Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 1958149, 9 pages
https://doi.org/10.1155/2020/1958149

Research Article

Validation of Text Data Preprocessing Using a Neural Network Model

HoSung Woo,[1] **JaMee Kim**,[2] **and WonGyu Lee**[3]

[1]*Department of Computer Science and Engineering, Graduate School, Korea University, Seoul 02841, Republic of Korea*
[2]*Major of Computer Science Education, Graduate School of Education, Korea University, Seoul 02841, Republic of Korea*
[3]*Department of Computer Science and Engineering, College of Informatics, Korea University, Seoul 02841, Republic of Korea*

Correspondence should be addressed to WonGyu Lee; lee@inc.korea.ac.kr

Received 24 March 2020; Accepted 1 May 2020; Published 14 May 2020

Guest Editor: Sanghyuk Lee

Copyright © 2020 HoSung Woo et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Many artificial intelligence studies focus on designing new neural network models or optimizing hyperparameters to improve model accuracy. To develop a reliable model, appropriate data are required, and data preprocessing is an essential part of acquiring the data. Although various studies regard data preprocessing as part of the data exploration process, those studies lack awareness about the need for separate technologies and solutions for preprocessing. Therefore, this study evaluated combinations of preprocessing types in a text-processing neural network model. Better performance was observed when two preprocessing types were used than when three or more preprocessing types were used for data purification. More specifically, using lemmatization and punctuation splitting together, lemmatization and lowering together, and lowering and punctuation splitting together showed positive effects on accuracy. This study is significant because the results allow better decisions to be made about the selection of the preprocessing types in various research fields, including neural network research.

1. Introduction

Recently, attempts have been made to increase work efficiency through studies using similarities between sentences. Examples include document classification, plagiarism detection, document summarization, paraphrasing, and automatic question-and-answer systems using a similarity measurement model between sentences [1].

Studying the similarity between sentences requires a deep understanding of the semantic and structural information of the language. Previous studies have extracted and used features in sentences [2], but the feature-extraction process was complicated, and the performance was irregular depending on the extracted features. Therefore, attempts have been made to learn a language model that computes probability distributions without extracting features. The linguistic model, which was based on statistical theory, used the conditional probability of a single word (unigram) or a sequence of multiple words (n-gram). In addition, a method has been proposed that combines a word-embedding method, in which information about the meaning or structure of a word is expressed in terms of a real-time multidimensional vector and a deep belief network structure that uses a prelearning method [3].

To improve the prediction accuracy of a high-performance neural-network-based sentence model or a natural-language-based study, confidence in the data should be the highest priority. The dataset of the public database is already purified. Data for research studies should be processed through a filtering step, in which the researcher himself conducts the preprocessing. Therefore, it is necessary to investigate the data preprocessing features that should be selected for machine learning [4] as well as the effects of various preprocessing tasks on the performance of classification models [5–7].

Data integration, refinement, reduction, discretization, feature selection, and data conversion can be used for data preprocessing. Recently, studies have been conducted in which various evaluation methods or preprocessing steps are performed automatically to select appropriate data features

[8]. However, most studies on machine learning to date do not include data preprocessing [9–14]. Furthermore, even in the studies where preprocessing was mentioned, only some parts of various processes, such as word normalization and elimination, were presented [15, 16].

The purpose of this study is to analyze the effect of text data preprocessing on the sentence model. If previous studies were aimed at improving the performance of the model through preprocessing, this study focuses on the effect of combinations of data preprocessing types on performance. Various preprocessing methods were compared and analyzed, such as the setting method for using the preprocessing technique or performance analysis where learning is performed according to the order of complexity of sentences.

The Materials and Methods section describes the different types of text data preprocessing. It also describes the research methods used and explains the sentence model, preprocessing type, and dataset. In the Results and Discussion section, the results of using combinations of different preprocessing types are discussed, and finally, the conclusions of this study are presented in the last section.

2. Materials and Methods

2.1. Text Data Preprocessing. The quality of the data plays an important role in the performance of the algorithm. If data are not preprocessed, the algorithm may behave unexpectedly due to inconsistent data, and performance may be affected.

Existing data preprocessing studies have been mainly conducted in the field of data mining. There have been studies that process web data to format them into an analytical form. These studies did not explain the effect of data preprocessing on the algorithm as a method included in the process of preparing data for analysis [17–19]. There is also a study that analyzed the effect of data preprocessing on predictive ability, limited to numerical data in neural network models [4, 20]. Feature selection, outlier data removal, dimension reduction, etc., were conducted; however, it is difficult to understand their effects on text data.

