AN EFFICIENT LSTM MODEL FOR FAKE NEWS DETECTION

Jayesh Soni

Knight School of Computing and Information Sciences Florida International University, Miami, FL, USA

ABSTRACT

Information spread through online social media or sites has increased drastically with the swift growth of the Internet. Unverified or fake news reaches numerous users without concern about the trustworthiness of the info. Such fake news is created for political or commercial interests to mislead the users. In current society, the spread of misinformation is a big challenge. Hence, we propose a deep learning-based Long Short Term Memory (LSTM) classifier for fake news classification. Textual content is the primary unit in the fake news scenario. Therefore, natural language processing-based feature extraction is used to generate language-driven features. Experimental results show that NLP-based featured extraction with LSTM model achieves a higher accuracy rate in discernible less time.

KEYWORDS

Fake News, Long Short Term Memory, Natural Language Processing, Word Embedding, Tensorflow

1. INTRODUCTION

The growth of social media has increased in the modern world. It has developed as an influential foundation of info and getting individuals together. For modern updates, trending articles, and other info, many people depend on social media news. This validates the deficiency with traditional news boards in capability. Discrimination of fake news from a real one is a challenging issue. People's sentiments, stock exchanges, and the news ecosystem are affected by the spread of misinformation on social media. The trustworthiness of such data is unknown, which hurts the public. There are several difficulties in detecting fake news. First, Fake news data collection with the proper labeling is challenging. Next, it is tough to identify them on the source of news content [1] since they are written purposely to deceive readers. As most news is bounded by time, it is challenging to clarify the validation. There are numerous techniques to solve the misinformation spread on social media. Statistical methods are employed to detect the features' correlation and further analyze the patterns. Machine learning-based algorithms are widely adopted to discriminate real with fake content. The rest of this paper is structured as follows: Section 2 discusses the literature review. Section 3 provides information regarding the dataset. Section 4 presents the proposed deep learning framework for fake news detection. Experimental results are discussed in Section 5. We conclude in section 6 with future work discussion.

2. RELATED WORK

Detection of fake news has developed an evolving topic in the exploration arena to overcome the spread of misinformation formed purposefully. Various features like text or images in news content on social media data can be used to detect such false news [2, 3]. This section discusses some of the current and significant work in this direction. Jain et al. employed whale optimized

algorithm using social network data. Gravanis et al. [4] presented the UnB Dataset and developed a learning-based detection of fake news. Their investigation study suggests that learning-based models provide great accuracy, given a proper linguistic feature set. Many scholars explored this arena based on the text [5]. Zhou et al. used semantic structures and projected a model for initial detection. The bag of words approach is used by Zhou et al. [6] for fake news detection. Bauskar et al. developed a hybrid NLP model by exploring features based on the content and social network. Alkhodair et al. detect the rumor on breaking news by creating a deep learning-based model. According to their study, a post without the verification of its trustworthiness is defined as a rumor which can be true or false. Liu et al. [7] detect spammers using the French satiric dataset. They put forward a novel work by considering the spread of messages in online media to classify them. Various scholars use the graph dataset rather than text to solve the issue [8, 9]. Zhang et al. [10, 11] use network-based content creators and article's text. Feng et al. [12, 13] aim at calculating the readability metrics for targeting the catchy headlines towards the detection of fake news. Most of the proposed standard machine learning algorithms in the literature work well, yet the results can be improved. Therefore, in this study, we proposed a deep learning-based algorithm. Our research indicates that using deep learning algorithms is more significant than the standard machine learning algorithms.

3. FAKE NEWS DATASET

Fake news dataset is an open-source dataset available on Kaggle. It consists of the following attributes:

1) Title: It contains the headlines of the news.
2) Text: It contains the actual content of articles.
3) Subject: It depicts the type of news.
4) Date: Published date of the news.

There are 23,481 articles labeled as fake and 21,417 as real. News, Politics, Government News, left-news, US_News, Middle East, and world news are the types of subjects. Figure 1 shows the total count for each type of news.

