing the data distribution over the various classes but also it's distribution within the classes.

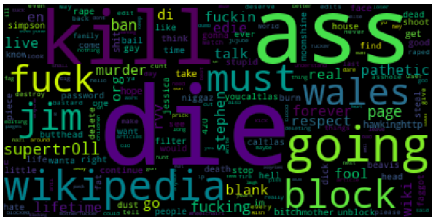
Toxic class



Severe Toxic class

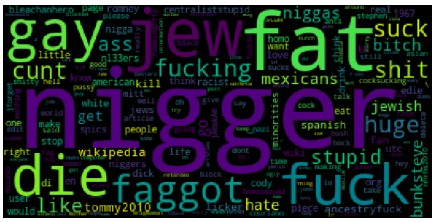
Obscene class

Threat class



Insult class

Identity Hate class



It is clear from the wordclouds that many words are common to all the classes. This makes it tough to train the model as the data is highly similar. The most commonly occurring words being fuck, shit, suck, nigger etc. Nothing much can be inferred from these wordclouds apart from the kinds of words we are dealing with here!

IV. Data Preprocessing and Feature Extraction

To make the training better we need to modify the data, which essentially means that the we need to convert the raw data we were given to more sophisticated data which would then be ready to undergo subsequent processing. Basically, we need to extract meaning out of raw data before we can use it to train our models. Models trained on meaningful data always have a upper hand to those trained on raw/less meaningful data. Hence, preprocessing the data is a very crucial step in training our model.

We used a couple of preprocessing techniques from packages like regex, re and nltk. We remove contraction from the words, like "don't" become "do not" and "I'm" becomes "I am". We further convert emoticons to words, like ":)" becomes "smile", ":(" becomes "sad", etc. Apart from modifying the data, we also need to clean the data. We removed all numbers and hyperlinks (which can be easily found by looking for "http" or "www" in the comments) because these won't be contributing to the toxicity of the comments. Further, as the data need not be grammaticaly correct, it just needs to make sense to the algorithm to train on, we removed any extra blank spaces and punctuations from the comments. This might make it hard for humans to read and understand these comments, but it would help the vecotizer deal with clean data. We noticed that there are a lot of redundant words, not by the frequency of occurrence of a particular word but by the shorthand way of writing, which is pretty common in texts. For example, "don't", "shouldn't", "hasn't", etc. These words have the same meaning as "do not", "should not" and "has not", but because they are different for a machine, they unnecessary increase the number of features the machine needs to train on. So we removed all such occurrences of the former forms from the comments and replaced them with their latter equivalents. We extended the above thought to parts of speech as well. Words with morphological affixes like "walking", "walked", "walker", etc. should all mean the same as "walk" in a machine's perspective to detect toxicity level in comments. For this, we used Snowball Stemmer (nltk) which removes all such different forms of words and replaces them with just the word stem. This too reduces the number of features to a large extent.

About TF-IDF Vectorizer: It is a numerical statistic that is intended to reflect the importance of a word in a document of a collection or corpus. TF-IDF vectors are calculated in the following way:

1) *Calculate term frequency (TF) in each document.*
2) *Calculate the inverse document frequency (IDF): Take the total number of documents divided by the number of documents containing the word.*
3) *Iterate each document and count how often each word appears.*
4) *Calculate TF-IDF: multiply TF and IDF together. [7]*

To extract the features from the comments we used TF-IDF vectorizer over the combined text of train and test in order to capture all the words occurring in the data with maximum features as 30,000 for an n gram of (1,1). In addition to using TF-IDF vectorizer for the words we also employed it on character n-grams of these words (1-4 including spaces) with maximum features of 23,000 to model the types of conscious or unconscious abbreviations or reductions of offensive words by the users [2].Then stacked both these matrices using scipy's hstack [8].

Then we tried dimentionality reduction using Truncated SVD. This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). Contrary to PCA, this estimator does not center the data before computing the singular value decomposition. This means it can work with scipy.sparse matrices efficiently [9]. But given the huge number of features extracted from TFIDF (53,000) over 1,59,571 datapoints, constructing a Truncated SVD of them became costly in terms of Memory and was throwing a Memory Error, hence we dropped the use of Truncated SVD. This was a tradeoff we had to make in the very early stages of the project while training as using a PCA would have reduced our training time considerably.

V. Model Selection

We used a one vs rest classifying approach in all our models. Our early stage models were brute force trying to go through different types of classifiers and ensemble classifiers and implementing them in our problem statement with out much of parameter tuning because of which models generated with algorithms like RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier and XGBClassifier were giving very less accuracy ranging from 82 to 94%. Later with the help of Grid Search over these classifiers one by one we were able to get their accuracy in the range of 97-98%, but given the number of features, doing a GridSearch followed by cross validation turned out to be really hard on the memory and time. Hence, even after achieving comparable results we didn't pursue them forward.

