IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.15, No.2, April 2021, pp. 177~188
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: 10.22146/ijccs.64916

Hate Speech Detection in Indonesian Twitter using Contextual Embedding Approach

Guntur Budi Herwanto*[1], Annisa Maulida Ningtyas[2], I Gede Mujiyatna[3], I Nyoman Prayana Trisna[4], Kurniawan Eka Nugraha[5]
[1,3,4,5]Department of Computer Science and Electronics, FMIPA UGM, Yogyakarta, Indonesia
[2]Department of Health Information and Services, UGM, Yogyakarta, Indonesia
e-mail: ***[1]gunturbudi@ugm.ac.id,**[2]annisamaulidaningtyas@ugm.ac.id,[3]demuji@ugm.ac.id, [4]kurniawan.eka.n@mail.ugm.ac.id ,[5]nyomanprayana@mail.ugm.ac.id

Abstrak

Ujaran kebencian muncul seiring dengan pesatnya perkembangan media sosial. Ujaran kebencian sering kali dikeluarkan karena kurangnya kesadaran publik tentang perbedaan antara kritik dan pernyataan yang dapat berujung pada ujaran kebencian. Oleh karena itu, sangat penting dilakukan deteksi dini terhadap kalimat yang akan dituliskan sebelum menimbulkan tindak pidana akibat ketidaktahuan masyarakat. Dalam penelitian ini, kami memanfaatkan pembelajaran mesin dalam untuk memprediksi apakah sebuah kalimat mengandung ujaran kebencian dan nada kasar. Kami menunjukkan kehandalan word embedding dan penyematan kontekstual (contextual embedding) untuk mendapatkan informasi semantik dalam kata-kata ujaran kebencian pada model yang dikembangkan. Selain itu, representasi penyematan dokumen melalui jaringan saraf berulang dengan gated recurrent unit digunakan sebagai arsitektur utama untuk memperkaya representasi dari penyematan dokumen. Dibandingkan dengan representasi sintaksis dari pendekatan sebelumnya, penyematan kontekstual dalam model kami terbukti memberikan peningkatan yang signifikan pada performa model dengan margin yang signifikan.

Kata kunci*—ujaran kebencian, pemrosesan bahasa natural, jaringan saraf dalam, penyematan kontekstual, jaringan syaraf berulang*

Abstract

Hate speech develops along with the rapid development of social media. Hate speech is often issued due to a lack of public awareness of the difference between criticism and statements that might contribute to this crime. Therefore, it is essential to do early detection of sentences written before causing a criminal act due to public ignorance. In this paper, we use the advancement of deep neural networks to predict whether a sentence contains a hate speech and an abusive tone. We demonstrate the robustness of different word and contextual embedding to represent the semantic of hate speech words. In addition, we use a document embedding representation via recurrent neural networks with a gated recurrent unit as the main architecture to provide richer representation. Compared to the syntactic representation of the previous approach, the contextual embedding in our model proved to give a significant boost to the performance by a significant margin.

Keywords*—hate speech, natural language processing, deep neural network, contextual embedding, recurrent neural network*

Received March 25th,2021; Revised April 25th, 2021; Accepted April 29th, 2021

1. INTRODUCTION

Hate speech is an expression, writing, action, or performance intended to provoke violence or discrimination against someone based on the characteristics of the community they represent, such as race, ethnicity, gender, sexual orientation, religion, and other characteristics [1]. Hate speech develops along with the rapid development of social media. It is a problem that affects the dynamics and interactions of the online social community. In the last two years, a criminal act of hate speech has been committed. Hate speech is often issued due to a lack of public awareness of the difference between criticism and statements that might contribute to this crime. Therefore, it is essential to do early detection of sentences written before causing a criminal act due to public ignorance.

Furthermore, Indonesia governs hate speech in the Electronic Information and Transactions (UU ITE) Law No. 11 of 2008, amended by Law No. 19/2016. The law includes prohibitions and criminal threats for offenders who make hate speech or fake news. Article 28 paragraph (1), under Article 45 of this Law, includes criminal threats against anyone who spreads false and misleading information that causes customer losses in electronic transactions knowingly and without authority [2]. One way to deal with hate speech found on online platforms is by reporting the content to the authorities and removing the content. Other actions in overcoming hate speech are by conducting surveillance, advocacy, and counter-speech [3]. However, these approaches are time-consuming and require human labour.

