Detecting Toxicity with Bidirectional Gated Recurrent Unit Networks

Vinayak Kumar[1], B.K.Tripathy[2]

[1]School of Computer Science and Engineering, [2]School of Information Technology and Engineering
Vellore Institute of Technology,Vellore,Tamil Nadu,India
[1]sivakumarvinayak@gmail.com , [2]tripathybk@vit.ac.in
Tel: 9481809871

Abstract. As large amounts of data keep being generated by users on social media platforms, some of the information can be considered as harmful. These kinds of textual information can be generated in forums, online discussions or any other communication exchange in an online medium. As such it is sometimes difficult to filter out what information is actually meaningful. Detecting these harmful pieces of information can help in providing a means on online moderation so that a safe discussion can be maintained, which helps in preventing issues such as cyber bullying. Using a labelled dataset of comments that are classified as toxic labels, we can implement deep learning models to implicitly extract textual features from the comments and solve this supervised learning problem. This paper focuses on using the variations of Recurrent Neural Networks, with the main focus on using Bidirectional Gated Recurrent Units, and evaluating their performances against each other.

Keywords: Recurrent neural networks, toxic, multi label text classification, long short term memory, gated recurrent units.

1 Introduction

Online moderation and content filtering has become more prevalent in the recent years due to more intelligent and automated systems being used in social platforms. However it is still quite difficult to identify and filter specific personal attacks on people in a public forum. The type of attack could also play a role in determining the severity of the attack. Based on a online public experiment, through Wikipedia, user generated comments had been collected and labelled through machine learning classifiers [10]. There are six labels for each comment, each determining the type of attack. The main label is the toxic label, based on which it could fall under the other labels.

There has been quite a substantial increase of research in deep learning due to its applicability in various areas, such as medical, artificial intelligence, computer vision, signal processing and many more. Another particular area of its application is in natural language processing. Deep learning provides a means to solve Natural Language Processing problems without worrying too much about additional processes to extract the information. Multi label text classification is a natural language interpretation problem where text of varying length is required to be labelled under multiple labels, where a single piece of text can belong to more than one label.

Recurrent neural networks(RNN) fall under a special type of deep neural networks, where the networks have an internal memory state that processes sequences of inputs and reuses the output processes in the hidden layers. In other words, it remembers the previous state and adjusts the network based on it as well as the current state and input state. This allows it to handle temporal data, as the previous information is compared to the current information. Based on this principle, there have been many variants of recurrent neural networks such as Long Short-Term Memory(LSTM) and Bidirectional Long Short-Term Memory, and have been used based on the sequential nature of the problem [4]. A newer variant which is similar to LSTM models called the Gated Recurrent Unit (GRU) model has also started to be used in deep networks [12] and behave very similar to LSTM. The Bidirectional part of the recurrent neural network basically passes the input in both the forward and backwards direction, hence making it able to predict the context of a long sequence of data. Using this we can apply it to the existing GRU, along with other hidden layers.

In section 2, we will compare text classification approaches proposed in different studies. In section 3, the proposed system for classifying the toxic data is explained and finally in section 4 we compare the classifier using various metrics.

2 Literature Survey

Yang, Z.[11] proposed a deep neural network model that focuses on the hierarchical nature of documents with attention mechanisms. This model is meant to classify documents by using the idea that not all documents are contextually relevant, hence finding certain parts that describes the context of the document can be achieved with attention level mechanisms on both the word and sentence level. GRU cells are used to encode the words, which is then sent to a word attention mechanism to extract the important context words. Then GRU cells are used to encode the sentences and are then sent to a similar sentence attention mechanism. Finally the document is then classified using a softmax function.

In another study, an abusive language detection system was made by sample

comments on several Yahoo! News and Finance pages [8]. Here more linear classifiers were used, and there was more emphasis on the different aspects of textual feature extraction. This included data preprocessing steps to handle linguistic, syntactic and semantic features by using n-gram models. Word embedding models were also used to create a larger comment embedding model which was based on the idea that surrounding word vectors as well as the comment the word is currently in, plays a large role in the occurrence of that word. The comments were represented by using low dimension vectors and the evaluation of the model was compared against other pretrained embedding models.

