Offensive Text Detection using Hybrid NLP Approches

1st Manish Shetty M
CSE Department
PES University
Bangalore, India
mmshetty.98@gmail.com

2nd Neelesh C.A.
CSE Department
PES University
Bangalore, India
neeleshca26@gmail.com

3rd Pallavi Mishra
CSE Department
PES University
Bangalore, India
pallavim98@gmail.com

***Abstract*—Text messaging through the Internet or cellular phones has become a major medium of personal and commercial communication. In the same time, flames (such as rants, taunts, and squalid phrases) are offensive/abusive phrases which might attack or offend the users for a variety of reasons. An automatic discriminative software with a sensitivity parameter for flame or abusive language detection would be a useful tool. Although a human could recognize these sorts of useless annoying texts among the useful ones, it is not an easy task for computer programs. In this paper, we describe an automated classifier using various models that finds if a text is offensive or not. One basic Naive Bayes model was chosen as the base model, on top of which other models like a GRU based model was trained. A CNN was trained on character embedding which achieved the best accuracy.**

***Index Terms*—Convolutional Neural Network, Glove, Part-of-Speech, Naive Bayes**

I. Introduction

Recently, pattern recognition and machine learning algorithms are being used in a variety of Natural Language Processing applications. Everyday we have to deal with texts (emails or different types of messages) in which there are a variety of attacks and abusive phrases. An automatic intelligent software for detecting offensive, sexist, racist or other abusive language would be useful and could save its users time and energy. Offensive phrases could mock or insult somebody or a group of people (attacks such as aggression against some culture, subgroup of the society, race or ideology in a tirade). Here are several types of offensive language in this category:

- Taunts: These phrases try to condemn or ridicule the reader in general.
- References to handicaps: These phrases attack the reader using his/her shortcomings
- Squalid language: These phrases target sexual fetishes or physical filth of the reader.
- Slurs: These phrases try to attack a culture or ethnicity in some way.
- Homophobia: These phrases are usually talking about homosexual sentiments.
- Racism: These phrases intimidate race or ethnicity of individuals.
- Extremism: These phrases target some religion or ideologies.

There are also some other kinds of flames, in which the flamer abuses or embarrasses the reader (not an attack) using some unusual words/phrases eg:Unrefined language where in some expressions that lack polite manners and the speaker is harsh and rude. Based on the above definitions, when we say offensive text detection, implicitly we are talking about every context that falls into one or more of the defined cases ie: Offensive or not.

A. Dataset

To maintain the natural language assumption we chose the dataset from a machine learning competition which uses scraped tweets. The dataset consists corpus of tweets labeled as 0 or 1 ie: not offensive and offensive.
Link to dataset : https://tinyurl.com/dataset-nlp

- **Train.csv** - For training the models, we provide a labelled dataset of **31,962 tweets**. The dataset is provided in the form of a csv file with each line storing a tweet id, its label and the tweet. The same file was split as 80:20 for train and test.
- **Test_tweets.csv** - The test data file contains only tweet ids and the tweet text with each tweet in a new line. This file was not used since it was not labeled to check for accuracy etc.

II. BACKGROUND CONCEPTS

Text classification is the process of assigning tags or categories to text according to its content. Its one of the fundamental tasks in Natural Language Processing (NLP) with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection.

Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature. Businesses are turning to text classification for structuring text in a fast and cost-efficient way to enhance decision-making and automate processes. Text classification models apply machine learning, natural language processing, and other techniques to

automatically classify text in a faster and more cost-effective way.

There are many approaches to automatic text classification, which can be grouped into three different types of systems:

- Rule-based systems
- Machine Learning based systems
- Hybrid systems

We chose to stick to machine learning and deep learning models to train on the dataset.

A. Machine Learning for Text Classification

Instead of relying on manually crafted rules, text classification with machine learning learns to make classifications based on past observations. By using pre-labeled examples as training data, a machine learning algorithm can learn the different associations between pieces of text and that a particular output (i.e. tags) is expected for a particular input (i.e. text).

The first step towards training a classifier with machine learning is feature extraction: a method is used to transform each text into a numerical representation in the form of a vector. One of the most frequently used approaches is bag of words, where a vector represents the frequency of a word in a predefined dictionary of words.

