**IARJSET** ISSN (O) 2393-8021, ISSN (P) 2394-1588

**International Advanced Research Journal in Science, Engineering and Technology**

**ISO 3297:2007 Certified ⁜ Impact Factor 7.105 ⁜ Vol. 9, Issue 7, July 2022**

**DOI: 10.17148/IARJSET.2022.9701**

# RNN AND CNN DEEP NEURAL MODELS FOR HATE SPEECH CLASSIFICATION

**Osuji Chinonso C.[1], Wani Nowshaba J.[2], Voronkov Ilya M.[3], Orefo Somtochukwu C.[4]**

Phystech School of Radio Engineering and Computer Technologies,

Moscow Institute of Physics and Technology, Moscow, Russia[1,2,3,4]

**Abstract**: The importance of creating a social media environment void of abuse, bullying and threats cannot be overemphasized. Using Twitter as a case study, this project intends to employ current technology to perform hate speech detection on social media. I propose a classification method that combines an embedding algorithm with Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN), such as Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), and Multichannel combinations of these networks, using a dataset from Twitter for the training, validation and testing of the neural network. This study implemented nine deep neural models with different architectures, and the multichannel RCNN model performed best among the other models.

**Keywords**: CNN, C-LSTM, GRU, LSTM, NLP, RNN, RCNN, ReLU.

## I. INTRODUCTION

Social media are an interactive online platform where users share and receive information, thoughts, ideas, messages, etc., across networks. Social media has grown to become part of our everyday life. We share every moment of our life, emotions, ideas, and opinions about different topics. Social media tries to break the boundary of location between its users, allowing people to connect and communicate seamlessly anywhere in the world.

Notwithstanding that social media is a tool for self-development, it has also become a tool for bullying through hate speech. Fortuna and Nunes (2018) define hate speech as any rhetoric that criticizes, denigrates or evokes violence or hatred against people depending on certain traits like distinctive features, religion, sexual preference, and gender identity, or even other characteristics, and it can take many different forms, such as nuanced expressions or jokes [1]. Hate speech has been seen to cause social upheaval, destruction, loss of life, and mental illness amongst the target group. Victims of social media bullying most of the time suffer from depression and, in worst cases, become suicidal. This project aims to detect and censor hate speech among social media users, thereby improving the mental health of social media users and aiding in fighting crime since, in some cases, there seems to be a connection between hate speech perpetrators to actual crimes [2].

Most of the online hate detection is done manually. Social media platforms flag comments as hate speech when users report such user accounts or comments that promote hate speech. This method of hate detection is slow compared to the internet speed presently, and the comment made may have already caused enough damage before the social media platform took it down. As a result, an approach is proposed to address the issues faced by the manual approach of hate speech identification. The solution is the use of automatic means of detection through the use of machine learning. Machines are taught to read, comprehend, and categorize text data using accessible tags. This process of training machines to understand and classify text is known as Natural Language Processing (NLP) in machine learning.

Language is one of the things that sets humans apart from animals. Though animals communicate amongst themselves, humans use a more sophisticated means of communication, language grammar. NLP is an interdisciplinary area of study consisting of different fields such as computer science, mathematics, statistics, electronic engineering, biology, philosophy, and psychology, with each field making its unique contributions [3]. This field of study focuses on the interactions between computer and human languages. It is defined as an area of research and application that seeks to find how computers can be employed to understand and manipulate natural language text or speech to do valuable things [4]. NLP creates a representation of texts, adding structure such as syntax and semantics to the unstructured natural language using linguistic rules [5]. Applications of NLP can be seen in machine translations such as google translate and Yandex translate; voice assistance such as Google assistant, Cortana Alexa, and Siri; classification and clustering such as spam filtering and sentiment analysis; chatbots in the company's website; text summarization and information extraction.

Hate Speech detection is an aspect of sentiment analysis, an application of natural language processing in machine learning and artificial intelligence. In this project, we will be focusing on the deep learning approach employed in hate speech detection using different neural networks to arrive at the best method of approach. We aim to build a model that will yield a higher accuracy result in detecting hate speech.

