2018 IEEE International Conference on Data Mining Workshops (ICDMW)

Overlapping Toxic Sentiment Classification using Deep Neural Architectures

Hafiz Hassaan Saeed[†]
Computer Science Department
Information Technology University
Lahore, Pakistan
phdcs17009@itu.edu.pk

Khurram Shahzad[†]
Computer Science Department
Information Technology University
Lahore, Pakistan
khurram.shahzad@itu.edu.pk

Faisal Kamiran
Computer Science Department
Information Technology University
Lahore, Pakistan
faisal.kamiran@itu.edu.pk

***Abstract*—We are living in an era where data is enjoying an unprecedented increase in its volume in each passing moment through online media platforms. Such a colossal amount of data is multifarious in its nature where textual data proves to be its vital pillar. Almost every sort of online media platform is producing textual data. Short posts (i.e. Twitter and Facebook) and comments constitute a significant part of this textual data. Unfortunately, this text data may contain overlapping toxic sentiments in terms of personal attacks, abuses, obscenity, insults, threats or identity hatred. In many cases, it becomes extremely important to track such toxic posts/data to trigger needed actions e.g. automated tagging of posts as inappropriate. State-of-the-art classification techniques do not handle the overlapping sentiment categories of text data. In this paper, we propose Deep Neural Network (DNN) architectures to classify the overlapping sentiments with high accuracy. Moreover, we show that our proposed classification framework does not require any laborious text pre-processing and is capable of handling text pre-processing (e.g. stop word removal, feature engineering, etc.) intrinsically. Our empirical validation on a real world dataset supports our claims by showing the superior performance of the proposed methods.**

***Index Terms*—Toxic comments, Focal Loss, Text Pre-processing, CNN, Bi-GRU, Bi-LSTM**

I. Introduction

Text classification is a well-known area of research in Natural Language Processing (NLP) and proved to be a vital constituent in many applications, such as online search, information filtering, topic categorization and sentiment analysis [1]. In this paper we study the deep neural network based techniques for overlapping toxic sentiment classification. By overlapping we mean that a comment (or a document) can belong to multiple overlapping sentiment classes, for example, a comment may be an insult, a threat and obscene simultaneously. Toxic sentiment classification is of great importance in order to maintain the unpartisan of online platforms and to protect the self-esteem of people's belonging to different cultures, societies, religions, ethnicities and countries etc. Many online platforms e.g. Facebook, provide their users with a feature to report a post in terms of nudity, violence, harassment, suicide or self-injury, fake news, spam, unauthorized sales and hate speech.

According to [2] only in 2017, daily Facebook posts volume surged to 4.3 billion and around 656 million tweets were posted on daily basis on Twitter. Such a large amount of data has galvanized both industry and academia in the last few years to elicit meaningful information from this textual data to interpret the sentiment of users about, for example, any particular product, incident or a topic. Many machine learning models are proposed by research community in order to extract useful information to fathom users' sentiment. Research community is actively proposing models for extracting sentiments from textual data like the model proposed in [3].

Sentiment analysis methods normally classify data into positive and negative classes. With the elapse of time, interest is growing beyond binary class classification, to identify and classify overlapping toxicity in textual data. In this paper, we use DNN architectures to classify textual data into overlapping toxic categories. We use a multi-label dataset freely available at Kaggle[1]. To identify overlapping toxicity in textual data, the problem turns into a text classification problem. Georgakopoulos et. al. address the toxic comment classification [4] and Google Jigsaw also launched perspective API for toxic comment classification. However, both handle simple binary text classification on the dataset whereas we have focused on overlapping classification of multi-label dataset, preserving the originality of the dataset.

The main contributions of this paper can be stated as follows:

- We propose a deep learning methodology which requires no pre-processing due to its inherent feature engineering capabilities.
- In contrast to the state-of-the-art binary classification, our method handles overlapping multi-label classification problem.
- Our extensive empirical validation on real world dataset shows the superior performance of our methods.

II. Related Work

A number of studies have been conducted on detecting hate speech, abuse detection or toxic comment classification in text. According to [5] the term 'hate speech' is interchangeably

[†] Both authors contributed equally to the work.

