Handling Out-Of-Vocabulary Problem in Hangeul Word Embeddings

Ohjoon Kwon[1†] **Dohyun Kim**[2†] **Soo-Ryeon Lee**[3] **Junyoung Choi**[3] **SangKeun Lee**[2,3]
[1]Samsung Research, Republic of Korea
[2]Department of Computer Science and Engineering, Korea University, Seoul, Republic of Korea
[3]Department of Artificial Intelligence, Korea University, Seoul, Republic of Korea
ohjoon.kwon@samsung.com
{dhkim1028, bonny05616, dnduddl2015, yalphy}@korea.ac.kr

Abstract

Word embedding is considered an essential factor in improving the performance of various Natural Language Processing (NLP) models. However, it is hardly applicable in real-world datasets as word embedding is generally studied with a well-refined corpus. Notably, in Hangeul (Korean writing system), which has a unique writing system, various kinds of Out-Of-Vocabulary (OOV) appear from typos. In this paper, we propose a robust Hangeul word embedding model against typos, while maintaining high performance. The proposed model utilizes a Convolutional Neural Network (CNN) architecture with a channel attention mechanism that learns to infer the original word embeddings. The model train with a dataset that consists of a mix of typos and correct words. To demonstrate the effectiveness of the proposed model, we conduct three kinds of intrinsic and extrinsic tasks. While the existing embedding models fail to maintain stable performance as the noise level increases, the proposed model shows stable performance.

1 Introduction

Word embedding refers to the process of generating vectors that contain semantic and syntactic information of languages. Machines can understand the meaning of words through the word embeddings. Several embedding methodologies have been proposed to improve various Natural Language Processing (NLP) models' performance, such as machine translation, sentence classification, and text generation (Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017).

Word2vec (Mikolov et al., 2013) generates word vectors using the contextual data of words based on the distributional hypothesis. Word2vec assigns a unique vector to each word, and has proved its effectiveness by improving various systems' performance. Similarly, GloVe (Pennington et al., 2014) that uses co-occurrence information of each word found throughout the corpus, generates word vectors that contain semantic and syntactic characteristics.

However, Word2vec and GloVe suffer from being dependent on the training corpus. In particular, they do not include the word embeddings that did not appear in the training process. To tackle this OOV problem, replacing unseen words with a unique token "UNK" is widely used as a solution. This method can allow the model to deal with OOV words. However, this countermeasure reduces the performance of the system. This is because several unseen words are recognized as having the same meaning. Hence, the importance of each word is ignored and recognized as noise.

This OOV problem amplifies when dealing with real-world datasets that contain a large number of typos and intended slang that has not been included in the training phase. In particular, if words relevant to the core meaning of a sentence are replaced as OOV, the system's performance will drastically reduce. For example, suppose an evaluation set for a sentiment classification task contains typos such as "funnnny" and "coooool". When predicting the sentiment of the sentence, the original meanings of the two words, "funny" and "cool", serve as decisive information. However, the sentiment classification accuracy drastically reduces when these words are processed as "UNK" and encoded to have the same meaning. Therefore, to increase the effectiveness of NLP models, robust word embeddings are strongly needed in noisy text.

Meanwhile, FastText (Bojanowski et al., 2017) presents a method to address the OOV problem. FastText follows the training process of Word2vec,

[†]Equal contribution

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 3213–3221
April 19 - 23, 2021.

but understands a word as a set of n-gram subword information. For example, the meaning of the word 'funny' is learned by decomposing the characters into n-grams such as 'fun', 'unn', and 'nny.' In this way, FastText can generate embeddings for OOV words that did not appear during the training phase by utilizing this subword information.

FastText is suitable for languages that have characteristics of isolated words (e.g., English). This is because the meaning of the word can be inferred through the summation of subword information: prefix, root, and postfix. However, in the case of agglutinative languages such as Hangeul (Korean writing system), it is not easy to fully understand the meaning of a word by merely referring to the summation of the subword information. A word in Hangeul consists of at least one to ten or more different morphemes, which make extracting internal information complicated. Therefore, to decipher a Hangeul word, it is necessary to be sensitive when looking for the vital subword information within the word.

