Future Computer Science Journal (FCSJ)
Vol. 1, No. 2, December 2023: 45-52
http://asasijournal.id/index.php/fcsj
http://doi.org/10.xxxxx/fcsj.xxxx.xxx

Detection of Cross-Site Scripting Attacks with Code Analysis Using Text Convolutional Neural Networks as a Step to Improve User Security

Nur Azis Kurnia Rianto[1*], Alamsyah[1]
[1,2]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang, Indonesia

Abstract
The main objective of this research is to build a reliable model capable of classifying Cross-Site Scripting (XSS) attacks through input analysis using the Convolutional Neural Network (CNN) method. The input in question is a JavaScript or HTML script indicated to be included in the XSS attack script. The development of the model architecture is based on the basic Text CNN architecture. The TensorFlow Keras library in Python is used to build the model architecture. The model is trained to study the correlation between data using data from the internet. The model's performance in data classification tasks will be evaluated using accuracy metrics and binary cross-entropy functions. The built model can accurately classify cross-site scripting attack data of 99.95% with a loss rate of 0.29%. To get the optimal model architecture, several experiments are needed to determine the correct number, components, and filter layer size. The Text CNN method for classifying XSS attacks is a new approach to detecting and preventing XSS attacks. The proposed CNN method is specifically for text processing that has been widely used in various fields and has proven performance. Input analysis is the foremost approach used and is crucial in preventing XSS attacks, considering that these attacks are generally carried out by code injection. The built model can accurately classify cross-site scripting attack data of 99.95% with a loss rate of 0.29%. The application of the Text CNN method makes the proposed model quite reliable and able to outperform previous methods in the XSS attack classification task.

Keywords:
Cross-site scripting (XSS); Convolutional Neural Networks; (CNN), and Text Classification;

Article History:
Received: December 19,2023
Revised: December 27,2023
Accepted: December 28,2023
Published: December 28,2023

Corresponding Author:
Nur Azis Kurnia Rianto
Computer Science Department,
Universitas Negeri Semarang
Indonesia
Email:
azaiskr305@students.unnes.ac.id

INTRODUCTION

Information system security has become a top priority in today's digital era. Hacking attacks, including cross-site scripting (XSS), have significantly threatened data integrity and confidentiality [1]. XSS attacks take advantage of vulnerabilities in web applications [2]to enter and run malicious code in a user's browser[3], which can lead to sensitive data theft, data manipulation, and system crashes [4].

Cross-site scripting (XSS) attacks are cyber attacks that threaten the security of web applications [5]. XSS attacks occur when an attacker inserts malicious code (HTML or JavaScript) into an otherwise secure web application[6], which makes the form or element accessible to the user. The attack execution flow typically involves sending data as input to a web application via an untrusted source, then combined into a dynamically constructed response and distributed to web users without being validated as malicious content [2]. This attack can result in data corruption [7], such as theft of sensitive data, taking control of software, and threatening user privacy.

Effective and reliable detection methods and techniques protect systems from XSS attacks. Many studies have been conducted to solve this problem [8][9],[2], [10], but there is still the problem of classifying safe and potentially dangerous inputs with high accuracy. Therefore, this research concentrates on creating a more sophisticated and accurate method of classifying XSS attacks.

The main objective of this research is to classify XSS attacks through input analysis using the Convolutional Neural Network (CNN) method, which has proven successful in various applications, including text

processing.[11][12], where CNNs can extract important information and learn complex patterns [13]. This research will use Text CNN to classify inputs as safe or dangerous. This method generates unique text features using convolution and max-pooling on a vector representation of words[14]. By using this method to perform input analysis, the built model is expected to distinguish with high accuracy between safe input and those containing XSS attacks.

METHOD

This section details all the steps taken in the research. The first step is to collect research data from existing data sets online. The data set used is the data set regarding Cross Site Scripting attacks in JavaScript or HTML code, along with the labels. Data preprocessing is then carried out on the data set to standardize and prepare the data before being processed by the model. The clean data is then used to train the Text CNN model that has been built. The model is then evaluated using test data and see how it performs. The model will continue to be optimized if the performance has not reached the specified target. The research steps carried out are shown in Figure 1.

