An attention ensemble based approach for multilabel profanity detection

Pratik Ratadiya
Dept. of Computer Engineering,
Pune Institute of Computer Technology,
Maharashtra, India.

Deepak Mishra
Dept. of Avionics,
Indian Institute of Space Science and Technology,
Kerala, India.

***Abstract*—The amount of user-generated content in the cyberspace keeps increasing in the 21st century. However, it has also meant an increase in the number of cyber abuse and bullying incidents being reported. Use of profane text by individuals threatens the liberty and integrity of the digital space. Manual moderation and reporting mechanisms have been traditionally used to keep a check on such profane text. Dependency on human interpretation and delay in results have been the biggest obstacles in this system. Previous deep learning-based approaches to automate the process have involved use of traditional convolution and recurrence based sequential models. However, these models tend to be computationally expensive and have higher memory requirement. Further, they tend to produce state of the art results in binary classification but perform relatively poorly on multilabel tasks, owing to less flexibility in architecture. In today's world, classifying text in a binary way is no longer sufficient and thus a flexible solution able to generalize well on multilabel text is the need of the hour. In this paper, we propose a multihead attention-based approach for detection of profane text. We couple our model with power weighted average ensembling techniques to further improve the performance. The proposed approach does not have additional memory requirement and is less complex as compared to previous approaches. The improved results obtained by our model on publicly available real-world data further validate the same. Flexible, lightweight models which can handle multilabel text well can prove to be crucial in cracking down on social evils in the digital space.**

***Index Terms*—Self attention, Profanity detection, Sentiment analysis, Natural language processing**

I. Introduction

There has been a constant rise in user activity as well as the amount of data being generated in the cyberspace. Today, there are approximately 4.42 billion internet users across the world [1]. Social media constitutes to be a major source of user-generated data on the web. As of July 2019, almost 510,000 new comments are uploaded on Facebook every minute with a presence of more than 1.59 billion daily active users [2]. Almost 680 million tweets are sent by users daily [3]. The information being generated is expected to only keep increasing in the coming years. This data could be put to good use in the domains of sentiment detection and classification.

With the extension of cyberspace, the scourge of cyberbullying and abuse has also assumed haunting proportions with a mass number of incidents of bullying, threatening and hate on social media being reported every day. Lack of serious and timely supervision has been a major factor in exacerbating this problem. Difference in opinions on various issues often lead to users resorting to the use of profane language. Use of such language threatens the liberty of other individuals online. Such profane text can be categorized into multiple categories like toxic, hate, insult etc and interest has now grown beyond binary categorization. Thus, the detection of profane text accurately across multiple categories in quick time is the need of the hour.

Previous approaches for profanity and cyber abuse detection have involved using supervised machine learning and deep learning algorithms [4], [5], [12]. Advanced deep learning-based approaches have made use of convolution and recurrence mechanisms. A main characteristic of LSTMs [6] and GRUs has been the use of a memory unit(cell state). The architectures also tend to have increased complexity. Although these models have been able to produce state of the art results in binary classification, they tend to generalize poorly for multilabel categorization.

In this paper, we propose the use of attention mechanism for the task in hand. Specifically, we make use of multihead attention in our architecture. We further make use of power weighted average ensembling to improve our results. The proposed architecture is found to be superior in handling multilabel text and provides optimum results on a standard public dataset[1] across various metrics. We also compare our obtained results with [17] presented at Sentire'18 and show the superior performance of the proposed approach. Our main contributions in this paper could be listed as:

1) We have been able to achieve quality results despite altogether skipping the recurrence mechanism which is used in most approaches.
2) We have made effective use of positional encoding to provide information related to sequence in absence of recurrence.
3) We have been able to demonstrate effective retention of maximum information present in a sequence despite padding by using concatenation.

The rest of the paper is structured as follows: Section 2 talks about related work in this area while the proposed methodology is explained in section 3. Dataset description and the results obtained are discussed in section 4. Our analysis of

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

the work conducted is presented in section 5 while the paper is concluded in section 6.