Joulin,A. proposed a text classification model [5] that focuses on its efficient architecture and performance by using regular linear baseline models with a large corpus. Word representations were taken and then averaged into sentence representations, along with a hiearchial softmax function to calculate the probability of the occurrence of the class. A fast and efficient n-gram model with hashing was used for the textual feature extraction. This system was named fastText and was meant to scale large amounts of data, comparable to deep learning models. Extensive performance testing was done on this system by comparing it's usage in sentiment analysis and tag prediction.

The system [7] uses LSTM and Convolutional layers to convert text from word vectors to a new "short-text" representation. In both instances, max pooling layers are done to obtain the final vector. This new short text vector is then fed to a feedforward Artificial Neural Network for that particular short text, which finally gives the probability distribution over a set of classes based on the input vector. This system was then used for dialog act prediction and was evaluated. A similar study [6] used a similar combination of deep neural network architectures, where recurrent cells were used in the convolution layer to capture the context of each word input, which was then passed to a max pooling layer to give an output. This model is meant to deal with the biasing problems of recurrent neural networks by being able to give more importance to certain word meanings, instead of just looking at the importance of a previous word input and the current word input. It was then compared to traditional machine learning models with feature extraction methods such as bag of words bigrams, as well as other popular neural networks.

3 Proposed System

3.1 Exploratory Data Analysis and Preprocessing the dataset

The dataset used to train and test the model is obtained from Kaggle. This was the Jigsaw Toxic Comment dataset which had several comments labelled as toxic, severe toxic, obscene, threat, insult or identity hate. The dataset had 159571

comments, out of which 15294 were labelled as toxic. Most of the comments had 50 words, while the few comments with the maximum number of words peaked at 200 words. This is important as this would help determine the feature vector of the input to the deep neural network.

The data was then tokenized based on a high average number of words per comment. This represents the input feature vector of the data. Unlike most linear classifiers, deep neural networks do not require more feature engineering, such as handling syntactic or semantic features. This is because we are able to use word embeddings based on the tokenized text.

3.2 Word Embedding Layer

Once the words are indexed with numbers, each comment becomes a vector with numeric features. This is fed into the deep neural network via the input layer. To be able to process the words for classification, we use an embedding layer that is able to project these words into a vector space, where similar words tend to occur next to each other, hence capturing the semantic meaning of that word. In other words, if a word is close to another word in this vector space, then it is highly likely that this word conveys a similar meaning. As such the output of the word embedding are coordinates for each word processed[9].This is also a better option generally instead of using an entire vocabulary of words and using a large dimension vector to indicate whether that word is present or not.

There are several approaches to word embeddings. There's the baseline approach where our own embedding layer is created based on the tokenized text. Pretrained models such as GloVe and word2vec are also good alternatives that use neural network layers or matrix factorization to try to predict the word representation based on the given data.

3.3 Bidirectional Gated Recurrent Units Layer

Recurrent Neural networks(RNN) consist of neural network layers which consist nodes in a directed graph, where the information of the input is fed recursively to the neuron. This is done through a transition function. These RNN units use dependent activations by generating the same weights and biases to all the layers, while giving the output as input to the next hidden layer [2]. The activation function of this current neural network neuron ht is dependent on the value of the previous state h_{t-1} and the current input neuron x_t. Based on this activation function, the layer output can be calculated using the output layer weights. Normally a tanh function is used as the activation function.

This was addressed in the another study [3] by introducing a new memory based learning unit in the recurrent layer called Long short-term network(LSTM). This

network was able to learn dependencies over a long period of time. This was achieved by having the LSTM unit consist of a separate memory unit that updates the content of the LSTM cell only when it is required. This is done by using gates, represented by sigmoid functions 'σ'. There are three gates, each represented by a sigmoid function. The first gate is the forget gate that reads h_{t-1} and x_t. Based on this value it creates a probability distribution between 0 and 1 on whether to discard this input or retain it. Similarly there is a input gate unit that decides which values to update, with the tanh function creating new candidate values. Finally there's the output gate which checks what parts of the cell state needs to be put in the output function. It is then finally sent to the activation function to create a new h_t.

Another system was used in [1] which also meant to remove the vanishing gradient problem, without the need of a memory unit though. This was called Gated Recurrent Units(GRU). Unlike LSTM, GRU has only two gates, and there isn't a memory storage unit generated as well. The figure for a GRU unit is given in Fig.1.