The sentence model uses word-based text data that include plural words, special characters, and numerals. Therefore, preprocessing for analysis is divided into transformation, in which the original form is transformed to a word-based form, and elimination, in which the words that are considered unnecessary for semantic interpretation are eliminated. The text preprocessing technique is shown in Table 1.

There are three types of normalization: lowering, which converts uppercase letters to lowercase letters, stemming, and lemmatization.

Stemming, which is a normalization technique that reduces the complexity of data, removes affixes and separates stems from words with modified word forms. Lemmatization is a technique that converts words used in various forms into dictionary forms [21, 22]. Table 2 compares stemming and lemmatization techniques.

In stemming, words with different roots are mapped to the same stem. Therefore, it is mainly used in search engines. Lemmatization extracts the original form of a word as the word is converted to a basic form. Therefore, lemmatization does not change the meaning of words [23–25].

As an example of punctuation, the method of removing "-" from the word "brute-force" and obtaining the two words "brute" and "force" is called splitting. Furthermore, if you get the word "bruteforce," it is called merging. Splitting is the same as tokenization, which divides sentences into words.

Elimination assumes that all words that make up a sentence, paragraph, or document do not have the same significance. In other words, according to this method, a word with a low frequency of occurrence or a word with a high frequency of occurrence in a document but with low semantic information, such as a stop word, a one-syllable character or a special character, is deleted.

2.2. Research Methods. This study was conducted to analyze the effects of data preprocessing on sentence models and not to examine fine-tuning or performance improvement. This section describes the setting of the preprocessing study, structure of the sentence model, and datasets used in the study. The procedure used in this study is shown in Figure 1.

This study, which aims to analyze the performance of combining types of text data preprocessing in a sentence model, can be divided into a typical preprocessing type and a preprocessing type that is developed according to the needs of the study. In the typical data preprocessing step, lowering, lemmatization, punctuation splitting and merging, and special character elimination were used by considering the preservation and accuracy of the part-of-speech information. However, splitting and merging, which transform based on punctuation, were also used.

This study considered that preprocessing steps such as normalization and punctuation could semantically damage the meaning of a sentence by making modifications. For example, technical terms can be important in a paragraph or sentence and can help readers to understand the meaning. Because terminology can consist of a single word or multiple words, the segmentation of all words may not reflect the essential meaning of the terminology. To analyze the method of using technical terms in the sentence model, this study developed a module that can identify technical terms composed of complex words and process them as multiple single words. In addition, an entropy-based sorting module was developed to check the effect of sentence complexity on accuracy.

This study applied various combinations of typical preprocessing types, and the techniques developed in this study were analyzed separately. Table 3 shows the preprocessing types used in the analysis.

The accuracy of the model was measured five times for each of the 25 preprocessing types, for a total of 125 measurements.

2.3. Preprocessing Techniques. The preprocessing techniques developed in this study are the sorting module, which sorts

Table 1: Text preprocessing technique.

Technique			Feature
Normalization	Lowering	Conversion to lowercase	Pros: search accuracy can be improved Cons: proper nouns composed of capital letters can be incorrectly classified as general nouns
	Stemming	Conversion to stems	Pros: time efficiency can be improved by reducing the size of the text Cons: dilution of meaning can affect accuracy
	Lemmatization	Conversion to headwords	Pros: part-of-speech information is converted into a preserved form, and search accuracy can be improved Cons: conversion time is long
Punctuation	Splitting	Word splitting	Pros: meaning can be preserved Cons: different rules should be applied depending on the purpose, and the rules are complicated
	Merging	Word merging	

Table 2: Stemming vs. lemmatization.

Word	Stemming	Lemmatization
Innovation	Innovat	Innovation
Innovations	Innovat	Innovation
Innovate	Innovat	Innovate
Innovates	Innovat	Innovate
Innovative	Innovat	Innovative