CNN captures locational and morphological information of words and characters, and many studies using CNN have achieved good performance in various NLP tasks (Zhang et al., 2015; Kim et al., 2016, 2018; Ma and Hovy, 2016). Inspired by this, in this paper, we propose a CNN-based channel attention word embedding model that generates robust Hangeul embeddings for noisy text. The proposed model utilizes CNN to extract n-gram pairs from Korean words. The model then applies an attention mechanism to features from each channel representing the inherent information in the n-gram pairs. An input word for training is replaced with intentionally generated typos with a certain probability of adapting to typos. This generated word vector enhances the semantics of a word by predicting the context words. In summary, the contributions of this paper are three-fold:

- We propose a robust word embedding model against typos by training it with intentionally generated typos.
- We introduce a channel attention mechanism to utilize the distinctive structure of Hangeul words.
- We demonstrate the effectiveness of the proposed model on the noisy text through word analogy, language modeling, and sentiment classification tasks.

The remainder of the paper is organized as follows. In Section 2, we investigate related work and discuss key differences from ours. In Section 3, we describe the proposed methodology in detail. Furthermore, we present the evaluation results and analyze the performance in Section 4 and 5, respectively. Finally, Section 6 contains the conclusion.

2 Related work

In this section, we review previous studies on 1) embedding generation methods that are robust to typos and 2) embedding generation methods that consider the unique characteristics of Hangeul.

2.1 Typos word embedding methods for English

There have been many studies for dealing with OOV problems (Bojanowski et al., 2017; Belinkov and Bisk, 2018; El Boukkouri et al., 2020). Unlike unseen words, typos require a different approach in that the originally intended word is likely to be known. RoVe (Malykh et al., 2018) and MOE (Piktus et al., 2019) explored the methodologies to deal with typos. RoVe (Malykh et al., 2018) utilized the English word's structural features as prefix, root, and postfix. The target word is encoded into Beginning, Middle, and End vectors. Embeddings generated in this way maintain robust model performance, although a noise level of the downstream task increases. However, RoVe performs poorly in generating agglutinative embeddings in languages such as Hangeul. As mentioned earlier, this is due to the feature of agglutinative languages where dozens of morphemes are combined to form a word. In other words, dividing morphemes into three parts – Beginning, Middle, and End – is not sufficient to comprehend Hangeul words.

MOE (Piktus et al., 2019) introduced supervised learning for misspelling patterns to FastText's training mechanism. The model was explicitly trained to infer the original meaning of the word from the irregular shape of typos by increasing the similarity between them. In this way, MOE was trained on typo data collected from real users on the web. Consequently, MOE showed stable performance for unseen words in intrinsic and extrinsic tasks. However, MOE generates the word embeddings with linear combinations of the constituent subwords. In other words, the word vectors do not reflect the differences in importance among the subword information. This is insufficient to fully

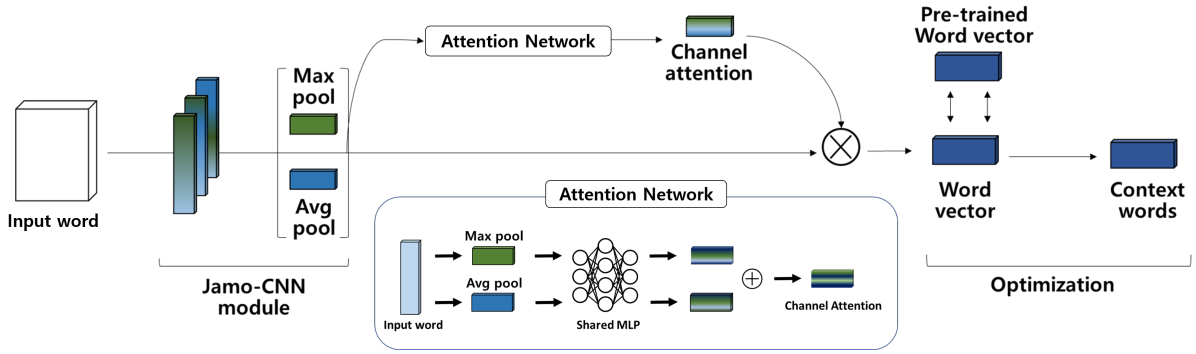

Figure 1: Overall framework for the word embedding generating system proposed in this paper.