II. Background and related work

Toxicity detection has been an active area of research in the field of Natural Language Processing. Numerous approaches have been tried in the past to solve the issues faced in the area of detection of cyber abuse, bullying, hate speech and profane text especially after the rise in the use of machine learning algorithms as well as availability of computational power and data. [4] was one of the first attempt to detect cyberbullying by making use of C4.5 decision trees and instance-based learner. [5] used various features like PoS tagging, Tf-Idf and label specific features for cyber abuse detection. Z. Zhang et al made use of a hybrid CNN+GRU based model along with elastic net regularization for detecting hate speech on twitter [7]. The authors also indicate how hate speech can often be different from profane and abusive text. Lai et al made use of recurrent CNNs for the task of text classification [8]. Zhang et al demonstrated the use of character-level CNN for text classification [9]. [10] proposed a Tf-Idf and N-gram based SVM and Logistic regression model for detection of cyber-aggressive comments. [11] combined bidirectional LSTMs with max pooling to enhance results in text classification. Badjatya et al proposed the use of multiple deep learning-based architectures like CNNs, LSTMs and Fasttext for the detection of hate speech in tweets [12]. Founta et al proposed a unified deep learning-based model which combined two different networks for abuse detection [13]. [14] empirically compared the improved performance of CNN over traditional machine learning algorithms in the detection of cyberbullying. Georgokopoulus et al worked in toxic comment classification using CNNs albeit they worked on a binary classification task [15]. Multiple natural language features were proposed to be used by Brand et al for classification of comments under online news. [16]

Majority of the work has been done by considering binary classification. Quite recently, Saeed et al worked on overlapping toxic sentiment classification using various deep neural architectures and concluded BiGRUs to be ideal for the same [17]. A notable observation is that almost all of these approaches have relied on convolution or recurrence for context understanding and classification. The relatively contemporary attention mechanism has not yet been actively used in this domain. While [18] does make use of attention for hate speech detection, they combine it with LSTM to derive the final architecture.

III. Methodology

Our task is to classify a given text sample into one or more of six categories of profanity-toxic, severe toxic, obscene, insult, threat and identity hate. We explain our solution by describing our approach for each phase of the standard text classification pipeline- preprocessing, text encoding, model training and prediction.

A. *Preprocessing of text*

The text is converted into lower case. It is stripped of any special characters, as well as numbers as these do not primarily add to the profanity of the sentence. Any abbreviations and short representations of verbs like shouldnt, cant are converted into full form for smoother and accurate embedding representation later. The cleaned text is also stripped of any stop words present in it. Stop words are the commonly used words which primarily serve to keep the sentence grammatically correct. The probability of occurrence of stop words in sentences containing profane text and clean sentences is almost the same and they do not contribute to the profanity proportion. The retained text is then converted into tokens based on the frequency of occurrence of individual words in the training data.

To improve the performance of the model, sequences are usually padded to a constant length. This is done by either appending characters or truncating the sequence at the start(referred as pre-padding) or at the end(referred as post padding). However in a task like profanity detection where toxic language is generally used by the user at the start or the end of sentence, there is a risk of information loss by padding sequences only in a particular way. We tackle this problem by passing two inputs to our model for every sample- one being the pre-padded sequence and the other being the post-padded sequence. The intermediate outputs are later concatenated together and passed ahead. This way we are guaranteed to retain the features of the entire sequence. This approach of handling sequence padding helps us significantly boost our results as shown later. It should be noted that the layers common for both the sequences share the same parameter values for the dimension of text encodings, positional encodings and the multi-head attention layer. Parameters aren't shared between the two columns, but the values used are the same.

B. *Text encoding*

We make use of word embeddings to convert the text into model friendly data. Embeddings are the distributed representation of text in n-dimensional space. They try to retain human understanding of words in the feature space. Every word is replaced by a n dimension vector. Various pre-trained embeddings like Word2Vec, GLoVe and Fasttext have been used by researchers in the past. For our use, we focus on the GLoVe and Fasttext word embeddings. We train four models differentiated only by the type of pre-trained embeddings that they use. The predictions of the four different models are then ensembled together to obtain the final prediction as described later. The four different embeddings used are:

1) Fasttext word embeddings: 300 dimension embeddings for 2 million words.
2) GloVe twitter embeddings: 200 dimension embeddings for 1.2 million words trained upon 27 million tweets.
3) Concatenated word embeddings: 500 dimension embeddings obtained by concatenating Fasttext embeddings and GloVe twitter embeddings in the same order.

