Learning to Generate Word Representations using Subword Information

Yeachan Kim, Kang-Min Kim, Ji-Min Lee and SangKeun Lee
Korea University, Seoul, Republic of Korea
{yeachan,kangmin89,jm94318,yalphy}@korea.ac.kr

Abstract

Distributed representations of words play a major role in the field of natural language processing by encoding semantic and syntactic information of words. However, most existing works on learning word representations typically regard words as individual atomic units and thus are blind to subword information in words. This further gives rise to a difficulty in representing out-of-vocabulary (OOV) words. In this paper, we present a character-based word representation approach to deal with these limitations. The proposed model learns to generate word representations from characters. In our model, we employ a convolutional neural network and a highway network over characters to extract salient features effectively. Unlike previous models that learn word representations from a large corpus, we take a set of pre-trained word embeddings and generalize it to word entries, including OOV words. To demonstrate the efficacy of the proposed model, we perform both an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. Experimental results show clearly that the proposed model significantly outperforms strong baseline models that regard words or their subwords as atomic units. For example, we achieve as much as 18.5% improvement on average in perplexity for morphologically rich languages compared to strong baselines in the language modeling task.

1 Introduction

Recently, distributed representations of words have become an inevitable component in systems of natural language processing (NLP), *e.g.,* chunking and named entity recognition (Turian et al., 2010), text classification (Kim, 2014), and question answering (Kumar et al., 2016). In such systems, word representations play a vital role by capturing useful information of words, leading to the improved performance of the systems. Most existing approaches to learning word representations are learned in an unsupervised learning manner from a large corpus and, for simplicity and effectiveness, they regard words as individual atomic units in a learning phase (Bengio et al., 2003; Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014).

However, word-based approaches typically ignore subword information in words. It prevents the approaches from taking into account the characteristic of words, *"words are lexically related when they share some morphemes or semantic elements"* (Williams, 1981). This constraint further makes representing out-of-vocabulary (OOV) words difficult. Even if lexically related words have been observed, word-based approaches fail to represent such words that are not included in a pre-defined vocabulary.

Perhaps the simplest way to sidestep this problem is to use one or more pseudo-words (*e.g.,* `<unk>`) to represent OOV words. However, this is not a fundamental remedy, as OOV words have their own meanings in many cases, such as variants, compounds, and misspellings. This problem is more evident when the domain of a text in which the word is learned differs from the domains to be applied to an NLP task. For example, the word *"talking"* and *"touchdown"* in the domain with generic text can be represented as *"tlking"* and *"touchdooown"* in the domain of social media. In this case, although these words share a similar sequence, representing words in other domain is difficult.

In this paper, we present a character-based approach to generate word representations. Unlike previous works that adopt unsupervised learning from a corpus, the proposed model learns to generate word representations based on the supervised learning of pre-trained word embeddings. We use a convolutional neural network (CNN) (LeCun et al., 1998) to extract subword features over characters and utilize a highway network (Srivastava et al., 2015) to relate salient subword features to pre-trained word embeddings.

The proposed model has several benefits. As we regard characters as our atomic units for learning word representations, we have an ability to generate representations for all kinds of words, including OOV words. Moreover, we take a set of pre-trained word embeddings and generalize it to represent all words. This enables the proposed model to learn word representations efficiently without having to train with a large corpus.

To evaluate the proposed model, we perform an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. We specifically conduct these tasks in various languages, as subword information is a characteristic of many languages such as French and Spanish (which have more than 40 different inflected forms for verbs). Experimental results show that the proposed model outperforms strong baselines that regard words or their subwords as atomic units, while representing OOV words quite well. In summary, we make the following contributions:

- We propose a character-based approach to generate word representations utilizing a convolutional neural network and a highway network.
- We leverage a set of pre-trained word embeddings and generalize it to all word entries. It allows us to efficiently learn word representations in a learning phase without training in a large corpus.
- We verify the efficacy of the proposed model in both an intrinsic and an extrinsic task with various languages. Our experimental results reveal that the word representations derived from our model considerably improve the performance on both of tasks in most languages.

The remainder of this paper is organized as follows. In Section 2, we discuss related works. In Section 3, we describe the proposed model. We report our performance evaluation results and analyze our methodology in detail in Sections 4 and 5, respectively. Finally, we conclude this paper in Section 6.

