A Scalable Hate Speech Detection System for Vietnamese Social Media using Real-time Big Data Processing and Distributed Deep Learning

Van-Co Dinh[*,†], Tran-Dai Vo[*,†], Mai-Phuong T. Nguyen[*,†], Trong-Hop Do[*,†]

[*] University of Information Technology, Ho Chi Minh City, Vietnam.
[†] Vietnam National University, Ho Chi Minh City, Vietnam.

***Abstract*—In this study, a system to detect hate and offensive social network comments in real-time using big data and distributed deep learning technology is presented. In the offline phase, state-of-the-art deep learning models are trained in a distributed manner using to BigDL library. The trained models are then integrated into the real-time big data processing component powered by Apache Spark, which a big data framework capable of processing a huge amount of comments in real-time. In the online phase, continuous stream of comments from Facebook are crawled and channeled through Kafka to this real-time big data processing component to output hate speech detection results. These results are then are then analyzed, and the statistical data is displayed in a web-app powered by Flask. Therefore, this work not only focuses on the accuracy but also emphasizes the system's practicality. Thanks to state-of-the-art deep learning models, the system can achieve high accuracy in the hate speech detection. With the deployed big data technology, the system can collect and process huge amounts of Facebook comments and produce statistical results in real-time.**

***Index Terms*—Hate speech detection, Distributed Deep Learning, Social network, Big data, Real-time.**

I. Introduction

The social network is a common space where people can share their emotions and express their opinions. With its benefits, the number of people using social networking sites is increasing daily and continuously. Because of that, the above data is growing, and controlling all the content, especially user comments, is not easy. One of the problems we can see is that users' comments bring harmful, offensive content to an individual, organization, or a specific region. Therefore, to ensure a healthy environment on social networks, it is necessary to have a system to detect toxic content in social networks automatically. For the system to be practical, it needs to be capable of processing large amounts of comments data and producing accurate results in real-time. Data is generated by users on social networking sites continuously. The amount of this streaming data generated is massive that conventional data processing systems cannot handle efficiently and timely. In recent years, various frameworks have been introduced for big data processing, of which Apache Spark is one of the most prominent. Using Spark, one can build a machine learning-based system that can process a large amount of streaming data from social networks and detect toxic content. However, using a big data framework like Spark is not the ultimate solution to social media data's hate speech detection problem as machine learning models currently integrated into Spark usually provide modest performances. In recent years, state-of-the-art deep learning models have performed outstanding tasks, including hate speech detection. However, training these models with a large amount of data and integrating the trained models into big data frameworks are very challenging. This study proposed a real-time hate speech detection system using big data processing and distributed deep learning for Vietnamese social media data. In the offline phase, several models using traditional machine learning and state-of-the-art deep learning approaches are trained in distributed manners using the BigDL library to detect Vietnamese hate speech contents. Thanks to distributed training, these models can fully utilize a large amount of training data. The trained models can be integrated into Spark Structured Streaming inside Spark to build a system that can process a large amount of streaming data of comments from social networks and produce accurate results in real-time to ensure the practicality of the system. In the online phase, a data collecting component to continuously crawl Vietnamese comments from Facebook and channel them via Kafka to trained models integrated into Spark structured streaming to output real-time result of hate speech detection. These results are then input to another analytical component to produce real-time statistical analysis which is displayed in web-app dashboards.

II. Related Works

The development of social networks in recent years makes the amount of data increase swiftly. Hence, this data is more and more difficult to control and analyze instantaneously. Recent works have shown user emotions from tweets on Twitter using machine learning and deep learning models. To be able to process a large amount of data in a short period of time and continuously, Hossam

Elzayady et al. [1] used Apache Spark framework as a tool to analyze these tweets.

For the Vietnamese, the classification of short texts using machine learning models [2] has also used Apache Spark as a tool to distribute the calculation process. Luu et al. [3] proposed a large dataset (Vi-HSD) in the problem of detecting hate and offensive user comments. In this work, the author used experimental deep learning architectures on their dataset as text classification problem. However, to apply it to the analysis of user comments from social networking sites is not an easy thing, especially with a large number of data and continuously. Therefore, in this work, we will use BigDL framework as a tool to distribute training process, along with Apache Spark and Kafka framework to analyze user comments that are generated directly from social networking sites. Thanks to these libraries, we can promptly train models on large amounts of data and use them in real-time analysis of the content of comments.