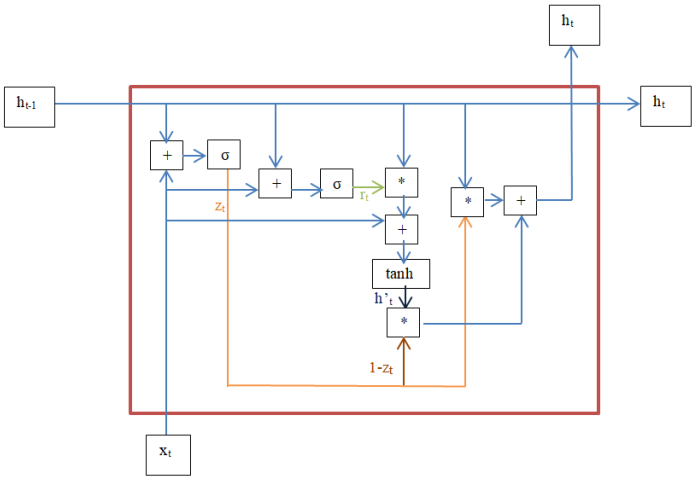

Fig.1. Single GRU Unit

The first gate(sigmoid function σ) is called the update gate. It is calculated from the following formula:

$$z_t = \sigma\left(W^{(z)} x_t + U^{(z)} h_{t-1} \right) \quad (1)$$

where x_t is the incoming input at instance t, and h_{t-1} is the state of the unit at instance t-1. $W^{(z)}$ and $U^{(z)}$ represent the weights of the input neuron and previous state at the update gate. They are added and then a sigmoid activation function is

used. The update gate z_t is used to identify how much of the previous information needs to be passed and processed.

The reset gate is the second gate and is calculated as follows:

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1} \right) \tag{2}$$

$W^{(r)}$ and $U^{(r)}$ represent the weights of the input neuron and previous state at the reset gate. A temporary memory content (h'_t) is created which uses the reset gate to store relevant past information. It is calculated as:

$$h'_t = tanh\left(Wx_t + r_t * Uh_{t-1} \right) \tag{3}$$

Note that '*' denotes the Hardmond matrix product operation. The element h'_t determines what to remove from the preceding time steps. Using this,we can find the current h_t state as:

$$h_t = z_t * h_{t-1} + (1 - z_t) * h_t^{'} \tag{4}$$

If the update gate z_t is a value close to 1, which represents that the new input is important, then $1\text{-}z_t$ will determine as how much of the current information to lose when calculating h_t.

Using these GRUs, we can then apply a mechanism called Bidirectional RNNs [13] to improve how the text information is going to be processed and classified. Here there are two layers of recurrent units, one processing the information in the forward direction and the other layer of units processing the information in the backward direction. When a prediction on what words to use is required, the past information as well as the future information are used. Using this mechanism we will use a bidirectional layer of GRU units.

3.4 Pooling and Dense Layers

We then use a max pooling layer to reduce the dimension for the next set of layers. In this layer , each batch of incoming input data is processed and the maximum value of each batch is taken. The output would then be a smaller batch of data which makes it simpler to process in the succeeding dense layers. These dense layers use the ReLU (Rectified Linear Unit) activation function. We also use dropout layers to try to constrain the network(avoiding overfitting of the neural network) and helps remove non-essential neurons from the network during that pass.

The final dense layer has a sigmoid activation function that generates a six dimension vector, representing the labels.

3.5 Final System Design

For training the model, the Jigsaw dataset was used and splitted into a pair of training and testing sets.It was then tokenized to be given as input to the neural network for training. After training the data, the classifier model should be able to label any text into the six labels by generating the probabilities of that label occurring on that specific text, as shown in Fig 2.

