A Survey on Effective Neural Networks for Toxicity Classification

Hitesh Goyal, Yash Tripathi
VIT Chennai, Tamil Nadu, India 600127
Email: hitesh.goyal2019@vitstudent.ac.in
yash.tripathi2019@vitstudent.ac.in

***Abstract*—With an increase in the need of online communication, multiple websites and applications have integrated an in-application messaging option. It is a very convenient and useful feature but it is also important to discourage any toxicity. Our aim is to make a toxicity detection module which can predict the amount of toxicity in a particular text. This module can help identify any toxic texts or even people which can then be reported to the authorities.**

1. Introduction

Word Embeddings in Natural Language Processing (NLP) are relatively new. Although there has been quite a bit of progress, since the development of GloVe by Jeffrey Pennington in August 2014, focused research is still required to further the field of NLP. Hence, in this research paper we are looking at different neural networks for processing GloVe Embeddings and in what ways they affect the accuracy.

2. Related Work

2.1. GloVe: Global Vectors for Word Representation

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. 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. The good part about this model is the amount of work done on GloVe. Stanford University has done an excellent job of providing us with large-scale vocabulary pre-trained which efficiently cuts down our model processing.

3. Method

3.1. Dataset

The dataset contains multiple features for each text. The features include toxic, severe toxic, obscene, threat, insult or identity hate. These attributes can help us make better and more useful predictions making it more effective.

3.2. Preprocessing methods

3.2.1. GloVe. It is an unsupervised learning algorithm developed by Stanford for generating word embeddings by aggregating a global word-word co-occurrence matrix from a corpus. The benefit of GloVe over any other embedding is the vast vocabulary trained and provided by Stanford for public use.

3.2.2. Vector Sum. The concept is as simple as the name itself. We simply add the individual vectors of the words in the document getting the *sum* of all of their individual *meanings*.

3.3. Model Architectures

3.3.1. Simple Neural Network with ReLU Activation. The Neural Network has an input layer with 128 nodes, a hidden layer with 64 ReLU activated nodes and an output layer of 6 nodes. The layers are batch-normalized and have a 20% dropout.

3.3.2. Simple Neural Network with TanH Activation. The Neural Network has an input layer with 128 nodes, a hidden layer with 64 TanH activated nodes and an output layer of 6 nodes. The layers are batch-normalized and have a 20% dropout.

3.3.3. LSTM with ReLU Feed Forward layers. The Neural Network has an input layer followed by 8 LSTM nodes, a hidden layer with 32 ReLU activated nodes and an output layer of 6 nodes. The layers are batch-normalized and have a 20% dropout.

3.3.4. LSTM with TanH Feed Forward layers. The Neural Network has an input layer followed by 8 LSTM nodes, a hidden layer with 32 TanH activated nodes and an output layer of 6 nodes. The layers are batch-normalized and have a 20% dropout.

3.4. Implementation

To find out how well different preprocessing methods work, we developed a homogeneous pipeline. The key aspects of the pipeline were:

1) Loading/Creating the word vectorization function.
2) Converting text from the dataset to vector form.
3) Training and evaluating models

3.4.1. Vectorization. The aim of vectorization is to convert any word into the form of a numeric vector. Once done, this vector can be used in place of the word to train the model. The Natural Language Processing department of Stanford provides multiple pre-trained vectorization corpora. They can be downloaded and word-wise vector mapping can be done to convert the words to vectors.

3.4.2. Data conversion. The vectorization mechanism is word-level, i.e., the functions take the input as a word or token and provide its vector. When dealing with intent detection, it becomes important to make a document-level vector.

We take the element-wise sum of individual words to combine their meanings. So, the vector for a document would be the element-wise sum of all the vectors whose corresponding words occur in the document.

3.4.3. Model training. A total of four models have been built. The models have as similar an architecture as is possible. The simple neural networks have the same kind of input and output. We input the document vector and predict the various classes.