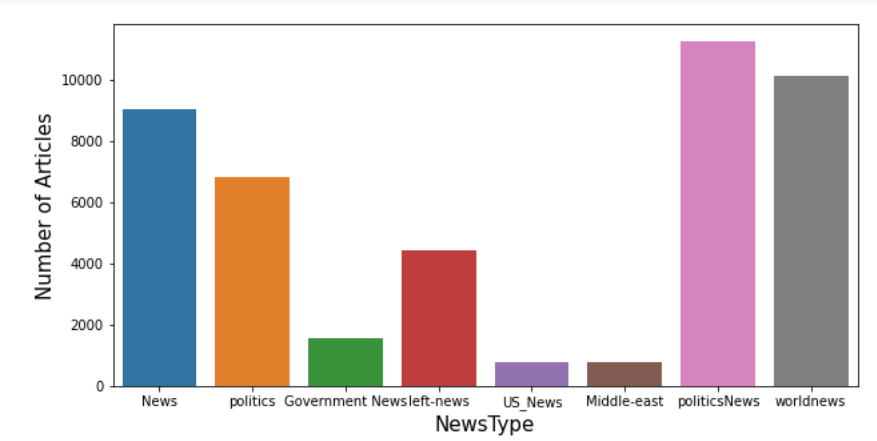

Figure 1. Number of Articles per NewsType

From Figure1, we can depict that this dataset mainly contains news about politics and world news. There are some repeated class names that have identical meanings. The total length for most of the news is in the range of 5000 words with a few near to 10000 words.

4. PROPOSED FRAMEWORK

The proposed framework is described in Figure 2. It comprises the following stages: Dataset Collection, Data Pre-Processing, and Algorithm Training with final test evaluation. Below we provide details for each of the stages in brief.

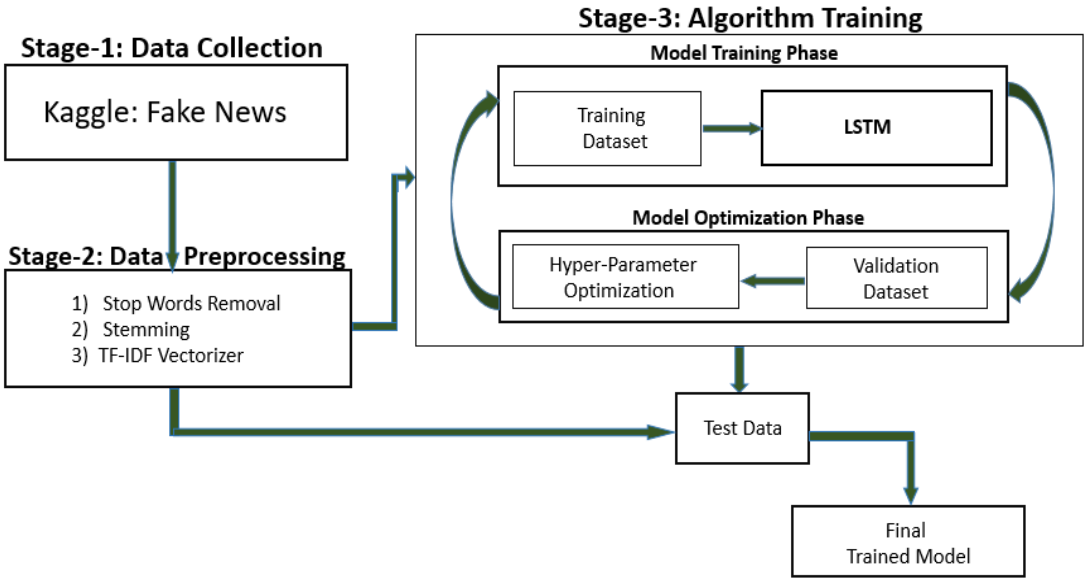

Figure 2. Proposed Framework

4.1. Data Collection

We use the Fake News dataset collected from [14, 15] as described in the previous section.

4.2. Data Preprocessing

In this stage, we perform the following data pre-processing task:

4.2.1. Stop Words Removal

Stop words are the words that are generally riddled out before processing the textual data. Prepositions, Conjunctions, Pronouns, adjectives, and so on are frequent words that do not add any valuable information for further processing. Some of the examples are "it", "the", “an”, “from”.

In any human language, such words are present in large quantities. Low-level contextual info is eradicated from the natural language text by eliminating such stop words. Henceforth, the priority is given to the most important words. There is no adverse effect on the training of the model on removing such words. Furthermore, it helps reduce the training time of the model with the reduced dataset containing less number of tokens (individual words). We use the NLTK library to perform this task [16].

4.2.2. Stemming

Stemming is the process of suffix removal and reducing the word to its root word. For instance: In the word “eating”, the suffix "ing" is removed first, and then it is transformed to its root word, which is "Eat". We use PorterStemmer to stem a word to its root words [17].

4.2.3. Term Frequency – Inverse Document Frequency (TF-IDF) Vectorizer

Transformation of textual data into a numeric vector is performed using the term frequency inverse document frequency. It is vectorizing the text. It consists of two modules.