Problem Statement: This project solves the problem of detecting and classifying social media posts made by users about hate speech by assigning hate or non-hate labels using CNN and RNN architectures.
Objectives: This project aims to painstakingly evaluate the performance of the existing natural language processing models toward detecting and predicting hate speech on social media using data extracted from Twitter. In the long run, this exercise will create a conducive social media environment for the users, free of hate and a better society at large.

There are so many approaches that have been employed in the task of detecting hate speech on social media. These approaches involve text classification and can be broken down into manual and automatic/machine learning-based system text classification [6]. Despite being costly and time-intensive, the former usually produces high-quality results since the process is carried out by a human annotator who analyzes the content of the text and categorizes it accordingly [7]. The latter can further be divided into traditional and deep learning text classification. Several works that are based on the traditional text classification focus on three main topics which are: feature engineering, feature selection, and using different types of machine learning algorithms [8]. TF-IDF (Term Frequency-Inverse Document Frequency), N-grams, and BoWV (Bag-of-Words Vector) are some examples of feature engineering. BoWV being the most popularly used feature [8], [9]. Feature selection reduces noisy features, enhances classification accuracy, and makes classifier training and deployment more efficient by reducing the size of the useful vocabulary. Machine learning algorithms use classifiers such as Logistic Regression, Naïve Bayes (NB), Nearest Neighbours (NN), Support Vector Machines (SVM), Decision Trees (DT), and Neural Networks [8].
Deep learning text classification uses multi-layer neural networks to abstract features from tokenized input raw data. Neural network architectures often employed in this area are Dense Networks, Convolution Neural Networks (CNN), and Recurrent Neural Networks (RNN) – LSTM or GRU [10], [11]. CNN is a feed-forward neural network with multiple layer perceptron trained to recognize or pick up patterns through space, while RNN is a looped neural network also trained to recognize patterns through time. Recently, these Deep Neural Networks (DNN) have been shown to perform better in Natural Language Processing (NLP) [10]–[14].

## II. PROPOSED APPROACH

### Dataset
The datasets to be used for this project are 'HatEval SemEval-2019 Task 5' [15], and a 'Twitter-Sentiment-Analysis' by Analytics-Vidhya [16]. The first dataset comprises 13,000 tweets for English in the fields of HS (hate speech), TR (target), and AG (aggression), which are 0.83, 0.70, and 0.73, respectively. However, this dataset's hate speech field will be used for this work. It consists of 4,400 hate speech tweets and 8,840 non-hate speech tweets. The second dataset comprises 31,962 tweets, '1' for 2,242 racist/sexist tweets, and '0' for 2970 not racist/sexist tweets. These datasets will be combined to form a single dataset.

### Data Pre-Processing
Raw data must be processed before being provided to the machine; otherwise, the model won't predict as precisely and won't be able to learn the data quickly. The following pre-processing steps will be taken to clean the data:
- Remove punctuation and special characters (@ # / , ; . : ? ¿ ¡ ! $ ^ &).
- Convert all the texts to lowercase.
- All links and URLs are removed.
- Words starting with a hashtag are broken down into individual words.
- Emoticons are to be removed and replaced with their appropriate meanings.

Before going ahead, it is necessary to load the data and convert it into a data frame using the Panda libraries. Empty entries will be detected and deleted in the dataset. Columns of the datasets with varying names are renamed, and unnecessary columns are dropped to match before the datasets are combined. The dataset is going to be divided into 80% for training and validation, 20% for testing.

### Word Embedding
Word embedding will be utilized to initialize our models, which might lead to better results. Each word is mapped onto a 300-dimensional real-valued vector, where each element is the weight dimension for that word. Assuming that there are 25 words in a sentence, the sentence will be transformed into a 25*300 tensor. The same word embeddings will be utilized for all the proposed models.