Material

This study uses data sets obtained from the Kaggle website[15]. The data set contains 13685 data in the form of XSS and non-XSS attack code, stored in the 'Sentence' column with the data index starting at 0. Furthermore, in the "Label" column, data included in the XSS attack is labeled 1, while data not included in the XSS attack code is labeled 0. The code is in the form of pure JavaScript or HTML program code with different lengths. A sample of the data in the data set used is presented in Table 1.

Table 1. Sample dataset

	Sentence	Label
0	<a href="/wiki/Mind%E2%80%93body_problem" ...	0
1	...	1
2	<col draggable="true" ondragenter="alert(1)">test</col>	1
3	<caption onpointerdown=alert(1)>XSS</caption>	1
4	<link rel="mw-deduplicated-inline-style"...	0

The research is carried out in three main steps: data collection, preprocessing, and system development. Evaluation is carried out on the system by reviewing the level of accuracy and efficiency of the system, as presented in Figure 1.

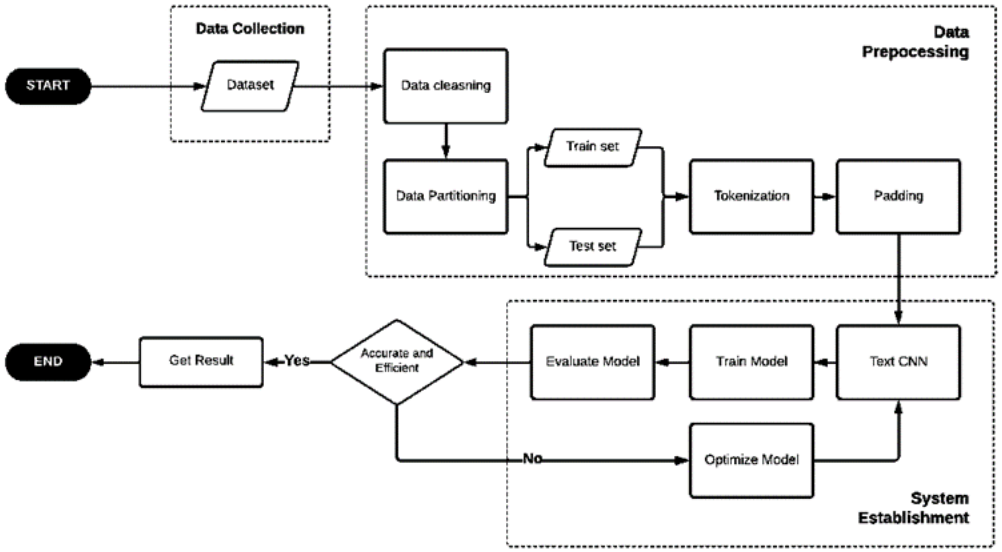

Figure 1. Research steps

Data cleansing is the first step in the data preprocessing stage. Data cleaning is done by deleting data rows based on indexes with no value for the "Sentence" or "Label" columns. The data list cleaned with rows with no value is divided into a train set and a test set. In division, "Sentence" and "Label" will be separated so that the data division will produce train_data, test_data, train_label, and test_label. For data sets, the division's " Label " results will match the model's computational results with their original values. Distribution is done randomly (not based on index data) with a ratio of 70% for the train set and 30% for the test set. The train set will be used for model training, while the test set will be used for model evaluation.

Tokenization converts text into a sequence of tokens or words in numerical form[16]. The tokenization process will convert text input into numeric data that Machine Learning models can process. Tokenization is done on train_data and test_data. To tokenize, some variables are set so that the data has the same form before being processed by the model. Variable settings include max_features = 20,000 and max_len = 300.

Setting the max_features variable determines the maximum number of words to be stored based on the times they occur. Words with high occurrence frequency will be preserved. Max_features will generate a kind of dictionary the model uses for machine learning.

The max_len variable determines the maximum length of the word sequence in each process. Sequences of words that have a length exceeding the max_len value will be truncated, while sequences that are less than max_len will be padded by adding a value of 0 so that word sequences are the same length.