[1] https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

2375-9259/18/$31.00 ©2018 IEEE
DOI 10.1109/ICDMW.2018.00193

used as profane, offensive or abusive language but they showed that hate speech can be different from abusive language. They did a little bit of pre-processing and used CNN+GRU network where the network was composed of an embedding layer, 1D convolution, 1D max-pooling, GRU, global max pooling and finaly a softmax layer with elastic net regularization. Zhang et al. have empirically shown that character level Convolutional Neural Network (CNN) perfectly works in classification of text [6]. Recurrent neural networks shown in [7] and Bi-LSTMs shown in [8] also give better performance in text classification. In [9] multiple classifiers such as Random Forests, Support Vector Machines, Gradient Boosted Decision Trees, Logistic Regression and Deep Neural Networks with GloVe word embeddings were used to detect hate speech in tweets. Related to our work, Cyberbullying is another term used by the research community that is defined as a post or an image that insults, humiliates or causes embarrassment through Internet. [10] used multiple classifiers like Support Vector Machines, Naïve Bayes, k-Nearest Neighbours, JRip, J48, Random Forest, Sentence Pattern Extraction arChitecture (SPEC) and CNNs for Cyberbullying detection and showed that CNN outperformed other classifiers by over 11% points in F-score. [11] also detected Cyberbullying across different social media platforms using four deep learning models namely CNN, LSTM, BiLSTM and BiLSTM with attention. They also used the concept of transfer learning during their training process on different datasets related to different social media platforms. The studies conducted on the dataset chosen in this research work mostly convert the dataset to a binary classification problem whereas the original released dataset is a multilabel classification problem. [4] used CNN architecture that was based on the architecture proposed in [12] and worked on binary classification for the dataset i.e., they converted multiple labels to either toxic or non-toxic comments. They also drew a comparison of CNN with other machine learning techniques used for classification of toxic comments. [13] also converted the dataset to binary classification but their work was to illustrate the importance or necessity of pre-processing that is considered to be one the very important ingredients in the natural language processing (NLP) pipeline. Another study, to classify news comments, proposes the use of multiple features such as length of comments, uppercase and punctuation, lexical features such as spelling, expletive and legibility by applying applied linear and tree based classifier [14]. FastText is a pre-trained word embedding model developed by Facebook Artificial Intelligence Research (FAIR) which is qualified to model text involving out-of-vocabulary (OOV) words [15].

III. Methodology

Our problem lies particularly in the domain of multi-label text classification. A multi-label classification problem is now being addressed. We are given a multi-label training dataset D which contains n pairs of documents and label vectors, where each document corresponds to a single comment in the dataset.

$$D = \{(X,Y)|X \in documentspace, Y \in \{0,1\}^L\}$$

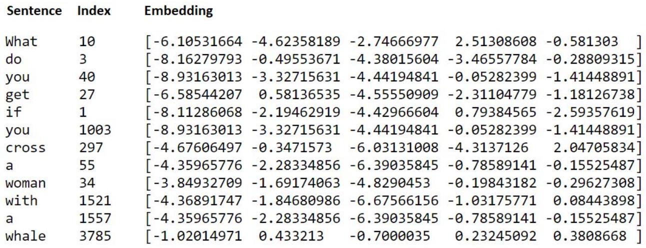
Here Y is the label vector and L=6 as we have 6 labels namely toxicity, severe toxicity, obscenity, threat, insult and identity hate. The goal is to learn multi-label learning model M that is a function Z of x_i and maps input document/comment x_i to y_i where each y_i is a label vector.

$$M = Z(X) : X \to Y$$

In the later sections of this paper, M will be referred to as the deep learning model, X will be input, Y will be the output and Z will be used for model predictions. The traditional text classification pipeline includes text pre-processing, text encoding, learning model and evaluation.