In addition to the previously mentioned countermeasures, some researchers have attempted to counteract hate speech through machine learning. Machine learning has proved to be a good tool for understanding human language. Machine learning disciplines that specifically deal with human language are called Natural Language Processing (NLP). Most of the current NLP approach uses a supervised learning algorithm. Supervised learning requires human intervention, which acts to label the sentences that are deemed to be hate speech or not. Differences in opinion among humans about whether a piece of writing is hate speech or not are part of the difficulties in determining hate speech. This represents the risk of misclassification in machine learning algorithms which are then trained using human labelling. For example, in a bag-of-words approach, we can have a dictionary of words that are classified as hate speech, such as "black", "gay", and "transgender". Currently, most of the language resources for NLP are developed for English. This poses an additional challenge for detecting hate speech in another language. This research aims to predict hate speech in Bahasa Indonesia, which gives another challenge to the language model.

As we mentioned in the previous paragraph, most of the research tackles hate speech detection problems as a supervised classification task by applying a machine learning approach. Spertus [4] utilizes machine learning with a decision tree algorithm to automatically detect messages containing offensive language on social media. Vigna and Warner [1], [5] use the Support Vector Machine (SVM) algorithm. This algorithm has an accuracy rate of 80% in the automatic detection of hate speech.

Several studies used simple linguistic features such as Bag of Words (BoW), n-gram, and Part-of-Speech (PoS) as fundamental features. Waseem and Hovey [6] conducted a study to detect hate speech on the Twitter platform. Researchers classified hate speech into two classes, namely racism and sexism. The author uses several features, such as n-gram characters, along with the user's demographic, linguistic, and geographic features. The results showed that the n-gram character with n = 4 gave the best results, and adding the user's gender feature could result in a slight improvement. Word embedding features, such as paragraph2vec [7], [8], are also used to classify user comments. Nobata et al. [8] combined the paragraph2vec feature with several features, such as n-gram features, linguistics and syntax, and semantic distribution. The addition of features shows an increase in the area under the curve (AUC) compared to only

using the paragraph2vec feature.

Aside from the supervised approach, Watanabe et al. [9] apply unsupervised machine learning with lexical features and word rules to detect sentiment in the text. The algorithm's focus is on word features, emoticons, hashtags, punctuation, and grammatical patterns. In addition, to detect harsh words in the text, they used a dictionary-based approach. Research on automatic hate speech detection using this grammatical feature is often used in English texts because English has a standard grammatical pattern. Meanwhile, the extraction of grammatical features in Indonesian, such as part-of-speech markers and automatic dependency parsers, remains limited.

Along with the development of research in the field of deep neural networks (DNN), some researchers [5], [10], [11] use DNN to solve the problem of automatic hate speech detection. The deep learning method uses word embedding to represent the features of the text. This feature can detect words that contain hate speech more effectively than syntax and linguistic features. The deep learning architectures that are often used are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), especially Long-Short Term Memory (LSTM). CNN's performance in the baseline dataset [6] has an F1 measure score of about 80% [12], [13]. Zhang et al. [14] combined the CNN architecture with Gated Recurrent Units (GRU) to improve CNN's performance. Several studies have shown that LSTM has better performance than CNN [10], [15], with an F1 score of 93%.

Our contribution to this paper is two-fold. Firstly, we used various contextual embedding and stacking between different contextual embedding approaches for hate speech detection in Indonesian. Secondly, we can outperform the performance of the prediction compared to the previous work. We use the advancement of deep neural networks to predict whether a sentence contains a hate speech sentiment. Furthermore, we predict whether a sentence contains abusive language. Finally, to provide richer representation, we use a text embedding representation through a recurrent neural network (RNN) with GRU as the main architecture.

This paper consists of 4 sections. The first section introduces the motivation for our work. We describe our method in detail in Section 2. The result of our experiment is described in Section 3, and finally, we conclude our work in Section 5.

2. METHODS

The method of detecting hate speech uses a supervised learning approach with contextual embedding and a recurrent neural network. Data that has been labelled hate speech and not hate speech is the primary input in machine learning. This study uses the recurrent neural network (RNN) representation that studies sequential patterns of text. Unlike previous studies [16], [17], we did not pre-process the data because we assumed that each word occurrence would determine a sequential pattern that could lead to hate speech or not. Before entering into the RNN document representation, we tokenize and pass each word token into the token embedding. Then document embedding will combine the token model into one document embedding vector. The document embedding representation becomes an input classifier that will determine whether the sentence is hate speech or not. The architecture of our model can be seen in Figure 1.