4) Reverse concatenated word embeddings: 500 dimension embeddings obtained by concatenating Fasttext embeddings and GloVe twitter embeddings in the reverse order i.e. twitter representations are followed by fasttext representations.

Separate representations are derived for both the pre-padded and post-padded sequences. Both Fasttext and Glove twitter embeddings are available only in the said dimensions. Here it should be noted that in case of concatenated and reverse concatenated word embeddings, if either of the original word embedding matrix do not contain the given word, the values are assigned as zero for the respective dimension range. In this way, we convert the original text into a matrix representation which is more suitable as input to the deep learning model.

C. Model architecture

A common architecture is used in case of all the four models. We propose the use of multi-headed self attention mechanism [20] as the principal component of the model architecture. By making use of attention, recurrence mechanism is skipped completely from the model and the memory requirement and complexity is reduced. As a result, though, no information regarding the order of sequence is present due to lack of recurrence. To tackle this problem, we introduce a positional encoding layer which provides information regarding the position of a word in the sequence. For each of the two inputs, the output of the embedding layer is passed to the positional encoding layer and subsequently to the multi-head attention layer. The representations obtained for the two sequences are then concatenated together and passed through an average pooling layer, a dropout layer and then to the output nodes. The model architecture is as shown in fig 1. The working of each layer is as follows:

1) Positional encoding:

This layer helps with providing information regarding the absolute and relative position of tokens in the sequence. It has the same dimension as that of the embedding layer thereby connecting properly. Researchers have earlier made use of both fixed as well as learned positional encodings. In our case, fixed positional encodings are used which are derived from sinusoidal functions of different frequencies:

$$PE_{(pos,2i)} = \sin(pos/10000^{(2i/d_{model})}) \tag{1}$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{(2i/d_{model})}) \tag{2}$$

where pos is the position, i is a particular dimension and d_{model} is the dimension of the embeddings. The advantage of using these functions is that they are able to address relative positions properly. Every $n + k^{th}$ positional encoding could be represented as a linear function of PE_n.