III. Proposed real-time big data processing and social listening problem

A. Real-time big data processing

1) Apache Kafka [4]: is a distributed data store optimized for ingesting and processing streaming data in real-time. Kafka combines two messaging models, queuing and publish-subscribe, to provide the key benefits of each to consumers. Queuing allows for data processing to be distributed across many consumer instances, making it highly scalable. In this study, Kafka is used to build the streaming data ingestion module which receives the continuous data from social network and send it to a streaming data processing component to output the result in real time.

2) Apache Spark [5]: is one of the most widely used open source frameworks when dealing with and working with BigData. Not only Spark can handle a huge volume of data but also process and output the result in real-time. In practice, the large amount of data constantly pouring from social networking sites like Facebook, Twitter, and Youtube needs to be processed immediately. Social listening models which can be built to be integrated into a component called Spark Structured Streaming inside Spark to process streaming data from social networks and output the results in real-time.

B. Distributed deep learning

In the distributed training problem, there are two common approaches: model parallelism and data parallelism. BigDL approaches data parallelism that means no distribution of the model across different workers [6]. This is not a limitation in practice. Because it is a distributed deep learning framework run on the Apache Spark/Hadoop cluster. This allows it directly processes the production data, and as a part of the end-to-end data analysis pipeline for deployment and management. BigDL implements a data parallel mechanism. Through BigDL, deep learning models can be trained synchronously with training data distributed across the cluster. In each iteration, multiple Spark jobs are executed to compute the gradients of the models using a partition of training data as shown in Fig. **??**. A Spark cluster consists of a single driver node and multiple worker nodes in Algorithm 1. The driver is responsible for coordinating tasks in a Spark job, while the workers are responsible for the actual computation. These gradients are then updated through a parameter server to create the new version of model parameters.

Algorithm 1 Data-parallel training in BigDL

```
1: for $i = 1$ to $M$ do
2:     //"model forward-backward" job
3:     for each task in the Spark job do
4:         read the latest weights;
5:         get a random batch of data from local Sample partition;
6:         compute local gradients (forward-backward on local model replica);
7:     end for
8:     //"parameter synchronization" job
9:     aggregate (sum) all the gradients;
10:    update the weights per specified optimization method;
11: end for
```

Algorithm 2 "Parameter synchronization" job

```
1: for each task $n$ in the "parameter synchronization" job do
2:     shuffle the $n^{th}$ partition of all gradients to this task;
3:     aggregate (sum) these gradients;
4:     updates the $n^{th}$ partition of the weights;
5:     broadcast the $n^{th}$ partition of the updated weights;
6: end for
```

To support efficient parameter synchronization, BigDL has taken a variety approach that directly implements an efficient AllReduce to mimic the functionality of a parameter server architecture as shown in Algorithm 2.

C. Feature extraction

1) Statistical based: There are many statistical-based methods to represent text features. In particular, TF-IDF [7] is a commonly used method, this technique will statistics and evaluate the relevance of a word to a document in a collection of documents. The TF-IDF value of a word in the document, depending on two factors: term frequency and important words.

$$TF(w,d) = \frac{n_d(w)}{n_d} \tag{1}$$

$$IDF(w) = \log_e \frac{n_D}{n_D(w)} \tag{2}$$

$$TF-IDF(w,d,D) = TF(w,d) \times IDF(w) \tag{3}$$

2) Deep learning based: The word embedding technique has been widely used in text representation. This technique will learn to represent relationships between words in the text, pre-trained models are usually trained on a large dataset. As a result, it can well describe the relationship, the semantic similarity of words and the context of text. For text classification in this problem, we use fastText word embedding to represent text helps deep learning models learn to predict data labels better.

D. Hate speech detection models

1) Machine learning models: Multinomial Naive Bayes [8] is a supervised learning method that uses probability and is focused on text classification cases. This method follows the principle of multinomial distribution in conditional probability.

Logistic Regression [9] is a regression model for the purpose of forecasting discrete output value. The principle operation of logistic rules is given from the beginning of the current module estimator. It passes a function trigger named sigmoid to all output values of the character function in the period [0,1].

Decision Tree [10] is a typical tree-based model, which is constructed by many nodes. Decision tree can be used in both classification and regression problem. It has a root and many leaves and decision tree model bases on the rules in these leaves to give prediction.

