Hindawi
Mobile Information Systems
Volume 2021, Article ID 6654029, 9 pages
https://doi.org/10.1155/2021/6654029

Research Article

Method of Profanity Detection Using Word Embedding and LSTM

MoungHo Yi,[1] MyungJin Lim,[2] Hoon Ko,[3] and JuHyun Shin [4]

[1]*Department of Software Convergence Engineering, Chosun University, 309 Pilmun-Daero, Dong-Gu, Gwangju 61452, Republic of Korea*
[2]*Department of Computer Engineering, Chosun University, 309 Pilmun-Daero, Dong-Gu, Gwangju 61452, Republic of Korea*
[3]*IT Research Institute, Chosun University, 309 Pilmun-Daero, Dong-Gu, Gwangju 61452, Republic of Korea*
[4]*Department of New Industry Convergence, Chosun University, 309 Pilmun-Daero, Dong-Gu, Gwangju 61452, Republic of Korea*

Correspondence should be addressed to JuHyun Shin; jhshinkr@chosun.ac.kr

Received 25 November 2020; Revised 22 December 2020; Accepted 10 February 2021; Published 25 February 2021

Academic Editor: Elio Masciari

With the rising number of Internet users, there has been a rapid increase in cyberbullying. Among the types of cyberbullying, verbal abuse is emerging as the most serious problem, for preventing which profanity is being identified and blocked. However, users employ words cleverly to avoid blocking. With the existing profanity discrimination methods, deliberate typos and profanity using special characters can be discriminated with high accuracy. However, as they cannot grasp the meaning of the words and the flow of sentences, standard words such as "Sibaljeom (starting point, a Korean word that sounds similar to a swear word)" and "Saekkibalgalag (little toe, a Korean word that sounds similar to another swear word)" are less accurately discriminated. Therefore, in order to solve this problem, this study proposes a method of discriminating profanity using a deep learning model that can grasp the meaning and context of words after separating Hangul into the onset, nucleus, and coda.

1. Introduction

Cyberbullying refers to a criminal act that inflicts mental and material harm on others by the repeated use of hostile expressions, such as text, images, and voices, online through IT devices including computers [1]. Of late, the age of cyberbullying perpetrators is gradually decreasing due to the proliferation of smart devices. Such perpetrators do not realize that cyberbullying is a crime and exhibit reckless behavior considering it as simple play [2]. In addition, there are cases of suicide by the victims of cyberbullying, rendering it a serious social problem [3]. According to the Communications Commission's 2019 Cyberbullying Investigation Results Presentation, the representative types of cyberbullying include cyber verbal abuse, cyber defamation, cyberstalking, cyber sexual harassment, personal information leakage, social exclusion online, cyber extortion, and cyber coercion [4]. Among them, cyber verbal abuse has the highest ratio of 36.7% [5–7]. In order to prevent cyber verbal abuse, the government has blocked certain sentences related to profanity [8]. However, Internet users are modifying the nucleus of a word to avoid blockage or using new profanity [9], even if profanity is modified accordingly. Many studies are being conducted to accurately determine profanity. Currently, deliberate typos and profanity using special characters can be discriminated. However, if the meaning of words and the flow of sentences are not understood, standard words such as "Sibaljeom (starting point, a Korean word that sounds similar to a swear word)" and "Saekkibalgalag (little toe, a Korean word that sounds similar to another swear word)" are less accurately determined as profane. Therefore, in this study, we attempt to accurately identify profanity in the following two cases: (1) when Internet users use profanity by modifying the nucleus of a word to avoid being blocked as profane and (2) when the morphology of a word is similar to profanity but is a standard language that is not profane in context. In order to improve the accuracy of profanity discrimination, we propose a method for discriminating profanity using the FastText model for word embedding by learning the meaning and form information of words and the LSTM model for learning the flow of context.

2. Related Work

2.1. Natural Language Processing. Natural language refers to a language that has formed and evolved naturally over a long period, such as Korean and English, and is commonly spoken and used [10]. Natural language processing analyzes the meaning of natural language to enable computers to process the language. Natural language processing is applied in areas such as text classification, sentiment analysis, summarization, and text clustering [11, 12]. This processing includes four steps: text collection, text preprocessing, word embedding, and machine learning modelling [13, 14].

