Emotion Detection in Text using Nested Long Short-Term Memory

Daniel Haryadi[1], Gede Putra Kusuma[2]
Computer Science Department, BINUS Graduate Program, Master of Computer Science
Bina Nusantara University, Jakarta, Indonesia, 11480

***Abstract*—Humans have the power to feel different types of emotions because human life is filled with many emotions. Human's emotion can be reflected through reading or writing a text. In recent years, studies on emotion detection through text has been developed. Most of the study is using a machine learning technique. In this paper, we classified 7 emotions such as anger, fear, joy, love, sadness, surprise, and thankfulness using deep learning technique that is Long Short-Term Memory (LSTM) and Nested Long Short-Term Memory (Nested LSTM). We have compared our results with Support Vector Machine (SVM). We have trained each model with 980,549 training data and tested with 144,160 testing data. Our experiments showed that Nested LSTM and LSTM give better performance than SVM to detect emotions in text. Nested LSTM gets the best accuracy of 99.167%, while LSTM gets the best performance in term of average precision at 99.22%, average recall at 98.86%, and f1-score at 99.04%.**

***Keywords*—Sentiment analysis; emotion detection; text mining; nested LSTM; machine learning**

I. Introduction

According to Liu, sentiment analysis is a field of study that analyzes opinions, sentiments, evaluations, judgments, behaviors, and emotions towards an entity such as products, services, organizations, individuals, issues, events, topics, and attributes [1]. Sentiment analysis analyzes each word or phrase and determines the orientation of the polarity of its sentiments, whether it is positive and negative [2]. Sentiment analysis is closely related to emotion detection. In computer science, text categorization in emotional states is known as sentiment analysis or emotion detection [3]. Text can trigger emotions when someone who reads the text and also can reflect or express the emotional state of the person who wrote it [4]. Humans have the power to feel different types of emotions because human life is filled with many emotions. Happy (joy), fear, anger, and sadness are some of the emotional states that can be found in everyday life [3].

Using a machine learning technique, we could use a computer to learn emotion from the text. In machine learning, computers are not taught to solve a problem by using a set of rules that have been programmed, but by making a model that can evaluate an example so it can predict a sentiment or emotion [5]. Part of machine learning is deep learning, which is also part of artificial intelligence [5]. Deep learning uses deep neural networks to study input data that can be a good representation, which can then perform a specific task [5]. Also, sentiment analysis (positive or negative) using deep learning has been showed to have a better accuracy compared to the traditional machine learning such as Naïve Bayes (NB) and SVM in [6].

Many studies about emotional detection have been carried out. Many of them are using machine learning techniques. One example of the study of emotional detection in texts conducted in Indonesia in 2012. In this study, emotions are grouped into 6 types, namely excitement, sadness, fear, anger, disgust, and surprise. It uses the methods of NB and K-Nearest Neighbors (KNN) [7]. In addition, there was also a study on evaluating traditional machine learning methods such as NB, SVM, KNN, and J48 for emotion detection in 2016 [8]. However, these studies do not use deep learning methods.

One of the methods of deep learning is LSTM. A study conducted in 2018 shows LSTM can be used to carry out sentiment analysis for classifying sentiment into positive and negative sentiments [9]. LSTM itself has various kinds of architectures such as Nested LSTM, Bi-LSTM, Gated Recurrent Unit (GRU), and Backpropagation Through Time (BPTT). The latest variant found in 2018 is Nested LSTM. Nested LSTM is claimed to be superior to stacking LSTM layers or commonly called Stacked LSTM [10]. Therefore, we are interested in examining the emotion detection found in the text by using deep learning, especially Nested LSTM.

II. Related Works

Before working on our research, we have reviewed some works that have been done related to our research. Summary of related works on emotion detection or emotion classification is shown in Table I.

In recent years, various methods of emotional detection have been proposed. In 2012, there was a study using the Multinomial Naïve Bayes (MNB) method and the classification of LIBLINEAR for emotion classification [11]. In this study, the effect of increasing datasets on the classification of MNB and LIBLINEAR was evaluated. They found out that increasing datasets could improve accuracy. The MNB reached 61.15% accuracy and LIBNEAR achieved 61.63% accuracy. Still, in the same year, an emotional detection study was also conducted by comparing the NB method with KNN [7]. In this study, KNN obtained better accuracy at 71.26% than NB at 58.01% for the classification task.

In 2014, a study was conducted using the KNN, PMI, and PMI-IR as classifiers [12]. KNN classifier is used to measure semantic and keyword similarities. When the KNN classifier fails to classify, then the PMI or PMI-IR classifier will be used

to classify again. In 2015, a study showed that emotion detection could be determined by proposing the Maximum Vector and Entropy Machine Support algorithm [13].

