SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Effect of Typos on Text Classification Accuracy in Word and Character Tokenization

Peter E. Shawky[1,*], Saleh Mesbah ElKaffas[1], Shawkat K. Guirguis[2]

[1] College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt
[2] Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, Alexandria, Egypt

ARTICLE INFO

Article history:
Received 3 August 2023
Received in revised form 7 November 2023
Accepted 5 December 2023
Available online 28 February 2024

Keywords:
Deep learning; CNN; LSTM; Sentiment analysis; Binary classification; Character Tokenization

ABSTRACT

To train a machine to "sense" a users' feelings through writings (sentiment analysis) has become a crucial process in several domains: marketing, research, surveys and more. Nevertheless in times of crisis like COVID. Typo is one of the underestimated challenges processing user-generated text (comments, tweets, ..etc), it affects both learning and evaluation processes. Word tokenization outcome changes drastically even with a single character change, hence as expected, experiments have shown significant accuracy decreases due to typo. Adding a spelling correction as preprocessing layer, building one for every language, is a very time and resources expensive solution, a huge challenge against large data and real-time processing. Alternatively, a CNN model consuming the same text, once tokenized on characters level and once on words level while inducing typo, showed that as the typo percentage approaches 10% of the text, the results with characters tokens surpasses words tokens. Finally, on %30 typo of the text, the model consuming characters tokenization outperformed itself with the word level by a significant %22.3 in accuracy and %24.9 in F1-Score, using the same exact model. This approach in solving the inevitable typo challenge in NLP proved to be of significant practicality, saving huge resources versus using a spelling-correction model beforehand. It also removes a blocker challenge in front of real-time processing of user-generated text while preserving acceptable accuracy results.

1. Introduction

Text classification, especially sentiment analysis, has become one of the very important NLP applications. It helps in processing millions of entries to deduct a sense of a mass satisfaction / dissatisfaction through users' produced text which helps companies and/or organizations and/or scientists to form perspective and/or decisions which are based on a big picture and developing applications that serves real-life solutions as discussed in [16,20,21].

Hence, much research studied sentiment analysis comparing different architectures to get to the one with highest performance. Although the results slightly differ in accuracy and F1-score results, CNN architecture seemed to be the one outperforming other architecture like LSTM and LSTM-CNN

* *Corresponding author.*
E-mail address: eng.peter.ezzatt@gmail.com

https://doi.org/10.37934/araset.40.2.152162

as shown by these studies [1-3,5,8,13-15,19,24]. Tokenization in one of the main data pre-processing before training or testing the CNN model. It splits the data into smaller chunks (tokens). The two main levels to tokenize text are to split words or characters (words and characters tokenization).

The core idea behind this study is the idea that typo, or typing error, is something to expect its occurrence in human-produced text, rather it is inevitable. It even happens with most experienced users because -after all- they are humans. Wouldn't that affect a model performance? Indeed.

Typo changes characters. For word tokenization, that will change a word's whole token as it's a different word than the original and correctly spelled word. The study looks through the effect of typo using a CNN model results, with both tokenization levels: words and characters. CNN has been shown to be one of -if not *the*- most efficient architecture for sentiment analysis or text classification in general [7-8,10-12,19]. The near-obvious expectation was typos would significantly affect accuracy for it significantly alter a word's token, learning and testing mis-spelled word [4,6,9].

A solution to tackle the typo problem is spelling-correction models as a preprocessing layer. This might be achieved by piping the training and testing text through another model as a preprocessing procedure before every training and evaluation process. That approach might sound intuitive but at the same time it creates a new set of greater challenges. The first one is the need to provide and maintain a specific model and lexicons for every single language and its dialects, which is already a huge challenge in NLP domain due the variant complexity of many languages and dialects regarding grammar and structure to begin with as these studies show [11, 21-23,25]. Then comes the challenge of the doubled resources needed, due to having two ML models (at least) running instead of one. Theses challenge gets more complicated and costly exponentially in regard of data/stream size and flow, added to the practical aimed processing timeframe for real-time purposes. This solution will consume resources and valuable time evaluating large datasets and/or high flow of streamed data. Aside from that, a spelling-correction model with a certain accuracy percentage would, again, replace human-generated typo with computer-generated ones, if not add to it.