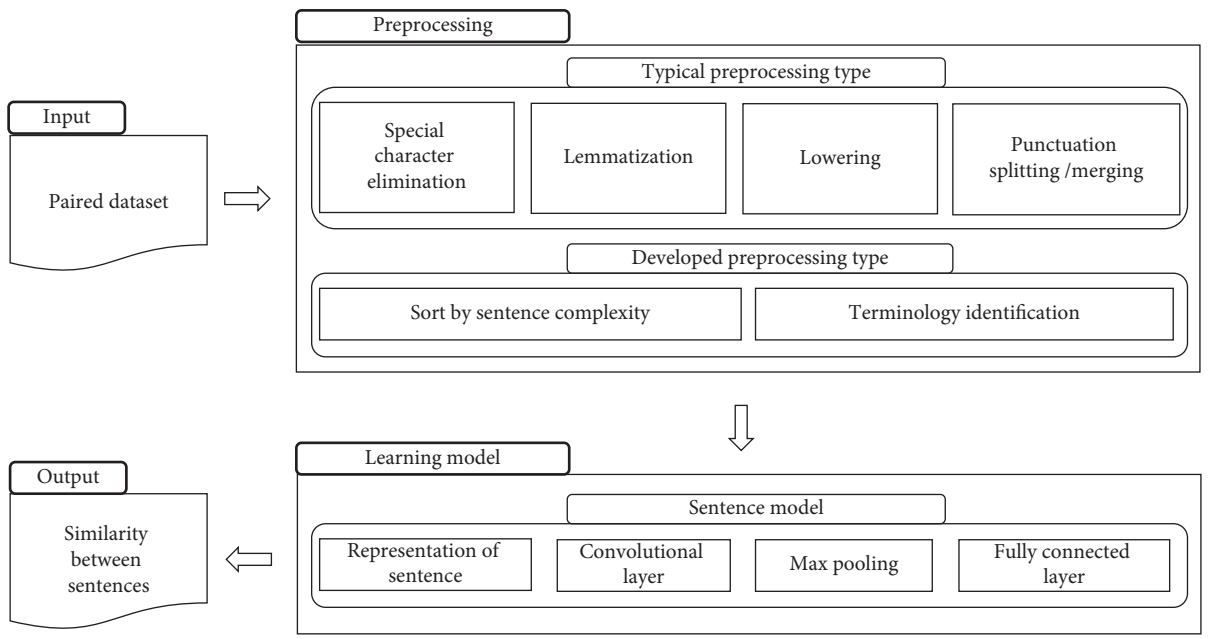

Figure 1: Structure of the similarity measurement model between sentences.

Table 3: Preprocessing types.

Type	Number
Preprocessing is not applied	1
Only one type of preprocessing is applied (a combination of traditional techniques + two proposed techniques)	8
Two types of preprocessing are combined	9
Three types of preprocessing are combined	5
Four types of preprocessing are combined	2
Total	25

the sentences according to an order of complexity of sentences, and the terminology-identification module. The details are described as follows.

2.3.1. Complexity Sorting Module. The characteristics of the data used in machine learning are an important factor in determining the efficiency of learning [26]. Documents are composed of various sentences, and each sentence has a different length, parts of speech, and complexity. Determining entropy for information complexity is a general-purpose technology that is commonly used for signal or video compression. The entropy of a sentence is calculated according to the distribution of syllables, and the calculated entropy is used to define the sentence complexity. This study also analyzed the relationship between the complexity, which is a characteristic of a sentence, and the accuracy of the model. In other words, based on the entropy, a sorting module that classified sentences according to their complexity was developed and confirmed.

The process of model development progressed as follows. Information entropy is an expected value (average) of information in data (as explained below), and when the expected value is high, it can be expressed as "much information." In other words, "much information" in a sentence indicates that the sentence is complicated on the surface. For a random variable for an event, $P(X)$, the information entropy, $H(X)$, is defined as follows:

$$H(p) = -\sum_{x \in X} p(x) \log p(x). \quad (1)$$

The operation algorithm of the complexity sorting module is shown in Figure 2.

In Figure 2, D^m indicates a set of sentences in the corpus and counts each sentence read from D^m according to the ASCII code value. In other words, it calculates the number of ASCII codes in a sentence. Then, the entropy is calculated based on the ASCII code value of a sentence. Finally, it returns the sentences sorted in the ascending or descending order based on the calculated entropy.

2.3.2. Module to Identify Technical Terms Composed of Complex Nouns. Technical term identification is a time-consuming and costly task that can be divided into statistical and rule-based methods. Statistical methods can have high portability because they are not affected by domain restrictions [27]. However, the low accuracy of the identified terms and the inclusion of noise pose difficulties in semantic interpretation. The rule-based method analyzes many terms and processes them through morphemes such as prefixes and suffixes. Although this method can have a low portability because the rules are manually defined and supplemented for each specific field, the accuracy of the identified terms can be high.