2 Related Works

Recently, many works have been proposed to improve the quality of word representations using subword[1] information. Some works used word and subword representations concurrently to produce better word representations. For example, (Qiu et al., 2014) proposed a model that jointly learns word and subword representations for word embeddings. They found that this joint model is useful in producing better representations for rare words. Similarly, (Mousa et al., 2013) proposed a deep neural network for language modeling, where the inputs to the network are a mixture of words and morphemes as well as their respective features. This work showed that utilizing morphemes could enhance lexical coverage and mitigate the problem of data sparseness for a n-gram language model.

In a similar manner, many works have proposed utilizing only subword information to produce word representations. For example, (Wieting et al., 2016) proposed embedding models for words and sentences using character n-gram count vectors. Although they require specific paragraph pairs for training, these models showed promising results in representing words while considering subword information. Recently, (Bojanowski et al., 2017) proposed FastText, which is an extension of the continuous skip-gram model (Mikolov et al., 2013). In this model, a word representation is replaced with the sum of character n-gram representations. This work showed that utilizing subword information allows the model to be trained efficiently by sharing the n-gram parameters between words. (Luong et al., 2013) utilized recursive neural networks in which inputs are morphemes of words. The proposed model showed better estimations of rare and complex words.

[1]We use the term "subword" for representing units smaller than a word, such as morphemes or character n-grams.

Some of the above-mentioned works (Luong et al., 2013; Mousa et al., 2013; Qiu et al., 2014), which used morphemes and normalized words, require segmentation tools for morphological segmentation. By contrast, we do not have to use segmentation tools as we use only characters to produce word representations. In addition, most works learn word representations from a large corpus using an unsupervised learning manner. However, we do not require training from a large corpus because we employ a set of pre-trained word embeddings.

Some works employed subword information, especially by using characters, to represent words in NLP tasks. For example, (Santos and Zadrozny, 2014) and (Ling et al., 2015) used a CNN and a recurrent neural network (RNN), respectively, over character representations for part-of-speech tagging. For a language modeling, (Kim et al., 2016) used a CNN and a highway network which achieved better performance than word-based models. That network shares a similar architecture with ours. However, our architecture is different from the early network (Kim et al., 2016) in that ours is designed with a simpler network architecture and a different output layer based on the pre-trained word embeddings. For neural machine translation, (Ruiz Costa-Jussà and Rodríguez Fonollosa, 2016) and (Luong and Manning, 2016) incorporated character representations to produce OOV word representations and achieved improved performance over models that used unknown tokens to represent OOV words.

Most closely related to our work is Mimick (Pinter et al., 2017), which is a bidirectional long short-term memory (LSTM)-based word representation model. It learns a function from characters to word embeddings to represent OOV words. In terms of the quality of OOV word representations, Mimick has shown promising results in syntactical properties. By contrast, we explore a CNN-based architecture to learn features from pre-trained word embeddings. As words consist of locally salient features such as prefixes, roots, and suffixes, a CNN architecture is expected to be effective at extracting these features.

3 Proposed Model

Figure 1 provides an overview of the proposed model. Our learning strategy is to *reconstruct* pre-trained word embeddings from character representations. To this end, we first extract subword features using a character-based convolution module. We then utilize a highway network to adaptively combine the subword features derived from the convolution module. In the optimization step, we make this resultant representation similar to its pre-trained word embedding.

3.1 Character-based Convolution Module

A CNN is designed to capture the most informative local aspects of a particular task (LeCun et al., 1998). CNNs have been widely used in not only computer vision but also recently in NLP (Kim, 2014; Cao and Lu, 2017) because of its ability to extract local context features such as n-grams. Accordingly, we apply this network on a character sequence to extract its subword features.

