Hate Speech Analysis on Roman Urdu Dataset

Dyass Khalid,Syed Mohammad Arham Noman

20100004@lums.edu.pk , 20100059@lums.edu.pk

April 2, 2020

Abstract

Hate speech detection has been a growing area of interest as the user base of social media websites has grown. Human content moderators can only do so much, hence automatic detection has become crucial. This paper explores the topic with a particular focus on Roman Urdu texts as these have not been explored.

1 Introduction

Due to the recent advancement in the field of ML particularly due to advent of deep learning networks, Natural Language Processing (NLP) has seen an increase in research papers. These advances have led to the adoption of NLP in automated speech recognition, question-answering systems, and more sophisticated NLP systems such as Siri and Alexa. However, recent applications such as exact voice copying with the voice generating hate-speech without even the people whose voice is being copied knowing is a big concern. To alleviate such accidents, there is a growing interest in hate-speech classification.

However, classifying hate speech accurately poses new challenges for the NLP community. First, due to multi-lingual nature of social media platforms, it become very difficult for the moderators to identify hate-speech. Furthermore, hate-speech for one community is not hate-speech for the other and vice-versa.

2 Methodology

A dataset containing Roman Urdu tweets was used for this paper. Three different soft computing algorithms were used: Neural networks, Genetic, and Neuro-Genetic.

2.1 Pre-Processing

An important step in preprocessing was removal of stop words in the dataset. However, the available stop words in roman Urdu did not sufficiently remove unnecessary words from the data. To further clean the data, we also removed words within an edit distance of 1 of the available stop words.We used the term-frequency inverse document(tf-idf) encoding to feed to the models we used.

2.2 Neural Network

The loss function used was binary cross-entropy. The neural network that was used contained 3 hidden layers with the configuration shown in the following figure.

Hidden layer number	Number of neurons	Activation Function
1	300	ReLU
2	50	ReLU
3	4	ReLU

Table 1: Neural Network parameters

2.3 Genetic Algorithm

The approach used for genetic algorithm is the same as the one described in[1]. Multiple algorithms were tested using Tpot library and the best one was selected. This approach allowed us to get the best genetic algorithm with optimized parameters and account for all types of classifiers. The tool was run for 5 generations to allow convergence.A total of 100 models were tested. They are provided in models.csv with relevant parameters reported.

2.4 Neuro-Genetic Algorithm

Genetic algorithm was used for hyper parameter tuning of Neural Network. The parameters tuned by genetic algorithm were batch size and epochs selection. The neural network has 3 hidden layers, the details of which are listed in the table below.

Hidden layer number	Number of neurons	Activation Function
1	300	ReLU
2	50	ReLU
3	4	ReLU

Table 2: Neural Network f parameters

3 Results

The results varied across the different approaches used, each approach had consistent results across multiple trials using different test and validation sets.

3.1 Neural Network

Epochs	Loss	Accuracy
25	5.67	0.3006

Table 3: Results of Neural Network

3.2 Genetic Algorithm

The detailed results of this algorithm can be found in the csv file that accompanies this paper, in the interest of saving space only key results are written here in the following table.

Maximum accuracy	0.607
Minimum accuracy	0.347
Average accuracy	0.528
Best classifier	Linear SVC with square hinge loss

Table 4: Results of Neural Network

3.3 Neuro-Genetic Algorithm

Neuro-Genetic results were by far the most promising. The use of genetic algorithm to select hyper parameters seemed to vastly improve the performance of a simple Neural Network.

Epochs	Loss	Validation Accuracy	Loss	Accuracy	Batch size
25	2.97	0.568	0.016	0.9918	10
50	2.86	0.577	0.015	0.9920	10
25	2.53	0.572	0.018	0.9915	20
50	2.71	0.572	0.016	0.9915	20
50	2.43	0.574	0.020	0.9917	30
50	2.66	0.566	0.0158	0.9913	30

Table 5: Results of Neuro-Genetic Algorithm

4 Conclusion

Neuro-Genetic had the best performance out of the three algorithms used in our approach, while Neural Networks had the worst performance with an accuracy of around 30%. Hybrid approaches seem to be better for this problem as the individual Genetic and Neural Network approaches did not provide results that were comparable to Neuro-Genetic approach.

4.1 Future work

It is clear that Neuro-Genetic is the best approach from the ones tested for this paper. Therefore future work should focus on hybrid models as they seem to work best for detecting hate speech. Even within Neuro-Genetic approach there is room for improvement, the genetic part can be extended to select activation functions, layers and number of neurons as well.

Another possiblity is the extension of this approach to different languages. The motivation for this paper was to build a detection system for a language that is not as popular as English for example.

4.2 Summary

Hate speech detection is a problem that will always need to be addressed. Diversity online is a reality and will only continue to grow as there are more users added everyday. This paper has attempted to highlight the need for diversity in hate speech detection by using one example of Urdu language, however there are

many more languages that do not have any work done in this regard. We hope that this will push for a larger change in this field.

References

[1] Randal S. Olson et al. "Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science". In: *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. GECCO '16. Denver, Colorado, USA: ACM, 2016, pp. 485–492.