In this study, an algorithm to apply the rule-based method and achieve high accuracy was developed. To extract the rules, 1,540 morphemes in the technical term corpus published by the Japan Information Processing Society in 2018 were analyzed. The analysis found that the number of parts-of-speech among the technical terms was 82, and these were composed of single words or combinations of morphemes. The technical term identification algorithm we developed is shown in Figure 3.

First, morphemes are analyzed for each word in the sentence. For analysis, NLTK's pos_tag module was used. Second, technical terms were identified from the learning data using the extracted rules. From the most common composition to the least common composition of the part of speech, there were 485 singular nouns, 396 adjectives + singular nouns, and 257 singular nouns + singular nouns, etc. Third, a search engine (Wikipedia API) was used to verify the identified technical terms. When a search result for a technical term exists in the search engine, the term is converted into an identified sentence. For example, in the sentence "People in a car_race," "car race" is identified to convert it into the sentence "People in a car_race." Fourth, the sentence in which technical terms have been processed is newly stored in Transformed_D. Learning of the sentence model was conducted using the dataset in Transformed_D.

2.4. Sentence Model. The model used to measure the similarity between sentences consists of an encoder/decoder method, which can process two sentences. Siamese networks include two identical subnetwork components to handle each of the two inputs. In other words, Siamese networks can be used as a method to measure the similarity between two sentences. This section describes the structure and performance of the Siamese networks' convolutional neural network- (CNN-) based sentence model for the study.

2.4.1. Structure of the Sentence Model. To measure the similarity between sentences, the CNN model was implemented based on Siamese networks [9]. The model developed by Kim et al. [9] for emotion analysis and question classification is a CNN structure using one layer, but the model proposed in this study is composed of two layers. In addition, hyperparameters for filter size and feature map size were properly tuned. The proposed model is shown in Figure 4.

In Figure 4, n is the number of words in the sentence, k is the dimension of the word vector, and h is the filter window size. Based on the CNN, sentences $X(i)$ and $X(j)$ are sequentially processed through the convolutional, pooling, and fully connected layers, and feature vectors are produced as outputs. For hyperparameters, the filter sizes were set to 2, 3, and 4, and they were set to have 50 feature maps. The dropout was set to 0.5. To compare two sentences, $X(i)$ and $X(j)$, the distance can be expressed as follows, by outputting a feature vector that is the encoding result of the same neural network structure.

If $x(i)$ and $x(j)$ are semantically similar, $\| f(X(i)) - f(X(j)) \|^2$ is small.

If $x(i)$ and $x(j)$ are semantically different, $\| f(X(i)) - f(X(j)) \|^2$ is large.

If the characteristics of the two sentences are well expressed, the distance between the vectors is small. Otherwise, the distance is large. In this study, the Manhattan

```
Input : D^m = {Sen_[0] , ..., Sen_[m]} # Set of sentences
Initialization : E^m = {0, ..., Sen_[m]} # Set of entropy complexity values
Output : Sorted_D^m = {Sen_[0] , ..., Sen_[m]} # Set of sentences sorted by entropy complexity

for i = 0, ..., N - m do
  Size = len (Sen_[i]) * 1.0
  for j = 0, ..., 128 do
     result + = Sen_i.count (chr (j)) / size * log (Sen_[i].count (chr (j)) / size, 2)
  E(i) = result * -1.0

E = dictionary (E)
Sorted_D = sort (D, E)

return Sorted_D
```

Figure 2: Complexity sorting algorithm.

```
Input : D^m = {Sen_[0] , ..., Sen_[m]} # Set of sentences
Initialization : Tagged_Sen^m = {{0, ... Sen_[0,n]}, ..., {0, ... Sen_[m,n]}} # Set of entropy
complexity values
Output : Transformed_D^m = {TSen_[0] , ..., TSen_[m]} # Set of sentences sorted by entropy
complexity
for i = 0, ..., N - m do
  Tagged_Sen_i = pos_tag (Sen_[i])
  Sentence = ‘’
  for j = 0, ..., len (Sen_[i,j]) do
    Terminology = Rule_based_Identifier (Sen_[I,j], Size_of_Terminology)
    Exists = Search_engine_API (Terminology)
    if Exists is True
      Sentence = Sentence + Terminology
      j = j + Size_of_Terminology
    else if Exists is not True
      Sentence = Sentence + Sen_I,j

  Transformed_D_i.Append (Sentence)

return Transformed_D
```