We first use a one-hot encoding to quantize the character inputs, which has a value of 1 at the index of the corresponding character, otherwise it has a value of 0. We concatenate these character representations to constitute a word, which is represented as $r \in \mathbb{R}^{|V_c| \times n}$ where V_c is a vocabulary of characters and n is the length of a word. Thus, the input representation for convolution has the following shape:

$$r = c_1 \oplus c_2 \oplus \cdots \oplus c_n = \begin{bmatrix} | & | & \cdots & | \\ c_1 & c_2 & \cdots & c_n \\ | & | & \cdots & | \end{bmatrix}$$

where c_i is a one-hot encoding for representing the i-th character in a word and $\oplus$ is a concatenation operator. We then insert zero-paddings to the side of these character representations r in order to perform the same number of convolutions for all characters. The number of zero-paddings on each side is $\frac{h-1}{2}$, where h is the width of convolution filter F. We perform the convolution operation on the padded representation r using the following equation:

$$s_i = tanh(F \cdot r_{i:i+h-1} + b) \tag{1}$$

where filter F is represented as $F \in \mathbb{R}^{|V_c| \times h}$ and $r_{i:i+h-1}$ is a sequence representation between character representation c_i to c_{i+h-1}. Then, we apply $tanh$ as the non-linear function to the convolution results.

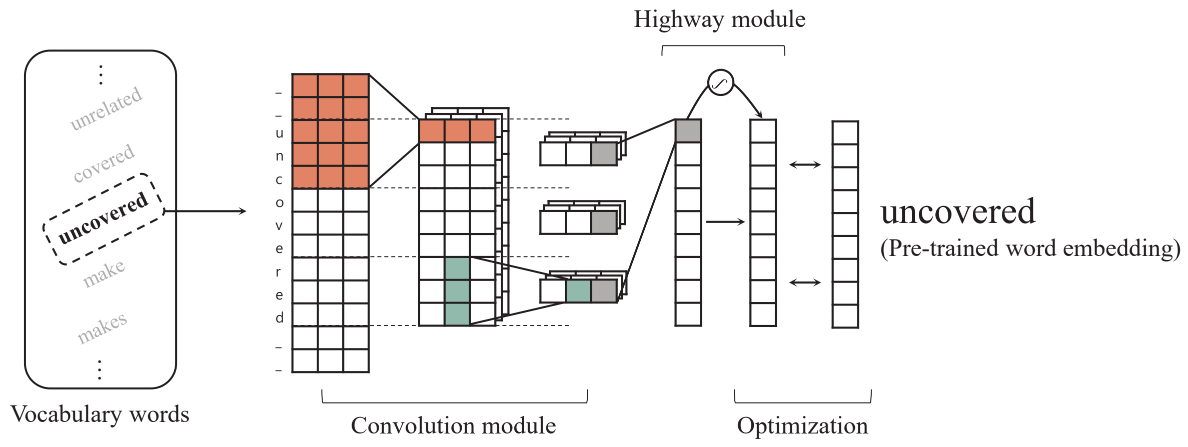


Figure 1: Overview of the learning phase in the proposed model. In this example, we process as input the word "*uncovered*" that is included in pre-trained word embeddings. In the convolution module, we use three filters with 5-width (thus, we add two zero-paddings) and other width filters, and we set the width of the stride to 3 for pooling. In the highway module, we use a single highway network. We then make this resultant representation similar to the pre-trained word embedding of "*uncovered*". After training converges, we can represent words that are lexically related to the input word but that are not included in pre-trained word embeddings, like "*uncovering*", as these words share similar character sequences.

The resultant vector s has the same length as the input word because of the zero-paddings. To extract salient features, we apply a max-pooling operation on the vector s. Specifically, we use max pooling with stride to derive locally salient features that can contain suffixes, roots, and prefixes. As a result, each feature derived from stride pooling has a summary of subword information in a character sequence. The equation for stride pooling is as follows:

$$e_i = \max\left[s_{k\cdot i:k\cdot(i+1)-1}\right] \tag{2}$$

where k is the length of a stride and $s_{k\cdot i:k\cdot(i+1)-1}$ is a sequence representation with one stride in s. As stride pooling results in features that vary in length according to the length of words, we therefore simply sum the elements of the vector e to produce fixed-sized features.

Thus far, we have described the procedure for a single filter. In our experiment, we use multiple filters with different widths to extract various subword features. We simply concatenate the results of each filter to merge the various features.