2.1 Dataset

This study used two datasets on hate speech on social media from Alfina [16] and Ibrohim [17]. Both of these datasets contain Indonesian tweets. In Alfina's dataset, 260 tweets have been annotated hate speech and 453 tweets that are not hate speech. Here are two examples of hate speech from the first dataset:

- *kebiadaban ahok cina kafir anak2 rakyat kecil pribumi dan ibu2 islam semua digusur dari rumahnya serta kehidupannya* (In English: The savagery of Chinese ahok kafir, the children of the indigenous small people and Muslim mothers are all evicted from their homes and their lives)
- *perempuan kaya lo mending mati aja deh, jelek aja, gausa sok jadi make up artist!* (In English: A woman like you should just die, it's ugly, I can't pretend to be a make-up artist!)

Here are two examples of non-hate speech data in the first dataset [16]:

- *Rencana Bapak yang di surga itu lebih indah yangg kita inginkan bapak ahok tetap semangat ya pak* (In English: Your plan in heaven is more beautiful. We want you to keep your spirits up, sir)
- *Itu yang ngomong jangan pilih Ahok Djarot pernah ngerasain banjir ga sih?* (In English: That's the one who says don't choose Ahok Djarot, have you ever experienced floods?)

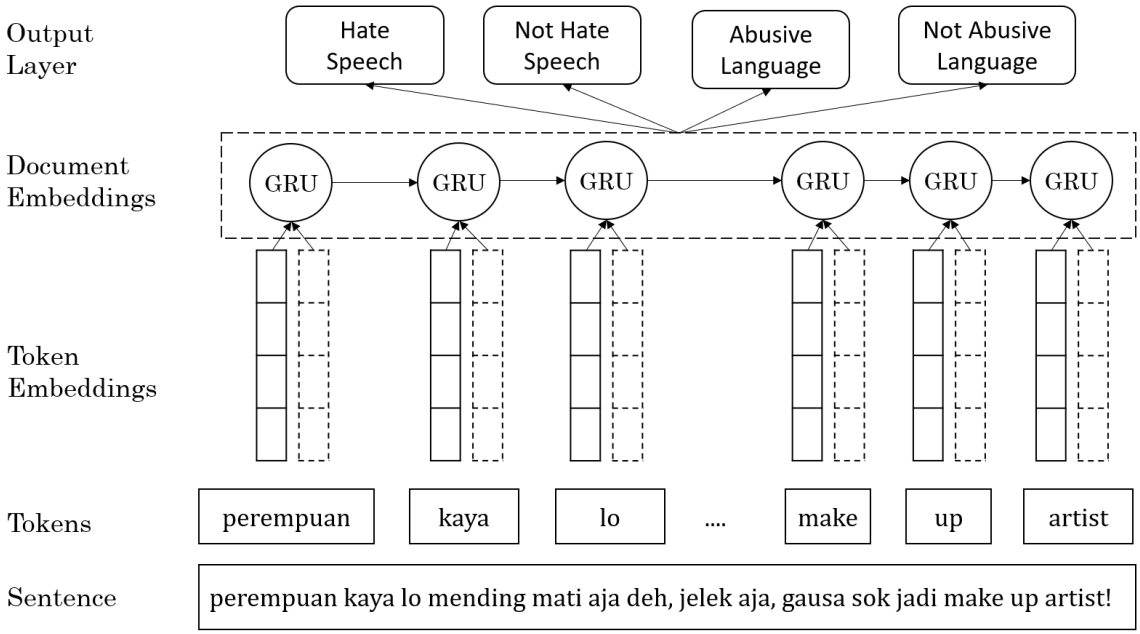

Figure 1. The architecture of our model. The model starts with the sentence, which then chunked into tokens. Aside from words, the tokens can be chunked into characters. These tokens are then represented in token embeddings. The dashed embeddings indicate that the embedding can be stacked with another embedding to provide a richer representation. The token embeddings were then combined with document recurrent neural network embedding. Finally, the model will predict the class's output, treated as a multi-label classification in this architecture.