No	The morpheme of the technical term	N
1	Singular noun	485
2	Adjective + singular noun	396
3	Singular noun + singular noun	257
4	Adjective	75
5	Adjective + singular noun + singular noun	53
6	Present participle	30
7	Present participle + singular noun	23
8	Singular noun + singular noun + singular noun	23
9	Plural noun + singular noun	17
10	Adjective + plural noun	16
11	Past participle + singular noun	11
12	Past tense verb + singular noun	11
13	Singular noun + preposition + singular noun	11

Figure 3: Technical term identification algorithm.

distance, which is a similarity function and has excellent performance, was applied (Jonas Mueller, 2016). This study was developed using Python 3.6.8 in the Linux 16.04 operating system environment.

2.4.2. Comparison of the Performance of Sentence Models. In the present study, a CNN-based model (Figure 4), Siamese LSTM model [28], and transformer model [29] were implemented for model selection. For the learning

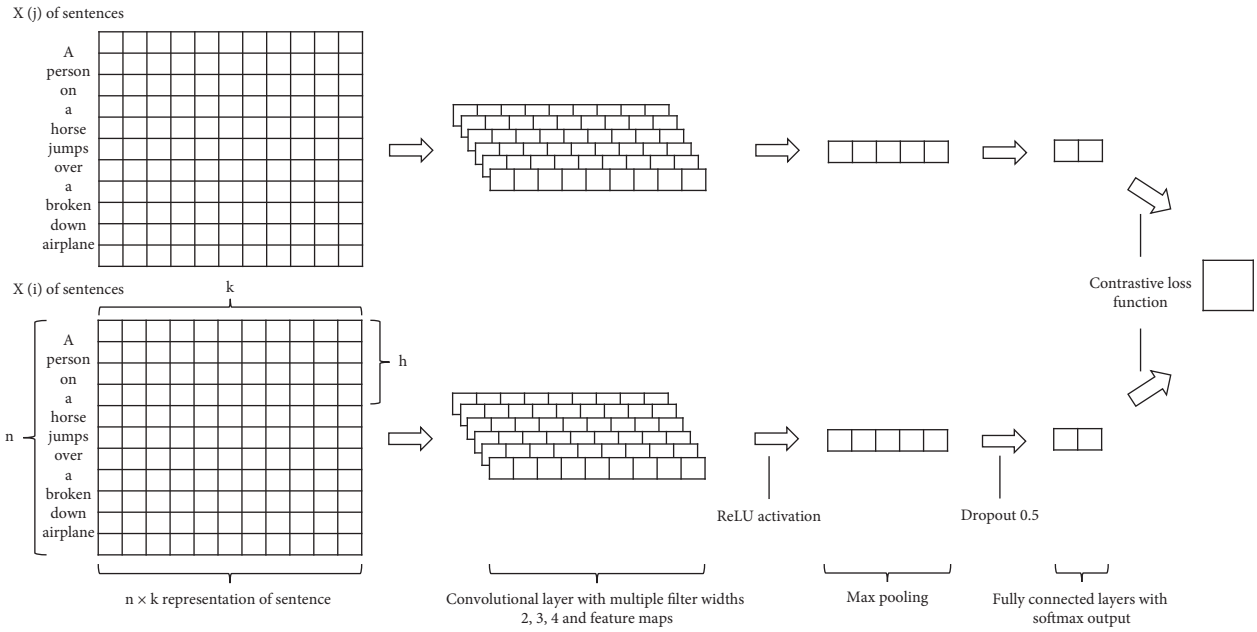

Figure 4: Siamese network based on the CNN model.

parameters of each model, the epoch was set to 10, batch size to 512, and learning rate to 0.001, and learning time and accuracy were as shown in Table 4.

To evaluate how the performance was affected using combinations of various types of preprocessing, a study was conducted using a CNN-based model, which had low accuracy but the fastest learning time.

2.5. Dataset. For model training, the Stanford Natural Language Inference (SNLI) corpus was used. The SNLI corpus is a dataset that expresses the logical relationship between two sentences, and it is used in research for various inferences [30, 31]. The corpus consists of 367,369 instances, of which 257,158 (70%) are training sets, 36,737 (10%) are validation sets, and 73,474 (20%) are testing sets. The structure of the SNLI corpus is shown in Table 5.

This structure includes two sentences and a corresponding label that is based on the similarity between the sentences. For example, "A person on a horse jumps over a broken down airplane" and "A person is at a diner, ordering an omelette" in the first line are not semantically/logically equivalent, and therefore the label is 0 (False).