TABLE. I. SUMMARY OF RELATED WORKS

No	Year of research	Title	Classification	Method / Algorithm	Reported Accuracy
1	2012 [11]	Harnessing Twitter 'Big Data' for Automatic Emotion Identification	Joy, sadness, anger, love, fear, thankfulness, surprise	Multinomial Naïve Bayes (MNB) dan LIBLINEAR	61.63%
2	2012 [7]	Classification of Emotions in Indonesian Texts Using K-NN Method	Anger, fear, sadness, joy, disgust, shame	Naïve Bayes dan K-Nearest Neighbor	71.26%
3	2014 [12]	Emotion Recognition from Text Based on Automatically Generated Rules	Happiness, sadness, surprise, disgust, anger, fear	KNN classifier, Point Mutual Information (PMI) classifier, and Point Mutual Information with Information Retrieval (PMI-IR)	-
4	2015 [13]	Emotion Detection from Punjabi Text using Hybrid Support Vector Machine and Maximum Entropy Algorithm	joy, surprise, anger, love, fear, sadness, disgust	Support vector machine and maximum entropy.	-
5	2016 [14]	Evaluation of Classification Methods for Indonesian Text Emotion Detection	Anger, disgust, fear, joy, sadness, and surprise.	Naïve Bayes, J48, K-Nearest Neighbor, Support Vector Machine Minimal Optimization (SVM-SMO)	85.5%
6	2017 [15]	Detecting Emotion from Text and Emoticon	25 emotion such as sad, hurt, happy, angry, confused, advice.	matching keyword analysis, keyword negation analysis, gathering proverbs, emoticon, simplify the word, dan exclamatory word	80%
7	2017 [16]	EmoTxt: A Toolkit for Emotion Recognition from Text	Love, joy, anger, sadness, surprise, fear	Support Vector Machine	-

In 2016, an evaluation of several classification methods for emotional detection in the text was carried out [14]. In the study, evaluations were carried out on the NB, J48, KNN, and SVM-SMO methods. Experiments were conducted using Indonesian texts, which consist of 1000 sentences containing 6 classifications, namely anger, disgust, fear, joy, sadness, and surprise. Preprocessing was done using tokenization, case normalization, stop word removal, stemming. The Term Frequency-Inverse Document Frequency (TF-IDF) was used to extract features. From the results of the study, it can be seen that the SVM-SMO has the highest accuracy compared to the other methods. The results show that the accuracy of NB, J48, KNN, and SVM-SMO are 80.2%, 80.8%, 68.1%, and 85.5% respectively.

Emotion detection study was also carried out in 2017 using keyword analysis matching method, keyword negation analysis, collection of proverbs, emoticons, short forms of words, and exclamations [15]. This study achieved 80% accuracy in classifying 25 emotions. Still, in the same year, a toolkit named EmoTxt was proposed to classify emotions. EmoTxt was developed using the SVM method [16].

From the review on related works, it can be concluded that SVM is a method that produces the best level of accuracy and has been successfully implemented. Thus, we chose SVM as a benchmark method. However, the achieved accuracy has not been satisfactory. Therefore, a deep learning approach is proposed in this work, since deep learning has been shown to be superior to the traditional machine learning methods.

One of the deep learning models is LSTM. The LSTM has been proven to be able to classify both positive and negative sentiments. Besides LSTM, there are also variants of LSTM. One of them is Nested LSTM, which is claimed to have better accuracy than LSTM in making predictions on the character-level prediction of Chinese poetry generation [10]. Therefore, in this study, LSTM and Nested LSTM are examined whether the methods can be used to make better predictions on emotion detection. The classification tasks are not only on positive and negative sentiments, but on multiple emotions classification, namely anger, fear, joy, love, sadness, surprise, and thankfulness.

III. RESEARCH METHOD

A. Dataset

Lots of people expressed their feeling on Twitter. That is the reason Twitter has many resources for opinion or idea about what people feel or think. On the previous study on big data for emotion identification, this work has successfully retrieved more than 2 million tweets on the Twitter site as their dataset in 2012 [11]. Unfortunately, not all of the data can be retrieved because of the Twitter privacy policy. They can share only the tweet id and classification. By utilizing Twitter API, we retrieve the text by tweet id. We have successfully retrieved 980,549 training data and 144,160 testing data. Table II shows the distribution of dataset, which is used in this experiment. Meanwhile, Table III shows some examples of the dataset. This table shows the sample text and the classification category that text belongs to.