The hypothesis behind this study is this: while, indeed, typo will equally change a word token as to a single character, yet a piece of text -obviously- includes more characters than words. Which means that for the whole piece of text, the overall tokens pattern will become significantly less altered using characters tokenization than word tokenization.

To test this hypothesis, rebuilding experiments with a typo-less text, an ordinary text classification, was the start to get a benchmark for the initial results [2]. Then gradually introduce and increase percentage of typo was the way to see how both tokenization levels performs. On each step, text is tokenized one time on word and one time on character and investigate the difference in performance.

Before introducing any synthetic typo, a solid ground of the results training a model with the clean dataset were needed, on both level: word and character. For the comparison, the start was to run the experiment on both tokenization levels: words and characters. Starting with these results as a benchmark, it was able to measure the accuracy decrease due to typo occurrence the same model as the percentage of typo occurrence in text increments gradually.

Upon recording the results on the typo-less dataset, adding synthetic typo to the training data and re-train the same model was the second step to get a demonstration of the effect of typo on the accuracy on both levels.

Finally, tweaking the CNN model to improve the accuracy despite the typo then train the new model to get the results of the tweaked model's accuracy.

2. Methodology

2.1 Dataset

This dataset is a large IMDb movie review dataset. The dataset contains 50,000 movie reviews categorized into either positive or negative. The dataset is available within Keras' library where each review is encoded as a sequence of word indexes. Then the dataset is split into a 70% for training and 30% for testing data. 10% of the training samples has been used as validation data duplicating the configurations conducted by previous studies [2,13]. The reviews length has been unified by zero-padding the shorter reviews so that is easy to train the architectures, which is padding.

2.2 Preprocessing

2.2.1 Padding

All the neural networks require to have inputs of the same shape and size to feed the input nodes. However, when it is pre-processed and used the texts as inputs for the model, not all the sentences have the same length. In other words, naturally, some of the sentences are longer or shorter. It was needed to have the inputs with the same size, this is where the padding is necessary.

The list of sentences that have been padded out into a matrix where each row in the matrix has an encoded sentence with the same length this is due to the:

- Additional zeros for short sentences and
- Truncating the sentences which exceed the max number of words which is declared by maxlen.

For the IMBD dataset, a script has been developed to find the maximum length to find out that the longest review. The result was 13,590 characters or 2,461 words.

2.2.2 *Synthetic typo*

To evaluate how well a CNN model works against text with typo, it was necessary to develop an algorithm to generate typo. Intuitively, a typo happens mostly when a finger slips to push a key that isn't supposed to be pushed, changing the word spelling. It's considered that there are three main factors during typo:

i) The switched characters' positions.
ii) The replacement character on each typo.
iii) The amount of typo is a variable and depends on how frequent typo happens, which determines the occurrence percentage in a text.

Typo happens randomly. So, it was needed to randomize the key switched while maintaining the typo percentage within the text. It was done by choosing one random character in a set of characters of length equals to the inverted pre-determined percentage. For instance, a %2 typo occurrence translates to a random position of a character among a set of 50 characters, a %4 translates to on character of 100 characters, and so on. Dividing text to sets of a certain length is likely to produce a single set with less than the intended length to randomize from. The final truncated set's increase in percentage as neglectable is neglected.

Secondly, the replacement character on each typo is also random. A list of all possible typos for each key is set by listing all surrounding keys around each character. For instance, the key "f" is

surrounded by the keys: ‘r’, ‘t’, ‘g’, ‘v’, ‘c’, ‘d’ and ‘e’. For each of the surrounding keys are listed, and after randomizing the character’s position to be replaced, a randomized pick among the set of surrounding keys happens, then the character is replaced.