In the first step (text collection), the texts to be processed are collected. The second step (text preprocessing) involves the standardization of unstructured texts to increase the accuracy of natural language processing. The text collected from SNS contains many elements that are difficult to analyze, such as typos, emoticons, abbreviations, and newly coined words. Most are expressed as if speaking in a carefree manner, in terms of the vocabulary or the structural order of the sentence. Therefore, after text processing, including changing the uppercase to lowercase, deleting special characters and emoticons, and text normalization such as word tokenization and stop word removal, preprocessing is performed according to the requirement. In the third step (word embedding), the words are converted to vectors such that computers can understand and process the natural language efficiently. In this study, we use the FastText model in which word embedding is performed by training using the morphology information of Korean letters, among several models. Finally, in the machine learning modelling stage, a supervised learning model is established, and training and prediction are performed using vectorized number-type data. In this study, we use the LSTM model for training and prediction, for profanity detection.

2.2. Embedding Algorithm. For a computer to understand and process natural language efficiently, the language needs to be transformed into numbers that can be processed by the computer. As the performance of natural language processing varies considerably depending on word representation, the conversion of words into number types is being extensively studied. Among these, the commonly used method is word embedding, which represents a word as a dense vector [15]. Methods of representing a word by a vector include sparse representation and dense representation. One-hot encoding is a method of representing a vector with sparse representation. The vector value expressed in spare representation mostly includes the number "0," and the number of dimensions is equal to the number of words to be trained. However, as sparse representation includes as many dimensions as the number of words for training, considerable space is wasted and the meaning of words cannot be appropriately represented. Of late, several algorithms that represent vectors through dense representation by improving the above disadvantages have emerged [16]. With dense representation, the vector dimension can be matched with the numbers set by the user, and the vectors, called dense vectors, have real values [16]. As indicated by the concept, word embedding refers to a method of representing words in the form of dense vectors [17]. Representative algorithm models that can adopt word embedding include Word2Vec [15], GloVe [18], and FastText. The Word2Vec model represents words in the vector space through distributed representation using their semantics and syntactic characteristics. However, this model is disadvantageous because the vector values cannot be obtained for OOV and training is not possible for infrequent words [19–21]. In the FastText model, training is performed by dividing words into character-level to supplement such limitations. As in the case of the Word2Vec model, the FastText model examines the preceding and subsequent contexts with reference to the target word and performs training on words; however, because it also learns words by dividing them to character-level, the model can also be trained on word morphology information. FastText model training is performed by representing words using the Bag-of-Words (BoW) or n-gram model. At the beginning and end of the words to be learned, "<", ">" is inserted as separators, and the entire word is also contained in the BoW with the separator to enable the model to learn the overall meaning [22]. A new scoring function using the BoW is defined as follows:

$$s(w, c) = \sum_{g \in g_w} z_g^T v_c, \tag{1}$$

where v_c is the vector of a word forming context and z_g^T is the vector corresponding to 1 BoW.

With the above change in the model, it can learn information on the semantics, syntax, and morphology of the target words [23]. In this study, for training the FastText model more effectively on the morphology information suited to the characteristics of Korean letters, each word is divided into onset, nucleus, and coda for analysis. In case of the absence of coda, symbol "–" is added.

2.3. LSTM Model. The LSTM model is a model devised to solve the long-term dependency problem of RNN. LSTM uses switchgear designed to allow you to forget previous information or store information for a long period of time through a memory cell, which is an internal node:

$$\begin{aligned}
\widetilde{c}_t &= \tanh\left(x_t U^g + h_{t-1} W^g + b_c\right), \\
c_t &= c_{t-1} \circ \mathrm{f_t} + \widetilde{\mathrm{c}}_\mathrm{t} \circ \mathrm{i_t}, \\
h_t &= \tanh\left(c_t\right) \circ \mathrm{o_t}, \\
i_t &= \sigma\left(x_t U^i + h_{t-1} W^i + b_i\right), \\
f_t &= \sigma\left(x_t U^f + h_{t-1} W^f + b_f\right), \\
o_t &= \sigma\left(x_t U^o + h_{t-1} W^o + b_o\right).
\end{aligned} \tag{2}$$

The combination of the input value (x_t) at the same time point and the hidden node value (h_{t-1}) at the previous time point is used to calculate the internal storage node candidate ($\widetilde{c}_t$), and the storage node candidate is combined with the storage node value (c_{t-1}) at the previous time point and

calculate the current value (c_t) of the internal storage node. At this time, the input gate (i_t) and forget gate (f_t) act as weights to adjust how the new information is passed and how the information is passed to the value in the previous state. Finally, using the tanh valid function, the value of the hidden node is output by adjusting how to pass the value of the current internal storage node by output gate (o_t). The value of each input gate, output gate, and forget gate is represented by a combination of a linear function of the input value at the current time point and the secret node value at the previous time point [24].