Padding is a process to uniform the word order that has been tokenized into the same length[17]. The padding process is needed because Machine Learning models require input with consistent dimensions when processing data related to word order[17]. Padding is performed on numeric sequences with a length less than the max_len limit by adding a 0 to the front of the sequence[17]. The padding step is performed on both tokenization results (train_data and test_data).

Text CNN is a deep learning architecture processing text data[18]. Text CNN uses a convolution layer to extract local features from a sequence of words[19]. Convolution is done by using a filter on the order of words to get the relevant features. Each filter has a width equal to the number of columns in the word order and a height specified by the hyperparameter[20]. The filter is shifted gradually over the entire sequence of words to produce features using the convolution operation[21]. The model can identify basic patterns in locally repeated text using convolution filters.

After convolution, layer pooling is used to reduce the dimensionality of the features produced by the convolution layer. Pooling can be done with operations such as max-pooling, which takes the maximum value of each feature in a given region. Max-pooling can take essential features in the region and ignore less critical information. Text CNN does not depend on the absolute position of the words in the sequence. The features generated by the convolution and pooling layers do not depend on the order of the words as a whole but only on the local patterns detected by the filter.

After the convolution and pooling layers, the resulting features will be connected to the fully connected layers to process and study the relationship between these features. Fully connected layers use neuron units that are fully connected to perform classification or regression.

The output layer on Text CNN depends on the task to be completed. For example, in a text classification task, the output layer could be a softmax layer that generates the probabilities for each class. The general architecture of the Text CNN is shown in Figure 2.

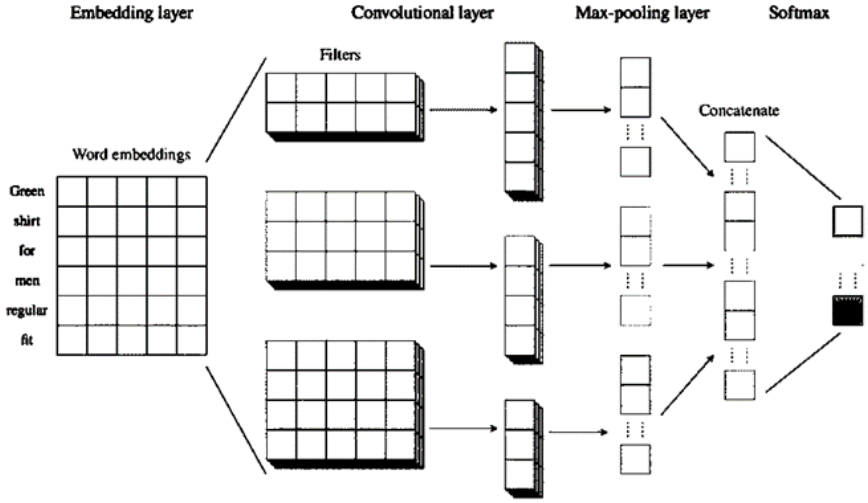

Figure 2. General CNN architecture for text classification

Proposed Model.

The CNN Text Model architecture used to classify Cross Site Scripting attacks based on input analysis is presented in Figure 3.

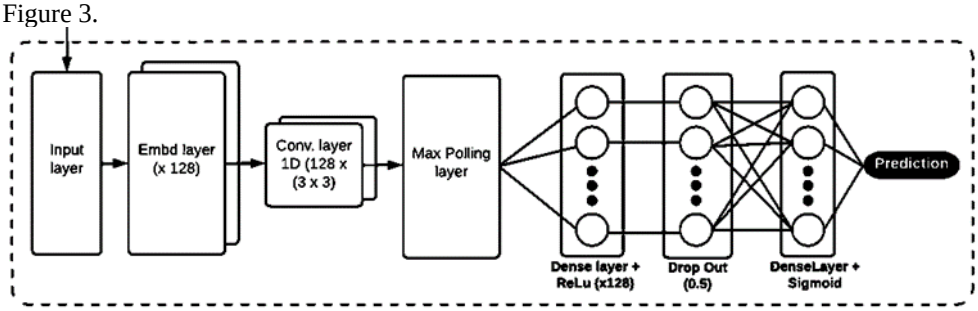

Figure 3. The Architecture of Proposed Text CNN Model

The model consists of 1 input layer, one output layer, and five processing layers. The explanation of each layer is as follows:

1. Input layers: is the input model layer to enter the tokenized sequence and padding with the size being the maximum length of the word sequence (max_len).
2. Embedding layers: used to convert the input sequence into a numeric vector. In the built model, the embedding layer has a vector size 128.
3. Convolutional 1D layers: The convolutional 1D layer functions to extract features from the embedded word sequence. Most convolutional 1D layers that build the model are 128 kernels with kernel size = 3. In this layer, the ReLu (Rectified Linear Unit) activation function is added to disable negative values. The ReLu activation function equation can be expressed in .$y(u) = \max(0, u)$
4. Max pooling layer: the type of max pooling layer used by the model is global max pooling. This layer is used to reduce the dimensions of the feature map produced by the previous layer. The global max pooling layer will take the maximum value as a representation of each feature's firm value[22] in one sequence to obtain a more concise representation.
5. Dense layer + ReLu: dense layer is a fully connected layer used to study the correlation between the features that have been obtained. The dense layer with the ReLu activation function in the model totals 128 neuron units.
6. Drop-out layers: The drop-out layer is added to the model to prevent overfitting during model training. The drop-out layer in the model will ignore the 0.5 part of the neuron when processing the results from the previous layer to the next layer.
7. Dense layers + sigmoid: fully connected layer with sigmoid activation function to generate probability prediction values for input data. The number of neurons for this layer is only one, and the sigmoid activation function will produce a predicted value from 0 to 1. Mathematically, thsigmoid activation function is expressed as $y(u) = \frac{1}{1+e^{(-u)}}$

The model summary shows each layer's size and its parameter composition. Presented in Figure 4.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 300)]             0

embedding (Embedding)        (None, 300, 128)          2560000

conv1d (Conv1D)              (None, 298, 128)          49280

global_max_pooling1d (Globa  (None, 128)               0
lMaxPooling1D)