TABLE. II. DATASET FOR EXPERIMENT

	Training Data	Testing data	Total
Anger	214,324	31,459	245,783
Fear	52,763	7,992	60,755
Joy	282,861	41,783	324,644
Love	121,830	17,812	139,642
Sadness	242,840	35,434	278,274
Surprise	9,739	1,452	11,191
Thankfulness	56,192	8,228	64,420
Total	980,549	144,160	1,124,709

TABLE. III. SOME EXAMPLES OF THE DATASET

No	Tweet	Classification
1	adam levinne is my #love #sohot	Love
2	Aaliyahs Christmas is almost done ;) #Excitement	Joy
3	I think I miss my boyfriend.. :(#lonely	Sadness
4	Two big parcels from Francais just arrived courtesy the nicest mailman in the world. #thankful	Thankfulness
5	@NathanTheWanted Please could you wish me good luck with my prelims? :D It would honestly mean so much! #nervous xx	Fear
6	She's not here again #surprise	Surprise
7	It doesn't really make sense that the Big EAST now has 4 teams from way WEST. #desperation	Sadness

B. Preprocessing

To be able to detect emotion, there are steps need to be done. The first step is preprocessing. Fig. 1 shows the illustration of the preprocessing for the LSTM and Nested LSTM. Preprocessing carried out in this study is eliminating punctuations and changing words into all lowercase letters. The punctuations such as [! " # $ % & ' (*) + , - . / \ : ; < = >? @\ ^ _ ` { } | ~)] are excluded. Then, each word in the sentence is represented by an integer, where the integer is unique for each different word. Then, the addition of padding "0" in the beginning so that each sentence has the same length. When that step is done, the integer will be an input for the neural network. In this preprocessing stage, we also calculate the unique number of words in the training data and the highest number of words in one sentence. There are unique words in the training data and the highest number of words in one sentence is 41 words. These parameters will be used later in the processing layer.

On the other hand, the benchmark method of SVM needs a feature extraction to be able to do the classification. We use TF-IDF as features for the SVM. However, for LSTM and Nested LSTM, we did not use it because a deep neural network does not need a feature extraction method. In this experiment, we use TFIDFVectorizer from the Sklearn library to implement preprocessing and convert the text into TFIDF weight.

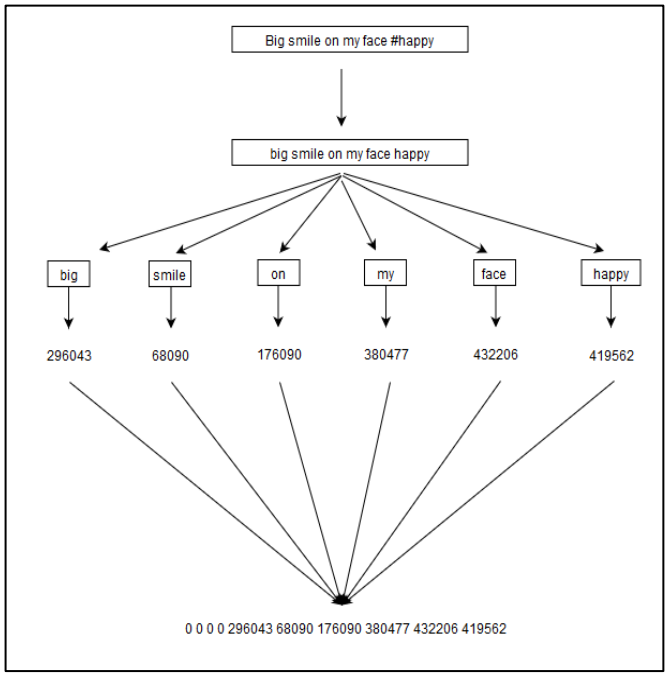

Fig. 1. Preprocessing for LSTM and Nested LSTM.

C. Emotion Classification

Fig. 2 shows an illustration of LSTM modeling using the Keras library. The followings are the specifications of the parameters used in conducting training:

a) Embedding layer

- *Input dimension* = 502,882 (number of unique words)
- *Output dimension* = 50 (size of embedding vector)
- *Input length* = 41

b) LSTM layer

- *Units* = 50
- *Dropout* = 0.2

c) Output layer

- Units = 7
- Activation = SoftMax

The embedding layer requires some parameters as its input. In the preprocessing step, we have calculated the highest number of words in one sentence that is 41 words. We put that value as the input length parameter for the embedding layer. That means there are 41 times steps of word embedding. The embedding layer only has one neuron. Every word that passed into this neuron will be transformed into a real-valued vector of length 50 (output dimension). Once the network has been trained, we can get the weights of the embedding layer, which in this case will be of size (502882, 50). It means that every word has one real-valued vector of length 50. The embedding process is implemented based on the word2vec embedding method of Mikolov *et al.* [17].