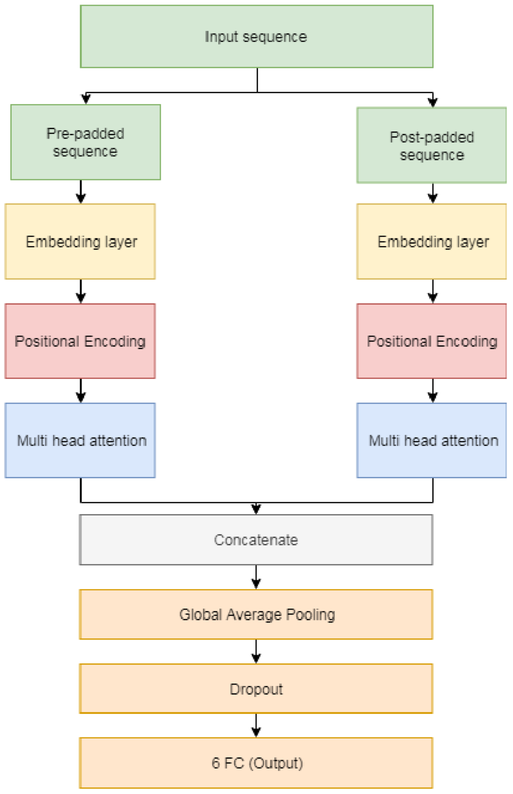

Fig. 1: Block diagram of the proposed model architecture

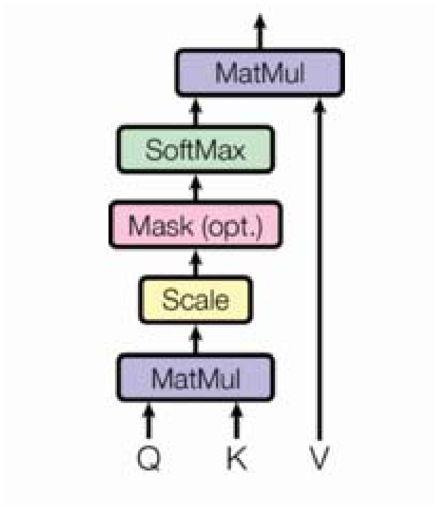

Fig. 2: Schematic representation of self dot product attention [20]

2) Multi head attention:

Attention function is the mapping of queries and key-value pairs to an output. Self attention is a variant of the attention mechanism where different positions of a sequence are related to calculate representations of the same sequence.

For predicting a new word, an attention vector is calculated which is based on correlation with other words present in the sequence.

Particularly, we make use of scaled dot product kind of self attention. For matrices Q, K and V for query, keys and values respectively, the attention function carries out a dot product of the query and key values and passes them through a softmax before obtaining final weights to be multiplied with the values V. The scaled dot product attention value is calculated as:

$$Attention\,(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (3)$$

where

- V, is the value vector
- Query, $Q = EW_q$
- Key, $K = EW_k$
- Value, $V = EW_v$

W_q,W_k,W_v are the weight matrices for the queries, keys and values respectively. $1/\sqrt{dk}$ acts as a scaling factor thus the name scaled dot product attention. As we are making use of self attention, Q, K and V are all the same wiz. representations provided by the positional encoding layer.

In multihead self attention, we do not restrict ourselves to making use of outputs of only one attention function. Instead for h different projections of the queries, keys and values, the outputs of the attention function are calculated. All these output values are then concatenated together and worked upon further. The multi-head self attention function is given as follows:

$$Multihead(Q,K,V) = Concat(head_1, ..., head_h)W^O \qquad (4)$$

where $head_i$ = $Attention(Q(W_i)^Q, K(W_i)^K, V(W_i)^V)$

Based on our empirical studies, we find h=8 to be the ideal number of parallel attention layers for our task. The values of these layers are concatenated together to deduce the final value as shown in figure 2.

After this stage, the outputs obtained for the two input sequences are concatenated together and passed to the next layers.

3) Average pooling and dropout:

The average pooling layer helps in reducing the total number of parameters. It does so by taking into consideration the average of all elements present in the pooling window and only taking this average value forward as representation of the window. Further to avoid overfitting, we pass the obtained values through a dropout layer. Dropout ignores a certain set of neurons at random during the training phase to ensure that no intra network co-dependency is developed [19]. During our experimentation, we found a dropout value of 0.3 to be ideal for the current architecture. The output of the dropout layer is then fed to a dense layer containing 6 nodes which represents the final output layer, with each node corresponding

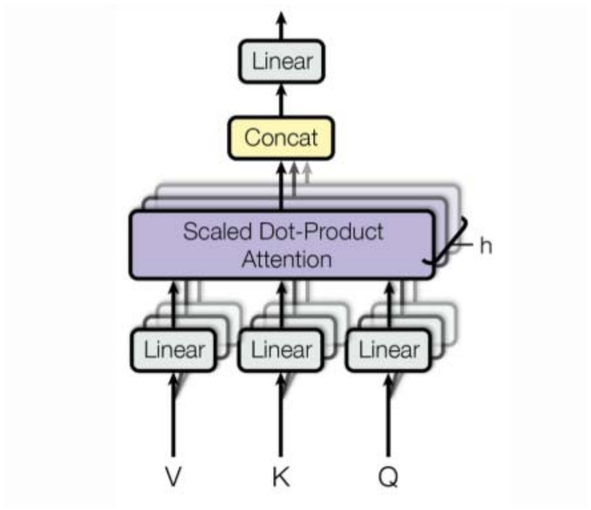

Fig. 3: Multi head attention consisting of parallel running attention layers [20]

to one category. Note that we train the model by optimizing the binary cross-entropy loss and not the categorical cross-entropy loss often used in case of multiple output nodes to satisfy the condition of multilabel classification.