dense (Dense)                (None, 128)               16512

dropout (Dropout)            (None, 128)               0

dense_1 (Dense)              (None, 1)                 129

=================================================================
Total params: 2,625,921
Trainable params: 2,625,921
Non-trainable params: 0
```

Figure 4. Model Summary

Train Models

The model will be configured/compiled first before being trained. The Adam optimizer is used as the optimization algorithm during model training in the model configuration. For each model training session, the loss value will be calculated with the binary_crossentropy loss function, and the model performance evaluation will use the metric accuracy parameter. The compiled model will be trained with several iterations (epochs).

Evaluate Models

Model evaluation is done by testing the model against test_data and test_label. Evaluation is done by calculating the metric parameters of loss, accuracy, precision, f1-score, and recall of the test_set.

- **Loss** refers to how big the model makes the prediction error.

The loss calculation on the model is calculated using the binary cross-entropy function. Binary cross-entropy measures the error between model predictions and target values in the form of probabilities. The smaller the loss value, the better the model performance..The binary cross-entropy function has the following mathematical form: Eq (1).

$$H_p(q) = -\frac{1}{N}\sum_{i-1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot log(1 - p(y_i)) \quad (1)$$

y_i : the actual label value

$\log\big(p(y_i)\big)$: probability prediction

$p(y_i)$: the probability value is 1

$1 - p(y_i)$: probability value is 0

- **Accuracy** is interpreted as how much data is correct from the total predictions made by the model[23].

Accuracy is calculated by dividing the number of correct predictions (True Positive and True Negative) by the total number of samples as presented in Eq (2).

$$accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (2)$$

True Positive (TP) is the number of correct predictions for positive data, True Negative (TN) is the number of correct predictions for negative data, False Positive (FP) is the amount of data that is predicted to be positive but is negative, and False Negative (FN) is the amount of data predicted negative but positive.

- **Precision** is expressed as how accurately the prediction is true of the model's positive data [24].

Precision is obtained from the division between the number of true positive predictions (True Positive) and the total number of positive predictions (True Positive and False Positive) as presented in Eq (3).

$$precision = \frac{(TP)}{(TP + FP)} \quad (3)$$

The range of precision values is 0 to 1. High precision indicates that the positive prediction made by the model is correct.

- **The recall** states how the model can correctly find or classify positive data from all actual positive data[24].

Recall measures the model's ability to avoid false negatives. A recall is stated in Eq (4).

$$recall = \frac{(TP)}{(TP + FN)} \quad (4)$$

The recall value ranges from 0 to 1. A high recall value indicates that the model can detect all true positive data.

- **F1-score** strikes a balance between precision and recall[25].

The calculation is done by taking the average value of the two parameters. F1-score is expressed in the following Eq (5).

$$f1\ score = 2 * \frac{(precision * recall)}{(precision + recall)} \quad (5)$$

An F1-score results in a value between 0 and 1. A value of 1 indicates a perfect F1-score (perfect precision and recall), and 0 indicates a poor F1-score (low one or both metrics).

RESULTS AND DISCUSSION

Results and Discussion should be an objective description of the results and should be in relation to the purposes of research. The discussion also needs to be supported by the reference list [14][15]. Results can be presented in figures, tables, and others that make the readers understand easily.

The model is built in Python on the Google Colab platform using the Tensorflow hardware library as the basis for building the model architecture as presented in Table 2

Table 2. Model training results

Epoch	time	accuracy	loss	val_accuracy	val_loss	val_precision	val_recall	val_f1
1	52s	0.9921	0.0562	0.9993	0.0039	1.0000	0.9986	0.9993
2	49s	0.9994	0.0029	0.9995	0.0031	1.0000	0.9991	0.9995
3	48s	0.9997	0.0000 19	0.9995	0.0029	1.0000	0.9991	0.9995
4	43s	0.9998	0.0000 067	0.9995	0.0027	1.0000	0.9991	0.9995
5	42s	0.9998	0.0000 031	0.9995	0.0029	1.0000	0.9991	0.9995

It can be seen that with each increase in epoch, the model's accuracy in making predictions also increases. This indicates that the model performs machine learning to recognize patterns and correlations in the data.

Evaluate Model (Test result).