3. Results and Discussion

Table 6 shows the results when the models with different combinations of preprocessing types were sorted by accuracy.

The variance of the average value of the results by the preprocessing type ranged from a minimum of 0.05 to a maximum of 0.34. Based on analysis of the results, we determined that a combination of two preprocessing techniques showed good performance. If only two preprocessing techniques can be used, lemmatization and punctuation splitting (No. 1) are good candidates. This combination showed the highest accuracy with 79.09%, which is 0.73% higher than the accuracy without preprocessing (No. 21). Furthermore, it was found that the use of the normalization techniques, lemmatization and lowering (No. 2) together or the use of lowering and punctuation splitting or merging (Nos. 3 and 4) also increased accuracy. It should be noted that, in any of these combinations, lemmatization, lowering, and punctuation splitting were used. Lemmatization and lowering are techniques that can improve accuracy by normalizing different words.

Table 4: Learning time and accuracy by the model.

Architectures	Learning time	Accuracy
Siamese LSTM model	2 : 14 : 02	79.66
Transformer model	1 : 54 : 52	79.61
CNN-based model	25 : 48	77.67

If only one technique was used (Nos. 9, 10, 11, 15, and 18), then the accuracy was higher than that without preprocessing. Among these, the accuracies associated with lemmatization (No. 9), lowering (No. 10), and punctuation splitting (No. 11) were similar and ranged from 78.876% to 78.842%. This implies that splitting a word into two words based on punctuation, such as during normalization or punctuation splitting, can extend the length of a sentence and have a positive effect on accuracy.

Characteristic features include the use of special character elimination and punctuation merging. When special character elimination was combined with lemmatization and lowering, the accuracy was increased (Nos. 5, 7, and 8). However, if lemmatization or lowering was used separately, the accuracy decreased (Nos. 12, 13, 16, 19, 20, and 22). In addition, except for the case where it was combined with lowering (No. 3), the accuracy decreased when punctuation merging was used. This contrasts with the fact that

Table 5: Structure of the SNLI corpus.

Premise	Hypothesis	Label
A person on a horse jumps over a broken down airplane	A person is at a diner, ordering an omelette	0
A person on a horse jumps over a broken down airplane	A person is outdoor on a horse	1
Children smiling and waving at the camera	There are children present	1
Children smiling and waving at the camera	The kids are frowning	0
A boy is jumping on the skateboard in the middle of a red bridge	The boy skates down the sidewalk	0
A boy is jumping on the skateboard in the middle of a red bridge	The boy does a skateboarding trick	1
Two blond women are hugging one another	The women are sleeping	0
Two blond women are hugging one another	There are women showing affection	1

Table 6: Accuracy based on the preprocessing type.

	No.	Type	Accuracy					
			1st	2nd	3rd	4th	5th	Mean (SD.)
Traditional method	1	[2] + [4]	79.17	79.01	79.31	78.93	79.03	79.090 (0.15)
	2	[2] + [3]	79	79.14	79.25	78.73	79.09	79.042 (0.20)
	3	[3] + [5]	79.15	78.98	79.17	78.89	78.97	79.032 (0.12)
	4	[3] + [4]	78.96	79.12	78.82	79.13	79.09	79.024 (0.13)
	5	[1] + [2] + [3] + [4]	79	78.96	79.12	78.93	79.01	79.004 (0.07)
	6	[1] + [4]	78.94	78.59	79.33	79.15	78.97	78.996 (0.28)
	7	[1] + [2] + [3] + [5]	78.8	78.86	79.09	78.79	78.98	78.904 (0.13)
	8	[1] + [2] + [3]	78.54	79.22	78.93	78.95	78.87	78.902 (0.24)
	9	[3]	79.02	78.97	78.75	78.73	78.91	78.876 (0.13)
	10	[4]	78.94	79	78.62	78.77	78.89	78.844 (0.15)
	11	[2]	78.83	78.78	78.83	78.87	78.9	78.842 (0.05)
	12	[1] + [3] + [4]	78.44	78.94	78.8	79.15	78.85	78.836 (0.26)
	13	[1] + [3] + [5]	78.41	78.8	78.67	79.34	78.8	78.804 (0.34)
	14	[2] + [5]	78.79	78.77	78.85	78.68	78.78	78.774 (0.06)
	15	[5]	78.34	78.67	78.86	78.94	78.7	78.702 (0.23)
	16	[1] + [2] + [5]	78.78	78.25	78.88	78.75	78.67	78.666 (0.24)
	17	[1] + [5]	78.51	78.38	78.72	78.96	78.59	78.632 (0.22)
	18	[1]	78.7	78.28	78.46	78.51	78.51	78.492 (0.15)
	19	[1] + [2]	78.22	78.61	78.27	78.56	78.36	78.404 (0.17)
	20	[1] + [2] + [4]	78.26	78.45	78.35	78.17	78.31	78.308 (0.10)
	21	[0]	78.42	78.36	78.28	78.11	78.34	78.302 (0.12)
	22	[1] + [3]	78.24	78.16	78.07	78.36	78.3	78.226 (0.11)
Developed method	23	[7-2]	78.35	78.68	79.1	78.65	78.34	78.624 (0.31)
	24	[7-1]	78.39	78.15	78.19	78.37	78.33	78.286 (0.11)
	25	[6]	73	72.96	72.43	72.59	72.8	72.756 (0.24)