The total count of an individual word in the document is term frequency. It indicates how significant a particular word is in the document. It considers the textual data a matrix where the total number of documents are represented as rows, and the total number of distinct words are represented as columns.

The total amount of documents comprehending a particular term is document frequency. It designates how common the term is.

Inverse document frequency gives a weightage to each term. The weight of a term is reduced if it is distributed throughout the documents. It can be calculated with the below equation:

$$\mathrm{IDF_j} = log\frac{n}{DF_i} \qquad (1)$$

Where n is the total count of documents, $\mathrm{IDF_j}$ is the IDF score for term j, and the total number of documents comprehending term j is $\mathrm{DF_j}$. IDF is inversely proportional to DF. Therefore, the greater the value of DF, the lower the value of IDF. When a particular term appears in every document, the IDF reaches zero since DF equals n. Thus, that individual term should be removed since it will not provide valuable information.

The TF-IDF score is formulated by multiplying term frequency with its inverse document frequency as shown below:

$$\mathrm{W_{j,k}} = TF_{j,k} * IDF_j \qquad (2)$$

Where $\mathrm{IDF_j}$ is IDF score for term j, term frequency for term j in document k is $\mathrm{TF_{j,k}}$, and $\mathrm{W_{j,k}}$ is TF-IDF score for term j in document k.

Scikit-learn Tf-IDF vectorizer is used to perform this operation.

4.3. Algorithm Training

This is the primary stage of the entire framework where we train the detection algorithm in two sub-phases as below:

4.3.1. Model Training Phase

Here we use the training dataset to train the deep learning-based Long short term memory algorithm [18].

LSTM is one of the kinds of recurrent neural networks (RNN). The vanishing gradient is one of the major problems with the RNN network. So, various gates are attached to the LSTM network to overcome the inaccuracies of RNN. Additionally, these LSTM has memory cells attached for retaining long-term dependencies [19, 20]. Figure 3 shows the LSTM cell. These gates act as a memory. The LSTM cell updates the memory on reading the input. The following gates are added to LSTM cells.

Forget gate: It panels the size of info to be detached.

Memory gate: It is used to generate a different memory.

Input gate: It regulates the total amount of latest memory info that needs to be updated. Output gate: The output gate updates the hidden state by extracting memory cell info.

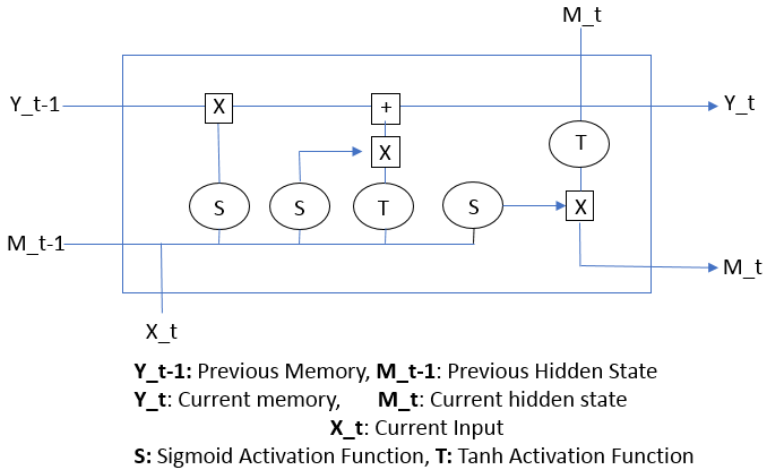

Figure 3. LSTM Cell

Given a sample dataset in the form of G = (g_1, g_2, g_3, g_{n-1}, g_n) as input to network, it learns the output variable by updating the values of the above four gates in hidden layer. It is updated as follows:

$$(g_t, b_{t-1}, s_{j-1}) \rightarrow (b_t, s_t) \quad (3)$$

$$i_x = \sigma\,(m_{ai} g_t + m_{di} b_{x-1} + m_{si} z_{x-1} + c_i) \quad (4)$$

$$f_x = \sigma\,(m_{af} g_x + m_{bf} b_{x-1} + m_{zf} z_{x-1} + c_f) \quad (5)$$

$$z_x = f_x * z_{x-1} + i_x * tanh(m_{az} a_x + m_{bz} b_{x-1} + c_z) \quad (6)$$

$$y_x = \sigma\,(m_{ay} g_x + m_{by} b_{x-1} + m_{zy} z_x + c_y) \quad (7)$$

$$b_x = y_t * tanh(z_x) \quad (8)$$

The bias units for the input gate, forget gate, output gate, and memory cell are denoted by c_i, c_f,c_y,c_z, respectively. Furthermore, m represents weight matrix, memory state as z, and hidden layer output as b. We also have tanh and sigmoid activation functions.