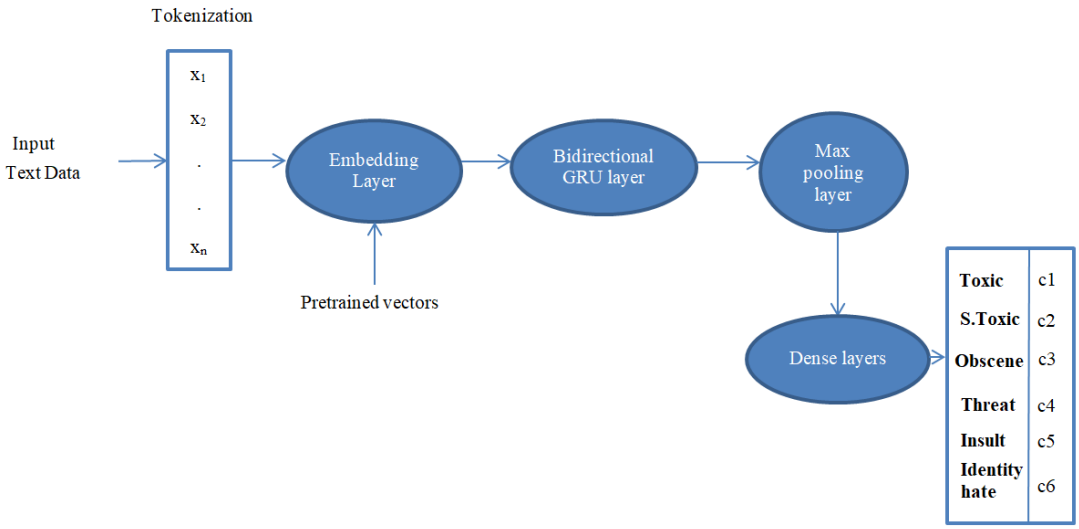

Fig.2. System Architecture

4 Results and Discussion

4.1 Hardware and Software Setup

The system used was a quad core Intel i5 processor running at 2.2 GHz with 8 Gb RAM,with a 2GB Nvidia 920M graphics card. Python 3.6 was used along with libraries Pandas,Numpy,TensorFlow,Keras and Sklearn.

4.2 Evaluation

We use the proposed design and tested the model against other variants of RNNs. We also compare them against traditional machine learning models.The loss and accuracy was based on the binary cross entropy function to handle the multiple labels. F1, precision and recall measures were also used to evaluate the model.

4.3 Results

When training the deep models, in all four cases, the testing loss results seemed to converge after two epochs. After the two epochs, the result became prone to overfitting. As seen from Fig.3, the model was still prone to overfitting since the training accuracy and loss improved over time while there was little improvement for the testing dataset. Regularization or more hyperparameter finetuning could be used to get a better testing accuracy and loss across the epochs during evaluation.

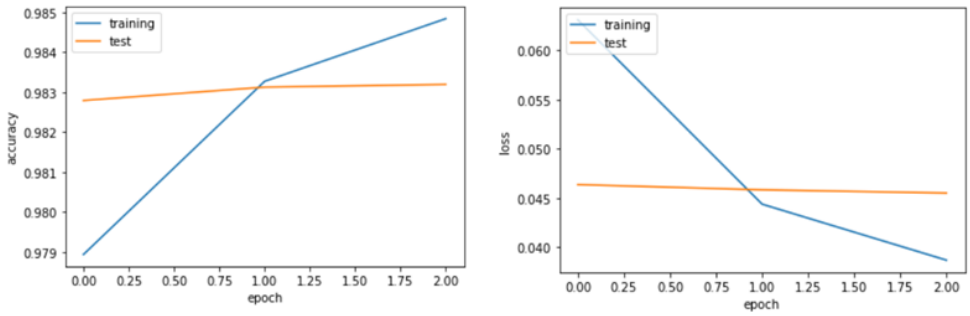

Fig.3. Bidirectional GRU Accuracy results (Left), Bidirectional GRU Loss results (Right)

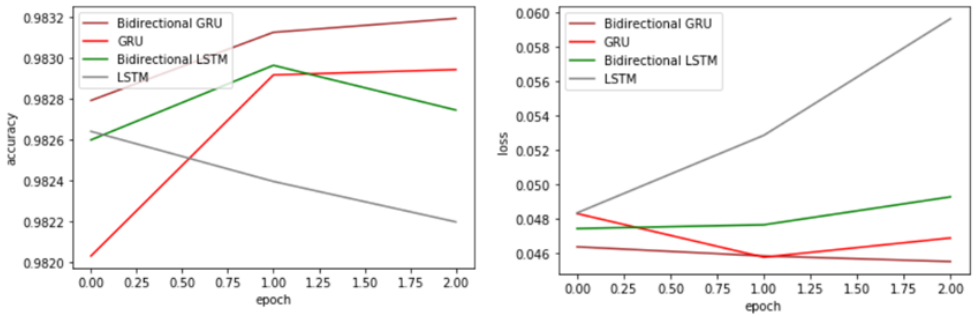

Fig.4 Testing Accuracy(Left), Testing Loss(Right)

As seen from Fig.4 all the deep learning models produced favorable results. Most of the differences in scores between the variations are minor. The bidirectional GRU was better than the remaining models in terms of its accuracy and loss. The LSTM models seemed to be better in terms of precision and recall scores.All the RNN variants seem to perform better than the other machine learning models.The

remaining results are compiled in Table 1.