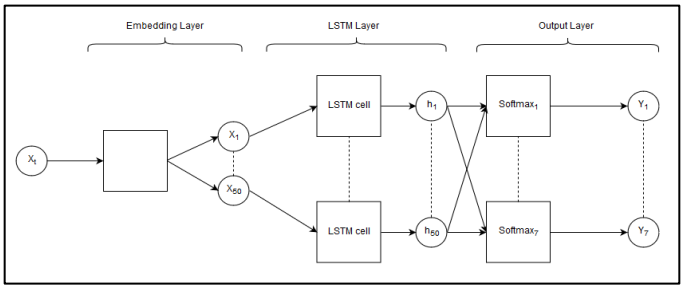

Fig. 2. Processing Layers of LSTM Model.

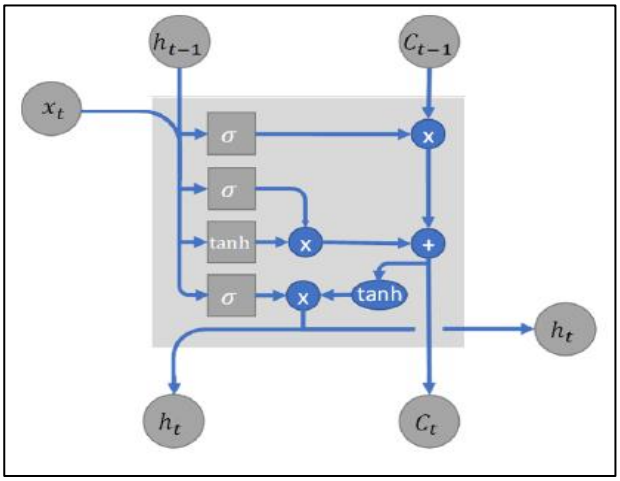

Fig. 3. LSTM Cell [9].

In this experiment, we set the LSTM cell to 50. Each LSTM cell will produce an output vector that is connected to the output layer. The LSTM cell can be seen in Fig. 3. Each LSTM cell consists of 4 gates that process each vector input. At the end of the process, each cell will produce an output that will be used for the output layer.

The first step of LSTM in Fig. 4(A) is a forget gate layer, which has sigmoid activation function that gives an output of 0 or 1. 0 means "let nothing through" and 1 "remember anything". Next, Fig. 4(B) is to decide which information will be stored. The sigmoid layer (input gate layer) to decide which value will be updated and the tanh layer is creating new candidate values between −1 and 1. Next step, Fig. 4(C), the old cell state is multiplied by output from forget gate, to forget the things that are not needed anymore, and the new information is added to the cell state. The final step in Fig. 4(D) is to decide the output. First, we run a sigmoid layer that decides what parts of the cell state we are going to output. Then, we put the cell state through tanh (to push the values to be between−1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to [18] [9].

At the output layer, there are 7 neurons where each of these neurons has SoftMax activation to produce values for each classification. The prediction will be based on the highest output value. In the training phase, we use Adam optimizer with a learning rate of 0.00001.

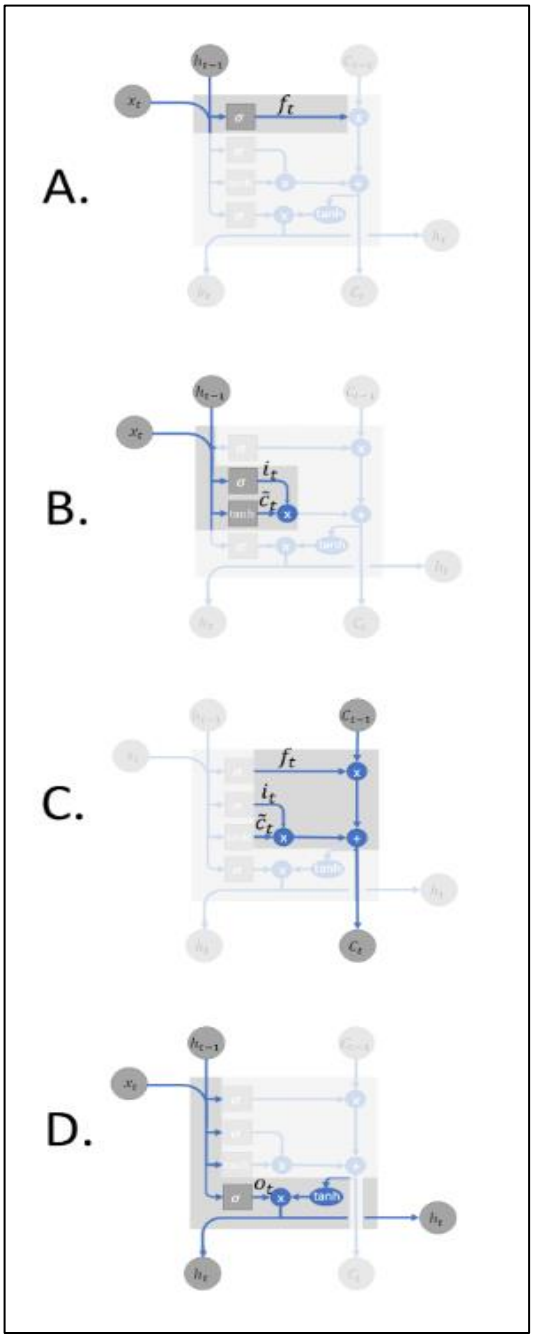

Fig. 4. LSTM Steps [9].