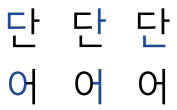

Figure 2: Example of the constituent of a Korean word "단어 (word)".

explain the meaning of words in complex agglutinative languages such as Hangeul. To generate effective Hangeul word embeddings, differentiating importance across n-gram pairs is necessary.

2.2 Word Embedding Methods for Korean

Ko-FastText (Park et al., 2018) and other morpheme-based studies (Lee et al., 2018; Nam and Kim, 2016) considered the characteristics of agglutinative languages during embedding. Ko-FastText performed better in intrinsic and extrinsic tasks with a Hangeul dataset by applying the FastText to each jamo (the smallest unit of a Hangeul word). However, this model did not fully utilize the sophisticated internal information of Hangeul words. Although the model refers to the morphological information revealed in the n-gram, lack of selectivity regarding the important morphological information of a word remains a problem.

Nam and Kim (2016) and Lee et al. (2018) explicitly considered the Korean words' morphological information. They decomposed words into character n-grams using morpheme analyzers. However, they encountered a fundamental problem that their models are dependent on the performance of the morpheme analyzer. In particular, the analyzer did not work appropriately for typos, so the performance of word embeddings cannot be guaranteed.

Unlike previous studies, we do not utilize a specific morpheme analyzer (or stemmer) or use typo data collected from the web. Instead, we propose a model that trains with our own generated typo set. Besides, we linearly combine the subword information by reinforcing the necessary information dynamically. Therefore, the model maintains robust performance against different grammatical errors as its embedding generation technique incorporates morphologically complicated Hangeul features.

3 Methodology

In this section, we describe the architecture of the proposed model. The model consists of three parts. The first part is the generation of Korean typos. We create rule-based typo datasets, and they reinforce the robustness of the embedding model for typos. Next, we create a jamo-level CNN with channel attention to represent the word vector. By applying channel attention to features extracted through convolution operation from each jamo, we selectively enhance the importance of the primary n-gram features. Based on a certain probability, the CNN learns to generate the original word vector from the typos, reducing the difference between the typos and the actual word embeddings. Finally, we perform the task of predicting the surrounding context words from the generated word embeddings. Through this process, we can proceed with the reinforcement of the semantics of words. Figure 1 shows the whole architecture of our model.

3.1 Generating Korean Typos

The smallest unit of Hangeul words that corresponds to the alphabet of English is jamo, consisting of 14 consonants and 10 vowels. In Hangeul, a syllable consists of three jamos: chosung, joongsung and jongsung. We divide each syllable into jamo units (Song, 2006). Figure 2 shows an example of dividing each syllable. The syllable "단" is divided into [ㄷ, ㅏ, ㄴ] and "어" is divided into

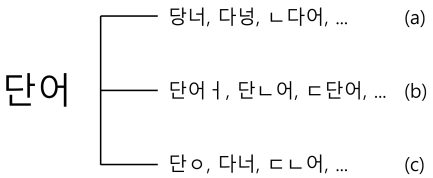

Figure 3: Examples of typos in Korean words. The typos generated in this paper are based on these cases.

[ㅇ, ㅓ]. In Hangeul, the chosung and joongsung jamos are always present while the jongsung jamo is only present or not (e.g., in "어").

After separating the Korean word into jamo, we make the following rules for creating a typo : (a) reversing the order of letters in the word, (b) adding arbitrary letters, or (c) dropping out a letter from the word. We generate our typo dataset following a well-known Korean typo distribution reported in the Korean information processing community (Jeon et al., 2010). Figure 3 shows examples of typos by the three rules. When following the rules, typos in Hangeul cause a breakdown of the word's form. This is because only consonants can be placed in the chosung position and only vowels can be placed in the joongsung position. Therefore, while typos in English can easily infer the original meaning (e.g., englihs, englis, ennglish), Hangeul typos are hard to understand even when they come from native speakers. Consequently, we generate at least 10 to 25 typo word pairs for each Korean word.