D. Predictions

To improve the performance, we ensemble the predictions provided by the four different models to get the final prediction. We make use of the following two ensembling techniques:

1) Weighted average ensembling:

In this method, prediction of each model is assigned a weight based on the accuracy of the model. Individual weight value lies between 0 and 1 and the sum of weights assigned to all models should be equal to 1. The sum of products of predictions with weights is considered as the final prediction. The weighted average predictions are thus given as:

$$WeightedAveragePrediction = \sum_{i=1}^{n} P_i W_i \qquad (5)$$

where P_i denote the predictions of the i^{th} model and W_i indicates the weight assigned to model i. n indicates the total number of models being used and

$$\sum_{i=1}^{n} W_i = 1$$

. In our case, n = 4 and we assign the weights of 0.3, 0.25, 0.25, 0.2 to the twitter, concatenated, reverse concatenated and fasttext embeddings based model predictions respectively. This helps us in getting a generalized output which is scaled effectively.

2) Power weighted average ensembling:

It is similar to weighted average ensembling except the fact that we take the power of every prediction before multiplying them to the weights and summing them up. This ensures that the model must be very confident of predicting the positive label(value = 1) so that the value is retained even after raising it to a power. The formula for power weighted average ensembling is given as:

$$P.W.A.\ prediction = \sum_{i=1}^{n} (P_i^k) W_i \tag{6}$$

where k indicates the power by which the predictions are to be raised and W_i indicates the individual class weight. Note that the higher the value of k will be, more skewed results will be obtained. After experimental results, we find out $k = 2$ to be the ideal value in our case. We then make the final predictions by averaging out the predictions given by the weighted average ensemble and the power weighted average ensemble together.

Algorithm 1 Ensembling of predictions

Input: Predictions of the four models P, Weights for each model W
Output: Final predictions fp

```
1: $fp = 0$
2: $weighted\_pred, pow\_weg\_pred;$
3: for all $p \in P$ do
4:     $weighted\_pred \leftarrow weighted\_pred + p * W_p$
5:     $pow\_weg\_pred \leftarrow pow\_weg\_pred + p^2 * W_p$
6: end for
7: $fp = (weighted\_pred + pow\_weg\_pred)/2$
8: return $fp$
```

IV. Dataset description and results

The proposed architecture is evaluated on the publicly available dataset on Kaggle released as part of the toxic comment classification challenge.

A. Dataset description

The dataset consists of a text field and six output labels namely toxic, severe toxic, obscene, threat, insult and identity hate. It is a multilabel dataset with samples being annotated into more than one of the six categories. The training dataset consists of 159,571 samples while the test set consists of over 63 thousand samples. A major issue we face in the training dataset is of heavy class imbalance with over 90% not containing any kind of profanity. This leads to the risk of model overfitting and we handle this issue by undersampling the completely clean data by 30%. After the preprocessing of text, the fixed sequence length is set to 205 based on the average and deviation values. The model is then trained on this data.

B. Performance metric

Majority of the training data is clean text and as a result, accuracy cannot be an appropriate metric for a task as it can give a false inference by favouring the majority class. We thus resort to other performance metrics such as mean AUROC, precision, recall and F1 score.

Receiver Operating Characteristics(ROC) curve maps the true positive rate to the false positive rate and area under the same indicates the probability of model rating a random positive instance higher than a random negative instance. Formula for the same is given as:

$$AUROC_{mean} = \frac{1}{L} \sum_{i=1}^{L} AUROC(l_i)$$

Precision refers to how precisely a model categorizes a given sample into a particular class. For prediction Z and truth value Y, the precision is calculated as:

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

Recall indicates how many of a particular class samples was the model able to predict. We calculate the average recall as:

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

The F1 score is the harmonic mean of precision and recall. The average F1 score is calculated as:

$$F1\ score = \frac{1}{L} \sum_{i=1}^{L} \frac{2 * Precision * Recall}{Precision + Recall}$$

In our case, L=6. The model was trained on Tesla K80 GPU on Google Colaboratory. The trained model was then evaluated over each of the above-mentioned metrics.