We divided the dataset into training and testing with 70% and 30%, respectively, and further developed and trained the LSTM model architecture shown in Figure 4. To get the optimal value for hyper-parameters, the K-fold cross-validation approach is employed. The proposed architecture has an embedding layer followed by an LSTM layer and a Dropout layer. The

dropout layer is utilized to avoid over fitting. LSTM layer has a total of 50 neurons. Finally, we have the dense layer to discriminate fake news from real news.

We used the following metrics to evaluate the model.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (9)$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad (10)$$

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (11)$$

$$\text{False Negative Rate (FNR)} = \frac{FN}{TP+FN} \quad (12)$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP+TN} \quad (13)$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \quad (14)$$

$$\text{F1-Score: } 2 * \frac{Precision*Recall}{Precision+Recall} \quad (15)$$

Where TP: True positive, FN: False Negative, TN: True Negative, and FP: False Positive.

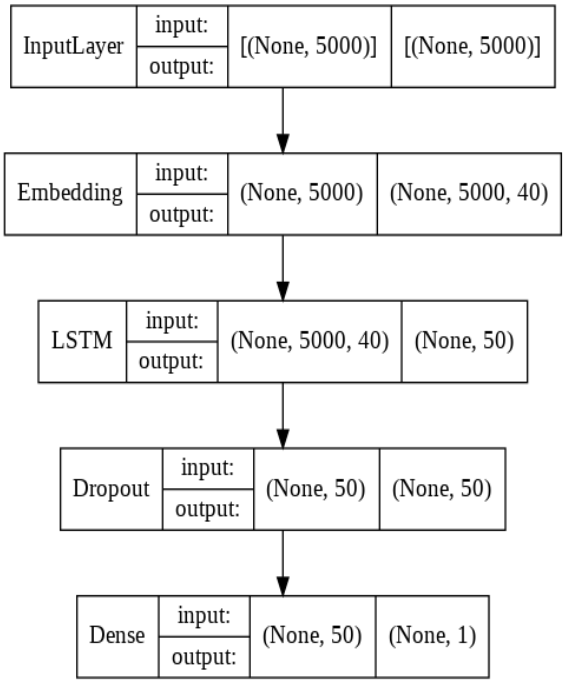

Figure 4. Model Architecture

4.3.2. Model Optimization Phase

To optimize the hyper-parameters of the LSTM model, a validation dataset is used. The following hyper-parameters are tuned:

-Epochs: Number of times a model is trained on the entire dataset.
-Batch-size: It is the total number of data points used by the model to update the weights.

Once we train our LSTM model with fine-tune optimization, we evaluate it with our test data and generate the final testing accuracy. This final trained model can be used for actual fake news detection.

5. EXPERIMENTAL RESULTS

The proposed LSTM model is trained with ReLu (Rectifier Linear Unit) as an activation function in hidden layers and SoftMax activation function in the dense output layer. Furthermore, to get the optimal weights for gradient calculation, an Adam optimizer is employed to update the gradients. Binary Cross Entropy metric evaluates the model. BatchSize is a hyper-parameter that needs to be tuned while training the deep neural network. Batch gradient descent, stochastic batch gradient descent, and mini-batch gradient descent are the three different variants for setting the batch size values. In batch gradient descent, we update the weights at the end of the training data. Next, in stochastic gradient descent, we update the weights at every data point. Lastly, in minibatch gradient descent, we update the weights at the end of every batch of data.

We used the mini-batch gradient descent algorithm and trained the LSTM network with batch sizes of {16, 32, 64, 128, and 256}. The accuracy value for each batch size is shown in Figure 5. We found out that batch size 128 gives the highest accuracy. The main advantage of this method is higher detection rate but on the other hand such learning based model requires huge volume of data with high computational power for analysis.

Our proposed approach was able to achieve the optimal accuracy rate with the minimum amount of training time. Thus it saves the computational power. The accuracy value at each epoch is shown in Figure 6, whereas Figure 7 shows the loss at each Epoch. Our trained model gives a 99% accuracy rate with a loss of 0.01%.