For example, if we have defined our dictionary to have the following words This, is, the, not, awesome, bad, basketball, and we wanted to vectorize the text This is awesome, we would have the following vector representation of that text: (1, 1, 0, 0, 1, 0, 0). Then, the machine learning algorithm is fed with training data that consists of pairs of feature sets (vectors for each text example) and tags (e.g. sports, politics) to produce a classification model:

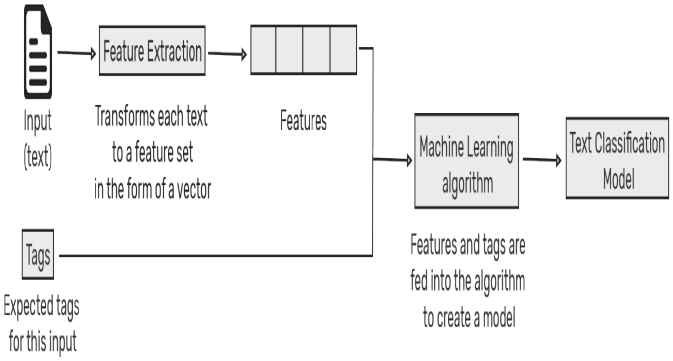

Fig. 1. Pipeline for text classification

III. Preprocessing

Some basic preprocessing was done to maintain the integrity of the natural text, yet not affect the model with noise. All the steps were not required for every model that was implemented. Some models required certain extra preprocessing to generate the feature vectors. The steps considered for preprocessing were

1) Sentence and Word Tokenize
2) Stop Word removal (if required for particular model)
3) Lemmatization of words
4) Removing non-alphabetic words

For one of the hybrid models an extra step of POS tagging was done. The POS tags were used to create count vectors.

IV. Proposed Methods

A series of basic models like Naive Bayes,random forest etc were implemented which provided good results as the data is not vast. Keeping in mind the large amounts of data,tweets or posts and the ever changing ways of communicating in natural languages, models that capture more information from the text than just co-occurrence and word level frequencies would do better. So our goal with this project was to come up with hybrid approaches that can do the same.

A. *Embedded POS Feature Vector with Glove and GRU*

A Neural Network was built with a long and short term memory component as the semantic similarity of words far away from each other could also contribute to the entire tweet being offensive or not. The following were the components used in the various layers of the network.

1) GloVe and POS tag counts: GloVe [7], coined from Global Vectors, is a model for distributed word representation.This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.
In our case a pre-trained Glove Vector model was used as an embedding layer of the GRU [9] based neural network. The layer would convert the given text into Glove vectors.

2) Hybrid Features using Hstacking: To this glove vector 5 extra text based features were appended. A number of extra text based features can also be created which sometimes are helpful for improving text classification models. Some examples are:

- Word Count of the documents : total number of words in the documents
- Character Count of the documents : total number of characters in the documents
- Average Word Density of the documents : average length of the words used in the documents
- Punctuation Count in the Complete Essay : total number of punctuation marks in the documents
- Upper Case Count in the Complete Essay : total number of upper count words in the documents
- Title Word Count in the Complete Essay : total number of proper case (title) words in the documents

Our model uses the approach of **Frequency distribution of Part of Speech Tags like Noun Count, Verb Count, Adjective Count, Adverb Count, Pronoun Count etc** appended to the glove vector to create a hybrid feature vector

for the tweet. NLTK [8] was used to to the above operations.

After the features were generated they were passed to a neural network with tweets into the initial GloVe embedding layer using a method called as Hstacking.