3.2 Highway Network Module

A highway network is an architecture that allows a model to train a very deep network efficiently (Srivastava et al., 2015). This network learns to route information using a gating mechanism. Recently, (Kim et al., 2016) utilized a highway network on a CNN for language modeling. It turned out that a highway network is helpful to a CNN, as it adaptively combines local features from individual filters. By following this line of work, we apply this network to adaptively combine subword features derived from a convolution module. It allows the proposed model to relate salient subword features with pre-trained word embeddings in a learning phase. The basic equation of a highway network is as follows:

$$y = T \odot H + C \odot e \tag{3}$$

where H is a non-linear transformation on its input e, T is the *transform gate* and C is the *carry gate*. In our work, we use C as a *(1-T)* for simplicity. For the function H and T, we use the following equations:

$$\begin{aligned} T &= \sigma(W_t \cdot e + b_t) \\ H &= \tanh(W_h \cdot e + b_h) \end{aligned} \tag{4}$$

where W_t, W_h and b_t, b_h are square matrices and biases[2], respectively. Finally, we set the size of resultant vector y as the pre-trained word embedding using a linear transformation for an optimization as follows:

$$w = W \cdot y + b \tag{5}$$

where w is a final word representation resulting from the proposed model, and W and b are parameters of linear transformation. Through the aforementioned procedures, we generate word representations while considering subword information.

3.3 Optimization with Pre-trained Word Embeddings

The proposed model reconstructs pre-trained word embeddings from characters. To make the word representation derived from our model learn the features of pre-trained word embeddings, we use the *squared Euclidean distance* as our objective function as follows:

$$E = \sum_{v \in V_w} \|w_v - \hat{w}_v\|^2 \tag{6}$$

where w_v is a word representation from our model and $\hat{w}_v$ is a pre-trained word embedding for a word v in the vocabulary. After training converges, we discard pre-trained word embeddings $\hat{w}$ and use w as our word representations for NLP tasks.

4 Experiments

To evaluate the proposed model, we perform both an intrinsic and an extrinsic task which are word similarity and language modeling, respectively. The word similarity evaluates the ability to capture the semantic similarity of words. The language modeling evaluates the usefulness of the proposed word representations in an NLP task. We specifically conduct both of tasks in various languages, as the subword information is a language-dependent feature.

4.1 Experimental Settings

We train the proposed model using pre-trained word embeddings. To simulate the settings having pre-trained word embeddings for various languages, we use word2vec (Mikolov et al., 2013), which is trained in Wikipedia[3] that was released on December 1, 2017 for seven languages: Arabic, Czech, German, English, Spanish, French, and Russian. The preprocessing for a corpus of each language is done using Natural Language Toolkit (NLTK) Tweet Tokenizer[4]. Based on (Lai et al., 2016), we assign to word vectors a 100 dimension, which has shown relatively better performance in both an intrinsic and an extrinsic task. In our experiments, we examine the following models:

- **word2vec** (Mikolov et al., 2013): This is a model that regards words as individual atomic units and adopts a window-based learning technique. Since this method is a word-based approach, it is incapable of representing OOV words. As a default for OOV words, we use null vectors for the word similarity task, and we randomly initialize and fine-tune them for the language modeling task. Of the two models of word2vec, CBOW and skip-gram, we use a skip-gram architecture for learning word representations, as it generally shows better performance than does CBOW (Lai et al., 2016).

- **FastText** (Bojanowski et al., 2017): This is an extension of word2vec and regards character n-grams as atomic units. The learning technique is similar to that of word2vec. We set the dimension of words to 100 that is same with other models and, for other settings, we follow the best settings described in the paper (Bojanowski et al., 2017).

[2]According to the suggestion of (Srivastava et al., 2015), we initialize b_t as -3 for making the networks initially biased towards *carry* behavior.

[3]https://www.wikipedia.org/

[4]https://www.nltk.org/

- **Mimick** (Pinter et al., 2017): This is a character-based model that learns a function from characters to word embeddings for representing words. For pre-trained word embeddings for this model, we use the previously mentioned skip-gram version of word2vec. In our experiments, we compare the best Mimick model, which uses pre-trained word embeddings for vocabulary words and generated embeddings for OOV words. We set the hidden dimension of an LSTM to 256 for better convergence, and other parameters are same as those of the paper (Pinter et al., 2017).

- **GWR** (proposed model): This is our proposed model, denoted as the generated word representation (GWR), and we use the word2vec for pre-trained word embeddings. We generate all words that are observed and use them for overall tasks.