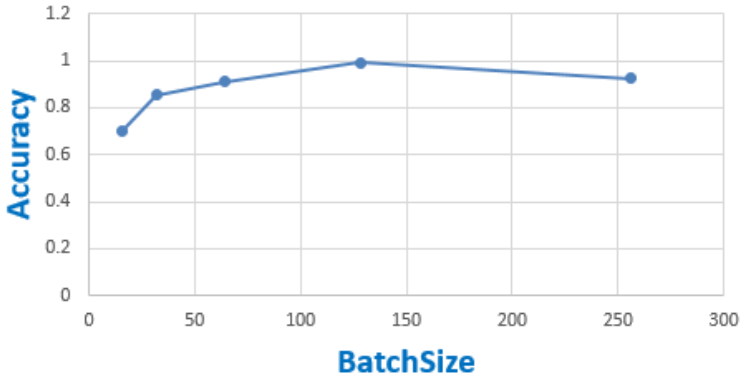

Figure 5. Accuracy with BatchSize

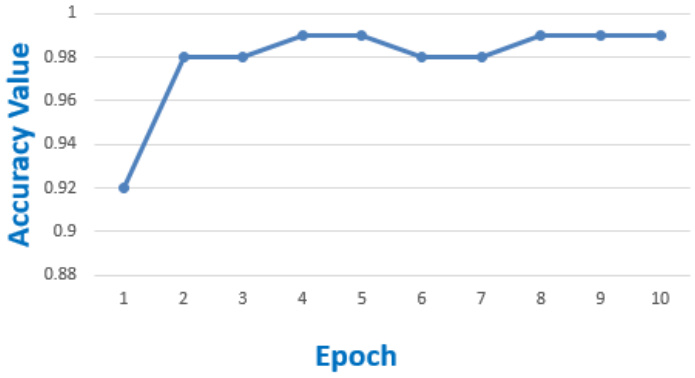

Figure 6. Accuracy w.r.t Epoch

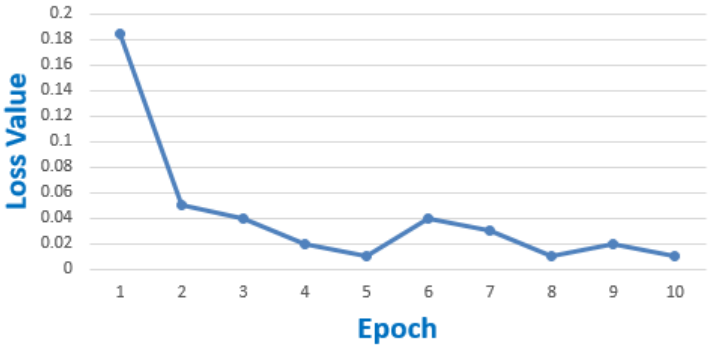

Figure 7. Loss w.r.t Epoch

Table 1. Evaluation Metrics

Recall	Specificity	Precision	FPR	FNR	Accuracy
0.986	0.986	0.959	0.002	0.103	0.991

Table 1 shows the achieved metrics. Google Colaboratory with GPU support trains the LSTM model with Keras [21] and TensorFlow [22] at the backend. Scikit Learn seaborn is used for visualization purposes [23, 24]. For each Epoch, it took 192 sec on average to train the model on GPU.

6. CONCLUSIONS

This paper applies the RNN based LSTM model to learn the semantics of news articles to discriminate fake articles from real articles. The model uses the fake news detection dataset. We transformed the textual data into an embedded vector set to feed them to the LSTM network. We employed a layered LSTM model with Adam optimizer to enhance the performance of the proposed model. Our trained model achieved 99% accuracy in distinguishing fake news from real news.

We can employ an ensemble approach by incorporating multiple algorithms to extend this work. Also, we can utilize the cryptography data for enhancement [25]. Furthermore, we can use Convolution Neural Network (CNN) to reduce feature dimensionality and then feed them to the LSTM layer to develop a robust model.

REFERENCES

[1] William Kai Shu, Amy Sliva , Suhang Wang , Jiliang Tang and Huan Liu. (2017). "Fake News Detection on Social Media: A Data Mining Perspective", SIGKDD Explorations: 19(1).

[2] Alkhodair, S. A., Ding, S. H., Fung, B. C., & Liu, J. (2020). Detecting breaking news rumors of emerging topics in social media. Information Processing & Management, 57(2), 102018.

[3] Bauskar, S., Badole, V., Jain, P., & Chawla, M. (2019). Natural language processing based hybrid model for detecting fake news using content-based features and social features. International Journal of Information Engineering and Electronic Business, 11(4), 1-10.