[0]: no applied preprocessing; [1]: special character elimination; [2]: lemmatization; [3]: lowering; [4]: punctuation splitting; [5]: punctuation merging; [6]: essential terminology preprocessing; [7-1]: ascending order based on entropy complexity; [7-2]: descending order based on entropy complexity.

punctuation splitting leads to high accuracy. The results are similar to those obtained for the technical term identification preprocessing technique that was developed in this study for comparison.

In No. 25, technical terms composed of complex nouns were recognized as one word for learning, and the associated accuracy was 72.76%. The accuracy was lower than that without preprocessing by 5.54%, and it was also relatively low compared with other preprocessing combinations. As suggested by the algorithm, there are many technical terms that contain more than two words, such as in the forms (adjective-singular noun), (singular noun-singular noun), and (adjective-singular noun-singular noun). As a result, the meanings of these terms cannot be correctly interpreted. In other words, because the technical terms consisting of two or more segments were processed as one word and the length of the sentence was shortened, the accuracy decreased.

When sentences were sorted according to their entropy complexity (Nos. 23 and 24), the accuracies were 78.624% and 78.286% for the descending and ascending order, respectively. This represents a difference of +0.322% and −0.016% compared with data that were not preprocessed. Therefore, ordering sentences according to their complexity may not affect the accuracy.

4. Conclusions

In neural network research, data, algorithms, and parallel hardware are essential elements. Even with good algorithms and high-performance hardware, studies cannot be conducted

if the quality of data is low or no data are available. Despite its importance, many existing neural network studies do not provide any information about data preprocessing.

This study analyzed the effect of preprocessing through text data preprocessing of sentence models. To this end, experiments were conducted to evaluate combinations of typical data preprocessing types. Furthermore, the effects of two new techniques on the accuracy of the model were analyzed: preprocessing of technical terms composed of compound words and determining the learning order based on data complexity.

Based on the results of this study, the following conclusions can be drawn. First, when only two preprocessing techniques are used, we recommended using lemmatization and punctuation splitting, lemmatization and lowering, or lowering and punctuation splitting. Second, when only one preprocessing technique is used, it is better to use lemmatization, lowering, or punctuation splitting. Third, to improve accuracy, it is generally not recommended to use a preprocessing type that shortens the lengths of sentences. Fourth, the use of special character elimination and normalization techniques does not contribute to improving the accuracy. Fifth, setting the learning order according to sentence complexity does not contribute to improving the accuracy.

Building predictive or sentence models from refined data can help improve the performance of the model. The accuracy of the preprocessing of text data in this study suggested a certain combination of preprocessing types could improve performance when various models are established. Consequently, this study is significant in that it allows better decision-making about which preprocessing type should be selected according to the purpose of the study or the type of the construction model.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIP) (no. 2019R1H1A1079885).

References

[1] D.-K. Lee, K.-J. Oh, and H.-J. Choi, "Measuring the syntactic similarity between Korean sentences using RNN," *The Korean Institute of Information Scientists and Engineers*, vol. 6, pp. 792–794, 2016.

[2] Y. Bengio, *Neural Probabilistic Language Models," Innovations in Machine Learning*, Springer, Berlin, Germany, 2006.

[3] T. Mikolov, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3111–3119, Lake Tahoe, NV, USA, December 2013.

[4] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245–271, 1997.

[5] S. F. Crone, S. Lessmann, and R. Stahlbock, "The impact of preprocessing on data mining: an evaluation of classifier sensitivity in direct marketing," *European Journal of Operational Research*, vol. 173, no. 3, pp. 781–800, 2006.