A. Text Pre-processing

There are several techniques that are used in text pre-processing such as converting all words to lower case, removal of stop words, punctuation marks, extra white spaces, non-printable characters, words with alpha-numeric characters, emoticons, URLs, hashtags, mentions, date/time, lemmatization, stemming, etc. Initially, we followed all the pre-processing techniques as described above but the results proved not to be much satisfactory because stop words and punctuation marks may contribute in the meaning of the text. Later on we ignored all pre-processing of data except the conversion all words to the lower-case. Results with no pre-processing proved to be promising than with pre-processing and by promising we mean that we do not lose anything if we do not pre-process the data. In results section we further discuss this topic.



Sentence	Index	Embedding
What	10	[-6.10531664 -4.62358189 -2.74666977 2.51308608 -0.581303]
do	3	[-8.16279793 -0.49553671 -4.38015604 -3.46557784 -0.28809315]
you	40	[-8.93163013 -3.32715631 -4.44194841 -0.05282399 -1.41448891]
get	27	[-6.58544207 0.58136535 -4.55550909 -2.31104779 -1.18126738]
if	1	[-8.11286068 -2.19462919 -4.42966604 0.79384565 -2.59357619]
you	1003	[-8.93163013 -3.32715631 -4.44194841 -0.05282399 -1.41448891]
cross	297	[-4.67606497 -0.3471573 -6.03131008 -4.3137126 2.04705834]
a	55	[-4.35965776 -2.28334856 -6.39035845 -0.78589141 -0.15525487]
woman	34	[-3.84932709 -1.69174063 -4.8290453 -0.19843182 -0.29627308]
with	1521	[-4.36891747 -1.84680986 -6.67566156 -1.03175771 0.08443898]
a	1557	[-4.35965776 -2.28334856 -6.39035845 -0.78589141 -0.15525487]
whale	3785	[-1.02014971 0.433213 -0.7000035 0.23245092 0.3808668]

Figure 1. Example of a 5 dimensional word embedding

B. Text Encoding

Learning models cannot take text data directly as input. It should be transformed into some encoding scheme like traditional schemes used TF.IDF weights for each word existing in the corpus. Most simple TF.IDF weights are calculated by:

$$TF.IDF = TF * log(\frac{N}{DF})$$

Here TF is the term frequency, DF is the document frequency and N is the total number of documents in the corpus. The term frequency refers to the number of times a word has appeared in a single document whereas document frequency refers to the number of documents in which that particular word has appeared. Most recent and state-of-the-art text encoding schemes are Word Embedding like Word2Vec by Google, Glove by Stanford and Fast-Text by Facebook. In this study we use Fast-Text[2] which is a pre-trained word embedding having 2 Million words where each word is a 300 dimensional embedding vector. An example of a sentence encoded using a 5 dimensional word embedding is given in Figure 1. The first column in the example is the actual sentence that has been tokenized, the second column is the tokenized index number corresponding to the word in the vocabulary and the third column shows 5 dimensional word vectors of each word in the sentence. Thus, a sentence consists of different words and the words are now vectors, and eventually a sentence has become a matrix of real numbers.

C. Learning Models

The first and the baseline architecture in this study is Convolutional Neural Network (CNN) with 1D convolutions. Then 4 other architectures are studied, in which two are based on CNNs, one on Long Short Term Memory (LSTM) and the other on Gated Recurrent Unit (GRU). The details of all these architectures are given in the next subsections. The common constituents of all these architectures are:

- The input of the embedding layer is a 200x300 dimensional matrix as we input only 200 words in a sentence where each word is represented by a 300 dimensional word embedding. All architectures contain a trainable embedding layer.
- After embedding layer, in all architectures, we used a 1D spatial drop out with 40% dropout rate. This 1D spatial drop out drops an entire 1D vector out with the given rate and stops the model from overfitting and enforces better generalizations for learning.
- In all the architectures we use exponential linear unit ELU except for LSTM, where we use tanh instead. ELU is a good alternative to ReLU because of its implicit batch normalization and it also decreases the bias shift by enforcing the mean activation towards zero [16]. ELU is defined as:
$$ELU = f(t) = \begin{cases} t & \text{if } t \geqslant 0 \\ \alpha(e^t - 1) & \text{if } t < 0 \end{cases}$$
- Focal loss is recently proposed in [17] handles class imbalance better than cross entropy. We use α=0.25 and γ=5.0 in all the architectures. Focal loss is given by:
$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma log(p_t)$$
- The last 4 layers are similar in all architectures. The fourth last layer is a drop out layer to prevent the model from overfitting and to get better generalizations. We use the drop out rate=10%. The third last layer is Fully connected layer and consists of 100 units with ELU as activation function, the second last layer has 50 units with ELU and 6 units of sigmoid constitute the last layer.