C. Results obtained

The results of the model on the test set are evaluated on each of the metrics. We focus more on the mean Area under ROC metric as it provides a better intuition of performance in case of class imbalance and multilabel classification. The same metric has been used in the challenge as well. The results obtained by various models and final ensembled model are as shown in table 1.

It can be seen that the final ensembled results outweigh the ones of individual models. Next, we compare our results with [17] who made use of Bi-GRU based network on the same dataset at Sentire'18. The comparison across mean AUROC score and accuracy is as shown in table 2.

It can be seen that our approach dominates the Bi-GRU based model. Further, we also compare the results obtained by the respective models per label across precision, recall and f1 score metrics as tabulated in table 3

It can be seen that our proposed approach has outperformed the other model on the majority of the classes across all the metrics. The areas where it has underperformed could be

Model	Mean AUROC
Attention(Fasttext embeddings)	97.89
Attention(Twitter embeddings)	98.01
Attention(Concatenated emb.)	98.02
Attention(Rev. concat. emb.)	98.07
Attention(Weighted average)	98.2
Attention(Power weighted average)	98.18
Attention(Final ensemble)	**98.3**

TABLE I: Profanity detection scores of various models

Model	Mean AUROC	Mean Accuracy score
Attention ensemble	**98.3**	**97.45**
Bi-GRU	98.23	92.39

TABLE II: Comparison of results across multiple metrics

Metric	Model	tox.	s_tox.	obsc.	thr.	ins.	i_hate
Precision	Attention	**0.69**	0.54	**0.73**	**0.62**	**0.77**	**0.82**
	Bi-GRU	0.65	**0.55**	0.68	0.53	0.73	0.67
Recall	Attention	**0.84**	**0.25**	**0.71**	0.18	**0.59**	0.3
	Bi-GRU	0.75	0.13	**0.71**	**0.46**	0.58	**0.48**
F1	Attention	**0.76**	**0.34**	**0.72**	0.28	**0.67**	0.44
	Bi-GRU	0.7	0.21	0.69	**0.49**	0.65	**0.56**

TABLE III: Comparison of models per label score(Attention refers to the Attention ensemble model)

attributed to the fact of class imbalance. Thus, our approach has given better results. We also carry out complexity analysis of various layer types as shown in table 4.

Layer Type	Complexity per layer
Self attention	O($n^2.d$)
Recurrent	O($n.d^2$)
Convolution	O($k.n.d^2$)

TABLE IV: Complexity analysis of various layers. n refers to the sequence length, d is the embedding dimension and k refers to the kernel size

Attention thus tends to be less complex than conventional approaches. Further, as a part of ablation studies, we also show the impact of concatenation of padded sequences in our architecture in table 5.

Finally we also show the impact of the amount of power in power weighted average ensembling on the final results as shown in table 6. Thus a detailed result analysis clearly shows the benefit of using attention mechanism over conventional methods along with variation in result based on changes in certain factors.

Type of padding used	Mean AUROC
Pre-padding	96.7
Post-padding	96.4
Concatenation of pre and post padding	**98.3**

TABLE V: Comparison of results for various padding types used

Power used in P.W.A. ensembling	Mean AUROC
k = 2	**98.3**
k = 4	98.2
k = 8	98.14

TABLE VI: Comparison of results for various values of power in power weighted average ensembling

V. Analysis

Our main observations after performing out the study have been:

- The complexity per layer as well as the minimum number of sequential operations required for self attention is less as compared to convolution or recurrent models thus making it a desirable choice.
- Concatenation of pre-padded and post-padded sequences leads to considerable improvement in results especially in case of models based on attention where positional information is fundamentally absent.
- The higher the power used for power weighted average ensembling, sharper the results would be. This would mean improved results in case of true predictions but also at the expense of heavy loss in case of false predictions.
- Recent models based on the transformer architecture like BERT [21] tend to give better results than the proposed architecture but they come at the expense of huge training cost as well as computation requirement. Also, these models have multi-head self attention as their building block only.