In the second dataset from Ibrohim, 5,540 tweets have been annotated with hate speech and 7,593 tweets that are not hate speech. Here are two examples of hate speech data in the second dataset [17]:

- *Amit amit itu mulut apa congor satwa yang namanya anjing sih* (In English: God forbid is the mouth of an animal called a dog)
- *hari hari makan babi berbentuk wang haram. muka pun mcm babi. perangai lebih babi dari babi. politikus ronggeng babi* (In English: everday eating a haram pig. even face like a pig. the temperament of a pig more than a pig. *ronggeng* pig politician)

Here are two examples of non-hate speech data in the second dataset [17]:

- *Anggota TNI, Polri, dan PNS Bisa Kredit Rumah Tanpa DP* (In English: Members of the TNI, Polri, and Civil Servants Can Home Loans Without Down payment)
- *Ada tas aneh yg di sinyalir ada bom di dalam nyaa ...Depok siaga lagi ini Hadeh* (In English: There is a strange bag that indicates there is a bomb in it ... Depok is on standby again, Gosh)

We got the two datasets from Alfina's Github [1] and Ibrohim's [2]. The two researchers did not provide the training and testing fold in their experiments. Thus, we took the initiative to divide the two datasets ourselves in training and testing to test and compare the model's performance with the two baselines that have been carried out [16], [17].

2.2 Token Embedding

The first step in our method is the creation of a token embedding. In this study, we did not pre-process the data so that every word in the input sentence was converted into tokens. We perform tokenization by dividing the sentence into words. In the flair embedding method, the token is character-based. Each token is then converted into a vector. We call this *token embedding*. In general, there are two types of token embedding that we used. The first type is a classic word embedding type, and the second is a contextual embedding type.

The first type is pre-trained embedding from Indonesian fastText [18], [19], which trained on the Indonesian Wikipedia and Common Crawl [19]. The model is trained with the aim that words that have similar semantics can have similar vectors. For example, Jakarta and Bandung's words will have a similar vector because they represent the same semantic, namely the city. In the fastText model, subwords are also considered to solve out-of-vocabulary problems during the pre-training data formation. This word embedding can be called one word, one embedding. So that word embedding of this type does not pay attention to context.

The second type of token embedding in our experiment is contextual embedding. Contextual embedding is a token embedding method that can encode semantic information relevant to the context of training data. In other words, the representations created by contextual embedding can differ depending on the sentence's context. In this study, we used a representation of Flair Embedding [20]. Flair embedding is contextual embedding which is trained by predicting the next character from a series of characters. This training model is proven to encode linguistic concepts such as words, sentences, and even sentiments in the context used. Flair Embedding is trained without explicit word feature information. It fundamentally models the word as a sequence of characters and is contextualized by the surrounding text, which means that the same word will have different embeddings depending on contextual usage. However, there are drawbacks to character-based approaches such as Flair Embedding. The drawback is that it is difficult to produce meaningful embeddings if there are character sets that are rarely used in a context. To overcome this shortcoming, the same researchers [21] proposed a method in which each unique character set will be dynamically combined. Then a pooling operation is used to filter the global word representation from all contextual instances. This method is called Pooled Flair Embedding [21].

2.3 Document Embedding

In contrast to token embedding, Document Embedding creates a single vector embedding for the entire text, while token embedding creates a vector embedding for each word or character. This is necessary to ensure that each sentence with a different number of words is represented identically. This study uses a recurrent neural networks (RNN) technique that trains the sequential token embedding pattern [22].

[1] https://github.com/ialfina/id-hatespeech-detection
[2] https://github.com/okkyibrohim/id-multi-label-hate-speech-and-abusive-language-detection

RNN is a form of neural network architecture in which processing is repeated for sequential data input. Because data is processed across multiple layers, RNN falls into the deep learning category. In long sequence patterns, the RNN has a problem with gradients which tend to have very small values, close to zero. This problem is often called a vanishing gradient. In this study, we used the Gated Recurrent Units (GRU). GRU is a variant of RNN and Long Short Term Memory. GRU can overcome the vanishing gradient by adding a gate mechanism in the RNN architecture [23]. GRU has been proven to overcome long sequence patterns and has a more straightforward gate mechanism than LSTM. The advantage of GRU is that the computation time is better and has competitive accuracy to avoid the problem of disappearing gradients. The two main gates in the GRU is the update gate and reset gate.