2) Deep learning models: Long Short-Term Memory (LSTM) [11] is a specific recurrent neural network (RNN) architecture that was designed to model temporal sequences and their long range dependencies more accurately than conventional RNNs. In this work, LSTM cells are used as the nodes of recurrent neural networks (see Fig. 1):

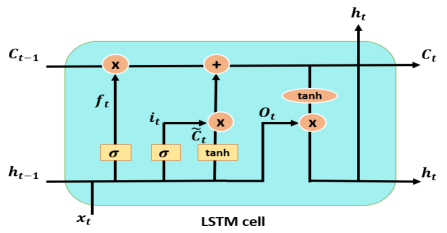

Figure 1: Structure of the LSTM cell.

Output includes c_t, h_t, where c is the cell state, h is the hidden state. And the input includes c_{t-1}, h_{t-1}, x_t, where x_t is the input in the tth state of the model, c_{t-1}, h_{t-1} is the output of the previous layer. h plays the same role as s in RNN, while c_t is new in LSTM.

f_t,i_t,o_t corresponding to forgate gate, input gate and outp gate.

- Forget gate:

$$f_t = \sigma\left(U_f * x_t + W_f * h_{t-1} + b_f\right) \tag{4}$$

- Input gate:

$$f_t = \sigma\left(U_i * x_t + W_i * h_{t-1} + b_i\right) \tag{5}$$

- Output gate:

$$o_t = \sigma\left(U_o * x_t + W_o * h_{t-1} + b_o\right) \tag{6}$$

$\mathbf{0} < f_t,\ i_t,\ o_t < 1,\ b_f,\ b_i,\ b_o$ is the bias coefficients; W is the recurrent connection between the previous hidden layer and current hidden layer. U is the weight matrix that connects the inputs to the hidden layer.

$$\widetilde{c}_t = tanh\left(U_c * x_t + W_c * h_{t-1} + b_c\right) \tag{7}$$

$\widetilde{c}_t$ is a candidate hidden state that is computed based on the current input and the previous hidden state.

$$c_t = f_t * \ c_{t-1} + i_t * \widetilde{c}_t \tag{8}$$

Forget gate decides how much to get from the cell state first and the input gate decides how much to get from state input and the hidden layer of previous.

$$h_t = o_t * \ tanh\left(c_t\right) \tag{9}$$

The output gate decides how much to take from the cell state to be the output of the hidden state.

Gated Recurrent Unit (GRU) [12] is the same as the RNN but the difference is in the operation and gates associated with each GRU unit. To solve the problem faced by standard RNN, GRU incorporates the two gate operating mechanisms called Update gate and Reset gate.

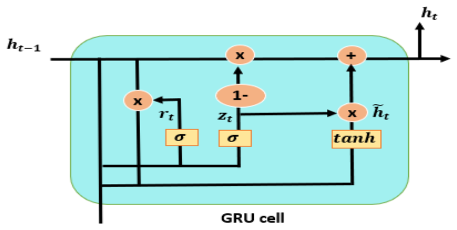

Figure 2: Structure of the GRU cell.

- Reset Gate: The candidate activation h_t is computed similarly to that of the traditional recurrent unit:

$$\widetilde{h}_t = tanh\left(Wx_t + U\left(r_t \odot h_{t-1}\right)\right) \tag{10}$$

  where r_t is a set of reset gates and $\odot$ is an element-wise multiplication.

The reset gate is similar to the forget gate in LSTM:

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{11}$$

- Update Gate: The activation h_t of the GRU at time t is a linear interpolation between the previous activation h_{t-1} and the candidate activation h_t:

$$h_t = (1 - z_t)h_{t-1} + z_t \widetilde{h}_t \tag{12}$$

where an update gate z_t decides how much the unit updates its activation, or content. The update gate is computed by:

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{13}$$

Text-CNN [13] is the model that use convolution to extract features in text to predict their labels. Suppose that we have a sentence of length n represented as follows:

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n \tag{14}$$

where $\oplus$ is the concatenation operator. A convolution operation involves a filter w is applied to a window of h words to produce a new feature. A feature c_i is generated from a window of words $x_{i:i+h1}$ by:

$$c_i = f(w \cdot x_{i:i+h1} + b) \tag{15}$$

with b is bias term and f is a non-linear function. This filter is applied to each possible window of words $x_{1:h}, x_{2:h+1}, ..., x_{nh+1:n}$ to create a feature map as:

$$c = [c_1, c_2, ..., c_{n-h+1}] \tag{16}$$

Then, apply a max-overtime pooling operation over the feature map and take the maximum value $\hat{c} = \max\{c\}$ as the feature corresponding to this particular filter. Finally, we get the valuable features in the text.