Evaluation of the Text CNN model that was built was carried out by testing the model on the test_set data. Testing is only done once to see how the model's reliability in classifying data differs from the data being trained. The results of the model test as an evaluation step are detailed in Table 3.

Table 3. Model testing results

Proposed model	Time(s)	total steps	Speed (ms/steps)	test accuracy	loss test
Text CNN	3	129	22ms/step	0.99951	0.002917

Based on Table 3, the model is quite reliable in carrying out classification tasks, as evidenced by achieving a model accuracy rate of 99.95%. The loss rate of the built model also shows a relatively low figure of 0.29%. In the classification task, the model only takes 3 seconds to classify test_set data with 4105 data.

The performance of the proposed Text CNN-based model is then compared with the methods that have been used before to find out how well the model performs when compared with other methods in the same task. Comparisons were made to the accuracy values obtained by the model in data classification. A comparison is presented in Table 4.

Table 4. Comparison of the model with the previous method

	approach model	accuracy (%)
[26]	AdaBoost classifier	99.92
[10]	Generic algorithm + reinforcement learning	99.89
[27]	Multilayer perceptrons	99.32
[28]	LSTM-attention	98.20
[29]	Bidirectional RNNs	96.00
[30]	DNN	99.60
proposed method	Text CNN	**99.95**

The proposed model can outperform the models used in previous studies regarding model accuracy in carrying out classification tasks with the same type of data.

Based on the results obtained, the proposed CNN-based model for text data classification can be used to calculate cross-site scripting attacks with the input analysis method. The proposed model can improve the accuracy of the XSS attack classification task compared to the previous methods [31]. The built CNN Text architecture is optimal, effective, and accurate in carrying out its duties. The model's optimality is determined by

the composition of the filter layers that build the model. Models with more filters will do their job more deeply but also take a long time.

CONCLUSION

This research aims to categorize cross-site scripting attacks using suggested input analysis techniques. This study uses a Convolutional Neural Network (CNN) for text classification, which has proven effective in processing text data and making accurate predictions. The results showed that the model built achieved an accuracy of 99.95%, a significant increase compared to previous studies. These results indicate that this model is the best in determining XSS attacks. These discoveries will be critical to advances in information systems security. The best model that correctly identifies XSS attacks protects the system from harmful attacks. This research contributes to significant advances in detecting and preventing XSS attacks.

Even though this study achieved outstanding results, there is still room for improvement. To improve the accuracy and reliability of the built models, one can learn to build models with various architectures or use a combination of other techniques. Additionally, additional research could be conducted to test this model on a larger scale and in various system environments.

REFERENCES

[1] Z. Liu, Y. Fang, C. Huang, and Y. Xu, "MFXSS: An effective XSS vulnerability detection method in JavaScript based on multi-feature model,"*Comput Security*, vol. 124, Jan. 2023, doi: 10.1016/j.cose.2022.103015.

[2] F. Caturano, G. Perrone, and S. Pietro Romano, "Discovering reflected cross-site scripting vulnerabilities using a multiobjective reinforcement learning environment, "*Comput Security*, vol. 103, Apr. 2021, doi: 10.1016/j.cose.2021.102204.

[3] L. Chen, C. Tang, J. He, H. Zhao, X. Lan, and T. Li, "XSS adversarial example attacks based on deep reinforcement learning,"*Comput Security*, vol. 120, Sept. 2022, doi: 10.1016/j.cose.2022.102831.

[4] AW Marashdih, ZF Zaaba, and K. Suwais, "Predicting input validation vulnerabilities based on minimal SSA features and machine learning,"*Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, p. 9311–9331, Nov. 2022, doi: 10.1016/j.jksuci.2022.09.010.

[5] GE Rodríguez, JG Torres, P. Flores, and DE Benavides, "Cross-site scripting (XSS) attacks and mitigation: A survey,"*Computer Networks*, vol. 166, Jan. 2020, doi: 10.1016/j.comnet.2019.106960.

[6] Q. Wang*et al.*, "Black-box adversarial attacks on XSS attack detection model," Comput Secur, vol. 113, Feb. 2022, doi: 10.1016/j.cose.2021.102554.

[7] M. Indushree, M. Kaur, M. Raj, R. Shashidhara, and HN Lee, "Cross Channel Scripting and Code Injection Attacks on Web and Cloud-Based Applications: A Comprehensive Review,"*Sensors*, vol. 22, no. 5. MDPI, Mar. 01, 2022. doi: 10.3390/s22051959.

[8] P. Chaudhary, BB Gupta, and AK Singh, "Securing heterogeneous embedded devices against XSS attacks in intelligent IoT systems,"*Comput Security*, vol. 118, Jul. 2022, doi: 10.1016/j.cose.2022.102710.