Finally, a person’s typing skills determine the frequency of error. To investigate typo effect on the model accuracy, the intended was to test the model against different amount of typo in data. It got varied the occurrence percentage between %2 to %30 on a %2 interval.

2.2.3 *Tokenization*

Tokenization is a way of separating a piece of text into smaller units called tokens. Usually, tokens can be either words, characters, or subwords. Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization. Tokenization prepares the text for the CNN embedding layer to be represented by vectors.

This is the core of the study, to investigate which tokenization level will do better with the presence of typo: words or characters tokenization.

2.2.4 *Convolutional neural network (CNN)*

Convolutional Neural Networks are like typical Neural Networks, these networks are consisting of neurons that own learnable weights and biases as demonstrated on the studies [17,18]. Some input was received by each neuron; a dot product was performed. CNN consists of one or more convolutional layers after that its followed by fully connected layers like the ordinary multilayer neural network. CNN works using three key concepts (local receptive fields, shared weight and biases, and activation and pooling). A small region of neurons in the input layer are connected to hidden layer neurons. These small regions are called local receptive fields. The network has neurons with weights and biases. During the training process, the model learns the weight and biases values however these values are the same for neurons in the hidden layer. The step of activation function applies the conversion to the output of each neuron by using Rectified Linear Unit (ReLU). ReLU is a commonly used activation function. The function of pooling step is to condense the output of the convolutional layer by half by reducing the dimensionality of the features map. Hence, CNN can be used for learning structure in paragraphs of words.

2.2.4.1 *Embedding layer*

Embedding is one of the main prerequisites for most of the NLP tasks where natural text is the subject. While working with any kind of text. Each word/character must be converted into an n-dimensional vector for fitting to any architecture.

Now, there is two ways of doing this. One is one-hot key encoding which includes Bag of words model and another one is word embedding. Bag of words is a very sparse representation which results in a waste of memory whereas Word embedding provides a compact representation of each word. Word embedding is done in a way that similar kinds of words are gathered in n’th dimensional space.

Here n represents the vector dimension of each embedded word. There are two pre-trained word embedding models. An embedding layer that has been provided by Keas has been used, a vocabulary of 8000 unique words is used and each word is embedded into a 100 dimensions vector space. Instead of using a pre-trained embedding word model, the embedding layer by has been trained using the training samples from IMDb movie review dataset.

2.2.4.2 *Convolution layer*

As per text is a 1-dimension data, the Keras' 1 dimension convolution layer has been used. This layer creates a convolution kernel that is convolved with the layer input over a single spatial (or temporal) dimension to produce a tensor of outputs.

2.2.4.3 *Dense layer*

The dense layer's neuron in a model receives output from every neuron of its preceding layer, where neurons of the dense layer perform matrix-vector multiplication. Matrix vector multiplication is a procedure where the row vector of the output from the preceding layers is equal to the column vector of the dense layer. The general rule of matrix-vector multiplication is that the row vector must have as many columns like the column vector.

For the activation function, the *sigmoid* function has been chosen because the matter in hand is a binary classification [16,20].

$$\text{sigmoid}(x) = 1 / (1 + \exp(-x))$$

Sigmoid is equivalent to a 2-element Softmax, where the second element is assumed to be zero. The sigmoid function always returns a value between 0 and 1, hence the binary result.

2.3 *Experiments*

Running the CNN model with different configurations, grouped together, helped in comparing results. All Experiments are repeated on both word and characters level, as mentioned above, hence the experiments consist of pairs of model compilation and evaluation. Figure 1 shows the Experiments architecture. For different typo percentage of the text, the experiment is run once with word tokenization and once with character tokenization for results comparison.