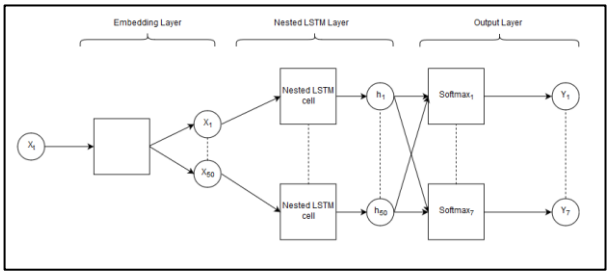

Fig. 5. Preprocessing Layers of Nested LSTM Model.

We also replaced the LSTM layer into Nested LSTM Layer in the experiments. Fig. 5 is an illustration of Nested LSTM modeling. The model consists of several layers, including:

a) Embedding Layer

- *Input dimension*=502,882 (number of unique word)
- *Output dimension*=50 (size of embedding vector)
- *Input length*=41

b) Nested LSTM Layer

- *Units*=50
- *Dropout*=0.2
- *Depth*=2

c) Output Layer

- Units=7
- Activation = SoftMax

The Nested LSTM is similar to the LSTM. The only difference between with LSTM and Nested LSTM is the second layer. Fig. 6 shows nested LSTM cell with depth = 2.

The Nested LSTM is a simple extension of LSTM. Instead of creating stacked LSTM, Nested LSTM created another LSTM via nesting. They called it inner LSTM. The inner memory cells of Nested LSTM form an internal memory, which is only accessible to other computational elements via the outer memory cells, implementing a form of temporal hierarchy. Inner LSTM gets the input from outer LSTM. Nested LSTM replaces the addition operation in Fig. 4(C) on LSTM steps to compute c_t in LSTM with a concatenation to be an input for inner LSTM [10].

As a comparison to the LSTM and Nested LSTM, we also train SVM with the same training data. To create the SVM model with large scale data, we use SGDClassifier [19] from Scikit-learn library [20], which is a linear classifier of SVM. All parameter used is default parameters, except for the *random_state* parameter that is set to 0. After all models are created, the models are then tested using the testing data.

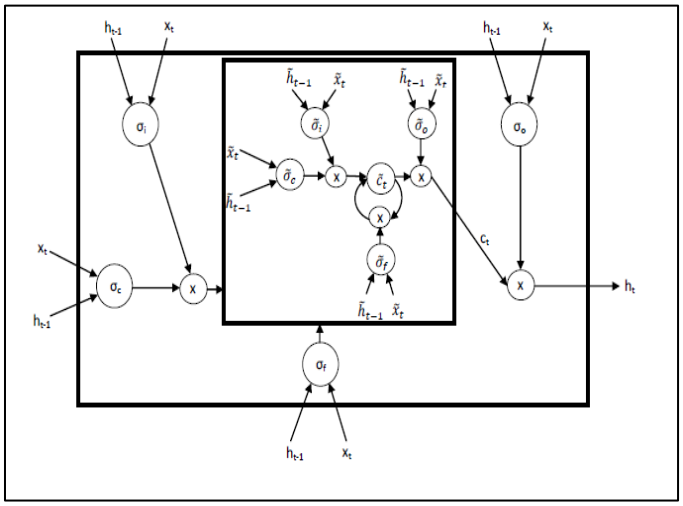

Fig. 6. Nested LSTM Cell [10].

IV. Experiment and Result

A. Experiment

During training LSTM and Nested LSTM model, we recorded every epoch of the model as a checkpoint to get accuracy and loss progress. We set the maximum number of the epoch at 50. The accuracy and loss outputs of the LSTM and Nested LSTM from the checkpoints are recorded.

Fig. 7 shows the accuracy and loss outputs of LSTM during the training phase. The loss value is decreased and the accuracy is increased at every epoch of the LSTM model training. The blue line (top) shows the accuracy and the orange line (bottom) shows the loss value. At the end of 50 epoch, the training reaches an accuracy of 100% and a loss value of 0.2%.