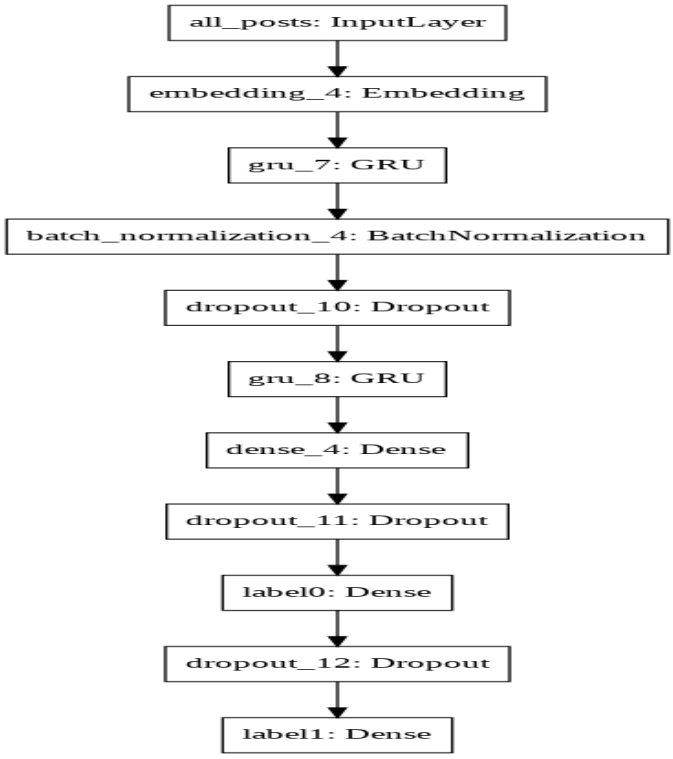

Fig. 2. GRU Neural Network Model

B. *Character level Convolutional Neural Network*

Convolutional neural networks are normally used in computer vision. CNN's are responsible for most computer vision systems, and are used in various applications ranging from photo detection like the one done by Facebook to self driving cars. Lately, research has been done to show that it can be used on textual data too with good results [1]–[6]. It seems like RNN's would be better suited to text classification since they take positions of the text into account, which CNN does not. CNN takes features as they are, and where they appear is not that important. But this is not a reason to not use them, as even something like a bag of words model which seems like a bad representation has been commonly used and to good effect. One of the big advantages of CNN's are that they are very fast. Convolutions are a central part of computer graphics and implemented on a hardware level on GPUs. Compared to something like n-grams, CNNs are also efficient in terms of representation. With a large vocabulary, computing anything more than 3-grams can quickly become expensive due to the increasing complexity.

Most of the models used for text classification are based at the word level. Usually something that can take order into account like n grams are used. There after, some combinatorics and statistics is used to find the resultant probability. But there is one issue with this kind of approach. In real world cases, there would be text that is misspelled, words never seen before, and things like emoticons, which while being part of language are not formally represented. Even though some form of pre-processing can take place, like minimum edit distance for misspelled words, add 1 smoothing for unseen words and hand written rules for emoticons, these are merely ways of overcoming deficiencies of the model and do not address the actual problem but rather move around it. A character level CNN can handle all of the above things using the text as it is, naturally learning them as part of the model, without external intervention.

A python keras model was used to implement the CNN. This requires the input to be in a numeric format, i.e. they had to be embedded. To ensure a limited size, some simple pre-processing like conversion to lowercase was done. There were a total of 70 characters including the unknown character. Reference [7] was used to design our model.

```
abcdefghijklmnopqrstuvwxyz0123456789
-,;.!?:’’’/\|_@#$%ˆ&*˜‘+-=<>()[]{}
```

Fig. 3. Characters considered for embedding

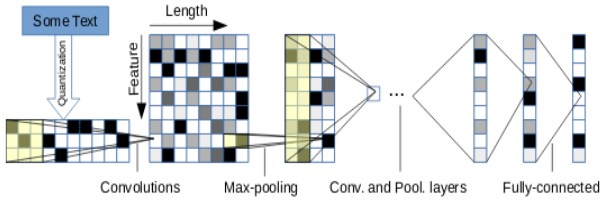

Fig. 4. Model used by Zhang et al. Our model is similar

As we do not have the necessary hardware to train models with many layers, we have 256 filters in our convolutional layer and 1024 output units in dense layer.

Layer	Large Feature	Small Feature	Kernel	Pool
1	1024	256	7	3
2	1024	256	7	3
3	1024	256	3	N/A
4	1024	256	3	N/A
5	1024	256	3	N/A
6	1024	256	3	3

Fig. 5. Convolutional layers used by Zhang et al.

Our model has two fully connected layers with 1024 neurons each. As it is a binary classification problem, the output layer has 2 neurons. To regularize the length of the text, it was restricted to 150. This was due to the tweet length restriction. This gave it a reasonably fast training time and accuracy.

Layer	Output Units Large	Output Units Small
7	2048	1024
8	2048	1024
9	Depends on the problem	

Fig. 6. Fully connected layers used by Zhang et al.