As far as the individual classifiers go, Logistic Regression and SGDClassifier (Stochastic Gradient Classifier) outperformed rest of the Naive Bayes generated models and ensemble algorithms mentioned earlier most of the times and given the extremely less time needed to train compared to others we decided to continue working with these two classifiers (The

best scores of each of these models is given in the following table). The least accuracy was shown by KNN Classifier with a mean of 75% at best. The best accuracy score of these individual models got to a saturation point of around 98.5%. Then we started looking into various ensembling techniques like VotingClassifier (scikit-learn [10]), Blending (custom made) and StackingClassifier (mlextend library) [11], EnsembleClassifier and BaggingClassifier after reading an article on MLWave.com which discusses about ensembling guides in detail [12].

We tried all of the above mentioned ensemble techniques with the previously mentioned algorithms like Logistic Regression, RandomForestClassifier, SGDClassifier, ComplementNB and AdaBoost. Given the complexity and time consumption we shifted most of our model experimentation and training to Kaggle and Google Collab Kernels. For some reason we weren't able to include XGBClassifier in VotingClassifier as it showed Memory Error, which we assumed to be due to insufficient RAM. Stacking and Blending didn't show promising results as these techniques generally require hundreds and thousands of models to work with for achieving extraordinary scores. Hence, rather than trying to work with such classifiers which are complex in terms of understanding and parameter tuning we chose to work with VotingClassifier which gives a weighted average of prediction probabilities of the estimator classifiers used in it. This gave the best result of 0.98702.

During the hackathon, we had to deal with an extremely skewed dataset. So, we learnt about oversampling and undersampling techniques by reading AnalyticsVidhya's article [13] to tackle this issue. A library called imblearn (imbalanced learn), built on top of scikit-learn makes it very easy to adopt those ensemble classifier techniques. As our accuracies for each class were almost close and were above 98% we didn't realize till then that even the present dataset is highly skewed. Hence, we started using EasyEnsembleClassifier, an undersampling classifier from imblearn which is a bag of balanced boosted learners with base estimator as Logistic Regression. Though this approach didn't give us a much better accuracy, we have been using it as it gives a bit of generalization to the model. Given that this is a type of Bagging classifier it's expected to be time consuming considering the number of estimators hence, we chose to have only one EasyEnsembleClassifier i.e., with LogisticRegression, and the rest of the classifiers like SGDClassifier, ComplementNB and RandomForestClassifier in the usual fashion in the VotingClassifier which gave us an accuracy of 0.987500 placing us on the top of the leaderboard for the second time.

Our most recent approach is a bit more systematic and has been adapted from the works of Cheng et al. [4] suggesting the use of Logistic Regression and RandomForestClassifier and Wang et al. [14] suggesting a simple but novel SVM/Logistic Regression variant using NB log-count ratios as feature values consistently performs well across tasks and datasets. Using Logistic Regression (solver being sag) suggested by Wang alone gave us an all time best accuracy score of 0.98765 but this model didn't perform well for a couple of classes namely, identity_hate and insult. To tackle this we wrapped this Logistic Regression with EasyEnsembleClassifier used another Logistic Regression with dual optimization being True (solver being liblinear), RandomForestClassifer and a SGDClassifier wrapped in EasyEsembleClassifier in the VotingClassifier with weights being 0.94 for EasyEnsebleClassifier(LogisticRegression),1.35 being EasyEnsebleClassifier(SGDClassifier),0.58 being LogisticRegression with dual optimization and 0.9 being RandomForestClassifier. The resultant model gave better results for those two classes giving us a huge leap in our accuracy (98.802%) placing us on top of the leaderboard again for the third time.

Why we didn't try multiclass classifiers?

With the use of TF-IDF Vectorizer the dimensionality of our matrix increased to 53,000 columns and as we know that the time taken for SVM to train over the data increases linearly with the datapoints and as the number of features is lesser than the number of observations, there were high chances of it not performing well. Even then we tried normal SVM and SVM with RBF kernel but weren't able to get the results as the kernel died out a few times. We understand that One vs Rest classifier algorithms don't always perform well and are not robust enough unlike the multiclass classifier but for a small number of classes it won't affect much. One vs Rest is a popular approach while solving multiclass - multilabel problem as in such cases multiple binary classifiers will work better given the number of classes is less.

TABLE I
RESULTS ON CROSS VALIDATION

Algorithm	toxic	severe toxic	obscene	threat	insult	identity hate	mean
LR [1]	0.9730	0.9875	0.9858	0.9839	0.9786	0.9765	0.9809
SGDC[2]	0.9729	0.9824	0.9857	0.9839	0.9792	0.9730	0.9795
CNB[3]	0.9742	0.9863	0.9853	0.9819	0.9789	0.9754	0.9803
RF[4]	0.9667	0.9790	0.9885	0.9402	0.9763	0.9722	0.9704
XGBoost	0.9679	0.9813	0.9853	0.9584	0.9775	0.9734	0.9739
V1[5]	0.9781	0.9893	0.9911	**0.9894**	0.9838	0.9837	0.9859
V2[6]	0.9787	0.9897	0.9921	0.9890	0.9844	0.9841	0.9864
NB-LR[7]	0.9814	0.9886	0.9925	0.9856	0.9850	0.9818	0.9858
NB-V1[8]	0.9821	0.9874	0.9926	0.9834	0.9852	0.9801	0.9851
Final	**0.9824**	**0.9901**	**0.9936**	0.9884	**0.9866**	**0.9843**	**0.9876**