Fig. 8 shows the accuracy and loss outputs of Nested LSTM during the training phase. The loss value is also decreased and the accuracy is also increased at every epoch of the Nested LSTM model training. The yellow line (top) shows the accuracy and the blue line (bottom) shows the loss value. At the end of 50 epoch, the Nested LSTM training reaches an accuracy of 99.9% and a loss of 0.3%.

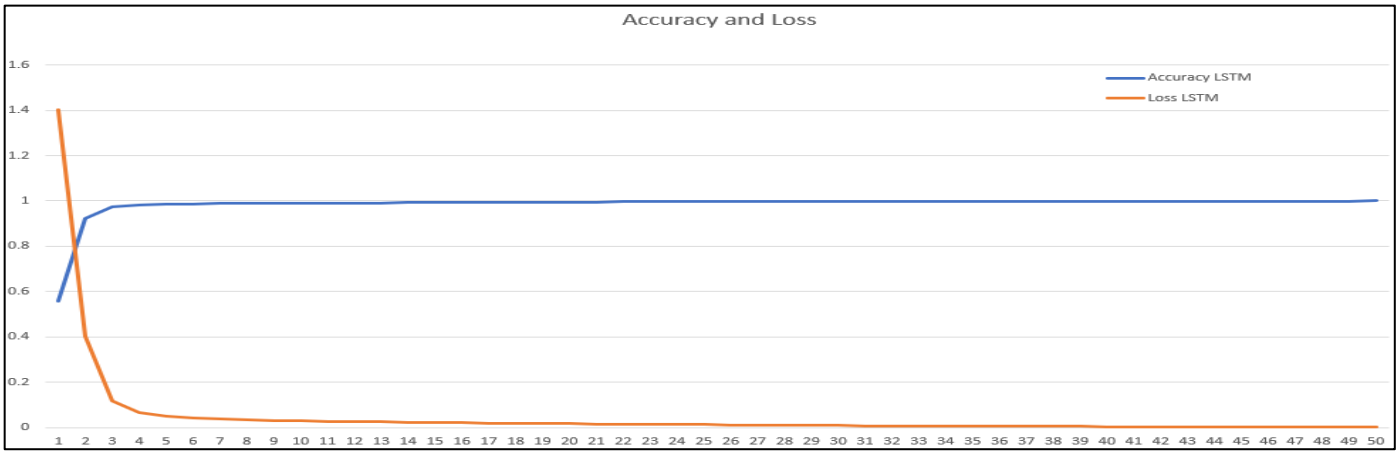

Fig. 7. Accuracy and Loss Outputs of LSTM During Training.

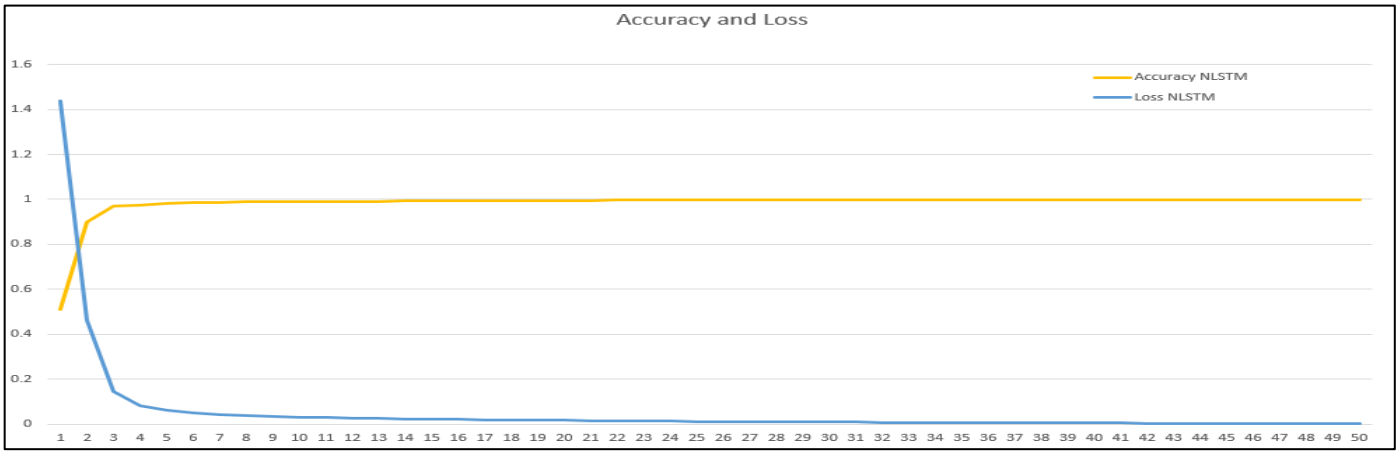

Fig. 8. Accuracy and Loss Outputs of Nested LSTM During Training.