The LSTM neural networks have the same input as the simple neural networks. We change the input shape to make it compatible to the network. Other than that, the output is exactly in the same form. The difference in all networks is the way they learn due to their internal differences in activation and prediction strategies.

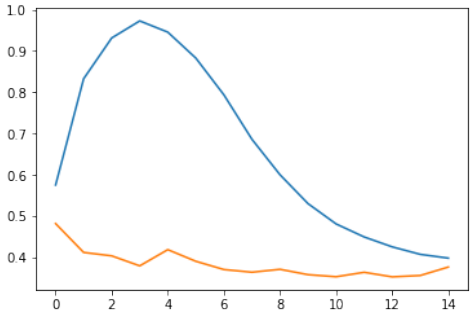

Figure 1. ANN ReLU Loss History

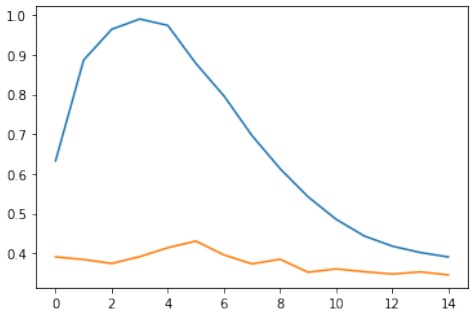

Figure 2. ANN TanH Loss History

3.5. Results

When the pipeline is followed and the models are created, a total of 4 models are created which can be compared to one another.

	Training	Validation	Testing
ANN, ReLU	0.3799	0.9818	0.9826
ANN, TanH	0.4562	0.9936	0.9941
LSTM, ReLU	0.9903	0.9923	0.9924
LSTM, TanH	0.9870	0.9927	0.9931

The simple neural networks seem to under perform as compared to the LSTMs in the training dataset. The LSTM models seem to out perform the Simple Neural Networks in case of training making them better and more reliable overall.

Among the two, ReLU activation performs slightly better than TanH. Since ReLU is also faster, we can say that ReLU is a better model.

4. Conclusion

Most Neural Networks are capable of learning with sufficient layers and training time. We were able to see how different types of neural networks learn faster or better with limited layers and training. We saw that changing activation function can cause a difference in what is learnt but may or may not be a major difference.

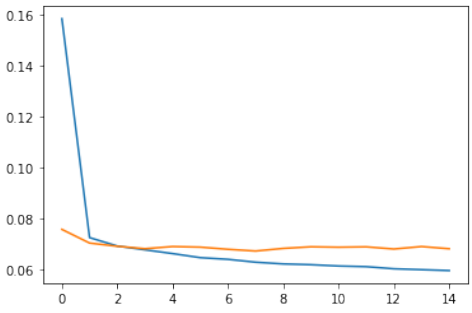

Figure 3. LSTM ReLU Loss History

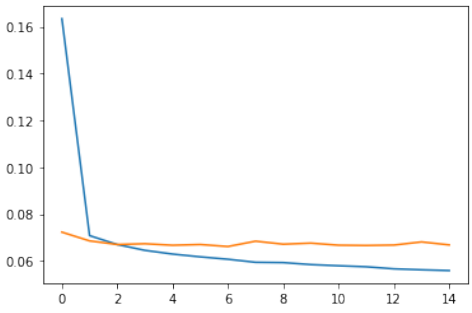

Figure 4. LSTM TanH Loss History

5. Future Work

Finding out the results of more preprocessing (vectorization) methods like the usage of ELMO and BERT to see their effectiveness is an obvious next step. Finally, using more Deep Learning models like Graph Neural Networks, other Recurrent Neural Network variants, Convolutional Neural Networks, etc.

6. Acknowledgement

This research work was supported by Vellore Institute of Technology, Chennai under Dr. Joshan A. We would like to thank him deeply for his support and motivation which helped us create this paper and perform research on this topic.

7. References

[1] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014 Oct (pp. 1532-1543).

[2] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013 Jan 16.

[3] NLP GloVe Dataset: nlp.stanford.edu/projects/glove/

[4] Toxicity Dataset: kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data