### Deep Neural Network
For this project, CNN and RNN are to be employed for the feature extraction and classification of hate speech. Outputs of the classification layer through the sigmoid layer are going to be "1" and "0" for "hate" and "no-hate" speech, respectively. For the compile stage of this work, the Adam optimization algorithm, a stochastic gradient descent method,

is used to update the weight matrix on ten epochs with varying batch sizes. Using estimations of the first and second moments of the gradients, this optimizer derived using adaptive moment estimation calculates distinct adaptive learning rates for various parameters [17]. A binary cross-entropy loss is used for the classification task. The word 'binary' in its name shows that this loss function is used for classifying tasks involving two labels (0's or 1's, True or False). The binary cross-entropy loss is calculated thus:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N} y_i . \log \hat{y}_i + (1 - y_i). \log( 1 - \hat{y}_i) \qquad (1)$$

Where $y_i$ is the corresponding target values with each problem having only two possible classes with target probabilities $\hat{y}_i$ and $(1 - \hat{y}_i)$ for class 1 and class 0 respectively. The N is the number of scalar values in the model output. To calculate for the loss in a class 1 observation the first part of the equation becomes active while the second part disappears, this is vice versa for a class 0 observation The metric used is the accuracy metric to determine the model performance. It is a function that produces two variables, total & count that calculate the frequency predictions made are equal to the labels [20].

**1. Dense Model:**
The dense model of this work comprises the input layer which is the word embedding layer and a flatten layer, 5 dense hidden layers with a ReLU activation function, and another output dense layer with a sigmoid activation. A dropout of 0.2 is placed after the second dense layer.

**2. CNN Models**
i. Simple CNN Model: This is a simple CNN model that consists of an input/embedding layer, a 1D convolution layer with a ReLU activation function, a 1D max-pooling layer, another 1D convolution layer with a ReLU activation function and a flatten layer, 5 dense layers with a ReLU with a dropout layer in-between, and finally, a dense output layer with a sigmoid activation function.
ii. Multichannel CNN Model: The multichannel CNN model of this work is inspired by [21], [22]. The idea of applying CNN for sentence classification using pre-trained word vectors as inputs was proposed in [21], the one-layer CNN architecture performed remarkably. But for this work, the model from [22], [23] which is an improvement of [21] model will be implemented. The CNN contains several layers:
• An input layer (embedding layer): This layer maps all tweets into 300-dimensional real vectors. An input function matrix with a form of a length of longest tweet m x 300 is then transferred by the embedding layer to a dropout layer at a rate of 0.2. The primary objective of the dropout layer is to reduce the over-fitting problem.
• A convolution layer: This layer receives the output of the embedding layer and it consists essentially of three parallel convolution layers of different kernel sizes (2, 3, and 4) on each layer and a rectified linear unit activation function. Each of the kernels has 100 filters resulting in 300 filters in total. The result of the convolution is called a feature map, 100 new feature maps are generated and passed unto a dropout layer with a rate of 0.2
• A 1D max-pooling layer: The convolution characteristics generated in the previous layer are fed for down-sampling as an input to a max-pooling layer, generating 3 (1 x 100) vector outputs. It is then followed by a flatten layer for the three vectors. These three vectors are then concatenated and passed on to the next layer.
• Two dense hidden layers: These layers consist of 250 and 50 neurons that are fully connected, and in-between is a dropout layer of rate 0.2. It receives the result of the concatenation as input.
• An output layer: The output is then fed to the sigmoid-activated dense output layer to generate the final predictions.

**3. RNN Models**
This deep learning model is inspired by the work of [10], the authors utilized an approach that employs a neural network solution composed of multiple or an ensemble of RNN classifiers such as LSTM, BiLSTM and GRU [10]. This model is implemented as a five-layer deep learning network, and is explained as follows:
• Embedding Layer: All the models share the same embedding layer, which is the same as the CNN model. A spatial dropout layer of 0.2 is implemented after the embedding layer for the BiLSTM and GRU models.
• RNN layer: The sigmoid activation was selected for the RNN layer. The dimensionality of the output space for this layer was set to 250 units to model long-range contextual information, producing an output with a shape of (1 x 250). This layer is fully connected to both the input and the subsequent layer.
• Two dense layers: The first dense layer receives the output of the RNN and runs through a fully connected layer with 50 neurons to improve the learning and obtain a more stable output. The ReLU activation function was used. After that, a dropout layer of 0.2 rate is used for regularization. Then follows a second dense layer with 25 neurons and ReLU activation function.
• The output layer: There are 2 neurons in this layer to provide output in the form of probabilities each for the classes, "hate" and "no-hate". A sigmoid activation function was utilized for the output of this layer.