VI. Conclusion

Thus by making use of attention mechanism, we have been able to achieve standard results in the area of multilabel profanity detection. While attention by itself cannot retain all information of a sequence, it can prove to be a powerful architecture when coupled with other features like positional encoding while cutting down on the complexity and memory requirement. While some earlier approaches have argued that preprocessing is not a necessary step, our results show it to be crucial to provide relevant and useful information to the deeper attention layers. The improved results across metrics over the previous approach are a testament to the power which attention mechanism holds. Further improvement in our work includes modifications in the post attention layers for better information transfer. Word embeddings can be replaced with

Byte Pair Encodings(BPE) and the results could be analyzed. Hyperparameter tuning can be explored further along with the issues of tackling class imbalance when making use of attention. Models like transformer, BERT based upon attention have already begun dominating traditional methods and the idea of using only recurrence for sequence modelling keeps becoming weaker day by day. Attention-based models have proven to be a better alternative and can be put to use in areas of sentiment classification and mapping effectively.

References

[1] ”Internet users in the world statistics”, 2019 Accessed: 01-08-2019. [Online] Available: https://www.internetworldstats.com/stats.htm

[2] ”The top 20 valuable facebook statistics”, 2019 Accessed: 01-08-2019. [Online] Available: https://zephoria.com/top-15-valuable-facebook-statistics/

[3] Irfan Ahmad, ”How Much Data Is Generated Every Minute? [Infographic]”, 2019 Accessed: 01-08-2019. [Online] Available: https://www.socialmediatoday.com/news/how-much-data-is-generated-every-minute-infographic-1/525692/

[4] Kelly Reynolds, April Kontostathis, Lynne Edwards, “Using Machine Learning to Detect Cyberbullying”, 10th International Conference on Machine Learning and Applications and Workshops, Dec 2011.

[5] Dinakar, K., Reichart, R. and Lieberman H, “Modeling the detection of textual cyberbullying”, In Proceedings of the International Conference on Weblog and Social Media 2011.

[6] Sepp Hochreiter,Jurgen Schmidhuber, ”Long short-term memory”, Neural Computation 9(8):1735-1780, 1997

[7] 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. 745760.

[8] S. Lai, L. Xu, K. Liu, and J. Zhao, ”Recurrent convolutional neural networks for text classification.” in AAAI, vol. 333, 2015, pp. 22672273.

[9] X. Zhang, J. Zhao, and Y. LeCun, ”Character-level convolutional networks for text classification”, in Advances in neural information processing systems, 2015, pp. 649657.

[10] Vikas Chavan, Shylaja SS, “Machine learning approach for detection of cyber-aggressive comments by peers on social media network”, International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2015.

[11] 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

[12] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, Vasudeva Varma, “Deep Learning for Hate Speech Detection in Tweets”, Proceedings of ACM WWW’17 Companion, Perth, Western Australia, Apr 2017

[13] Antigoni-Maria Founta, Despoina Chatzakou, Nicolas Kourtellis, Jeremy Blackburn, Athena Vakali, Ilias Leontiadis, ”A Unified Deep Learning Architecture for Abuse Detection”, arXiv:1802.00385 [cs.CL]

[14] 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. 1925.

[15] 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.

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

[17] Hafiz Saeed, Khurram Shahzad, Faisal Kamiran, ”Overlapping Toxic Sentiment Classification using Deep Neural Architectures”, International Conference on Data Mining Workshops (ICDMW), 2018.

[18] Gretel Liz De la Pena Sarrac en, Reynaldo Gil Pons, Carlos Enrique Muniz Cuza, Paolo Rosso, ”Hate Speech Detection using Attention-based LSTM”,Vol-2263, Ceur-WS.

[19] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”, Journal of Machine Learning Research, 1929 1958, 2014.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention is All you Need”, Advances in Neural Information Processing Systems(NIPS) 30, 2017.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, arXiv:1810.04805 [cs.CL]