3. Profanity Detection with Word Embedding and LSTM

3.1. System Architecture. Figure 1 illustrates the system architecture of the proposed profanity detection method, which is divided into the Training Data Process that trains data, and the Testing Data Process that detects profanity in the texts written by users.

In Training Data Process, a model is trained for profanity detection. This process involves preprocessing, FastText model training, and LSTM model training. In the text preprocessing step, normalization is performed on the collected texts, and in order to include the morphology information of Hangul in the training of the FastText model, each character is divided into the onset, nucleus, and coda. Therefore, the FastText model is trained using such divided text data. The trained FastText model contains the vector information of the text data, and word embedding is performed on the text data using the vector information. In the LSTM model learning step, binary classification supervised learning is performed to detect whether a sentence contains profanity. While training the LSTM model, the presence of profanity in a sentence is first checked, and labeling is performed, where "1" is assigned to a sentence with profanity and "0" is assigned to a sentence without profanity. Further, training is performed using the numerical data with word embedding.

In the Testing Data Process, when a sentence is input, the model detects the presence of profanity and provides the output. For result prediction with the trained model, the data format should be the same as that used for training. Therefore, the same text preprocessing step performed in the Training Data Process is repeated, and through the word embedding step, profanity is predicted using the LSTM model. In the predicted result, "0" indicates a sentence without profanity and "1" indicates a sentence with profanity.

3.2. Preprocessing and Word Embedding. This section describes in detail the (1) preprocessing step that facilitates computerized analysis on texts collected from Twitter and Naver movie review posts and the (2) word embedding step using the FastText model.

3.2.1. Text Preprocessing. First, remove special symbols and emoticons. Text data were collected from Twitter posts and Naver movie reviews. Many of the collected data began with special characters, emoticons, "kieukkieukkieukkieuk (LOL)," and "@," and such texts were removed as they hinder profanity detection.

Second, remove one syllable. With reference to word spacing, the collected sentences contained numerous one-syllable words such as "geos (thing)," "a (Ah)," "geu (that)," "tto (again)," and "i (this)." On examining these one-syllable words, it was determined that, in many cases, these words were typed due to mistakes in word spacing when the user was writing the posts; furthermore, there was no profanity comprising one-syllable. Considering these aspects, one-syllable words were excluded from the analysis.

Third, remove sentences with less than five words. The texts collected from Twitter included many sentences that did not have proper spacing between words, such as sentence "nunapaije lenjeuppaenda (I am taking the lens out because my eyes are sore, written without spacing)." Preprocessing can be performed only by knowing the rules of the sentences that do not have proper spacing. Investigation of the rules of sentences revealed that there were fewer words in these sentences, compared to normal sentences when the number of words was counted with reference to spacing. In addition, for training the LSTM model, the number of words in all the sentences needs to be adjusted; sentences with less number of words need to be filled with "0," resulting in sentences with many "0"s. Therefore, sentences with less than five words were removed through preprocessing. Thereby, high accuracy was realized in profanity detection.

Fourth, the onset, nucleus, and coda are separated. In this study, the FastText model was trained after separating Hangul into the onset, nucleus, and coda to enable profanity detection, even when the user deliberately makes a typo and changes the morphology of the profanity. Here, the absence of coda was indicated by adding "-." Table 1 presents the source code that separates the onset, nucleus, and coda.

3.2.2. Word Embedding Using the FastText Model. The preprocessed text data were used for training the FastText model. This training was performed with skip-gram, at a learning rate of 0.05, dimension (vector space) of 100, window size of 5, epoch of 50, and n-gram of 1–6. As shown in Table 2, in the training result, the character vectors substrings are formed with n-gram and the character vectors strings are formed by adding the character vector substrings.

In the FastText model, character vector strings are formed by adding the substrings of the character vectors; similarly, the morphology information is used for training. The FastText model trained based on this method generates a vector similar to the word before it was changed, even if the word nucleus is changed.