3.2 Jamo-level CNN with Channel Attention

After decomposing the input word into jamo units and generating typos, the convolution operation is used to extract the most prominent features of the word. Each one-hot encoded jamo representation is concatenated to set in a form suitable for convolution operations. We use zero padding to fix the input word vector's length to the size of the longest word. Then we obtain jamo representation $K \in \mathbb{R}^{m \times d}$, where m is the maximum length of the input word, and d is the embedding dimension. A convolution operation involves a filter $w \in \mathbb{R}^{h \times d}$, where $h \in \{1, 2, ..., 10\}$ is a filter size. We apply the convolution operation and activation function with these generated jamo representations. For example, a feature c is generated from a window of jamos $w_{i:i+h-1}$ by

$$c_i = ReLU(K^h \cdot w_{i:i+h-1} + b) \quad (1)$$

where $b \in \mathbb{R}$ is a bias. This convolution filter extracts the major feature of the jamos $\{w_{1:h}, w_{2:h+1}, ..., w_{m-h+1:m}\}$ to produce a feature map

$$c = [c_1, c_2, ..., c_{m-h+1}], \quad (2)$$

with $c \in \mathbb{R}^{m-h+1}$. We generate n feature maps in this way. Hence, we obtain concatenated feature map $C \in \mathbb{R}^{(m-h+1) \times n}$, where n is the number of the feature maps. To extract the features for the attention score from the convolution operation results, we employ average and max pooling in channel level as follows:

$$C_{avg} = average(C), \quad (3)$$

$$C_{max} = max(C), \quad (4)$$

where $C_{avg}, C_{max} \in \mathbb{R}^n$. Next, to extract the attention-applied feature map, we compute the attention scores between the channels.

$$C_{score} = W_2(W_1(C_{avg} + C_{max})), \quad (5)$$

where $W_1, W_2 \in \mathbb{R}^{n \times n}$ are the first and second weight matrix of the attention network for calculating the attention score, respectively. C_{score} is the result that comes from passing through the attention network. We apply a softmax function to create a standardized channel attention map. We then broadcast C_{score} into the same shape of C. The channel attention map is multiplied by the continuous jamo pair features to generate a feature map of the input word reflecting an attention score for each channel. Then max pooling is applied to the attention score as follows:

$$C_{att} = max(ReLU(C \times softmax(C_{score}))). \quad (6)$$

We obtain C_{att} for each filter size following this method, and consequently we obtain $C^h_{att} \in \mathbb{R}^{10n}$.

Then C^h_{att} is connected to the fully-connected layer, and we obtain final word representation w as follows:

$$w = C^h_{att} \cdot W + b, \quad (7)$$

where $W \in \mathbb{R}^{10n \times d}$, $b \in \mathbb{R}$. We have selectively enhanced the substructure of the word and created word vector $w \in \mathbb{R}^d$ that reflects contextual information. We then train the jamo-level CNN so that the word vector w resembles the representation of the vector $\hat{w}$ from the pre-trained word. In this step,

we use the squared Euclidean distance with each normalized word vector for the objective function as follows:

$$Loss = \sum_{v \in V_w} \| w_v - \hat{w}_v \|^2 . \quad (8)$$

This equation approximates the generated word vector w to the pre-trained word vector $\hat{w}$. To create word embeddings using the proposed model, we split a word to jamo-level and insert it into the model as an input following a similar approach in the training phase. When the input tensor flows through the model and reaches the transformation layer, it is considered as the newly created embeddings of the word.

4 Experiments

In this section, we present our experimental results by comparing the performance of the proposed model with other embedding methods. We measure the performance of the models through word analogy, language modeling, and sentiment classification tasks.

4.1 Experimental Settings

We use an Adam optimizer with a learning rate of 0.001. The convolution filter has a size from one to nine, and each filter has 100 channels. The context window size for calculating semantic loss is set to an arbitrary number of less than five. The corpus used for learning is a Korean Wikipedia dump dataset released on June 1st, 2019. No preprocessing is performed other than removing special characters and replacing numbers with "N" tokens. We tokenize the corpus in word units for all models in the experiment. To ensure fairness, we set different random seeds in the training and evaluation phases.

4.2 Baselines

To measure the performance of the proposed model against widely used and state-of-the-art models, we used the following methods:

- *Word2vec* (Mikolov et al., 2013): *Word2vec* uses a representative embedding technique based on the distributional hypothesis. To find the meaning of a word in context, *Word2vec* learns the meaning of the target word from context words. Following this, if a word that is not registered at the training phase appeared at the evaluation phase, we replaced the word as "UNK" token. We used skip-gram architecture since it is generally proven to be better than other methods.