**4. Hybrid Model**

I. Simple C-LSTM: This model is similar to the proposed LSTM model but here, we propose a model that combines the simple CNN model and the LSTM model. Here, a two-layer CNN network with a 1D max-pooling layer in-between comes before the LSTM layer and the dense layers proposed in the LSTM model.

ii. Multichannel C-LSTM: A combined architecture of CNNs and RNNs was proposed in [24]; here, the authors tried combining CNNs with different RNNs like GRUs and LSTMs. The previous models, Multichannel CNN and LSTM, will be joined together for this model. The LSTM layer will be applied before the fully connected layer and after the concatenated 3 feature vectors layer. This model architecture is similar to that used in [11], [23] for their hate speech detection project implementation.

iii. Multichannel RCNN: This model is similar to the Multichannel CNN model in structure but instead of the proposed 3 parallel CNN layers, here one CNN layer with a 1D max-pooling layer, an LSTM and GRU layer of 250 units are connected in parallel, flattened, concatenated and passed on 2 dense ReLU activation layers of 50 and 25 units respectively with a flatten and dropout layer in-between. The output layer is a 2-unit dense layer with a sigmoid activation to yield output according to the classifier used.

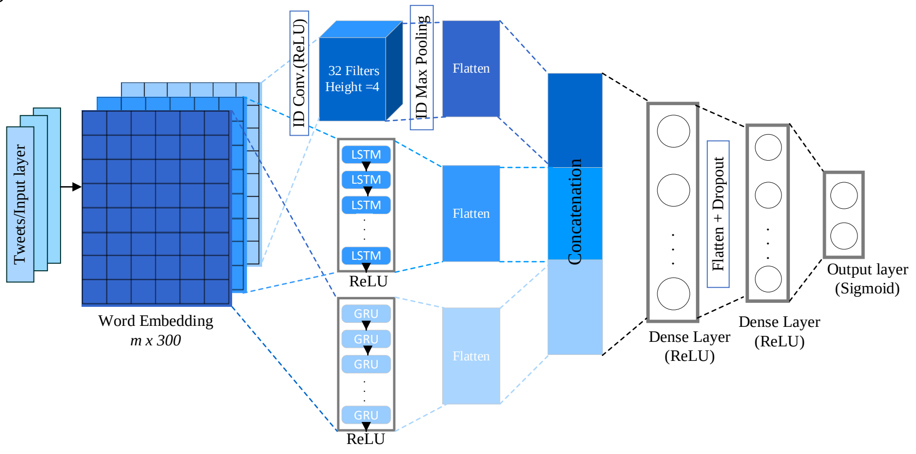

**Figure 1 Multichannel RCNN Model**

## III. Results

During the dataset analysis, the combined datasets realized a sum of 45,202 tweets, 38,560 non-hate tweets, and 6642 hate tweets with a ratio of 5.81 to 1, respectively. It can be deduced that the dataset is not balanced between the two classes. Hence, the hate tweets are extracted and concatenated twice to the combined dataset to realize a new value of 58,486 tweets, 38,560 non-hate tweets, and 19,926 hate tweets with a 1.94 to 1, respectively. The dataset is shuffled, and primary data pre-processing is carried out to feature the data for the project's next stage. Using a Scikit-learn, the dataset was split into a 0.8 train set and a 0.2 test set with a random state of 22. The resulting values are 46,788 tweets for the training set and 11,698 tweets for the test set, with uneven distributions of hate and non-hate tweets. The table I below explains the various data distributions.

**TABLE I DATASET DISTRIBUTION**

| | Non-hate | Hate | Sum |
|---|---|---|---|
| Train Set | 30,847 | 15,941 | 46,788 |
| Test Set | 7,713 | 3,985 | 11,698 |
| Total Dataset | 38,560 | 19,926 | 58,486 |

The pre-processed dataset is passed onto the word embedding stage, where they are tokenized using TensorFlow-Keras-Tokenizer and ready to be fed into the different neural network architectures as input. During the training of the models, the simple CNN and Bi-LSTM models took the shortest and longest time for training, respectively. During the training of all the models, a measure of their training, validation, and testing accuracy was measured for evaluation. Table II shows the record of these values for the respective models employed in this work. From the table below, it can be inferred that the LSTM model performed the least in training, validation, and testing, while the multichannel RCNN model performed best in training and testing but was outperformed by the Dense model in the validation category by a small margin of 0.001 in comparison to all other models. The bar plot of their training performance is shown in figure 2.