3.3. Profanity Detection with the LSTM

3.3.1. Data Set Configuration for LSTM Training. To conduct supervised learning with the LSTM model, word embedding in which data are mapped to vectors must be performed, for which the trained FastText model can be used. As shown in Table 3, a sentence consists of several

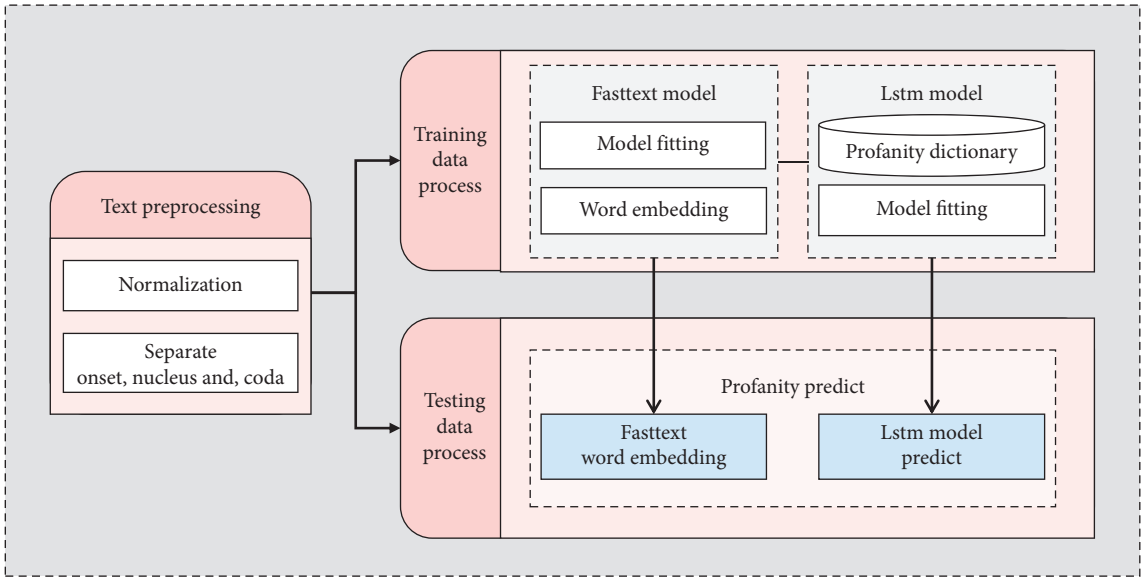

Figure 1: Overview of the system architecture.

Table 1: Separation of the onset, nucleus, and coda.

```
:>>> def run (x)
>>>    consonant_ord_list = [ord (char) for char in Korean consonants]
>>>    choseong_list = [char for char in Korean consonants]
>>>    jungseong_list = [char for char in Korean vowels]
>>>    jongseong_list = [char for char in Korean consonants and double consonants]
>>>    result = []
>>>    for char in s:
>>>       if ord (char) = = 32:
>>>          result.append (char)
>>>       elif 48 ≤ ord (char)≤57:
>>>          result.append (char)
>>>       elif consonant_list.count (char) = = 0:
>>>          character_code = ord (char)
>>>          if (55203 < character_code or character_code <44032):
>>>             continue
>>>          code = 44032
>>>          choseong_index = (character_code−code)//21//28
>>>          jungseong_index = (character_code−code−(choseong_index * 21 * 28))//28.
>>>          jongseong_index = character_code−code−(choseong_index * 21 * 28)−(jungseong_index∗28)
>>>          result.append (choseong_list [choseong_index])
>>>          result.append (jungseong_list [jungseong_index])
>>>          result.append (jongseong_list[jongseong_index])
>>>       else:
>>>          choseong_index = consonant_ord_list.index(ord(char))
>>>          result.append(choseong_list[choseong_index])
>>>          result.append ("−")
>>>          result.append ("−")
>>>       return "" .join (result)
```

Table 2: FastText model vector training result [25].

Character vector substring	$n = 2$	<ieung = [0.121, −0.187, 0.002, . . ., −0.028] ieung ah = [−0.071, −0.158, 0.101, . . ., −0.004]. . .
	$n = 3$	<ieung ah = [−0.095, 0.189, 0.082, . . ., 0.113] ieung ah rieul = [0.108, −0.158, 0.462, . . ., −0.054]. . .
	$n = 4$	<ieung ah rieul = [0.003, −0.139, 0.372, . . ., 0.319] ieung ah rieul rieul = [0.091, 0.133, 0.421, . . ., 0.104]. . .
Character vector string	alladin = <ieung + ieung ah + . . . + ieung ah rieul + . . . + digeut ee nieun> = [0.234, −0.322, 0.401, . . ., 0.159]	

Table 3: Data word embedding [25].