- *Ko-FastText (Ko-ft)* (Park et al., 2018): *Ko-ft* uses the jamo n-gram as the smallest unit. Therefore, it has an advantage that unseen words can be generated by a combination of jamo n-grams generated in the training phase. This model has proved to have the best performance in Hangeul embedding methods. $ch4$ means the model use one to four jamo-level n-grams, and $ch6$ means the model use one to six n-grams.

4.3 Word Analogy Task

We first conduct an embedding evaluation using the word analogy task as an intrinsic task. We measure how the word embedding process has learned semantic and syntactic meaning. Following (Park et al., 2018), we use 3COSADD based metric as an evaluation metric.

4.3.1 Dataset

We use the dataset introduced in (Park et al., 2018). The dataset is divided into two categories of semantic and syntactic meaning in the English version. Then they are translated into Korean and divided into ten subcategories. In this way, it is suitable to evaluate the performance of Hangeul embedding.

4.3.2 Results

Table 1 shows the analogy task results[1]. *Word2vec* has a big gap compared to *Ko-ft* and *misK*, which shows the limitations of the embedding technique from the viewpoint of words. In general, *misK* outperforms *Ko-ft*, indicating that the *misK* learned the semantic and syntactic meaning of each word better with the help of the well-designed attention mechanism in Figure 1. Also, *misK-ft* generally performs better than the model that learned the shape of words from *Word2vec* (*misK-w2v*). This demonstrates the importance of pre-trained embedding knowledge when creating new embeddings (Pinter et al., 2017).

4.4 Language Modeling Task

We conducted a language modeling experiment to check the embedding performance as an extrinsic

[1] Although our word embedding is designed to work in the noisy environment, we examine how it works in the clean (i.e. typo-free) environment.

Model	Semantics					Syntactics				
	Capt	Gend	Name	Lang	Misc	Case	Tense	Voice	Form	Horn
Word2vec	0.471	0.574	0.528	0.466	0.587	0.478	0.561	0.633	0.672	0.665
Ko-ft (ch6)	0.446	0.512	0.524	0.364	0.512	**0.194**	0.417	0.461	0.524	0.362
Ko-ft (ch4)	**0.441**	0.517	0.521	0.368	0.521	0.197	0.422	0.468	0.529	0.369
misK-w2v	0.463	**0.482**	0.513	0.331	**0.486**	0.328	**0.382**	0.436	0.389	0.326
misK-ft	0.460	0.488	**0.498**	**0.327**	0.491	0.311	0.388	**0.421**	**0.361**	**0.294**

Table 1: Result of a word analogy task experiment. *misK-w2v* and *misK-ft* are our proposed models trained from *Word2vec* (Mikolov et al., 2013) and *Ko-FastText* (Park et al., 2018), respectively. Lower score is better. Best results are highlighted in bold font.

Model	Noise level			
	0%	10%	20%	30%
Word2vec	248.3	259.7	269.1	281.4
Ko-ft (ch6)	**206.4**	**218.1**	229.5	241.7
Ko-ft (ch4)	210.2	223.1	231.9	246.3
misK-w2v	213.8	220.7	224.7	229.8
misK-ft	211.6	219.4	**224.6**	**228.4**

Table 2: Perplexity measured from the language modeling. The noise level is the probability that each word in the dataset will be replaced by typos. Best results are highlighted in bold font.

Model	Noise level				
	0%	10%	20%	30%	40%
Word2vec	64.1	62.9	60.4	59.3	58.4
Ko-ft (ch6)	**68.8**	67.3	66.3	64.1	62.3
Ko-ft (ch4)	68.4	66.4	65.2	63.4	61.7
misK-w2v	68.3	**67.6**	**66.7**	65.4	**64.2**
misK-ft	68.1	67.2	66.4	**65.5**	64.1

Table 3: Result of the sentiment classification experiment. The noise level is the probability that each word in the dataset will be mistyped. Best results are highlighted in bold font.

task. Also, we replaced the word embeddings of the basic LSTM network with dropout regularization (Kim et al., 2016). All embedding dimensions are set to 300 and the dropout probability is set to 0.5. Each embedding model is measured using perplexity, which is widely used to measure language modeling performance (Katz, 1987). The lower the perplexity value measured, the more likely the model predicts the next word correctly. Besides, in the downstream task, we used intentionally generated typo data in the experiment to verify the robustness of embedding for noisy text.