[9] T. Scholte, W. Robertson, D. Balzarotti, and E. Kirda, "Preventing input validation vulnerabilities in web applications through automated type analysis," in *Proceedings - International Computer Software and Applications Conference*, 2012, pp. 233–243. doi: 10.1109/COMPSAC.2012.34.

[10] I. Tariq, MA Sindhu, RA Abbasi, AS Khattak, O. Maqbool, and GF Siddiqui, "Resolving cross-site scripting attacks through genetic algorithm and reinforcement learning,"*Expert System Appl*, vol. 168, Apr. 2021, doi: 10.1016/j.eswa.2020.114386.

[11] W. Luo, "Research and Implementation of Text Topic Classification Based on Text CNN," in*2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, 2022, pp. 1152–1155. doi: 10.1109/CVIDLICCEA56201.2022.9824532.

[12] AY Muad*et al.*, "An effective approach for Arabic document classification using machine learning," Global Transitions Proceedings, vol. 3, no. 1, pp. 267–271, 2022, doi: https://doi.org/10.1016/j.gltp.2022.03.003.

[13] MP Akhter, Z. Jiangbin, IR Naqvi, M. Abdelmajeed, A. Mehmood, and MT Sadiq, "Document-Level Text Classification Using Single-Layer Multisize Filters Convolutional Neural Network,"*IEEE Access*, vol. 8, pp. 42689–42707, 2020, doi: 10.1109/ACCESS.2020.2976744.

[14] EF Ayetiran, "Attention-based aspect sentiment classification using enhanced learning through cnn-Bilstm networks,"*Knowl Based System*, vol. 252, p. 109409, 2022, doi: https://doi.org/10.1016/j.knosys.2022.109409.

[15] SSH SHAH, "Cross-site scripting XSS dataset for Deep learning,"*www.kaggle.com*, 2020.

[16] Z. Alyafeai, MS Al-shaibani, M. Ghaleb, and I. Ahmad, "Evaluating Various Tokenizers for Arabic Text Classification,"*Neural Process Lett*, 2022, doi: 10.1007/s11063-022-10990-8.

[17] F. Alrasheedi, X. Zhong, and P.-C. Huang, "Padding Module: Learning the Padding in Deep Neural Networks,"*IEEE Access*, vol. 11, p. 7348–7357, 2023, doi: 10.1109/ACCESS.2023.3238315.

[18] J. Wen, X. Zhou, P. Zhong, and Y. Xue, "Convolutional neural network based text steganalysis,"*IEEE Signal Process Lett*, vol. 26, no. 3, pp. 460–464, 2019, doi: 10.1109/LSP.2019.2895286.

[19] M. Dong, Y. Li, X. Tang, J. Xu, S. Bi, and Y. Cai, "Variable Convolution and Pooling Convolutional Neural Network for Text Sentiment Classification,"*IEEE Access*, vol. 8, pp. 16174–16186, 2020, doi: 10.1109/aCCESS.2020.2966726.

[20] B. Guo, C. Zhang, J. Liu, and X. Ma, "Improving text classification with weighted word embeddings via a multi-channel TextCNN model,"*Neurocomputing*, vol. 363, pp. 366–374, 2019, doi: https://doi.org/10.1016/j.neucom.2019.07.052.

[21] Y. Luan and S. Lin, "Research on Text Classification Based on CNN and LSTM," 2019.

[22] R. Wang, Z. Li, J. Cao, T. Chen, and L. Wang,*Convolutional Recurrent Neural Networks for Text Classification*. Hungary: IEEE, 2019. [Online]. Available: http://www.ieee.org/publications

[23] Y. Zhou, J. Li, J. Chi, W. Tang, and Y. Zheng, "Set-CNN: A text convolutional neural network based on semantic extension for short text classification,"*Knowl Based System*, vol. 257, Dec. 2022, doi: 10.1016/j.knosys.2022.109948.

[24] H. Dalianis, "Evaluation Metrics and Evaluation," in *Clinical Text Mining*, Springer International Publishing, 2018, pp. 45–53. doi: 10.1007/978-3-319-78503-5_6.

[25] N. and SS Sokolova Marina and Japkowicz, "Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation," in *2006: Advances in Artificial Intelligence*, B. Sattar Abdul and Kang, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1015–1021.

[26] P. Roy, R. Kumar, P. Rani, and TS Joy, "XSS: Cross-site Scripting Attack Detection by Machine Learning Classifiers," in*2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, 2022, pp. 1535–1539. doi: 10.1109/SMART55829.2022.10046960.

[27] FMM Mokbal, W. Dan, A. Imran, L. Jiuchuan, F. Akhtar, and W. Xiaoxi, "MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique,"*IEEE Access*, vol. 7, pp. 100567–100580, 2019, doi: 10.1109/ACCESS.2019.2927417.

[28] L. Lei, M. Chen, C. He, and D. Li, "XSS Detection Technology Based on LSTM-Attention," in*2020 5th International Conference on Control, Robotics and Cybernetics (CRC)*, 2020, pp. 175–180. doi: 10.1109/CRC51253.2020.9253484.

[29] Y. and XY Liu Zhonglin and Fang, "Cross-site Scripting Threat Intelligence Detection Based on Deep Learning," *Frontiers in Cyber Security*, F. Ahene Emmanuel and Li, Ed., Singapore: Springer Nature Singapore, 2022, pp. 89–104.

[30] M. Krishnan, Y. Lim, S. Perumal, and G. Palanisamy, "Detection and defending the XSS attack using a novel hybrid stacking ensemble learning-based DNN approach,"*Digital Communications and Networks*, 2022, doi: https://doi.org/10.1016/j.dcan.2022.09.024.

[31] Walid and Alamsyah, "Recurrent Neural Network For Forecasting Time Series With Long Memory Pattern", 2017 J. Phys.: Conf. Ser. 824 012038, doi: https://doi.org/10.1088/1742-6596/824/1/012038.