B. *Testing Result*

Table IV shows a confusion matrix for the LSTM method. In this table, A is for Anger; B is for Fear; C is for Joy; D is for Love; E is for sadness; F is for surprise, and G is for thankfulness. The LSTM model is tested using 144.160 testing data. The LSTM achieves an overall accuracy of 99.154%.

Table V shows the precision, recall, and f1-score for the LSTM method. The performances of the LSTM method on each class and its average performances are calculated based on the confusion matrix. The LSTM method yields an average precision of 99.22% and an average recall of 98.86%, and therefore an average f1-score of 99.04%.

Table VI shows a confusion matrix for the Nested LSTM method. The Nested LSTM model is also tested using 144.160 testing data. The Nested LSTM achieves an overall accuracy of 99.167%.

TABLE. IV. CONFUSION MATRIX FOR LSTM

		Predicted						
		A	B	C	D	E	F	G
Actual	A	31247	12	62	21	107	0	10
	B	29	7870	54	8	29	0	2
	C	79	28	41539	56	61	1	19
	D	27	11	54	17680	32	0	8
	E	183	31	125	44	35032	2	17
	F	9	3	7	7	9	1416	1
	G	7	7	33	10	14	0	8157

TABLE. V. PRECISION, RECALL, AND F1-SCORE OF LSTM

Class	Precision	Recall	F1-Score
Anger	98.94%	99.33%	99.13%
Fear	98.84%	98.47%	98.66%
Joy	99.20%	99.42%	99.31%
Love	99.18%	99.26%	99.22%
Sadness	99.29%	98.87%	99.08%
Surprise	99.79%	97.52%	98.64%
Thankfulness	99.31%	99.14%	99.22%
Average	**99.22%**	**98.86%**	**99.04%**

TABLE. VI. CONFUSION MATRIX FOR NESTED LSTM

		Predicted						
		A	B	C	D	E	F	G
Actual	A	31206	22	76	16	133	1	5
	B	20	7855	66	12	37	1	1
	C	38	23	41587	45	76	5	9
	D	21	6	74	17672	32	1	6
	E	118	49	148	44	35072	1	2
	F	6	5	13	0	12	1416	0
	G	11	5	31	15	15	0	8151

Table VII shows the precision, recall, and f1-score for the Nested LSTM method. The performances of the Nested LSTM method on each class and its average performances are calculated based on the confusion matrix. The Nested LSTM method achieves an average precision of 99.21% and an average recall of 98.83%, and therefore an average f1-score of 99.02%. Meanwhile, Table VIII shows a confusion matrix for the SVM method. The SVM model is also tested using 144.160 testing data. The SVM achieves an overall accuracy of 98.679%. Thus, the SVM model yields the lowest accuracy compared to the LSTM and Nested LSTM models.

Table IX shows the precision, recall, and f1-score for the SVM method. The performances of the SVM method on each class and its average performances are calculated based on the confusion matrix. The SVM method achieves an average precision of 98.53% and an average recall of 98.22%, and therefore an average f1-score of 98.37%. Its performances are lower than the LSTM and Nested LSTM methods.

Comparison of accuracy, precision, recall, and f1-score from all three models can be seen in Table X. Nested LSTM produces the best accuracy of 99.167% among the three methods, even though the difference in accuracy is not significantly different from the LSTM. But on average, LSTM obtained precision, recall, and f1-scores that are better than the Nested LSTM. The Nested LSTM and LSTM get better scores, in terms of accuracy, precision, recall, and f1-score, compared to the SVM.

TABLE. VII. PRECISION, RECALL AND F1-SCORE OF NESTED LSTM

Class	Precision	Recall	F1-Score
Anger	99.32%	99.20%	99.26%
Fear	98.62%	98.29%	98.45%
Joy	99.03%	99.53%	99.28%
Love	99.26%	99.21%	99.24%
Sadness	99.14%	98.98%	99.06%
Surprise	99.37%	97.52%	98.44%
Thankfulness	99.72%	99.06%	99.39%
Average	**99.21%**	**98.83%**	**99.02%**

TABLE. VIII. CONFUSION MATRIX FOR SVM

		Predicted						
		A	B	C	D	E	F	G
Actual	A	31325	7	16	7	51	1	0
	B	4	7874	17	5	18	1	0
	C	21	40	41211	315	134	12	15
	D	52	40	386	17330	288	14	29
	E	47	28	57	43	34919	3	5
	F	2	3	24	5	8	1419	1
	G	8	0	72	107	16	2	8178

TABLE. IX. PRECISION, RECALL AND F1-SCORE OF SVM

Class	Precision	Recall	F1-Score
Anger	99.57%	99.74%	99.66%
Fear	98.52%	99.43%	98.98%
Joy	98.63%	98.71%	98.67%
Love	97.29%	95.54%	96.41%
Sadness	98.55%	99.48%	99.01%
Surprise	97.73%	97.06%	97.39%
Thankfulness	99.39%	97.55%	98.46%
Average	**98.53%**	**98.22%**	**98.37%**