4.4.1 Datasets

The language modeling dataset was released by the Korean Wikipedia, published on June 1, 2019. Each embedding technique was trained based on this corpus in its way. The code to extract and refine the text from the Wikipedia database dump followed the open-source software[2].

4.4.2 Results

The results of the language modeling experiments are shown in Table 2. Regardless of the noise level, *Word2vec* has the lowest performance. In other words, it can be seen that the word-level embeddings separated by blank spaces are not suitable for Korean text irrespective of the response to misspelled words. *Ko-ft* performs best when the noise level is below 20%. Meanwhile, the performance of *misK* is better than the other models as the noise level rises to more than 10%. The fact that *misK* showed superior performance means it can stably respond to various typos in Hangeul text.

4.5 Sentiment Classification Task

The performance of each model is compared using the sentiment analysis experiment on movie reviews. We represent the reviews by averaging the word embeddings, constructing a review, and training through the logistic regression classifier for a fair comparison.

4.5.1 Datasets

For sentiment classification, we used the Naver Sentiment Movie Corpus[3] as our classification data. For the typos used in the real world, we did not proceed with any form of preprocessing. Therefore, it includes typos or slang used by Koreans. The experiment was also conducted by increasing noise

[2]https://github.com/attardi/wikiextractor

[3]https://github.com/e9t/nsmc/

	In-Vocabulary				Out-Of-Vocabulary			
	컴퓨터	대한민국	대학교	자동차	커뮤펕	댕나민국	댕가교	자도앚
Word2vec	개인용	한국	대학교의	승용차	-	-	-	-
	그래픽	국내	대학교에서	자동차의	-	-	-	-
	컴퓨터의	대한민국의	대학교의	자동차가	-	-	-	-
FastText	컴퓨터의	대한민국은	대학교는	자동차를	커뮤니티	**중화민국**	침례교	자
	컴퓨터에	대한민국을	대학교와	자동차도	객실	**대한민국**	외교	재차
	컴퓨터로	대한민국이	대학교의	자동차에	앱	**대한민국과**	폐교	자작
misK-ft	컴퓨터는	한국	대학교의	자동차에	코뮌	**대한민국**	**대학원을**	**자동차**
	컴퓨팅	대한민국에서	대학의	자동차의	**소프트웨어**	**중화민국**	**대학원의**	**자동차의**
	소프트웨어	국내	대학교는	자동차가	**컴퓨터는**	웹	**학사**	광고

Table 4: In OOV experiments, words that are related to the original word are highlighted in bold font. 커뮤니티-community, 객실-room, 앱-app, 소프트웨어-software, 침례교-baptist church, 외교-diplomacy, 폐교-closing schools, 학사-bachelor, 자-ruler, 재차-again, 자작-oneself.

level of the evaluation dataset and intentionally producing misspelled words to compare robustness to the typos.

4.5.2 Results

Table 3 shows that *Ko-ft* has the highest performance in the sentiment classification experiment results in the clean environment. Besides, *Word2vec* showed significantly reduced performance than *Ko-ft*. This is because several OOVs were included in the test set. However, as the noise level increases, the performance of *misK* overtakes that of the *Ko-ft*. It confirms that the performance of *misK* decreases relatively smoothly as the amount of noise in the data rises. In other words, *misK* is more robust to the noisy data than the others.

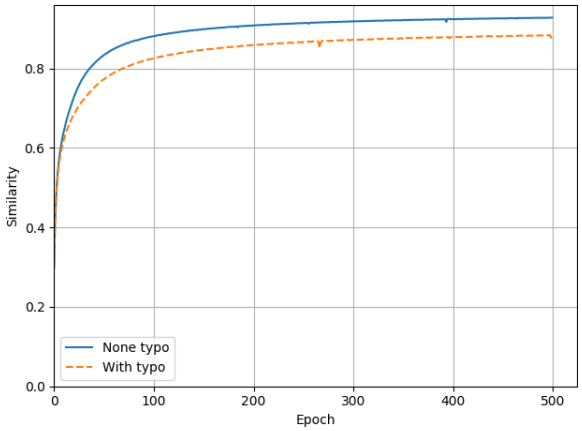

Figure 4: Cosine similarity between the generated word vector and pre-trained word vector.

5 Analysis

In this section, we analyze the proposed methodology. We find the nearest neighbor words mapped close to the misspelled word to evaluate robustness of the typo embeddings qualitatively.