Entire data	['ieung ah rieul rieul ah-digeut ee nieun ieung yuh ieung hieut wah–ssang-giyok oo rieul jieut ae mieum', 'bieup oo tieut ieung ee–giyok oh-nieun ah–nieun ee–hieut uh–hieut ah nieun siot oh giyok ieung ee–ieung oo rieul rieul uh ieung', 'chieut oh–ieung ee rieul-giyok giyok ee–bieup ah–rieul oh–siot ee–jieut ah giyok'. . .]
One-word vector (1D)	[0.234, −0.322, 0.401, . . ., 0.159]
One-sentence vector (2D)	[[0, . . ., 0], [0, . . ., 0], [0.234, . . ., 0.159], . . ., [0.532, . . ., 0.216]]
Vector of all the sentences (3D)	[[[0.234, . . ., 0.159], [0.532, . . ., 0.216], . . ., [0.032, . . ., 0.659]], [[0.175, . . ., 0.011], [−0.431, . . ., 0.382], . . ., [0.233, . . ., 0.301]], [[0, . . ., 0], . . ., [0.335, . . ., 0.053], [0.092, . . ., −0.382]]]

words based on the spacing, and a word becomes a one-dimensional vector when vectorized using the trained FastText model.

In this case, the number of vectors for one word is set as the dimension value when training the FastText model. Therefore, as the vector of a sentence is a combination of several words, this vector is two-dimensional, and the vector of all the sentences is three-dimensional. In the word embedding algorithm, the number of words in a sentence is matched with the number of vectors. When the number of vectors is different, "0"s are prepended to fill the number of lacking words, for matching with the number of vectors. In addition, for supervised learning of the LSTM model, each data requires a label. In this study, if a sentence contained profanity, the label was set as "1," and if there was no profanity, the label was set as "0".

3.3.2. LSTM Training Method. To train the LSTM model, a 3D vector was set as X, and a label comprising "0" and "1" was set as y. For the LSTM hyperparameters, as shown in Figure 2 and Table 4, the units were set to "1," the time step was set to 25, which is the number of words, and the feature was set to 100, which is the number of dimensions used for training FastText. For the hyperparameters of Dense, the units were set to "1," and the Sigmoid function was set as the activation function.

3.3.3. Profanity Detection. The following tables demonstrate that the profanity detection results are accurate. Table 5 shows the results of sentences with and without profanity. In Table 6, even for cases where the nucleus of the word "Ssibal (f**k)" is transformed to words such as "Ssuibal" or "Ssuibal-namah," the results show correct profanity detection.

Finally, if a standard word and not profanity is used in context, the results accurately display the message, "Profanity is not contained." Table 7 shows the standard word detection result for "Sibaljeom (starting point)" and "Saekkibalgalag (little toe)."

4. Results and Discussion

In this section, the process of collecting texts and data sets used for testing is described. For demonstrating the effectiveness of the profanity detection method proposed in this study and for comparative analysis with a prior study, the accuracy, recall, and precision were measured for performance assessment utilizing the classification performance assessment indicators.

4.1. Data Collection. For the application and performance assessment of the profanity detection method proposed in this study, SNS posts and Naver movie review data were collected. SNS data include features written freely without being restrained by any specific format; among the various types of SNS, Twitter, which has more pronounced free writing, was used for data collection. The data collected from Twitter were searched with specific keywords and the search period was set from January 01, 2016. The number of sentences collected from Twitter was approximately 4.4 million. Naver movie review data were relatively refined; therefore, approximately 500,000 sentences were collected.

4.2. Data Sets. The data sets used for testing the models are described here. The number of data sets collected from Twitter was approximately 4.4 million, as shown in Table 8; however, as they contained many duplicate tweets, such duplicates were removed. As a result, the number of Twitter posts decreased to approximately 1.3 million.

Due to the characteristics of the FastText model, it can learn the meaning and morphology information of words more accurately as the number of data increases. Hence, 1.3 million tweets and 500,000 Naver movie reviews were used as data to train the FastText model. The data set for training the LSTM model did not contain all the collected data, and for accurate profanity detection, the data sets were composed with a ratio of 1 : 1 between the number of sentences with profanity and the number of sentences without. Thereby, the composed data sets included 600,000 sentences, each, with and without profanity.