Although the amount of data for deep learning methods is usually larger than what we have taken, the results were still good.

V. RESULTS

The Basic models themselves did a great job using tf-idf or count based vectors as the classification problem was fairly simple. We also observed that having hybrid feature vectors does not necessarily increase the accuracy. The CNN model was the most optimal in terms of run-time compared to the other neural network models on text and at the same time provides good performance too. We observe that the GRU based model that used embedded feature vector layer is highly dependant on the features chosen for the scenario. Feature selection itself can be a form of hyper parameter tuning for this model.

TABLE I
MODEL COMPARISON

Model	Test Accuracy
Naive Bayes	0.95
GRU with POS embedding	0.94
Character level CNN	0.97

VI. CONCLUSION

This project was completed keeping in mind the use of hybrid approaches more than a way of improving accuracy of an existing approach. All models tested provided good results and some new approaches were discussed in this paper. The CNN model due to it's speed of execution suggests us to be the best model of choice considering the applications of finding offensive text. These can be used as recommendation systems in suggestive text or to monitor children picking up harsh language via social media. The applications to a small portion of sentiment analysis is thus vast.

VII. FUTURE WORK

Some models with more weighted features can be implemented with the CNN or the GRU embedding. The sentiment of each individual word can be found using existing unsupervised approaches (if any) and weighted with the vector representing that word. This could provide more insight into how much effect each word has to the entire sentence being offensive.

ACKNOWLEDGMENT

We acknowledge Professor Ashwini Joshi for her support and this opportunity.

REFERENCES

[1] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun.2018. An empirical evaluation of generic convolu-tional and recurrent networks for sequence model-ing.CoRR, abs/1803.01271.

[2] Nal Kalchbrenner, Edward Grefenstette, and Phil Blun-som. 2014.A convolutional neural network formodelling sentences. InProceedings of the 52ndAnnual Meeting of the Association for Computa-tional Linguistics, ACL 2014, June 22-27, 2014,Baltimore, MD, USA, Volume 1: Long Papers, pages655665. The Association for Computer Linguis-tics.

[3] Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, HengZhang, Fangyuan Wang, and Hongwei Hao. 2015.Semantic clustering and convolutional neural net-work for short text categorization.InProceed-ings of the 53rd Annual Meeting of the Associationfor Computational Linguistics and the 7th Interna-tional Joint Conference on Natural Language Pro-cessing of the Asian Federation of Natural LanguageProcessing, ACL 2015, July 26-31, 2015, Beijing,China, Volume 2: Short Papers, pages 352357. TheAssociation for Computer Linguistics.

[4] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun.2015. Character-level convolutional networks fortext classification.InAdvances in Neural Infor-mation Processing Systems 28: Annual Conferenceon Neural Information Processing Systems 2015,December 7-12, 2015, Montreal, Quebec, Canada,pages 649657.

[5] Rie Johnson and Tong Zhang. 2015.Effective useof word order for text categorization with convolu-tional neural networks. InNAACL HLT 2015, The2015 Conference of the North American Chapterof the Asso-ciation for Computational Linguistics:Human Language Technologies, Denver, Colorado,USA, May 31 - June 5, 2015, pages 103112. TheAs-sociation for Computational Linguistics.

[6] Mohit Iyyer, Varun Manjunatha, Jordan L. Boyd-Graber, and Hal Daum e III. 2015. Deep unorderedcomposition rivals syntactic methods for text clas-sification. InProceedings of the 53rd Annual Meet-ing of the Association for Computational Linguisticsand the 7th International Joint Conference on Nat-ural Language Processing of the Asian Federationof Natural Language Processing, ACL 2015, July26-31, 2015, Beijing, China, Volume 1: Long Pa-pers, pages 16811691. The Association for Com-puter Linguistics.

[7] Xiang Zhang, Junbo Zhao, Yann LeCun. Character-level Convolutional Networks for Text Classification. Advances in Neural Information Processing Systems 28 (NIPS 2015). Poster. Datasets. Code. Errata.

[8] Loper, Edward, and Steven Bird. "NLTK: the natural language toolkit." arXiv preprint cs/0205028 (2002).

[9] Tang, Duyu, Bing Qin, and Ting Liu. "Document modeling with gated recurrent neural network for sentiment classification." Proceedings of the 2015 conference on empirical methods in natural language processing. 2015.