TABLE II CUMULATIVE TRAINING RESULTS OF THE MODELS

| Models | Train | Validation | Test |
|---|---|---|---|
| Dense | 0.998504 | **0.960062** | 0.965635 |
| CNN | 0.997585 | 0.949231 | 0.965293 |
| M_CNN | 0.997606 | 0.951992 | 0.967345 |
| LSTM | 0.974994 | 0.811455 | 0.931869 |
| Bi-LSTM | 0.986492 | 0.921380 | 0.938622 |
| GRU | 0.990297 | 0.866148 | 0.935801 |
| M_C-LSTM | 0.998611 | 0.948077 | 0.958027 |
| M_RCNN | **0.999252** | 0.958685 | **0.973243** |
| C-LSTM | 0.996794 | 0.934698 | 0.949820 |

Table III displays the confusion matrix results and the cumulative results of the nine models obtained after testing. The multichannel RCNN model has the best result for the f1-score, precision, and accuracy, while the multichannel C-LSTM has the best result in the recall metrics. The LSTM has the least result in the f1-score, precision, and accuracy metrics but outperformed the Bi-LSTM in the recall metrics by a small margin of 0.0005. It can be inferred that the multichannel RCNN model performed best and the LSTM model the least performed in the experiment. The bar plot of their metric performance is shown in figure 3.

TABLE IIIII CUMULATIVE METRIC RESULTS OF THE MODELS.

| **Models** | **TP** | **FN** | **FP** | **TN** | **F1-Score** | **Precision** | **Recall** | **Accuracy** |
|---|---|---|---|---|---|---|---|---|
| Dense | 7388 | 325 | 77 | 3908 | 0.951083 | 0.923222 | 0.980678 | 0.965635 |
| CNN | 7429 | 284 | 122 | 3863 | 0.950074 | 0.931517 | 0.969385 | 0.965293 |
| M_CNN | 7446 | 267 | 115 | 3870 | 0.952967 | 0.935460 | 0.971142 | 0.967345 |
| LSTM | 7102 | 611 | 186 | 3799 | 0.905063 | 0.861451 | 0.953325 | 0.931869 |
| Bi-LSTM | 7183 | 530 | 188 | 3797 | 0.913619 | 0.877513 | 0.952823 | 0.938622 |
| GRU | 7116 | 597 | 154 | 3831 | 0.910733 | 0.865176 | 0.961355 | 0.935801 |
| M_C-LSTM | 7291 | 422 | 69 | 3916 | 0.941007 | 0.902720 | **0.982685** | 0.958027 |
| M_RCNN | 7476 | 237 | 76 | 3909 | **0.961505** | **0.942836** | 0.980928 | **0.973243** |
| C-LSTM | 7226 | 487 | 100 | 3885 | 0.929759 | 0.888609 | 0.974906 | 0.949820 |