5.1 Nearest Neighbor of Words

We check the nearest neighbor of words to compare the performance of each word embedding. This evaluation is to identify which words are located around the represented target words (컴퓨터-computer, 대한민국-Korea, 대학교-university, 자동차-automobile) and typos transformed from these words.

Table 4 shows the experimental results. The results with In-Vocabulary show that all three models produce the appropriate embeddings. However, the results from the OOV confirm that *Word2vec* cannot deal with the typos. In contrast, *Ko-ft* and *misK* can generate embeddings for these unseen words to investigate these nearest words. In general, the typo embeddings generated by the *Ko-ft* are inaccurate (note the meaning of the words described in the caption). On the other hand, we deduce that words with appropriate meanings are mapped with our proposed model, *misK*.

5.2 Similarity between Generated and Pre-trained Word Vector

Experiments show superiority in the noisy text of our proposed embedding generation model. We measure the cosine similarity between the pre-trained word embeddings and those generated by our proposed model. As shown by the dotted curve in Figure 4, when training data includes intentional typos, the cosine similarity between the pre-trained word embeddings and those generated by the proposed model could not be increased beyond 0.86. On the other hand, in the absence of typos, the similarity increases to around 0.95. Besides, there is a

tradeoff between the ability to map typos closely to the original word and the ability to mimic pre-trained word vectors.

6 Conclusion

In this paper, we proposed a robust Hangeul embedding model against typos. For this purpose, we intentionally generated typos to train word embeddings. A channel attention mechanism is used to utilize the structure information found in Korean words selectively. This reflects the distinctive characteristics of Hangeul, which combines several morphemes to form a single word, without depending on the performance of a morpheme analyzer. Based on this, the training progresses from the extracted jamo n-gram feature of the target words through the convolution layer to aim the pre-trained word embeddings. The proposed embedding model outperformed the current state-of-the-art in the intrinsic task. Also, we verify that the original research goal was achieved by functioning a stable embeddings even in the extrinsic tasks. We plan to apply the proposed methodology to various downstream tasks (e.g., POS tagging and machine translation) and other agglutinative languages, such as Japanese and Turkish.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This research was supported by the Basic Research Program through the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2020R1A4A1018309), the NRF grant funded by the Korea government (MSIT) (No. 2018R1A2A1A05078380), and Institute of Information communications Technology Planning Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-00079, Artificial Intelligence Graduate School Program (Korea University)).

References

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *Proceedings of International Conference on Learning Representations*.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters. In *Proceedings of the International Conference on Computational Linguistics*, pages 6903–6915.

Hee-Won Jeon, Daniel Huang, and Hae-Chang Rim. 2010. Analyzing of hangul search query spelling error patterns and developing query spelling correction system based on user logs. In *Proceedings of Annual Conference on Human and Language Technology*, pages 15–21.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3):400–401.

Yeachan Kim, Kang-Min Kim, Ji-Min Lee, and SangKeun Lee. 2018. Learning to generate word representations using subword information. In *Proceedings of the International Conference on Computational Linguistics*, pages 2551–2561.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the American Association for Artificial Intelligence Conference on Artificial Intelligence*.

Dongjun Lee, Yubin Lim, and Taekyoung Kwon. 2018. Morpheme-based efficient korean word embedding. *Journal of Korean Institute of Information Scientists and Engineers*, 45(5):444–450.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1064–1074.

Valentin Malykh, Varvara Logacheva, and Taras Khakhulin. 2018. Robust word vectors: Context-informed embeddings for noisy texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 54–63.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3111–3119.

Kyeong-Min Nam and Yu-Seop Kim. 2016. A word embedding and a josa vector for korean unsupervised semantic role induction. In *Proceedings of the American Association for Artificial Intelligence Conference on Artificial Intelligence*, pages 4240–4241.

Sungjoon Park, Jeongmin Byun, Sion Baek, Yongseok Cho, and Alice Oh. 2018. Subword-level word vector representations for korean. In *Proceedings of*

the Annual Meeting of the Association for Computational Linguistics, pages 2429–2438.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234.

Yuval Pinter, Robert Guthrie, and Jacob Eisenstein. 2017. Mimicking word embeddings using subword rnns. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 102–112.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 649–657.