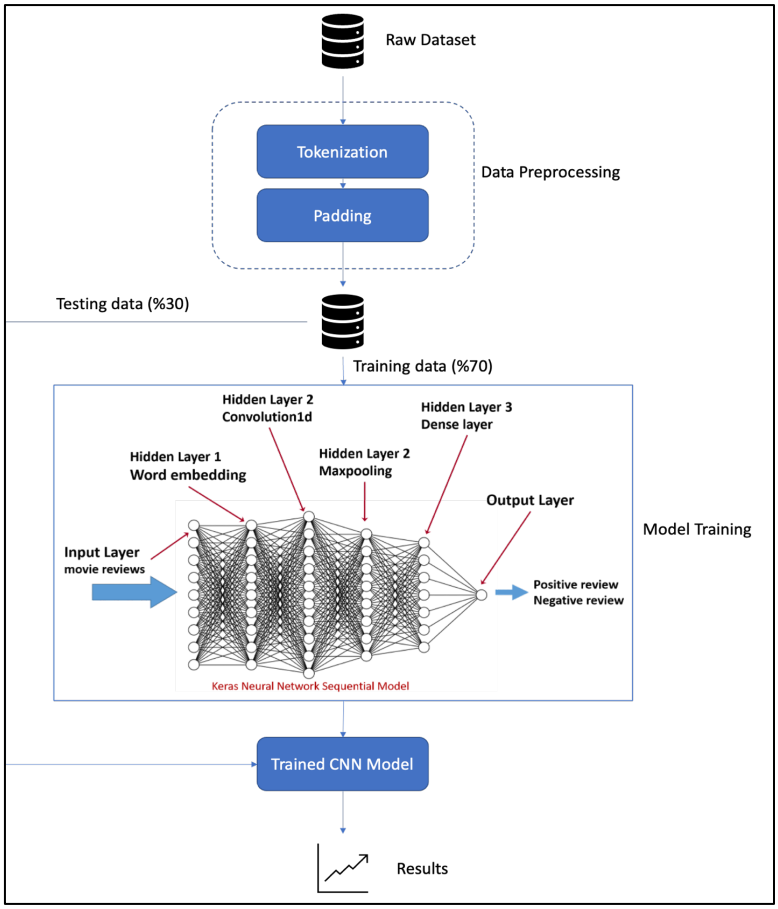

Fig. 1. Experiments architecture

3. Results

Results showed significant decrease in the model accuracy result with the introduction of synthetic typo using both tokenization levels (word and character) on every metric (Table 1). Increasing the percentage of typo occurrence in the text demonstrated that the character tokenization was significantly affected by typo versus the word tokenization, hence better performance with higher percentage. Up to a certain percentage of typo (~10%) the model running with tokenized data on the character level outperforms the same model running with tokenized data on the word level.

At 0% of typo (before inducing any synthetic typo), the experiment result showed the same accuracy result as this study [2]. This result put a benchmark to have a trust on the following rounds of the experiments raising the typo percentage up.

Table 1.
Model performance running with Word tokenization VS Character tokenization