[6] C. A. Gonçalves, R. Camacho, and E. C. Oliveira, "The impact of pre-processing on the classification of MEDLINE documents," pattern recognition in information systems," in *Proceedings of the 10th International Workshop on Pattern Recognition in Information Systems, PRIS 2010*, Funchal, Madeira, Portugal, June 2010.

[7] A. K. Uysal and S. Gunal, "The impact of preprocessing on text classification," *Information Processing & Management*, vol. 50, no. 1, pp. 104–112, 2014.

[8] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: a new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[9] Y. Kim, "Convolutional Neural Networks for Sentence Classification," 2014, http://arxiv.org/abs/1408.5882.

[10] 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*, Lisbon, Portugal, September 2015.

[11] T. Lei, R. Barzilay, and T. Jaakkola, "Molding cnns for text: non-linear, non-consecutive convolutions," 2015, http://arxiv.org/abs/1508.04112.

[12] H. Chen, "Neural sentiment classification with user and product attention," *Proceedings. Of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, TX, USA, November 2016.

[13] Y. Xiao and K. Cho, "Efficient character level document classification by combining convolution and recurrent layers," 2016, http://arxiv.org/abs/1602.00367.

[14] K. Kowsari, "Hdltex: hierarchical deep learning for text classification," 2017, http://arxiv.org/abs/1709.08267.

[15] Z. Yang, "Hierarchical attention networks for document classification," in *Proceedings. Of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, Human Language Technologies, San Diego, CA, USA, June 2016.

[16] S. Lai, "Recurrent convolutional neural networks for text classification," *AAAI*, vol. 333, 2015.

[17] T. Kuzar and P. Navrat, "Preprocessing of slovak blog articles for clustering," in *Proceedings of the 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Toronto, ON, USA, September 2010.

[18] J.-S. Lee, "A study on the data mining preprocessing tool for efficient database marketing," *Journal of Digital Convergence*, vol. 12, no. 11, pp. 257–264, 2014.

[19] S. K. Dwivedi and B. Rawat, "A review paper on data preprocessing: a critical phase in web usage mining process," in *Proceedings of the 2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, Greater Noida, Delhi, India, October 2015.

[20] H.-S. Shin, Y. Jin, and C.-S. Park, "Influence of data preprocessing on a machine learning model," *Architectural Institute of Korea*, vol. 37, no. 1, pp. 491-492, 2017.

[21] M. Kasthuri and Dr. S. Britto Ramesh Kumar, "A framework for language independent stemmer using dynamic programming," *International Journal of Applied Engineering Research*, vol. 10, pp. 39000–39004, 2013.

[22] K. Abainia, S. Ouamour, and H. Sayoud, "A novel robust Arabic light stemmer," *Journal of Experimental & Theoretical Artificial Intelligence*, no. 1–17, 2016.

[23] P. Han, S. Shen, D. Wang, and Y. Liu, "The influence of word normalization in english document clustering," in *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference*, Zhangjiajie, China, May 2012.

[24] M. Attia and J. Van Genabith, "A jellyfish dictionary for Arabic," in *Proceedings of the Electronic lexicography in the 21st century: thinking outside the paper: proceedings of the eLex 2013 conference*, Tallinn, Estonia, Europe, October 2013.

[25] R. J. Prathibha and M. C. Padma, "Design of rule based lemmatizer for kannada inflectional words," in *Proceedings of the Emerging Research in Electronics, Computer Science and Technology (ICERECT), 2015 International Conference*, Mandya, India, December 2015.

[26] S. Samsani, "An RST based efficient preprocessing technique for handling inconsistent data," in *Proceedings of the 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1–8, Chennai, India, December 2016.

[27] P. Pantel and D. Lin, "A statistical corpus-based term extractor," in *Proceedings of the Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pp. 36–46, Ottawa, Canada, June 2001.

[28] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proceeding AAAI'16 Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 2786–2792, Quebec, Canada, May 2000.

[29] A. Vaswani, N. Shazeer, N. Parmar et al., "Attention is all you need,," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 6000–6010, Cambridge, MA, USA, December 2015.

[30] A. Talman and S. Chatzikyriakidis, "Testing the generalization power of neural network models across NLI benchmarks," in *Proceedings of the Second BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium, November 2018.

[31] A. Conneaum, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from Natural Language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 670–680, Copenhagen, Denmark, September 2018.