We compared and evaluated five deep neural models in this research study. The architectures of these models are briefly explained as:

1) CNN-1D: The first and the baseline model used in this study is the convolutional neural network with 1D convolutions. By 1D convolutions, we mean that the kernel used for convolution was one dimensional vector. In this architecture, the first layer is the embedding layer with a 1D spatial drop out. The next were 4 identical blocks in a sequence connected to the embedding layer after the 1D spatial drop out layer. Each block had two convolutional layers and one max pooling layer. Each convolutional layer had 32 kernels with kernel size=3. Kernel size determines the field of view of the convolution. The last 4 layers are already described above.

2) CNN-V: This CNN architecture has 2D convolutional kernels or filters and is inspired from a very famous VGG16 proposed in [18]. Our limited computational resources deterred us from training full VGG architecture, therefore we modified it to a shorter version. Three convolutional blocks in a sequence are used with the embedding layer after 1D spatial drop out where the first two blocks has 2 convolutional layers with one max pooling layer and the last block has 3 convolutional layers and a max pooling layer. In the first block convolutional layers had 16 kernels of size 3x3, the second block has 32 kernels of size 3x3 and last block consists of 64 kernels of size 3x3. Due to the inspiration from VGG we named this model as CNN-V.

3) CNN-I: The base of this architecture is the Inception model proposed in [19] and was also used by Yoon Kim in [12] for text classification. After the embedding and 1D spatial drop out layers, 6 convolutional blocks comprised of a convolutional layer and a max pooling layer were attached in parallel. The convolutional kernels used in each block are two dimensional and they differ in each of the six blocks i.e., the kernel size in the first convolutional block is 1x300, in the second convolutional block the kernel size is 2x300 and in the last block the kernel size is 6x300. We used 32 kernels in each of the parallel convolutional blocks. The output of all of these blocks is then concatenated and passed on to last 4 layers as described before.

4) Bidirectional LSTM: This architecture uses two parallel blocks of bidirectional Long Short Term Memory (LSTM) where the the term bidirectional means that the input sequence is given to the LSTM in two ways, first way is to input the sequence as-it-is and the second way is to input the sequence in reverse order. A simple LSTM is a variation of a recurrent neural network that has an input gate, an output gate, a forget gate and a cell. We used two parallel bidirectional LSTM blocks having 128 units in one and 64 units in the other. The output of both of these blocks is concatenated and then two separate blocks are connected with the concatenation layer. One of these two separate blocks are used for global average pooling and the other one is used for global max pooling. Both these blocks are then concatenated and passed on to last 4 layers. Hard sigmoid and tanh were used for recurrent activations and hidden units respectively.

5) Bidirectional GRU: Unlike LSTMs, gated recurrent units (GRU) do not have output gate. They were first introduced in 2014 in [20]. GRUs have an update gate and a reset

[2] https://fasttext.cc/docs/en/english-vectors.html

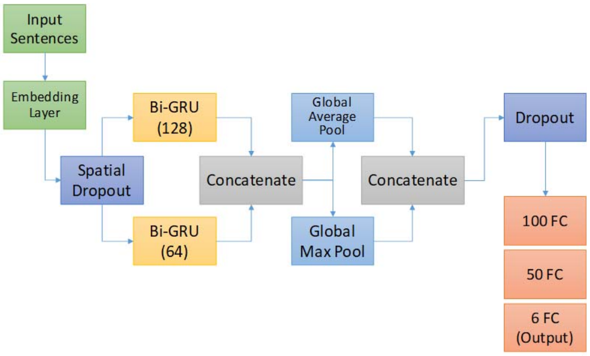


Figure 2. Bi-directional GRU Architecture Block Diagram

gate. The reset gate is responsible of combining new input with the previous one, and the update gate is responsible of how much the previous memory is required to be kept.