%		Accuracy	F1 Score	Precision	Loss	Recall
0	C	0.8569	0.8588	0.8317	0.3296	0.8966
	W	0.895 [2]	0.8923	0.8911	0.3795	0.9004
2	C	0.8489	0.842	0.8607	0.3413	0.8328
	W	0.8772	0.8739	0.8717	0.452	0.8836
4	C	0.8353	0.8645	0.8611	0.3673	0.8709
	W	0.8600	0.8556	0.8572	0.4983	0.8614
6	C	0.8064	0.7992	0.8106	0.6953	0.7983
	W	0.8427	0.8389	0.8426	0.3582	0.8447
8	C	0.7967	0.7912	0.795	0.7512	0.7976
	W	0.8244	0.8147	0.8435	0.3862	0.7974
10	C	0.8095	0.7893	0.8624	0.4166	0.7387
	W	0.7967	0.7912	0.795	0.7512	0.7976
12	C	0.8112	0.8155	0.7839	0.4089	0.86
	W	0.762	0.759	0.7567	0.8922	0.7732
14	C	0.8088	0.7946	0.8405	0.4199	0.7634
	W	0.7452	0.7398	0.7412	0.751	0.9406
16	C	0.8020	0.8068	0.7719	0.4281	0.8559
	W	0.7290	0.7256	0.7251	0.9698	0.7395
18	C	0.8070	0.8027	0.8018	0.4156	0.8138
	W	0.7345	0.7291	0.7328	0.9597	0.7376
20	C	0.7929	0.7813	0.8094	0.4396	0.7657
	W	0.7135	0.7086	0.7101	1.0053	0.7212
22	C	0.7965	0.7884	0.8067	0.4309	0.7813
	W	0.7109	0.7083	0.7076	1.0747	0.7233
24	C	0.7838	0.7906	0.7567	0.4545	0.84
	W	0.6742	0.6708	0.6719	1.1482	0.6833
26	C	0.79094	0.7849	0.7915	0.4458	0.7901
	W	0.6762	0.6731	0.6725	1.1435	0.6878
28	C	0.7813	0.7684	0.799	0.4599	0.7516
	W	0.668	0.6606	0.6695	1.2086	0.6668
30	C	0.7683	0.7777	0.736	0.484	0.8356
	W	0.6282	0.6265	0.6257	1.2072	0.6419

The results show that as the typo percentage raises, even though the metrics decline for both experiments, yet the one with character tokenization shows more resilience to the effect of typo. To clearly and visually demonstrate the flip of performance, charts have been generated using Table 1 to demonstrate the change in metrics: Accuracy (Figure 2), F1 score (Figure 3), Precision (Figure 4), Loss (Figure 5) and Recall (Figure 6).

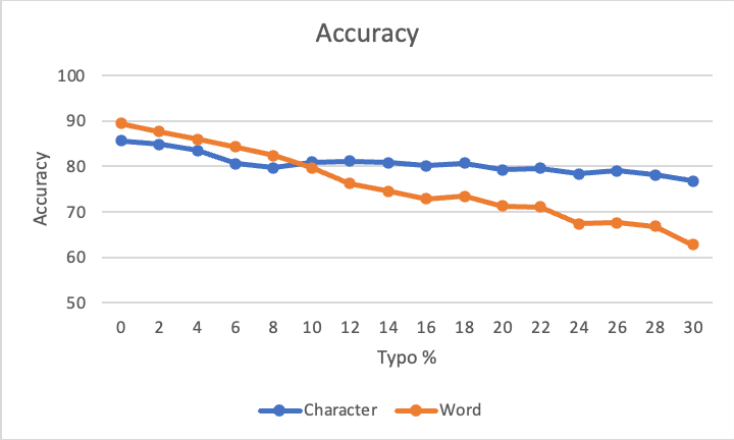

Fig. 2. Model Accuracy using word vs character tokenization Word tokenization accuracy outperforms the word tokenization as the typo percentage increases

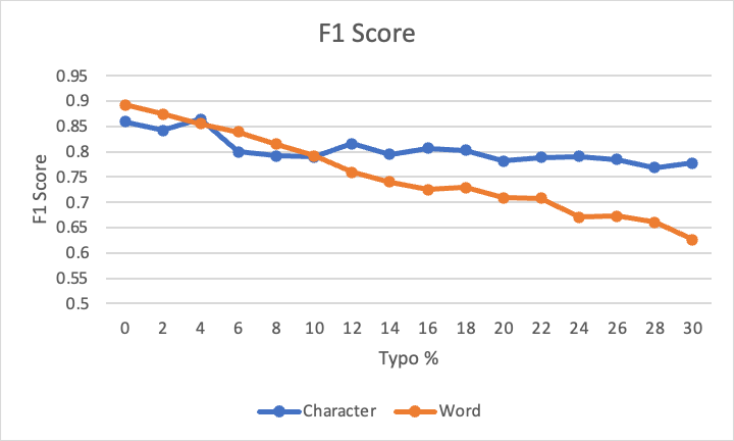

Fig. 3. Model F1 Score using word vs character tokenization Word tokenization F1 score outperforms the word tokenization as the typo percentage increases