IV. Experiment

A. Experimental procedure

The experimental procedure is illustrated in Fig. 3. The raw dataset was applied with various preprocessing steps to create a clean dataset, which was then splitted into training and testing datasets. The training dataset is used to train six hate speech detection models, including Multinominal Naive Bayes (MNB), Decision Tree (DT), Logistic Regression (LR), Bi-LSTM, Bi-GRU, and Text-CNN. The trained models were then applied to the test dataset to evaluate their performances in terms of Accuracy and F1-score. The best model was then chosen to build the real-time HSD application. The streaming data of social network comments were crawled, preprocessed, and fed into this application to provide real-time statistical analysis of these comments.

B. Dataset

We use the Vi-HSD dataset [3] in this work. This is a dataset containing comments collected from Facebook posts and YouTube videos with content related to entertainment, celebrities, and socio-political issues. In addition, dataset annotated processes are very strict to meet the research requirements in classifying offensive and hate comments. The dataset includes 33400 comments, each comment in the dataset has been annotated one label such as HATE, OFFENSIVE, and CLEAN. CLEAN are comments without any harassment. OFFENSIVE are comments that contain harassment content, even profanity, but do not attack any specific object. HATE are comments have harassment and abusive content and direct aim at an individual or a group of people based on their characteristics, religion, and nationality.

In Table I, we have illustrated some comment sentences of three labels extracted from the dataset.

Table I: Sample sentences from the dataset.

#	**Comment**	**English translation**	**Label**
1	Nghe giọng nứkg quá Abe :))	Listen to voice feels sexually aroused Abe))	offensive
2	Luôn ủng hộ K-ATM	Always support K-ATM	clean
3	Bảo dân trí thấp thì tự ái :)	Getting defensive when being told ill-educated	offensive
4	<Name> sủa dơ	<Name> bark loudly	hate
5	Hot girl méo gì nhìn xấu vkl thề :)))	What a fucking ugly chick :)))	hate

C. Data preprocessing

We use several techniques to preprocess the data to improve performance. These data preprocessing methods have an essential role, especially in text classification, and the problem we are working on is classifying offensive and hate comments on Vietnamese text.

Some of the steps in this preprocessing phase include tokenization using the PyVi tool, removing words that don't make much sense with the stopword technique, converting uppercase letters to lowercase letters, and removing content that is not user's comments like URLs, hashtags, emojis,... After this process, we have the user comments data cleaned before training the model.

D. Distributed model training

The machine learning models and deep learning models including MNB, LR, DT, Bi-LSTM, Bi-GRU, Text-CNN were trained in distributed manner. We only use the first 100 words of each sentence to train the models.

Machine learning models: For the NB, the hyperparameter *smoothing* is set to 1. For the LR, the hyperparameter *maxIter* is set to 10, *regParam* is 0.3 and 0.8 for *elasticNetParam*.