$$z = \sigma(W_z h_{t-1} + U_z x_t)$$
$$r = \sigma(W_r h_{t-1} + U_r x_t)$$
$$c = tanh(W_c(h_{t-1} \otimes r) + U_c x_t)$$
$$h_t = (z \otimes c) \oplus ((1-z) \otimes h_{t-1})$$

GRU is faster than LSTM because it needs less computations to update its hidden state. We used two bidirectional GRU blocks in parallel like we did in the BiLSTM architecture. The first block has 128 units in it and the second block consists of 64 units. This architecture is shown in Figure 2. The basic idea of using GRUs is to learn long-term dependencies or sequences.

D. Evaluation

To measure the performance of the architectures we used exact match accuracy, mean area under the receiver operating characteristic (ROC) curve score, mean precision, mean recall and mean F1 score. A prediction from a multi-label classifier can be partially correct. The exact match accuracy ignores the partially correct predictions and counts only those predictions to be correct that exactly match with the label vector in all dimensions. The exact match accuracy is given by:

$$accuracy = \frac{exactly\, matched\, instances}{total\, instances}$$

ROC curve is a plot of true positive rate vs false positive rate and the area under its curve gives us the probability that the classifier will rank higher to a random positive instance than to a random negative instance. We report average of ROC-AUC of all labels given by:

$$ROCAUC_{Mean} = \frac{1}{L}\sum_{i=1}^{L} ROCAUC(l_i)$$

Precision is the proportion of correctly predicted instances to the total number of predicted as correct instances. Average precision used in this study is the average of precision of each label in Y.

$$Precision_{Mean} = \frac{1}{L}\sum_{i=1}^{L} \frac{|Y_i \cap Z_i|}{|Z_i|}$$

Recall is the proportion of correctly predicted instances to the total number of actual correct instances. Average recall used in this study is the average of recall of each label in Y.

$$Recall_{Mean} = \frac{1}{L}\sum_{i=1}^{L} \frac{|Y_i \cap Z_i|}{|Y_i|}$$

F1 score is the harmonic mean of precision and recall and the average F1 score is given by:

$$F1_{Mean} = \frac{1}{L}\sum_{i=1}^{L} \frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$

IV. Experimentation & Results

We used CNN with 1D convolutions as our baseline architecture which is then compared with two variants of CNN, Bidirectional LSTM and Bidirectional GRU. The two CNNs are inspired from VGG-16 and Inception architectures. We evaluated our architectures on the dataset openly available at Kaggle. This dataset has a huge number of comments from talks page edit of Wikipedia. It is a multi-labelled dataset and contains six labels in total named as toxic, severe toxic, obscene, threat, insult and identity hate. Usually, the labels in the dataset are converted to binary classification task by combining all the labels into either toxic or non-toxic but we did not change the originality of the problem and focused on multi-label task. This dataset has a class imbalance problem that is why we used focal loss in our all architectures. The average words in the dataset sentences is 67, however, we used fixed 200 words for our input embeddings matrix. All positive labels in the severe toxic column are simultaneously toxic as well in both train and test sets provided by Kaggle. All the results shown here are on the test set instead of the train set. All the results, demo, data and source codes are publicly available[3].

Exact match accuracy, mean ROC-AUC, mean precision, mean recall and mean F-1 scores are reported for each of the proposed models (see Figure 3). It is evident from the plot shown in the figure that the accuracy in each model is almost equal. The data distribution is highly skewed (towards non-toxicity), accuracy metric here is misleading as predicting only the majority class can get us approximately 90% accuracy. Therefore, we cannot rely on accuracy and have to resort to other evaluation metrics like precision, recall and F1 scores for all the labels.

The baseline architecture CNN-1D gave lowest ROC-AUC score while all other four models gave better ROC-AUC score. The ROC-AUC scores for CNN-V, CNN-I, Bi-LSTM and Bi-GRU were approximately equal. CNN-I gave lowest precision but still is comparable to other models in terms of precision.