4.3. Test Assessment and Analysis

4.3.1. Test Assessment Method. To evaluate the accuracy, recall, and precision of profanity detection and to determine the superiority of the proposed method, the edit distance algorithm proposed in a previous study and the CNN, GRU, and LSTM models used in this study were comparatively analyzed. Equation (3) was used for calculating the accuracy, recall, and precision using the confusion matrix:

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}, \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}. \end{aligned} \tag{3}$$

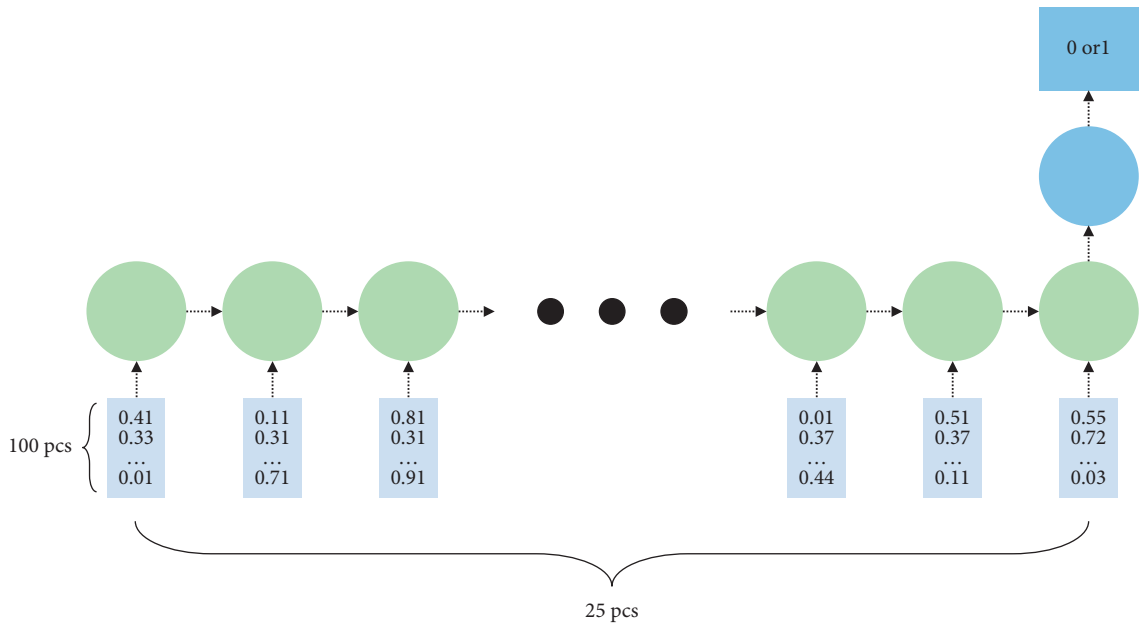

Figure 2: LSTM model.

Table 4: LSTM model.

el = Sequential ()

>>> mod
>>> model.add (LSTM (units = 1, input_shape = (25, 100)))
>>> model.add (Dense (1, activation = 'sigmoid'))
>>> model.compile (loss = 'binary_crossentropy',optimizer = 'RMSprop',metrics = ['accuracy']).
>>> model.fit (X_train, y_train, epochs = 2, validation_data = (X_test, y_test))

Table 5: Profanity detection results.

Txt	Profanity detection
Ya jigeum il-eonassneunde neomu jollyeo.	*X*
Ya ssibal jigeum il-eonassneunde neomu jollyeo.	*O*

※ *O*: detected as profane; *X*: detected as not profane.

Table 6: Profanity detection result with the transformed nucleus.

Txt	Profanity detection
Ya ssuibalnomah jigeum il-eonassneunde neomu jollyeo.	O
Ssibal	O
Ssuibal	O

Table 7: Standard word detection results for 'Sibaljeom' and 'Saekkibalgalag.'

Txt	Profanity detection	Txt	Profanity detection
Sibaljeom	*X*	Saekki	*O*
Yeogiga sibaljeom-iji	*X*	Saekkibalgalag	*X*
Yeogiga sibal	*O*		

Table 8: Data sets.

Search keywords	Number of posts	Search keywords	Number of posts	Search keywords	Number of posts
Gaesaekki (son of a b***h)	853,431	Jotna (f***king)	560,440	Sibal (f**k)	359,837
Byeongsin (dickhead)	211,181	Jonna (f***king)	275,631	Annyeong (hello)	285,554
Saekki (bastard)	387,500	Ssibal (f**k)	532,741	Eumsik (food)	169,428

Table 9: Comparison of the CNN, GRU, and LSTM model results.