[4] Gravanis, G., Vakali, A., Diamantaras, K., & Karadais, P. (2019). Behind the cues: A benchmarking study for fake news detection. Expert Systems with Applications, 128, 201-213.

[5] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD explorations newsletter, 19(1), 22-36.

[6] Zhou, X., Jain, A., Phoha, V. V., & Zafarani, R. (2020). Fake news early detection: A theorydriven model. Digital Threats: Research and Practice, 1(2), 1-25.

[7] Wu, L., & Liu, H. (2018, February). Tracing fake-news footprints: Characterizing social media messages by how they propagate. In Proceedings of the eleventh ACM international conference on Web Search and Data Mining (pp. 637-645).

[8] Soni, J., Prabakar, N., & Upadhyay, H. (2019). Feature extraction through deepwalk on weighted graph. In Proceedings of the 15th international conference on data science (ICDATA'19), Las Vegas, NV.

[9] Soni, J., & Prabakar, N. (2018). Effective machine learning approach to detect groups of fake reviewers. In Proceedings of the 14th international conference on data science (ICDATA'18), Las Vegas, NV (pp. 3-9).

[10] Zhang, X., & Ghorbani, A. A. (2020). An overview of online fake news: Characterization, detection, and discussion. Information Processing & Management, 57(2), 102025.

[11] J. Zhang, B. Dong and P. S. Yu, "FakeDetector: Effective Fake News Detection with Deep Diffusive Neural Network," 2020 IEEE 36th International Conference on Data Engineering (ICDE), 2020, pp. 1826-1829, doi: 10.1109/ICDE48307.2020.00180.

[12] Feng, L., Jansche, M., Huenerfauth, M., & Elhadad, N. (2010). A comparison of features for automatic readability assessment.

[13] Flesch, R. F. (1951). How to test readability.

[14] Ahmed, H., Traore, I., & Saad, S. (2018). Detecting opinion spams and fake news using text classification. Security and Privacy, 1(1), e9.

[15] Ahmed, H., Traore, I., & Saad, S. (2017, October). Detection of online fake news using n-gram analysis and machine learning techniques. In International conference on intelligent, secure, and dependable systems in distributed and cloud environments (pp. 127-138). Springer, Cham.

[16] Loper, E., & Bird, S. (2002). Nltk: The natural language toolkit. arXiv preprint cs/0205028.

[17] Hardeniya, N., Perkins, J., Chopra, D., Joshi, N., & Mathur, I. (2016). Natural language processing: python and NLTK. Packt Publishing Ltd.

[18] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems, 28(10), 22222232.

[19] Soni, J., Prabakar, N., & Upadhyay, H. (2019). Comparative Analysis of LSTM SequenceSequence and Auto Encoder for real-time anomaly detection using system call sequences.

[20] Soni, J., Prabakar, N., & Upadhyay, H. (2019, December). Behavioral Analysis of System Call Sequences Using LSTM Seq-Seq, Cosine Similarity and Jaccard Similarity for Real-Time Anomaly Detection. In 2019 International Conference on Computational Science and Computational Intelligence (CSCI) (pp. 214-219). IEEE.

[21] Ketkar, N. (2017). Introduction to keras. In Deep learning with Python (pp. 97-111). Apress, Berkeley, CA.

[22] Dillon, J. V., Langmore, I., Tran, D., Brevdo, E., Vasudevan, S., Moore, D., ... & Saurous, R. A. (2017). Tensorflow distributions. arXiv preprint arXiv:1711.10604.

[23] Soni, J., Prabakar, N., & Upadhyay, H. (2020). Visualizing High-Dimensional Data Using tDistributed Stochastic Neighbor Embedding Algorithm. In Principles of Data Science (pp. 189206). Springer, Cham.

[24] Stančin, I., & Jović, A. (2019, May). An overview and comparison of free Python libraries for data mining and big data analysis. In 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 977-982). IEEE.

[25] Shaik, Cheman, Preventing Counterfeit Products Using Cryptography, QR Code and Webservice (February 18, 2021). Computer Science & Engineering: An International Journal (CSEIJ), Vol 11, No 1, February 2021, Available at SSRN: https://ssrn.com/abstract=3787844

AUTHOR

Jayesh Soni is a Ph.D. candidate in the Knight School of Computing and Information Sciences department at Florida International University, Miami. The primary focus of his research is to detect anomalies at the system level by leveraging Artificial Intelligence techniques. His research interests include Cyber Security, Big Data, and Parallel Processing.