[3] https://github.com/hafizhassaan/Toxic-Comments

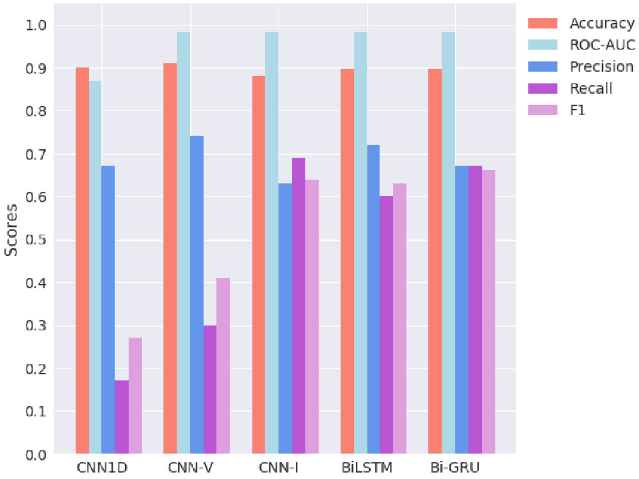


Figure 3. Comparison of Model Scores

The baseline CNN-1D gave poor results in recall and F1 scores. Moreover, CNN-V gave highest precision score but a lower recall and F1 score. CNN-I had the highest recall and and a good precision and F1 score. Bi-LSTM also had a very high precision, high recall and high F1 score. Bi-GRU had highest F-1 score and a very good precision and recall. Overall, Bi-GRU seemed to perform the best among all the proposed architectures of this study.

Table I
COMPARISON OF MODELS PER LABEL SCORE

Model	**Scores**	**toxic**	**s_toxic**	**obscene**	**threat**	**insult**	**id_hate**
CNN-1D	Precision	0.68	0	0.78	0	0.79	0
	Recall	0.21	0	0.19	0	0.15	0
	F1	0.32	0	0.31	0	0.25	0
CNN-V	Precision	0.72	0.48	0.75	0.33	0.85	0.57
	Recall	0.4	0.08	0.37	0.01	0.14	0.01
	F1	0.52	0.14	0.5	0.02	0.24	0.01
CNN-I	Precision	0.58	0	0.64	0.58	0.76	0.78
	Recall	0.84	0	0.74	0.18	0.53	0.4
	F1	0.68	0	0.69	0.28	0.62	0.52
BiLSTM	Precision	0.66	0.27	0.72	0.62	0.84	0.84
	Recall	0.74	0.01	0.66	0.44	0.45	0.11
	F1	0.7	0.02	0.69	0.51	0.58	0.2
BiGRU	Precision	0.65	0.55	0.68	0.53	0.73	0.67
	Recall	0.75	0.13	0.71	0.46	0.58	0.48
	F1	0.7	0.21	0.69	0.49	0.65	0.56

As stated earlier, the dataset was highly skewed. In such scenarios it is better to look at scores for individual labels instead of taking averages. For further analysis, Precision-Recall-F1 score was separately calculated for each label (see table I). These results were quite interesting and gave in-depth overview of how each model was performing. CNN-1D was not predicting any severe toxic, threat and identity hate labels at all. CNN-I which seemed to perform very well was also not predicting any severe toxic label. However other architectures were able to predict severe toxic and identity hate labels. By looking at the overall per label scores of precision, recall and F1, Bi-GRU performed exceptionally well for all the labels. It gave highest Precision-Recall-F1 score for all the labels.