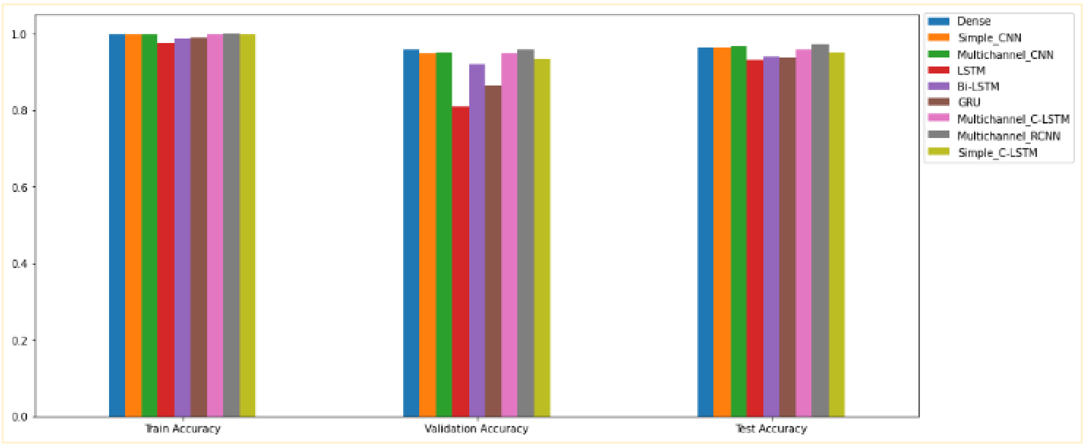

Figure 2 A bar plot of the Model Training Evaluation

DOI: 10.17148/IARJSET.2022.9701

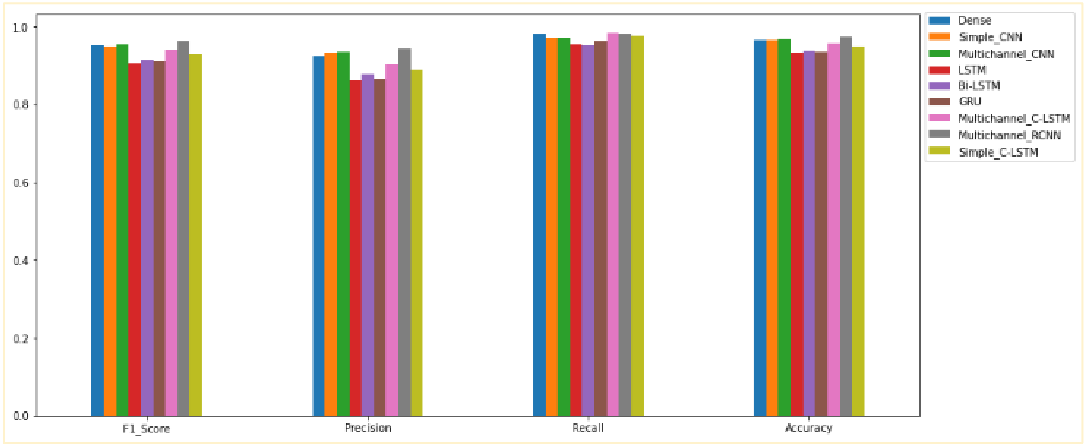

**Figure 3 A bar plot of the Model Performance Evaluation**

## IV. CONCLUSION

The purpose of this study which is to track the performance of CNN, RNN, and the merged CNN-RNN network, was achieved by detecting hate speech using text data extracted from Twitter. The results of the experiment prove that a merged multichannel CNN and RNN consisting of LSTM and GRU is the best model for this task. The result of the LSTM model is not the best in comparison with the other models; however, we cannot neglect its contribution to this exercise.
For future research, I recommend using pre-trained word embeddings such as FastText, Glove and word2vec for the embeddings. I also recommend adding pre-trained NLP models like ELMo, and Transformer neural networks such as BERT and GPT-2 &3 to the list of the models to infer their contributions and performance. Finally, I recommend modifying these models to yield better results as there is varying overfitting in the models.