[1]Logistic Regression
[2]SGDClassifier
[3]ComplementNB
[4]RandomForestClassifier
[5]Voting of LogisticRegression, SGDClassifier and ComplementNB
[6]Voting of LogisticRegression, SGDClassifier, RandomForestClassifier and ComplementNB
[7]Naive-Bayes LogisticRegression
[8]Niave-Bayes V1

VI. Training Details

We have been working on cross validation of cv=5. Though this means that the algorithm has to train on 80% of the data for five times for generating the best model which is time consuming but it gives a very good rough estimate of the test accuracy. This had been an important judging criteria of our submissions files given that we only had two submissions per day.

VII. Conclusion

We would like to conclude that we were able to come up with an efficient model to predict the toxicity of a given sentence. Such projects have a potential scope to make the internet a better place for open discussions.

Challenges and Future scope

One of the main challenge dealing with such sensitive topics is the way these labels are generated. The toxicity of the online comments might sometimes depend nationality or ethnic background of the users and given that these comments are labelled with the help of the crowd sources, it is tough to standardize the level of toxicity as most of the time it is the intuition/ perspective of the reader that helps decide the label. In such cases, Machine Learning is not the most novel approach but gives us comparitively faster results which is more generalised.

With the help of more features like who are the attackers and victims, we can come to a better conclusion of which communities are targeted, the patterns in the comments of the attackers can be analysed to come up with policies to regulate such unacceptable behaviour on the internet. This sort of analysis is fairly helpful when it comes to cyberbullying and harassment. Using neural networks and pretrained vectors we could visualize deeper correlations and achieve better results.

Acknowledgment

We would like to thank Professor G. Srinivas Raghavan and our Machine Learning Teaching Assistants, Ravi Theja, Chandana, Shayaan, Bhavuk and Pratheeksha for giving us the opportunity to work on the project and help us whenever we were struck by giving us ideas and resources to learn from. We would also like to thank Team Narang for being a great competitor and setting a benchmark time by time for the rest of us which acted as a driving fuel for us to constantly work hard and surpass them. A special thanks to Team 899129 (Satvik Ramprasad, Raghavan GV and Ananth Shrikumar), Shreyas Gupta and Tanishq Gupta for having a healthy discussions regarding various approaches in preprocessing, feature selection, algorithm optimizations etc.

We would gladly say that we had a great learning experience while working on the project. Leaderboard was great motivation to work on the project and the competition forced us to read up various articles and papers which gave us ideas and enthusiasm for the project.

Project Link

The link for our project files are available at https://drive.google.com/open?id=1kvBS6Vl7cYEalTegJaj_nMLKr7bI2qwz

References

[1] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," *CoRR*, vol. abs/1610.08914, 2016. [Online]. Available: http://arxiv.org/abs/1610.08914

[2] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," in *Proceedings of the 25th International Conference on World Wide Web*, ser. WWW '16. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016, pp. 145–153. [Online]. Available: https://doi.org/10.1145/2872427.2883062

[3] D. Yin and B. D. Davison, "Detection of harassment on web 2 . 0," 2009.

[4] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Antisocial behavior in online discussion communities," *CoRR*, vol. abs/1504.00680, 2015. [Online]. Available: http://arxiv.org/abs/1504.00680

[5] S. Pieschl, C. Kuhlmann, and T. Porsch, "Beware of publicity! perceived distress of negative cyber incidents and implications for defining cyberbullying," *Journal of School Violence*, vol. 14, no. 1, pp. 111–132, 2015. [Online]. Available: https://doi.org/10.1080/15388220.2014.971363

[6] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying." in *The Social Mobile Web*, ser. AAAI Workshops, vol. WS-11-02. AAAI, 2011. [Online]. Available: http://dblp.uni-trier.de/db/conf/icwsm/smw2011.html#DinakarRL11

[7] Kaitlyn, "Calculating tf-idf with python," 2017, [Online; posted 3-June-2017]. [Online]. Available: \url{https://methodi.ca/recipes/calculating-tf-idf-python}

[8] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed <today>]. [Online]. Available: http://www.scipy.org/

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[11] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[12] A. S. Hendrik Jacob van Veen, Le Nguyen The Dat, "Kaggle ensembling guide," 2015, [Online; posted 6-February-2018]. [Online]. Available: \url{https://mlwave.com/kaggle-ensembling-guide/}

[13] Upasana, "How to handle imbalanced classification problems in machine learning?" 2017, [Online; posted 7-March-2017]. [Online]. Available: \url{https://www.analyticsvidhya.com/blog/2017/03/imbalanced-classification-problem/}

[14] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ser. ACL '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 90–94. [Online]. Available: http://dl.acm.org/citation.cfm?id=2390665.2390688