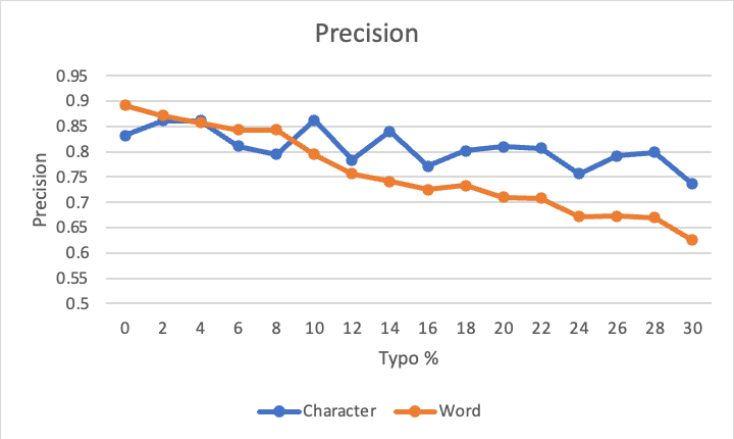

Fig. 4. Model Precision using word vs character tokenization Word tokenization precision outperforms the word tokenization as the typo percentage increases

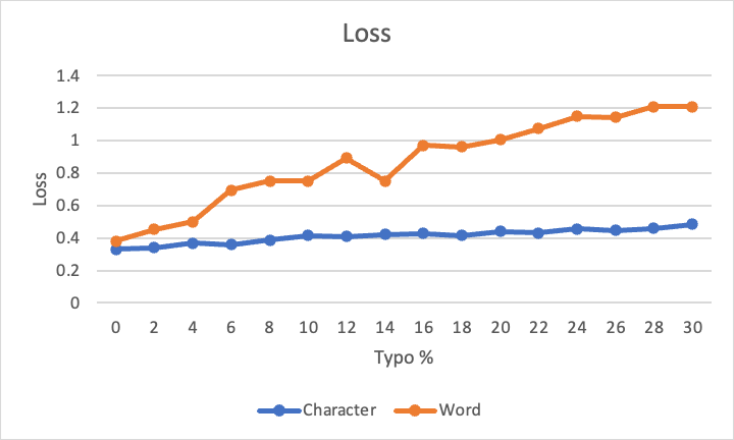

Fig. 5. Model Loss using word vs character tokenization
Word tokenization loss is more stable compared to the word tokenization as the typo percentage increases

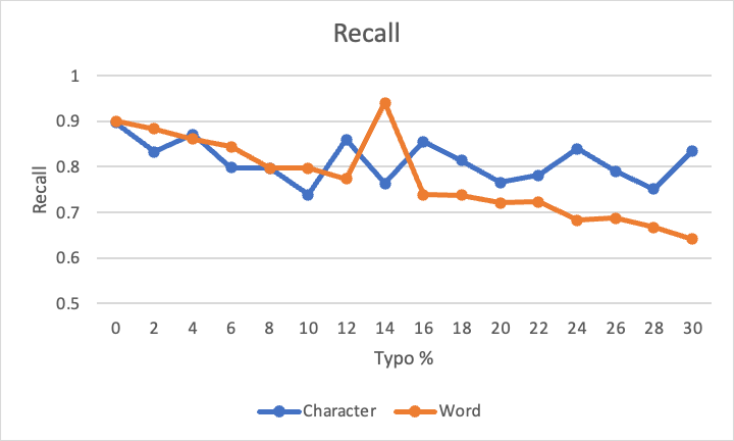

Fig. 6. Model Recall using word vs character tokenization
Word tokenization recall outperforms the word tokenization as the typo percentage increases

4. Conclusion

From the data and the chart, it is shown that tokenizing on the word level exceeds character at 0 typos as clearly shown by the following studies [1-3,6,9], a perfectly text with no spelling mistakes. But realistically, as typos occur, tokenizing text on characters level helps the CNN model to have better accuracy results. Spelling mistakes has less effect on vectoring characters one by one, hence making better prediction. AS IF the model is guessing the write spelling from the characters' vector representation.