	CNN	GRU	LSTM (%)
Accuracy	53%	94.74%	96.15
Recall	13%	95.18%	96.29
Precision	79%	94.32%	96

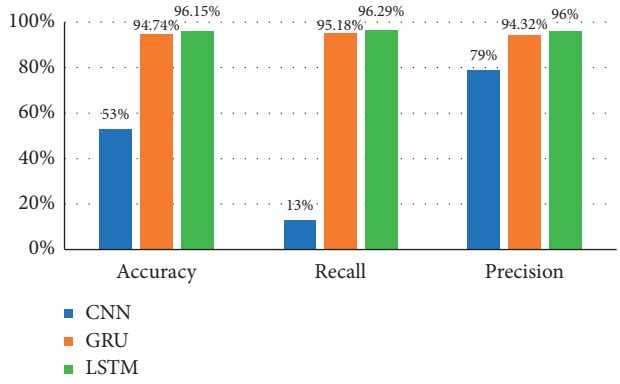

Figure 3: CNN, GRU, and LSTM model results.

Table 10: Profanity detection result comparison.

Category	Word	Edit distance algorithm	CNN	GRU	Proposed method LSTM
Transformed profanity	Ssuibal	O	O	O	O
	Junna	O	O	O	O
	Yabal	O	X	O	O
	Sseuba	O	X	O	O
Standard word	Sinbal	O	O	X	X
	Sibaljeom	O	O	X	X
	Saekkibalgalag	O	O	O	X

※ O: detected as profane; X: detected as not profane.

4.3.2. Test Result Analysis. In this study, the correct answer for a sentence containing profanity was set as "1," and for a sentence without profanity, it was set as "0"; thus in the test, Positive = 1 and Negative = 0. Table 9 and Figure 3 outline the analysis results. For prediction using the CNN model, the accuracy is 53%, recall is 13%, and precision is 79%. It can be observed that the overall accuracy and recall of the CNN model for profanity detection are considerably lower compared to other models. The GRU model shows high performance with an accuracy of 94.74%, recall of 95.18%, and a precision of 94.32% but is still slightly inferior to the LSTM model. The LSTM model proposed in this study has the highest performance with an accuracy of 96.15%, recall of 96.29%, and precision of 96%. The cross-verification score using the LSTM model is 96%.

Table 10 shows the results of the comparative analysis between the edit distance algorithm proposed in a previous study, and the CNN, LSTM, and GRU models tested in this study, demonstrating the superior performance of the proposed method.

Transformed profanities such as "Ssuibal" and "Junna" and "Yabal" and "Sseubal" are accurately detected by the previously proposed algorithm as well as the proposed method in this study. Words such as "Sinbal," "Sibaljeom," and "Saekkibalgalag" have forms similar to profanities "Sibal" and "Saekki (bastard)"; hence, they are detected as profane by the previously proposed algorithm, but the LSTM model proposed in this study correctly detects it is not profane because it has been trained on the flow of context.

5. Conclusions

In this study, we proposed a profanity detection method using word embedding and the LSTM model. The proposed method divides the text for training into the onset, nucleus, and coda. Further, it considers not only the semantics of the words but also the morphology information using the FastText model. Moreover, by training on the flow of context using the LSTM model, it can detect profanity that cannot be detected by the methodologies proposed in previous studies. Among the 40,005 sentences with profanity and 40,254 sentences without profanity, 40,126 sentences were predicted as profane and 40,133 sentences were predicted as not profane by the proposed method. According to the classification performance test indicators, the accuracy was 96.15%, recall rate was 96.29%, and precision was 96%, indicating high performance. The result of comparative analysis between the edit distance algorithm proposed in a previous study and the proposed method in this study confirmed that the proposed method was capable of more accurate profanity detection.

For follow-up research, it is necessary to add a more diverse range of profanities, in addition to the seven profanities "Sibal (F**k)," "Ssibal (F**k)," "Byeongsin (dickhead)," "Jonna (f***king)," "Jotna (f***king)," "Gaesaekki (son of a b***h)," and "Saekki (bastard)" used in this study, in order to find a method that can detect all the profanities with improved accuracy. Furthermore, an extended study needs to be performed that can detect profanity considering emoticons and special characters with meanings as well as social issues. If immense text data can be collected, all the profanities can be accurately detected through the proposed method. Furthermore, cyber verbal abuse can be expected to decrease by the introduction of a profanity blocking system based on the findings of this study.

