INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact factor: 6.078
(Volume 7, Issue 1)
Available online at: https://www.ijariit.com

Profanity detection in social media text using a hybrid approach of NLP and machine learning

Raktim Chatterjee
raktimchatterjee.2018@gmail.com
Guru Nanak Institute of Technology, Kolkata, West Bengal

Sukanya Bhattacharya
sukanyabhattacharya99@gmail.com
Guru Nanak Institute of Technology, Kolkata, West Bengal

Soumyajeet Kabi
soumyajeetkabi12@gmail.com
Guru Nanak Institute of Technology, Kolkata, West Bengal

ABSTRACT

Profanity is socially offensive language, which may also be called cursing, cussing, swearing, or expletives. Nowadays where everything is digitally managed, there are lots of online platforms and forums which people use. If we take an example of any social media platform like Twitter, their privacy policy suggests that users cannot share or write any obscene/vulgar language on a public platform. Several corporate and research organizations discuss how such content is found and controlled, such as computer vision research has developed to detect illegal practices in public spaces, NLP has progressed to detect profanity in social media texts. However, existing profanity detection systems still remain flawed because of various factors. In this paper, we define and analyze the system which will use NLP and Machine learning approach to solve this. It is usually framed as a supervised learning problem. Generic features such as Bag-Of-Words or embeddings systematically deliver fair success in classification. Lexical resources in combination with models such as Linear Support Vector Machine (SVM); feature modeling specific linguistic constructs making it more effective in classification.

Keywords: *Linguistic, Classification, Features, Profane, Learning*

1. INTRODUCTION

In the current digital world, Social Media is a place to find news, friendly chat and meaningful information or breaking news. It also has some drawbacks because it is a forum for some users for knee-jerk reactions, trolls and persecution and that can be listed under the label of “Profanes”. According to research, on an average, one tweet out of 13 tweets will contain at least one cursing word on Twitter [1]. The reason is because the identity of the person remains anonymous in virtual world. They can share certain unhealthy content that they wouldn’t share in the real world. A “Sentiment Analysis” attempt, which has its drawbacks, showed that while negative feelings won out in swearing tweets, passion and playfulness(like two friends saying “you whore”) appeared in the data as actual signals, 22% and 17% of cursing tweets were characterized by rage and sorrow, while 7% appeared to express affection. Detection and deletion of profanity is often considered to be a simpler task relative to the difficulties of identifying profanes. Many existing mechanisms to track identification of profanity manage new content against large predefined databases of profane words. However, these systems are flawed in one major way. Relatively simple to beat are the predefined term-lists. By replacing one or more letters with punctuation marks and special characters (e.g., “f%@k”, “a$$h0!e”), users often impersonate profanity. Intentional or unintended misspellings (e.g., “bieetch”, “sluut”) and the usage of derogatory terms (e.g., “asshole”) that grow rapidly and sometimes have local variants often challenge profanity lists to be more extensive and adaptive than they may fairly be. Therefore, these pre-existing systems face recall problems; they are unable to capture certain instances of profanity [2]. In this paper we make two primary contributions to research on profanity detection using NLP and Machine Learning. First, we briefly address the existing systems present and which follows the predefined list based approach and discuss on the conditions on which they fail to perform accurately. Secondly, on the basis that lack of precision in terms of profane identification and time complexity is a big oversight in these systems, we investigate how our proposed method is different in terms of the factors listed above. In this section, we examine these through analysis of Twitter and Wikipedia data set of hate speech and comments from social media websites.

2. RELATED WORK

In this section, on offensive content filtering in social media, we review current approaches and software libraries.

2.1 profanity-A python library to check for (and clean) profanity in strings [3]

The method of the profanity package is to use normal censor characters (!@#$%) to censor terms. To build a more normal censor string, a pool of censor characters is used and it censors all instances of a given term with the same censor string for simple correlation. The whole package operates with 32 predefined word lists. It clearly defines profanity by searching for one of these terms.

2.2 Better_profanity-Blazingly fast cleaning swear words in strings [4]

It is the subset of the package "profanity" mentioned above; this library follows the same approach except for the fact that it uses a 140 predefined list of words. Also, this library is much faster as compared to its original version.

2.3 Profanityfilter-A universal python library for detecting and/or filtering profane words [5]
profanityfilter package too uses the approach but with 418 predefined list of words. The above mentioned systems are not favorable because profanity detection based on predefined wordlists are extremely subjective. For example: all the above three packages includes the word "suck" in their wordlist. Is it so that any sentence containing the word "suck" is profane? Furthermore, any hard-coded list of bad words will inevitably be incomplete-Is it so profanity's 32 or 140 or 418 list of bad words are the only ones out there?

2.4 profanity-filter-A python library for detecting and filtering profanity [6]
profanity-filter uses Machine Learning for profanity detection in text. The library detects not only the exact profane word matches but also derivative and distorted profane words using the Levenshtein automata, ignoring dictionary words, containing profane words as a part. But the problem with profanity-filter is that it's really slow! Below is a benchmark comparing (1) profanity-filter and (2) profanity *(*the one with list of 32 words*)*

Table 1*:* Benchmark comparing profanity-filter and profanity.

Package	1 Prediction (ms)	10 Predictions (ms)	100 Predictions (ms)
profanity-filter	60	1200	13000
profanity	0.3	1.2	24

From the above result, it's clear that even a human could likely perform the task quickly than profanity-filter can. profanity-filter is very slow to perform many forecasts in real time.