## REFERENCES

[1] P. Fortuna and S. Nunes, "A Survey on Automatic Detection of Hate Speech in Text," *ACM Comput. Surv.*, vol. 51, pp. 1–30, 2018.
[2] A. Fino, "Defining Hate Speech," *J. Int. Crim. Justice*, vol. 18, no. 1, pp. 31–57, 2020, doi: 10.1093/jicj/mqaa023.
[3] B. Manaris, "Natural Language Processing: A Human-Computer Interaction Perspective," *Adv. Comput.*, vol. 47, no. C, pp. 1–66, 1998, doi: 10.1016/S0065-2458(08)60665-8.
[4] G. G. Chowdhury, "Natural language processing," *Annu. Rev. Inf. Sci. Technol.*, vol. 37, pp. 51–89, 2003, doi: 10.1002/aris.1440370103.
[5] K. Verspoor and K. B. Cohen, "Natural Language Processing," *Encycl. Syst. Biol.*, pp. 1495–1498, 2013, doi: 10.1007/978-1-4419-9863-7_158.
[6] S. MacAvaney, H. R. Yao, E. Yang, K. Russell, N. Goharian, and O. Frieder, "Hate speech detection: Challenges and solutions," *PLoS One*, vol. 14, no. 8, pp. 1–16, 2019, doi: 10.1371/journal.pone.0221152.
[7] Monkey Learn, "Text Classification: What it is And Why it Matters," *Monkey Learn*, 2022. https://monkeylearn.com/text-classification/ (accessed Jun. 28, 2022).
[8] S. Lyu and J. Liu, "Convolutional recurrent neural networks for text classification," *J. Database Manag.*, vol. 32, no. 4, pp. 65–82, 2021, doi: 10.4018/JDM.2021100105.
[9] D. Sarkar, "Understanding Feature Engineering: Deep Learning Methods for Text Data - KDnuggets," 2021. https://www.kdnuggets.com/2018/03/understanding-feature-engineering-deep-learning-methods-text-data.html (accessed Jun. 28, 2022).
[10] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," *Appl. Intell.*, vol. 48, no. 12, pp. 4730–4742, 2018, doi: 10.1007/s10489-018-1242-y.
[11] Z. Zhang, D. Robinson, and J. Tepper, "Hate Speech Detection Using a Convolution-LSTM Based Deep Neural Network," *Proc. ACM Web Conf. (WWW 2018)*, pp. 1–10, 2018, [Online]. Available: https://doi.org/10.475/123_4
[12] Y. Zhou, Y. Yang, H. Liu, X. Liu, and N. Savage, "Deep Learning Based Fusion Approach for Hate Speech Detection," *IEEE Access*, vol. 8, pp. 128923–128929, 2020, doi: 10.1109/ACCESS.2020.3009244.
[13] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," *26th Int. World Wide Web Conf. 2017, WWW 2017 Companion*, pp. 759–760, Jun. 2017, doi: 10.1145/3041021.3054223.
[14] G. L. De la Peña Sarracén, R. G. Pons, C. E. M. Cuza, and P. Rosso, "Hate speech detection using attention-based LSTM," *CEUR Workshop Proc.*, vol. 2263, no. December 2018, pp. 1–5, 2018, doi:

10.4000/books.aaccademia.4784.

[15] M. Zampieri and Google, “OLID - OffensEval Shared Task,” *OffensEval Shared Task*, 2019. https://sites.google.com/site/offensevalsharedtask/olid (accessed Jun. 28, 2022).

[16] “Twitter Sentiment analysis | Kaggle.” https://www.kaggle.com/datasets/arkhoshghalb/twitter-sentiment-analysis-hatred-speech (accessed Jun. 28, 2022).

[17] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Dec. 2014, doi: 10.48550/arxiv.1412.6980.

[18] Peltarion, “Binary crossentropy loss function | Peltarion Platform,” *Peltarion*. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/binary-crossentropy (accessed Jun. 28, 2022).

[19] S. Saxena, “Binary Cross Entropy/Log Loss for Binary Classification,” *AnalyticsVidhya.com*, 2021. https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/ (accessed Jun. 28, 2022).

[20] J. Liu, S. Japip, and T. S. Chung, “Accuracy Metrics,” *International Journal of Hydrogen Energy*, vol. 43, no. 43. pp. 19998–20003, 2018. Accessed: Jun. 28, 2022. [Online]. Available: https://keras.io/api/metrics/accuracy_metrics/

[21] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” *EMNLP 2014 - 2014 Conf. Empir. Methods Nat. Lang. Process. Proc. Conf.*, pp. 1746–1751, 2014, doi: 10.3115/V1/D14-1181.

[22] Y. Zhang and B. Wallace, “A Sensitivity Analysis of (and Practitioners’ Guide to) Convolutional Neural Networks for Sentence Classification,” pp. 253–263, 2015, Accessed: Jun. 28, 2022. [Online]. Available: http://arxiv.org/abs/1510.03820

[23] R. Alshalan and H. Al-Khalifa, “A Deep Learning Approach for Automatic Hate Speech Detection in the Saudi Twittersphere,” *Appl. Sci. 2020, Vol. 10, Page 8614*, vol. 10, no. 23, p. 8614, Dec. 2020, doi: 10.3390/APP10238614.

[24] X. Wang, W. Jiang, and Z. Luo, “Combination of convolutional and recurrent neural network for sentiment analysis of short texts,” in *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, 2016, pp. 2428–2437. Accessed: Jun. 28, 2022. [Online]. Available: https://aclanthology.org/C16-1229/