Data Availability

The data used to support the findings of this study have been deposited in the NAS (Network Attached Storage) repository (http://gofile.me/5RqDr/b85bbrzd0).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT (MSIT)) (no. 2019R1F1A1057325) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (no. 2017R1A6A1A03015496).

References

[1] "Cyberbullying–wikipedia", wikipedia. accessed Mar 19, 2020, https://en.wikipedia.org/wiki/Cyberbullying.

[2] H. Jeong, *Cases of Domestic and Overseas Cyber Violence and Countermeasures of Each Country*, pp. 31–47, Internet & Security Focus, Japan, 2013, https://www.kisa.or.kr/uploadfile/201311/201311081657227967.pdf.

[3] "Students who ended their lives in 'cyber violence'... The perpetrator who mocked even death.", SBS News. accessed Mar 21, 2020, https://news.sbs.co.kr/news/endPage.do?news_id=N1004932661%20&plink=COPYPASTE&cooper=SBSNEWSEND.

[4] H. Jeong, "A study on the actual condition and countermeasure of cyber violence in school environment," *Korean Juvenile Protection Review*, vol. 20, pp. 205–241, 2012.

[5] Korea Communications Commission, *Announcement of 2019 Cyber Violence Survey Results*, Korea Communications Commission, South Korea, 2020.

[6] "From 'data shuttle' to 'kakao-talk prison'... A society bruised by 'cyber violence'" MoneyToday. accessed May 23, 2020, https://news.mt.co.kr/mtview.php?no=2019051316474551634.

[7] "Are you crazy because you didn't bother me these days? Endless SNS violence, no exit." Asian economy. accessed Mar 22, 2020, https://www.asiae.co.kr/article/2018091316082814018.

[8] "Blocking 'Kakao-talk' profanity." Women's newspaper. accessed Apr 2, 2020, https://www.women news.co.kr/news/articleView.html?idxno=55721.

[9] "The meaning and meaning of 'Yabal' drip." Naver blog. accessed Apr 2, 2020, https://m.blog.naver.com/PostView.nhn?blogId=rara4000&logNo=221648276145&proxyReferer=https:%2F%2Fwww.google.com%2F.

[10] "Natural language processing." Namu wiki. accessed Mar 25, 2020, https://namu.wiki/w/jayeon%20eon-eo%20cheoli.

[11] "What is natural language processing? How do you use it for your business?", CIO. accessed Mar 26, 2020, http://www.ciokorea.com/news/37457.

[12] "[Special report] what is natural language processing (NLP)... The technology and market.", aritificial intelligence times. accessed Mar 26, 2020, http://www.aitimes.kr/news/articleView.html?idxno=15036.

[13] B. Cho, *A Comparative Study on Requirements Analysis Techniques Using Natural Language Processing and Machine Learning*, M.S. thesis, Ajou University, Suwon, South Korea, 2020.

[14] "Natural Language Processing", MonkeyLearn. Accessed Mar 26, 2020, https://monkeylearn.com/natural-language-processing/.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient Estimation of Word Representations in Vector Space*, 2013, http://arxiv.org/abs/1301.3781.

[16] T. Young, D. Hazarika, S. Poria, and E. Cambria, *Recent Trends in Deep Learning Based Natural Language Processing*, 2017, http://arxiv.org/abs/1708.02709.

[17] "Introduction to natural language processing using deep learning.", Wikidocs. accessed Apr 2, 2020, https://wikidocs.net/33520.

[18] J. Pennington, R. Socher, and D. Christopher, "Global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October 2014.

[19] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2017, http://arxiv.org/abs/1607.04606.

[20] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, http://arxiv.org/abs/1607.01759.

[21] A. Joulin, E. Grave, P. Bojanowski et al., "Compressing text classification models," 2016, http://arxiv.org/abs/1612.03651.

[22] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and J. Armand, "Advances in pre-training distributed word representations," 2017, http://arxiv.org/abs/1712.09405.

[23] H. Jo and S.-goo Lee, *Korean Word Embedding Using FastText*, pp. 705–707, The Korean Institute of Information Scientists and Engineers, Seoul, South Korea, 2017.

[24] S.M. Ahn, Y. Chung, J. Lee, and J. Yang, *Korean Sentence Generation Using Phoneme-Level LSTM Language Model*, pp. 71–88, The Korea Intelligent Information Systems Society, Seoul, South Korea, 2017.

[25] "Apperance and Sound of Hangul", Korean study. accessed Mar 2, 2020, https://www.cinecafe.kr/?mid=Reference&document_srl=322418&listStyle=list.