Table 1. Evaluation of Testing Data

Model	Evaluation Metrics				
	Accuracy	Loss	F1 Score	Precision	Recall
Bidirectional GRU	0.9830	0.0459	0.6651	0.7585	0.6293
GRU	0.9826	0.0470	0.6613	0.7541	0.6259
Bidirectional LSTM	0.9828	0.0481	0.6583	0.7654	0.6131
LSTM	0.9824	0.0536	0.6589	0.7414	0.6293
Logistic Regression	0.9711	0.0808	0.2549	0.6998	0.1682
Random Forest Classifier	0.8789	0.1665	0.4093	0.6575	0.3411
Naïve Bayes	0.9135	0.0769	0.2748	0.5969	0.1954

5 Conclusion

As we have seen, the deep learning models have provided favourable results despite not focusing on the data pre-processing part and feature engineering that traditional classifiers need in order to perform decently. A lot of improvements can be made to the model, such as adding an attention based mechanism to better handle the toxicity.

Using this kind of learning system, we can identify specific aspects of toxicity in a system and then probe into dealing with those issues. Toxicity isn't just limited to abusive behavioural comments as well. Anything unrequired in an information exchange could also be considered as toxic, and by understanding how to use deep learning in this system, we can help improve public discussions and online moderation.

References

1. Chung, J., Gulcehrel, C., Cho, K., Bengio,Y.: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, arXiv:1412.3555, Cornell University (2014)
2. Elman,J.: Finding structure in time, Cognitive Science A Multidisciplinary Journal, 14(2),pp.179-211 (1990)

3. Hochreiter,S.,Schmidhuber,J.: Long Short-Term Memory, Neural Computation,9(8),pp.1735-1780 (1997)
4. Huang,Z.,Xu,W.,Yu,K.: Bidirectional LSTM-CRF Models for Sequence Tagging, arXiv:1508.01991,Cornell University (2015)
5. Joulin,A., Grave,E., Bojanowski,P., Mikolov,T.: Bag of Tricks for Efficient Text Classification, arXiv:1607.01759, Cornell University (2016)
6. Lai,S.,Xu,L.,Liu,K.,Zhao,J.: Recurrent convolutional neural networks for text classification, AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence,pp.2267-2273 (2015)
7. Lee,J.,Dernoncourt,F.: Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies ,Association for Computational Linguistics, 10.18653/v1/N16-1062,pp.515-520 (2016)
8. Nobata,C., Tetreault,J., Thomas,A., Mehdad,Y.,Chang,Y.: Abusive language detection in online user content, Proceedings of the 25th International Conference on World Wide Web, WWW '16, , Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee, pp. 145-153 (2016)
9. Shen,D.,Wang,G.,Wang,W.,Min,M.,Su,Q., Zhang,Y.,Li,C.,Henao,R.,Carin,L.: Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms, Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics , P18-1041,pp.440-450 (2018)
10. Wulczyn, E., Thain,N., Dixon,L.: Ex Machina: Personal Attacks Seen at Scale ,26th International Conference on World Wide Web (WWW '17). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, pp.1391-1399 (2017)
11. Yang,Z.,Yang,D.,Dyer,C.,He,X.,Smola,A.,Hovy,E.: Hierarchical Attention Networks for Document Classification, Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics , N16-1174, pp.1480-148 (2016)
12. Zhao,R .,Wang,D., Yan,R., Mao,K.,Shen,F., Wang,J.: Machine health monitoring using local feature-based gated recurrent unit networks, IEEE Transactions on Industrial Electronics ,65.2 , pp.1539-1548, (2018)
13. Zhou,P.,Shi,W.,Tian,J.,Qi,Z.,Li,B.,Hao,H.,Xu,B.: Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification, Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics , P16-2034 ,pp.207-212 (2016)