Perhaps adding another NLP model to correct spelling before text classification model with word tokenization would make the word surpasses the character [14]. But obviously, this would slowdown the whole process immensely. That's why this finding can be practical to have better acceptable accuracy on public text classification without the need to correct spelling.

Acknowledgement
This research was not funded by any grant. The author would like to thank Arsani Boshra for his contribution and for providing very helpful and insightful comments.

References

[1] Ali, Nehal Mohamed, Marwa Mostafa Abd El Hamid, and Aliaa Youssif. "Sentiment analysis for movies reviews dataset using deep learning models." *International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol* 9 (2019). https://doi.org/10.5121/ijdkp.2019.9302

[2] Haque, Md Rakibul, Salma Akter Lima, and Sadia Zaman Mishu. "Performance analysis of different neural networks for sentiment analysis on IMDb movie reviews." In *2019 3rd International conference on electrical, computer & telecommunication engineering (ICECTE)*, pp. 161-164. IEEE, 2019.

[3] Luan, Yuandong, and Shaofu Lin. "Research on text classification based on CNN and LSTM." In *2019 IEEE international conference on artificial intelligence and computer applications (ICAICA)*, pp. 352-355. IEEE, 2019. https://doi.org/10.1109/ICAICA.2019.8873454

[4] Yoon, Hoijin. "Finding unexpected test accuracy by cross validation in machine learning." *International Journal of Computer Science & Network Security* 21, no. 12spc (2021): 549-555.

[5] Hassan, Abdalraouf, and Ausif Mahmood. "Efficient deep learning model for text classification based on recurrent and convolutional layers." In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1108-1113. IEEE, 2017.

[6] Minaee, Shervin, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. "Deep learning--based text classification: a comprehensive review." *ACM computing surveys (CSUR)* 54, no. 3 (2021): 1-40. https://doi.org/10.1145/3439726

[7] Li, Qian, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. "A survey on text classification: From shallow to deep learning." *arXiv preprint arXiv:2008.00364* (2020).

[8] Wu, Hongping, Yuling Liu, and Jingwen Wang. "Review of Text Classification Methods on Deep Learning." *Computers, Materials & Continua* 63, no. 3 (2020).

[9] Zulqarnain, Muhammad, Rozaida Ghazali, Yana Mazwin Mohmad Hassim, and Muhammad Rehan. "A comparative review on deep learning models for text classification." *Indones. J. Electr. Eng. Comput. Sci* 19, no. 1 (2020): 325-335. https://doi.org/10.11591/ijeecs.v19.i1.pp325-335

[10] Li, Qian, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip S. Yu, and Lifang He. "A survey on text classification: From traditional to deep learning." *ACM Transactions on Intelligent Systems and Technology (TIST)* 13, no. 2 (2022): 1-41. https://doi.org/10.1145/3495162

[11] Lu, Hongxia, Louis Ehwerhemuepha, and Cyril Rakovski. "A comparative study on deep learning models for text classification of unstructured medical notes with various levels of class imbalance." *BMC medical research methodology* 22, no. 1 (2022): 181. https://doi.org/10.1186/s12874-022-01665-y

[12] Zhou, Hai. "Research of text classification based on TF-IDF and CNN-LSTM." In *Journal of Physics: Conference Series*, vol. 2171, no. 1, p. 012021. IOP Publishing, 2022. https://doi.org/10.1088/1742-6596/2171/1/012021

[13] Varshitha, Konda Sai, Chinni Guna Kumari, Muppala Hasvitha, Shaik Fiza, K. Amarendra, and Venubabu Rachapudi. "Natural Language Processing using Convolutional Neural Network." In *2023 7th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 362-367. IEEE, 2023.