The update gate is used to determine the amount of information from the previous units to the next unit. This mechanism can help the model to prevent the vanishing gradient problem. The update gate is computed by equation (1). The equation is almost the same as the linear layer in the vanilla neural network, which multiplies the weight $W^{(z)}$ with the network unit x_t in timestep t. However, it is added by the weight information $U^{(z)}$ multiply by the network of the previous unit h_{t-1}. Finally, the result is passed to the sigmoid activation function.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{1}$$

The second gate is the reset gate. The reset gate is used to determine how much information from the past should be discarded. The reset gate is computed by equation (2). The equation is identical to the update gate. However, the usage of each of the outputs will be different in the later step.

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{2}$$

The final step of the GRU is to calculate the current memory content and the final memory at the current time step. The reset gate will be used in the current memory content, which will calculate how much information to be discarded. The equation to calculate the current memory content can be seen in (3).

$$h_t^{'} = \tanh\left(W_{X_t} + r_t \odot U h_{t-1}\right. \tag{3}$$

The update gate will be used in the final memory at the current time step to decide what information should be passed to the next unit. The equation to obtain the final memory at the current time step can seen in (4).

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h_t^{'} \tag{4}$$

2.4 Output Layer

The output of the document embedding will be passed to a linear output layer. In this study, we did not add an additional hidden layer after the embedding layer. We assume that a combination of token embedding and document embedding can provide representations that describe semantic patterns and sequential patterns that lead to hate speech. In the output layer, we use cross-entropy loss because our classifier is binary. The formula for cross-entropy loss can be seen in (5).

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i)) \tag{5}$$

Where $H_p(q)$ is cross-entropy, and y_i shows the class label of hate speech or non-hate speech and $p(y_i)$ is the probability that a sentence is hate speech or not from all sentences in the corpus.

3. RESULTS AND DISCUSSION

In this section, we present the setup of our experiment performance, followed by our model's performance.

3.1 Experimental Setup

Our experiments were conducted on three different sets of data. Unlike our previous work [24], we did not combine the two datasets to provide a benchmark for improving our model's performance. The first set of data is from Alfina [16], which contains two labels only, hate speech (HS) or *not hate speech (NHS)*. The second and third sets of data are from Ibrohim and Budi [17]. For the second dataset, we only took the hate speech label. Thus, we treat the second dataset as a binary classification problem. For the third dataset, we took the *hate speech (HS)* and *abusive language (AB)* columns to benchmark the performance with their original paper [17]. As a result, we treat the third dataset as a multi-label classification problem which resulted in four possible labels. The last label we haven't introduced is *not abusive language (NAB)*.

We separated the data into training and test sets for each dataset. We allocate 80% of the data for training and 20% of the data for testing. The training data will be further divided into two parts: training data and validation data, where the validation data plays a role in testing the model's performance during training. Table 1 shows the distribution of training data and testing data for the first and second data sets, while Table 2 shows the third data sets' distribution.

Table 1. Training and Testing for Hate Speech Detection

Dataset	Training Set		Testing Set	
	HS	NHS	HS	NHS
Alfina et.al. [16]	204	366	56	87
Ibrohim and Budi [17]	4.440	6.095	1121	1513

Table 2. Training and Testing for Hate Speech and Abusive Language Detection

Dataset	Training Set				Testing Set			
	HS	NHS	AB	NAB	HS	NHS	AB	NAB
Ibrohim and Budi [17]	4.433	6.102	4.030	6.505	1.128	1.506	1.013	1.621

We conducted our model training on an Intel(R) Core(TM) i9-7900X CPU @ 3.30GHz machine with 20 logical central processing unit cores and a GeForce GTX 1080 Ti Graphical Processing Unit. The significant difference between the two datasets makes the training times much longer for the second datasets. We use the Flair framework for the implementation of our model [25].

3.2 Experiment Result

The experiment aims to determine which embeddings have the best prediction results for each dataset. Precision, recall, and F1-Measure are the three key measures we use in our experiment. The precision equation can be found on (6), recall on (7), and F1-Measure on (8). We show the performance for every class, Hate Speech (HS) and non-Hate Speech (NHS). We add two more classes for the third experiment: Abusive Language (AB) and non-Abusive Language (NAB).