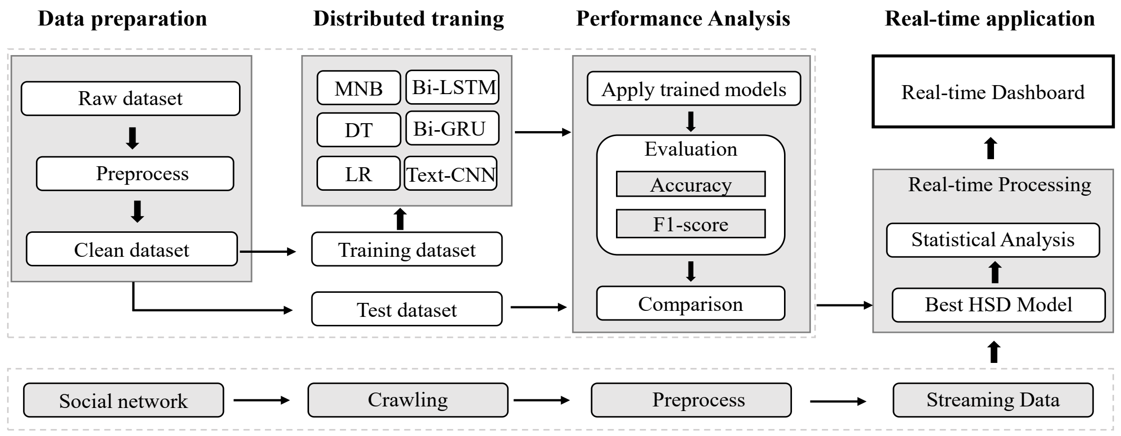

Figure 3: Experimental procedure

Deep learning models: models are trained with 40 epochs and batch size of 256. We add one more reversed direction layer in LSTM and GRU. There are 80 *units* in the layers and the *spatial dropout* is 0.2 for both of these models. With Text-CNN, using 2D Convolution layer 32 *filters* and *sizes* are 2, 3, 5 and 0.5 is set for the *dropout* ratio.

We use the MLLib library to train machine learning models and the BigDL framework is used to train deep learning models.

E. *Evaluation metric*

We use two metrics including F1 score and Accuracy to evaluate the performance of the models. Because the given datasets have significantly imbalanced classes, the F1-macro score, which weights accuracy and recall equally for each label before finding their unweighted mean, is the best measurement for this task.

F. *Experiment results*

The experiment results are shown in Table II. To evaluate the performance of models, Accuracy and F1-macro metrics is chosen as the primary measurement. Because the Vi-HSD dataset has an imbalance between labels, CLEAN is a label that accounts for a more significant proportion than the other two labels. The best model for the results is Text-CNN with a result of 61.76% by F1-macro metric and 86.84% by Accuracy metric. If only in traditional machine learning models, the Logistic Regression model has better results than the remaining models with 58.42% by F1-macro and 82.38% by Accuracy metric.

V. Proposed Large-Scale Real-Time Hate Speech System

A. *The architecture of the proposed application*

The architecture of the proposed application is illustrated in Fig. 4. Several hate speech detection models are trained distributedly using BigDL and the best model is integrated into Spark streaming. The user comments will be crawled from a page searched for a keyword on Facebook using the Selenium framework. Then, the data is fed into Spark Streaming to perform data preprocessing. Next, perform steps such as word splitting and normalizing to lowercase letters. Finally, the cleaned user comments are encoded from text to numbers to match the input of the predictive model. After preprocessing, Spark Streaming will pass the encoded data to the model to predict the label for each comment bridge. Finally, the prediction results will be statistically visualized on the website interface.

Table II: Experimental results

	Model	Accuracy (%)	F1-score (%)
ML models	MNB	76.08	53.96
	LR	81.50	56.64
	DT	84.40	43.36
DL models	Bi-LSTM	85.51	59.04
	Bi-GRU	85.34	59.46
	Text-CNN	**86.84**	**61.67**

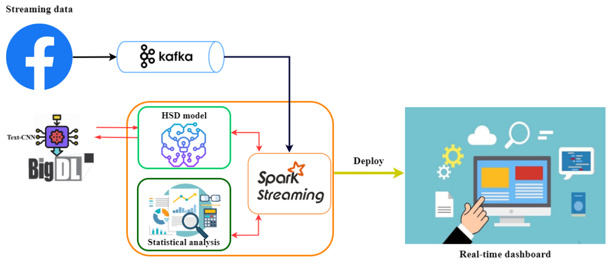

Figure 4: The architecture of the proposed system

The data of users pouring in is a lot, and the speed is fast. To test our system, we will collect data continuously

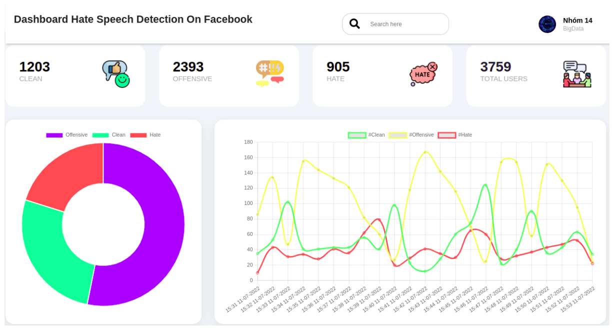

Figure 5: Real-time hate speech detection system

from users on Facebook. First, the Selenium framework is used as the tool for data retrieval. Our system allows searching for a keyword on Facebook and filters the search content to just the pages. For each page from the search results, the comments of each article will be retrieved and stored as JSON for each comment sentence. The information per user comment that we store includes **Timestamp** (at the time that the comment was taken), **User** (user's name), and **Comment** (comment content).