TABLE. X. SUMMARY OF PERFORMANCE RESULTS

Model	Accuracy	Precision	Recall	F1-score
LSTM	99.154%	**99.22%**	**98.86%**	**99.04%**
Nested LSTM	**99.167%**	99.21%	98.83%	99.02%
SVM	98.679%	98.53%	98.22%	98.37%

V. CONCLUSION

In this paper, we presented our study on emotion detection from text. Based on the discussion and evaluation carried out in the previous section, LSTM, Nested LSTM, and SVM methods can be used for multi-classes emotion detection. Nested LSTM has the best accuracy among the three methods with the accuracy of 99.167%. This accuracy is not significantly different from the LSTM, which gets an accuracy of 99.154%. LSTM has better average performances in terms of precision, recall, and f1-score, at 99.22%, 98.86%, and 99.04% respectively. In future works, we plan to employ and evaluate other more sophisticated deep learning models to find out the best method for emotion detections tested in a more challenging dataset.

REFERENCES

[1] Liu, Sentiment Analysis and Opinion Mining, San Rafael: Morgan & Claypool, 2012.

[2] H. Rahmath P, "Opinion Mining and Sentiment Analysis - Challenges and Applications," International Journal of Application or Innovation in Engineering & Management, vol. 3, no. 5, pp. 401-403, May 2014.

[3] J. Kaur and R. J. Saini, "Emotion Detection and Sentiment Analysis in Text Corpus: A Differential Study with Informal and Formal Writing Styles," International Journal of Computer Applications, vol. 101, pp. 1-9, 09 2014.

[4] S. M. Kim, "Recognising Emotions and Sentiments in Text," University of Sydney, Sydney, 2011.

[5] N. Buduma, Fundamentals of Deep Learning, Sebastopol: O'Reilly Media, Inc, 2017.

[6] P. Singhal and P. Bhattacharyya, "Sentiment Analysis and Deep Learning: A Survey," 2016.

[7] Arifin and K. E. Purnama, "Classification of Emotion in Indonesian Texts Using K-NN methods," International Journal of Information and Electronics Engineering, pp. 899-903, 2012.

[8] J. Riany, M. Fajar and M. P. Lukman, "Penerapan Deep Sentiment Analysis penerapan Deep Sentiment Analysis," Jurnal Sisfo, vol. 06, p. 147–156, 2016.

[9] F. Miedema, "Sentiment Analysis with Long Short-Term Memory networks," 2018.

[10] J. R. A. Moniz and D. Krueger, "Nested LSTMs," Computing Research Repository, vol. 1801.10308, 2018.

[11] W. Wang, l. Chen, K. Thirunarayan and A. P. Sheth, "Harnessing Twitter "Big Data" for Automatic Emotion Identification," Proceedings of the 2012 ASE/IEEE International Conference on Social Computing and 2012 ASE/IEEE International Conference on Privacy, Security, Risk and Trust, pp. 587--592, 2012.

[12] S. Shaheen, W. El-Hajj , H. Hajj and S. Elbassuoni, "Emotion Recognition from Text Based on Automatically Generated Rules," IEEE International Conference on Data Mining Workshops, pp. 383-392, 2015.

[13] E. U. Jain and A. Sandu, "Emotion Detection from Punjabi Text using Hybrid Support Vector Machine and Maximum Entropy Algorithm," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 11, 2015.

[14] Muljono, N. A. S. Winarsih and C. Supriyanto, "Evaluation of Classification Methods for Indonesian Text Emotion Detection," International Seminar on Application for Technology of Information and Communication, pp. 130-133, 2016.

[15] R. Rahman, T. Islam and M. H. Ahmed, "Detecting Emotion from Text and Emoticon," London Journal of Research in Computer Science and Technology, vol. 17, no. 3, 2017.

[16] F. Calefato, F. Lanubile and N. Novielli, "EmoTxt: A Toolkit for Emotion Recognition from Text," 2017 Seventh International Conference on Affective Computing and Intelligent Interaction Workshops and Demos, 2017.

[17] T. Mikolov, I. Sutskever, K. Chen, G. Corrado and J. Dean, "Distributed Representations of Words and Phrases and Their Compositionality," in Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, Lake Tahoe, Nevada, 2013.

[18] C. Olah, "Understanding LSTM Networks," 27 August 2015. [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed 5 May 2019].

[19] L. Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," Proc. of COMPSTAT, pp. 177-186, 2010.

[20] P. Fabian, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot and É. Duchesnay, "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, pp. 2825-2830, 2011.