Table II
TOXIC SENTIMENT SCORES (WITHOUT PRE-PROCESSING)

Sentences	**toxic**	**s_toxic**	**obscene**	**threat**	**insult**	**id_hate**
All Muslims are terrorists	yes	-	-	-	-	yes
all Americans must die	yes	-	-	yes	-	yes
Get the f**k out of this room.	yes	-	yes	-	yes	-
sh**t man	yes	-	yes	-	-	-
shyt man	yes	-	yes	-	-	-
sh++t man	yes	-	yes	-	-	-
i hate you you motherf*****r	yes	yes	yes	-	yes	-
He is a kind man	-	-	-	-	-	-
you are stupid	yes	-	-	-	yes	-
@ss hole	yes	-	yes	-	-	-
go kill yourself	yes	-	-	yes	yes	-
mothafucka i will kill you and wear your skin	yes	-	yes	yes	yes	-

Furthermore, a very short list of sentences is given in table II along with the predictions from Bi-GRU. The predictions are thresholded at 0.5. Either 'Yes' or '-' is shown in the labels where 'Yes' means that the predicted probability is greater than or equal to 0.5 otherwise it is '-'. The predictions reported here indicate that even with the raw input, deep learning models are capable of giving good results.

V. CONCLUSION

This study draws a comparison among DNN models when there is an overlapping multi-label text classification problem. Based on the empirical results, considering overlapping text classification, we recommend not to spend too much time in data pre-processing. As a matter of fact, stop words, punctuation, etc. proved to be a vital constituent of data when it comes to the training of the models. Every DNN model shows improvement in model performance measurement metrics when trained using unprocessed data. We also recommend the use of focal loss to deal with imbalanced classes. Focal loss alleviated the skewed class problem, though not much significantly, but still it did not exacerbate the skewness problem. Use of multiple model performance metrics can prove decisive in the choosing and rating DNN models. In overlapping multi-label text classification it is observed that the per-label measurement of performance metrics corroborates the better understanding of model performance. Here five performance metrics, accuracy, ROC-AUC, precision, recall and F1 guided us towards the choice of Bi-GRU as the best model for overlapping multi-label toxic text classification.

REFERENCES

[1] C. C. Aggarwal and C. Zhai, "A survey of text classification algorithms," in *Mining text data*. Springer, 2012, pp. 163–222.

[2] J. Schultz, "How much data is created on the internet each day?" 2017, accessed: 01-08-2018. [Online]. Available: https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day

[3] H. Meisheri, K. Ranjan, and L. Dey, "Sentiment extraction from consumer-generated noisy short texts," in *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*. IEEE, 2017, pp. 399–406.

[4] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos, "Convolutional neural networks for toxic comment classification," *arXiv preprint arXiv:1802.09957*, 2018.

[5] Z. Zhang, D. Robinson, and J. Tepper, “Detecting hate speech on twitter using a convolution-gru based deep neural network,” in *European Semantic Web Conference*. Springer, 2018, pp. 745–760.

[6] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in neural information processing systems*, 2015, pp. 649–657.

[7] S. Lai, L. Xu, K. Liu, and J. Zhao, “Recurrent convolutional neural networks for text classification.” in *AAAI*, vol. 333, 2015, pp. 2267–2273.

[8] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, “Text classification improved by integrating bidirectional lstm with two-dimensional max pooling,” *arXiv preprint arXiv:1611.06639*, 2016.

[9] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, “Deep learning for hate speech detection in tweets,” in *Proceedings of the 26th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 2017, pp. 759–760.

[10] M. Ptaszynski, J. K. K. Eronen, and F. Masui, “Learning deep on cyberbullying is always better than brute force,” in *IJCAI 2017 3rd Workshop on Linguistic and Cognitive Approaches to Dialogue Agents (LaCATODA 2017), Melbourne, Australia, August*, 2017, pp. 19–25.

[11] S. Agrawal and A. Awekar, “Deep learning for detecting cyberbullying across multiple social media platforms,” in *European Conference on Information Retrieval*. Springer, 2018, pp. 141–153.

[12] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

[13] F. Mohammad, “Is preprocessing of text really worth your time for online comment classification?” *arXiv preprint arXiv:1806.02908*, 2018.

[14] D. Brand and B. Van Der Merwe, “Comment classification for an online news domain,” 2014.

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *arXiv preprint arXiv:1607.04606*, 2016.

[16] D. Pedamonti, “Comparison of non-linear activation functions for deep neural networks on mnist classification task,” *arXiv preprint arXiv:1804.02763*, 2018.

[17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *arXiv preprint arXiv:1708.02002*, 2017.

[18] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[20] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.