The output from HSD model is fed into another component to give statistical analysis of comments. This analysis is displayed through real-time dashboards.

B. *Application interface*

We use Flask framework in python to build a web application quickly. Figure 5 shows the web application interface of the system. User's comment data is predicted and updated continuously after a short period (typically 15 to 20 seconds).

Hate speech detection models integrated inside Spark Structured Streaming will receive streaming data of Facebook's comments from Kafka and predict labels for these comments. Prediction results will be pushed to the dashboard, including statistics of the number of users, and the number of comments with clean, offensive, and hate content. Besides, we have visualized statistics with the doughnut, line, and word cloud charts. A doughnut chart shows the ratio of the number of comments between the predicted labels, a line chart shows the change in the number of comments on the labels over time. With these statistics and charts, users can easily capture information about the status of user comments on Facebook from the past to the present. In addition, the ten latest user comments and predictable labels are also visible on the application. Users with the highest number of comments with offensive and hate labels are also pointed out.

VI. Conclusion

This study presents a real-time hate speech detection application for Vietnamese social network comments. Compared to previous works on the same topic, which only focus on building hate speech detection models, this study focus on the practicality of the system. It is related to the feasibility of training a large amount of data, the capability of applying the trained model to process a huge amount of comments, and producing output in real-time. In this study, a distributed deep learning library called BigDL is used to train several machine learning and deep learning based hate speech detection models. The trained model with the best performance is integrated into Spark Structured Streaming, a big data framework to create a system capable of processing a huge amount of data which is continuously crawled from social network and streamed via Kafka to provide real-time hate speech detection result. The results are then statistically analyzed and displayed through web-app based dashboards in real-time.

References

[1] H. Elzayady, K. M. Badran, and G. I. Salama, "Sentiment analysis on twitter data using apache spark framework," in *2018 13th International Conference on Computer Engineering and Systems (ICCES)*, 2018, pp. 171–176.

[2] H. X. Huynh, L. X. Dang, N. Duong-Trung, and C. T. Phan, "Vietnamese short text classification via distributed computation," *International Journal of Advanced Computer Science and Applications*, 2021.

[3] S. T. Luu, K. V. Nguyen, and N. L. Nguyen, "A large-scale dataset for hate speech detection on vietnamese social media texts," *CoRR*, vol. abs/2103.11528, 2021.

[4] C. Martín, P. Langendoerfer, P. S. Zarrin, M. Díaz, and B. Rubio, "Kafka-ml: connecting the data stream with ml/ai frameworks," *Future Generation Computer Systems*, vol. 126, pp. 15–33, 2022.

[5] E. Shaikh, I. Mohiuddin, Y. Alufaisan, and I. Nahvi, "Apache spark: A big data processing engine," 11 2019, pp. 1–6.

[6] J. J. D. et al., "Bigdl: A distributed deep learning framework for big data," *CoRR*, vol. abs/1804.05839, 2018.

[7] C.-H. Chen, "Improved TFIDF in big news retrieval: An empirical study," *Pattern Recognition Letters*, vol. 93, pp. 113–122, Jul. 2017.

[8] S. A. F. Arif Abdurrahman Farisi, Yuliant Sibaroni, "Sentiment analysis on hotel reviews using Multinomial Naïve Bayes classifier," *The 2nd International Conference on Data and Information Science*, 2019.

[9] K. N. P. A. T. George Antonogeorgos, D. B. Panagiotakos, "Logistic Regression and Linear Discriminant Analyses in Evaluating Factors Associated with Asthma Prevalence among 10- to 12-Years-Old Children: Divergence and Similarity of the Two Statistical Methods," *International Journal of Pediatrics*, 2009.

[10] L. Rokach and O. Maimon, "Decision Trees," vol. 6, pp. 165–192, Jan. 2005.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder–decoder approaches," in *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, Doha, Qatar, Oct. 2014, pp. 103–111.

[13] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014, pp. 1746–1751.