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (8)$$

We use a variety of embeddings in each experiment. We also stack or combine traditional word embedding with contextual embedding. There are seven different types of embeddings in total. The first three embeddings use classical word embeddings trained on sub-word information from fastText. For the first type, we use the fastText pre-trained model on Wikipedia (*FastText Wiki*). In the second model, we use the pre-trained model on Common Crawl (*FastText crawl*). In the third model, we perform stacking between those two embeddings (*FastText wiki+crawl*).

We use contextual embeddings in the fourth and fifth experiments: *Flair* embeddings and *Pooled Flair* Embeddings. For both the flair and pooled flair embeddings, we use both forward and backward models. The embedding was pre-trained on the JW300 corpus, which contained the Bahasa Indonesia language [26]. Lastly, we stack the classical word embedding with contextual embeddings to provide a richer representation. We only use the Wikipedia model for the classical word embedding, which performed better in our experiment [24].

Table 3 shows the performance of our model in the first dataset [16]. We also shows our baseline model's performance from Alfina et al. [16], which trained on the random forest decision trees with the combination of features. These features are word unigram, word bigram, char trigram, char quadragram, and negative sentiment. One of our models achieves better performance compared to the baseline. The best model is using FastText representation trained on Wikipedia. Overall, the first dataset performed better on the classical word embedding rather thank contextual word embedding. The second-best performance is still using the same embeddings, stacked with the FastText crawl. Thus, the FastText common crawl gives no improvement into the basic FastText Wikipedia.

Table 3. The performance of the classification on the first dataset [16]. We also show the performance for the baseline model

Features and Embeddings	**Precision**			**Recall**			**F1-Measure**		
	HS	**NHS**	***avg***	**HS**	**NHS**	***avg***	**HS**	**NHS**	***avg***
Sent+Word+Char+RFDT [16]									89.8
FastText wiki	**92,5**	**92,2**	**92,3**	**87,5**	**95,4**	**91,5**	**89,9**	**93,8**	**91,9**
FastText crawl	84,9	87,8	86,3	80,3	90,8	85,6	82,6	89,3	85,9
FastText wiki+crawl	89,8	87,2	88,5	78,6	94,3	86,6	83,8	90,6	87,2
Flair	84,6	86,8	85,7	78,6	90,8	84,7	81,5	88,8	85,2
Pooled Flair	83,0	82,3	82,6	69,6	90,8	80,2	75,7	86,3	81,0
FastText Wiki + Flair	93,0	84,0	88,5	71,4	96,5	83,9	80,8	89,8	85,3
FastText Wiki + Pooled Flair	86,1	81,0	83,5	66,1	93,1	79,6	74,8	86,6	80,7

In contrast to the first dataset, the second dataset performs better on contextual word embeddings. The last fourth experiment, which uses contextual word embeddings, outperforms the classical word embeddings. However, stacking the classical word embeddings (FastText Wiki) gave a boost of performance to the Flair Embeddings. The best performance was achieved by Pooled Flair Embeddings, which give 87,3 % F1-Measures on average. We cannot benchmark the second experiment because the previous research [2] does not perform binary classification.

Table 4. The performance of the binary classification of hate speech label on the second dataset [17]

Embeddings	Precision			Recall			F1-Measure		
	HS	**NHS**	***avg***	**HS**	**NHS**	***avg***	**HS**	**NHS**	***avg***
FastText wiki	83,0	86,1	84,6	80,8	87,8	84,3	81,9	86,9	84,4
FastText crawl	82,5	82,7	82,6	75,0	88,2	81,6	78,6	85,4	82,0
FastText wiki+crawl	84,0	88,2	86,1	84,1	88,1	86,1	84,1	88,2	86,2
Flair	83,9	84,4	84,2	84,4	88,0	86,2	84,2	88,2	86,2
Pooled Flair	**85,8**	**88,9**	**87,4**	**84,9**	**89,6**	**87,3**	**85,3**	**89,2**	**87,3**
FastText Wiki + Flair	82,9	90,4	86,7	87,6	86,7	87,2	85,2	88,5	86,9
FastText Wiki + Pooled Flair	86,6	87,6	87,1	82,8	90,5	86,7	84,6	89,0	86,8

In the third experiment, we conduct multi-label text classification. The classes are either hate speech (HS) or non-hate speech (NHS) and abusive (AB), or non-abusive (NAB). Due to space limitation, we only show the average of HS and NHS and are denoted µHS. The same treatment is for AB and NAB; we only show the average of those two with µAB. Then both result gets aggregated to shows the overall average. Based on the experiment, we can show that our best model (FastText Wiki + Pooled Flair Embedding) significantly outperformed the baseline [17].