3. PROPOSED METHOD

3.1 Dataset

We merged two open access datasets for our experiment:

- The "Twitter" hates speech and abusive language dataset containing web tweets that were scrapped from Twitter [7].
- The "Wikipedia" dataset released by Conversational AI team of Alphabet, including contributions from the edits of the Wikipedia talk page [8].

Both the datasets contains text samples hand-labeled by humans through crowd sourcing.

3.2 Approach

We cleaned and combined both the datasets using cutting-edge technologies and furthermore the following methods were used to train the model:

3.2.1 Vectorization (Bag-Of-Words): In this step, any text string turns into a vector by computing the frequency of each word that is present. This is known as Bag-Of-Words (BoW) representation. For example, if the only words in the English languages were the, boy, sat and hall, a possible vectorization of the sentence the boy sat in the hall might be

Table 2*:* Vectorization example.

the	boy	sat	hall	??
2	1	1	1	1

The ?? represents any unknown word, which for this sentence is in. Any sentence can be represented in this way as counts of the, boy, sat, hall, and ??

Table 3*:* Sentence vs BoW Representation.

Sentence	BoW Representation
the boy sat	[1,1,1,0,0]
hall on a boy	[0,1,0,1,2]
boy boy boy boy boy	[0,5,0,0,0]

Obviously there are more English words in the vocabulary, so we used fit transform method [9]:

Fit: learns a vocabulary by looking at all words that appear in the dataset.

Transform: turns each text string in the dataset into its vector form.

3.2.2 Training (Linear Support Vector Machine): The model we implemented is a Linear Support Vector Machine (SVM).The goal of a support vector machine is to find the optimal separating hyper plane which maximizes the margin of the training data and further helps in classification. Here is an example to support the use of Linear Support Vector Machine in this case: The model learns through the training process which words are "profane" and how often they are "profane" because these words appear most frequently in offensive messages [9]. Thus, it is as if the training phase takes out the words "profane" from all imaginable variations of words and uses them to make potential predictions. This is way better than just depending on arbitrary dataset of word list chosen manually. We use the LIBLINEAR package [10] in particular, which has been shown to be useful for problems like this for text classification. For example, for the role of Natural Language Recognition with Machine Learning, which also comes under text classification methods, it has been demonstrated to be a very powerful classifier.

4. RESULT

The results of our experiment are as follows:

4.1 Confusion Matrix

Table 4*:* Confusion Matrix of our experiment.

Actual Predicted	Not Profane (0)	Profane (1)
Not Profane (0)	697 (TN)	20 (FP)
Profane (1)	87 (FN)	45 (TP)

4.2 Accuracy

Table 5*:* Accuracy of our experiment.

Package	Proposed Method	profanity-filter	profanity
Test Accuracy	87.4%	91.8%	85.6%
Balanced Test Accuracy	65.7%	83.6%	65.1%
Precision	69.2%	85.4%	91.7%
Recall	34.1%	70.2%	30.8%
F1 Score	0.46	0.77	0.46

4.3 Performance

Table 6: Performance of our experiment.

Package	Proposed Method	profanity-filter	profanity
1 Prediction (ms)	0.2	60	0.3
10 Predictions (ms)	0.5	1200	1.2
100 Predictions (ms)	3.5	13000	24

5. CONCLUSION

In this study, we investigated existing software packages that are used for profanity detection in social media text. Packages like profanity-filter use more polished methods that work more accurately but with a bottleneck of performance (time complexity). The proposed method that we have showcased is anywhere from 300-4000 times faster than profanity-filter in this benchmarking. We also provided an analysis of how profanity is detected. Our final conclusion is to never treat any forecasts from this proposed system as completely accurate, because it does and will make mistakes. Instead, we recommend using this as empirical system.

6. REFERENCES

[1] fastcompany, "140 Characters Of F*ck, Sh!t, And @ss: How We Swear On Twitter," [Online]. Available: https://www.fastcompany.com/3026596/140-characters-of-fck-sht-and-ss-how-we-swear-on-twitter.

[2] S. &. A. J. &. C. E. Sood, "Profanity use in online communities," in Conference on Human Factors in Computing Systems - Proceedings, 2012.

[3] B. Friedland, "https://pypi.org," Python Software Foundation, 2013. [Online]. Available: https://pypi.org/project/profanity.

[4] S. N. Thanh, "https://pypi.org," Python Software Foundation, 2020. [Online]. Available: https://pypi.org/project/better-profanity.

[5] Beigh, "https://pypi.org," Python Software Foundation, 2018. [Online]. Available: https://pypi.org/project/profanityfilter.

[6] R. Inflianskas, "https://pypi.org," Python Software Foundation, 2020. [Online]. Available: https://pypi.org/project/profanity-filter.

[7] T. Davidson, "https://github.com," GitHub, In, 2019. [Online]. Available: https://github.com/t-davidson/hate-speech-and-offensive-language/tree/master/data.

[8] Conversation AI, "https://www.kaggle.com," Kaggle, 2018. [Online]. Available: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data.

[9] V. Zhou, "victorzhou.com," 2019. [Online]. Available: https://victorzhou.com/blog/better-profanity-detection-with-scikit-learn.

[10] M. Z. Shervin Malmasi, "Challenges in Discriminating Profanity from Hate Speech," in Journal of Experimental & Theoretical Artificial Intelligence, 2018.