[14] Wang, Wei, and Jianxun Gang. "Application of convolutional neural network in natural language processing." In *2018 international conference on information Systems and computer aided education (ICISCAE)*, pp. 64-70. IEEE, 2018. https://doi.org/10.1109/ICISCAE.2018.8666928

[15] Song, Peng, Chaoyang Geng, and Zhijie Li. "Research on text classification based on convolutional neural network." In *2019 International conference on computer network, electronic and automation (ICCNEA)*, pp. 229-232. IEEE, 2019. https://doi.org/10.1109/ICCNEA.2019.00052

[16] Shakil, Mahmudul Hasan, and Md Golam Rabiul Alam. "Hate Speech Classification Implementing NLP and CNN with Machine Learning Algorithm Through Interpretable Explainable AI." In *2022 IEEE Region 10 Symposium (TENSYMP)*, pp. 1-6. IEEE, 2022.

[17] Albawi, Saad, Tareq Abed Mohammed, and Saad Al-Zawi. "Understanding of a convolutional neural network." In *2017 international conference on engineering and technology (ICET)*, pp. 1-6. Ieee, 2017. https://doi.org/10.1109/ICEngTechnol.2017.8308186

[18] Wang, Jiaying, Jing Shan, Yaxin Li, Yi Ren, Duo Zhai, and Jinling Bao. "Text Classification Using Scope Based Convolutional Neural Network." In *2019 IEEE International Conferences on Ubiquitous Computing & Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*, pp. 402-407. IEEE, 2019. https://doi.org/10.1109/IUCC/DSCI/SmartCNS.2019.00093

[19] Lavanya, P. M., and E. Sasikala. "Deep learning techniques on text classification using Natural language processing (NLP) in social healthcare network: A comprehensive survey." In *2021 3rd international conference on signal processing and communication (ICPSC)*, pp. 603-609. IEEE, 2021.

[20] Gu, Tong, Guoliang Xu, and Jiangtao Luo. "Sentiment analysis via deep multichannel neural networks with variational information bottleneck." *IEEE Access* 8 (2020): 121014-121021.
[21] Hossain, Md Rajib, Mohammed Moshiul Hoque, and Nazmul Siddique. "Leveraging the meta-embedding for text classification in a resource-constrained language." *Engineering Applications of Artificial Intelligence* 124 (2023): 106586. https://doi.org/10.1016/j.engappai.2023.106586
[22] Hossain, Md Rajib, and Mohammed Moshiul Hoque. "Toward Embedding Hyperparameters Optimization: Analyzing Their Impacts on Deep Leaning-Based Text Classification." In *The Fourth Industrial Revolution and Beyond: Select Proceedings of IC4IR+*, pp. 501-512. Singapore: Springer Nature Singapore, 2023. https://doi.org/10.1007/978-981-19-8032-9_35
[23] Hossain, Md Rajib, Mohammed Moshiul Hoque, Nazmul Siddique, and Iqbal H. Sarker. "CovTiNet: Covid text identification network using attention-based positional embedding feature fusion." *Neural Computing and Applications* 35, no. 18 (2023): 13503-13527. https://doi.org/10.1007/s00521-023-08442-y
[24] Hossain, Md Rajib, and Mohammed Moshiul Hoque. "Covtexminer: Covid text mining using cnn with domain-specific glove embedding." In *International Conference on Intelligent Computing & Optimization*, pp. 65-74. Cham: Springer International Publishing, 2022. https://doi.org/10.1007/978-3-031-19958-5_7
[25] Hossain, Md Rajib, and Mohammed Moshiul Hoque. "Towards Bengali word embedding: corpus creation, intrinsic and extrinsic evaluations." (2020). https://doi.org/10.20944/preprints202012.0600.v1