Table 5. The performance of multi-label classification on the second dataset [17]

Embeddings	Precision			Recall			F1-Measure		
	µHS	**µAB**	***avg***	**µHS**	**µAB**	***avg***	**µHS**	**µAB**	***avg***
FastText wiki	83,62	89,79	86,70	83,34	89,29	86,32	83,47	89,53	86,50
FastText crawl	83,53	89,18	86,36	82,66	87,72	85,19	82,98	88,33	85,65
FastText wiki+crawl	84,99	91,44	88,21	85,08	90,86	87,97	85,03	91,14	88,08
Flair	85,60	91,16	88,38	85,39	88,70	87,05	85,49	89,62	87,55
Pooled Flair	85,48	92,39	88,94	86,19	92,68	89,43	85,58	92,53	89,05
FastText Wiki + Flair	87,13	92,61	89,87	87,19	92,40	89,79	87,16	92,50	89,83
FastText Wiki + Pooled Flair	**86,56**	**93,11**	**89,84**	**87,07**	**93,56**	**90,31**	**86,75**	**93,33**	**90,04**

Based on the three experiments we have conducted, contextual embedding has proven to be robust in a larger dataset. Flair contextual embedding can capture the linguistic information, including subclauses [20] which can be in the form of Indonesian slang language. In the multi-label classification task, stacking classical word embedding with contextual embedding can give the best result. It consistently followed with the second-best model, which also the stack embedding. Stack Embedding able to provide the information about the global context of a word provided by fastText trained on Wikipedia, and contextual information was given by Flair embedding.

4. CONCLUSIONS

In this paper, we build the prediction model for hate speech and abusive language prediction focusing on the social media dataset. We demonstrated the usage of the word and contextual embedding approach to provide a semantical representation of the tokens. To prove the robustness of the embeddings, we did not conduct any pre-processing of the data. In other words, the model can still work well without any pre-processing using contextual embedding. Thus, we leave this gap for future work. We are also experimenting with stacking one embedding with another embedding to provide a richer representation of the sentence.

Moreover, we use the document recurrent neural network embedding to capture the sequence information from the sentence. Our model proved to improve the dataset provided by Alfina et al. [16] and a significant improvement on the larger dataset by Ibrohim and Budi [17].

In the future, we suggest using stopwords elimination, slang substitution, stemming, and other pre-processing techniques. We believe that several pre-processing methods will boost the model's efficiency due to the high noise in social media data. We also recommend testing out new transformer models, including Bidirectional Encoder Representations (BERT) and Generative Pre-trained Transformers (GPT).

ACKNOWLEDGEMENTS

This research was supported by Gadjah Mada University through capacity building for young lecturer grants. We are grateful to our colleagues from the Department of Computer Science and Electronics who provided the high-performance computing machine that was very helpful for training our model.

REFERENCES

[1] W. Warner and J. Hirschberg, "Detecting hate speech on the world wide web," in *Proceedings of the second workshop on language in social media*, 2012, pp. 19–26, [Online]. Available: https://www.aclweb.org/anthology/W12-2103/.

[2] L. S. Widayati, "Ujaran Kebencian: Batasan Pengertian dan Larangannya," *Info Singk. Kaji. Singk. terhadap isu Aktual dan Strateg.*, 2018, [Online]. Available: http://berkas.dpr.go.id/puslit/files/info_singkat/Info Singkat-X-6-II-P3DI-Maret-2018-186.pdf.

[3] J. Garland, K. Ghazi-Zahedi, J.-G. Young, L. Hébert-Dufresne, and M. Galesic, "Countering hate on social media: Large scale classification of hate and counter speech." 2020, [Online]. Available: https://arxiv.org/abs/2006.01974.

[4] E. Spertus, "Smokey : Automatic cogniti ostile Messages," 1997, [Online]. Available: https://www.aaai.org/Papers/IAAI/1997/IAAI97-209.pdf.

[5] F. Del Vigna, A. Cimino, and F. D. Orletta, "Hate me , hate me not : Hate speech detection on Facebook Hate me , hate me not : Hate speech detection on Facebook," no. May, 2017, [Online]. Available: http://ceur-ws.org/Vol-1816/paper-09.pdf.

[6] Z. Waseem and D. Hovy, "Hateful Symbols or Hateful People? Predictive Features for Hate Speech Detection on Twitter," in *Proceedings of the NAACL Student Research Workshop*, Jun. 2016, pp. 88–93, doi: 10.18653/v1/N16-2013.

[7] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. Bhamidipati, "Hate Speech Detection with Comment Embeddings," in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 29–30, doi: 10.1145/2740908.2742760.

[8] C. Nobata, J. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive Language Detection in Online User Content," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 145–153, doi: 10.1145/2872427.2883062.

[9] H. Watanabe, M. Bouazizi, and T. Ohtsuki, "Hate Speech on Twitter : A Pragmatic Approach to Collect Hateful and Offensive Expressions and Perform Hate Speech Detection," *IEEE Access*, vol. 6, pp. 13825–13835, 2018, doi: 10.1109/ACCESS.2018.2806394.

[10] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep Learning for Hate Speech Detection in Tweets," no. 2, 2017, [Online]. Available:

https://dl.acm.org/doi/abs/10.1145/3041021.3054223.

[11] T. L. Sutejo and D. P. Lestari, "Indonesia Hate Speech Detection using Deep Learning," *2018 Int. Conf. Asian Lang. Process.*, pp. 39–43, 2018, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8629154/.

[12] B. Gambäck and U. K. Sikdar, "Using Convolutional Neural Networks to Classify Hate-Speech," in *Proceedings of the First Workshop on Abusive Language Online*, Aug. 2017, pp. 85–90, doi: 10.18653/v1/W17-3013.

[13] J. H. Park and P. Fung, "One-step and Two-step Classification for Abusive Language Detection on {\{}T{\}}witter," in *Proceedings of the First Workshop on Abusive Language Online*, Aug. 2017, pp. 41–45, doi: 10.18653/v1/W17-3006.

[14] Z. Zhang, D. Robinson, and J. Tepper, "Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network," 2018, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-93417-4_48.

[15] S. Agrawal and A. Awekar, "Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms." 2018, [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-76941-7_11.

[16] I. Alfina, R. Mulia, M. I. Fanany, and Y. Ekanata, "Hate speech detection in the Indonesian language: A dataset and preliminary study," in *2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, 2017, pp. 233–238, doi: 10.1109/ICACSIS.2017.8355039.

[17] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter," in *Proceedings of the Third Workshop on Abusive Language Online*, Aug. 2019, pp. 46–57, doi: 10.18653/v1/W19-3506.

[18] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *CoRR*, vol. abs/1607.0, 2016, [Online]. Available: http://arxiv.org/abs/1607.04606.

[19] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov, "Learning Word Vectors for 157 Languages," 2018, [Online]. Available: https://arxiv.org/abs/1802.06893.

[20] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual String Embeddings for Sequence Labeling," in *Proceedings of the 27th International Conference on Computational Linguistics*, Aug. 2018, pp. 1638–1649, [Online]. Available: https://www.aclweb.org/anthology/C18-1139.

[21] A. Akbik, T. Bergmann, and R. Vollgraf, "Pooled Contextualized Embeddings for Named Entity Recognition," in *Proceedings of the 2019 Conference of the North {A}merican Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jun. 2019, pp. 724–728, doi: 10.18653/v1/N19-1078.

[22] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing*, 2015, pp. 1422–1432, [Online]. Available: https://www.aclweb.org/anthology/D15-1167.pdf.

[23] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv Prepr. arXiv1412.3555*, 2014, [Online]. Available: https://arxiv.org/abs/1412.3555.

[24] G. B. Herwanto, A. M. Ningtyas, K. E. Nugraha, and I. N. P. Trisna, "Hate speech and abusive language classification using fastText," in *2019 International Seminar on*

Research of Information Technology and Intelligent Systems (ISRITI), 2019, pp. 69–72, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9034560/.

[25] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "FLAIR: An easy-to-use framework for state-of-the-art NLP," *NAACL HLT 2019 - 2019 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. - Proc. Demonstr. Sess.*, pp. 54–59, 2019, [Online]. Available: https://www.aclweb.org/anthology/N19-4010.pdf.

[26] Ž. Agić and I. Vulić, "JW300: A Wide-Coverage Parallel Corpus for Low-Resource Languages," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Jul. 2019, pp. 3204–3210, doi: 10.18653/v1/P19-1310.