We implemented the proposed architecture using TensorFlow frameworks (Abadi et al., 2016) and trained our model in a single machine equipped with Intel Core i5, 16GB of RAM, and an NVIDIA GeForce GTX 1080 with 8GB of RAM. The details of the training parameters are shown in Table 1. We chose these listed parameters by validating the word similarity task.

Type	value
optimizer	Adam (Kingma and Ba, 2014)
learning rate	0.001
batch size	50
the width of filter	[1,2,3,4,5,6,7]
the width of stride	3
# of filters	max(200, 50*width)
# of highway networks	2

Table 1: Experimental settings of the proposed model

4.2 Word Similarity

We first perform word similarity task to evaluate the ability of models to capture the semantic similarity among words. The performance of word similarity is measured by computing the Spearman's correlation coefficient between human judgment and the cosine similarity between word pairs.

4.2.1 Datasets

We collect word similarity datasets for six languages: Arabic (Ar), German (De), English (En), Spanish (Es), French (Fr), and Russian (Ru). For English, we compare the models on the WS353 (Finkelstein et al., 2001) and SL999 (Hill et al., 2014) datasets. For German, we conduct experiments on the GUR350 and ZG222 (Zesch and Gurevych, 2006) datasets, and we evaluate the models on the MC30 and WS353 datasets for Arabic and Spanish that are translated by (Hassan and Mihalcea, 2009). We evaluate French word vectors on the translated dataset RG65 (Joubarne and Inkpen, 2011), and for Russian, we evaluate the vectors on the HJ dataset introduced by (Panchenko et al., 2016).

4.2.2 Results

Table 2 shows the overall results. We first observe that subword-based learning methods (FastText, Mimick, GWR) show superior performance compared to that of the word-based method (word2vec) on most languages and datasets. This indicates that considering subword information is effective at representing words and useful with many languages. Among the subword-based methods, GWR outperforms Mimick in all languages except for English. We conjecture that this result can be explained by the fact that a CNN architecture is helpful in extracting local aspect features. In addition, GWR shows competitive performance with FastText that learns word representations from a large corpus (*i.e.,* GWR shows better performance in six out of nine datasets).

Language	Dataset	word2vec	FastText	Mimick	GWR$^-$	GWR
Ar	WS353	34.9	**52.1**	44.9	36.3	43.8
De	GUR350	34.6	48.7	47.6	35.4	**53.2**
	ZG222	16.2	35.4	36.5	16.4	**38.7**
En	SL999	29.5	27.7	**30.3**	28.6	29.8
	WS353	63.1	64.1	63.7	63.9	**64.6**
Es	MC30	40.6	**72.1**	66.8	41.4	69.8
	WS353	29.5	50.0	40.7	30.8	**50.2**
Fr	RG65	46.8	58.6	51.3	48.1	**60.5**
Ru	HJ	30.3	**60.1**	42.1	31.4	45.5

Table 2: Word similarity results on entire datasets (Spearman's $\rho \times 100$). GWR$^-$ implies a model using null vectors for OOV words. Best results are in bold.

To confirm that OOV word representations derived from GWR indeed improve the performance, we assess the performance of our model when used with null vectors for OOV words (GWR$^-$). The results show that the model generating OOV words (GWR) considerably outperforms the model that does not generate them (GWR$^-$). This improved performance indicates that the proposed model effectively expands the lexical coverage and generates representations of OOV words quite well. In addition, the GWR$^-$ model shows slightly better performance than that of the word2vec. It also shows its effectiveness at considering subword information in representing words. The differences between these two representations for vocabulary words are described in Section 5.2.

4.3 Language Modeling

As a second experiment, we perform language modeling as an extrinsic task to verify the usefulness of the proposed model when performing an NLP task. To evaluate the performance of word embeddings, we use an LSTM network with dropout regularization, which is used in (Bojanowski et al., 2017). Through the development set in (Botha and Blunsom, 2014), we set the hidden dimension of an LSTM to 256 and the probability of dropout to 0.5. In our experiments, we replace only the component of word embeddings to solely evaluate word embedding performance. The performance of language modeling is computed based on perplexity, which is a widely used metric in evaluating the usefulness of language modeling. Lower perplexity means a better model.

4.3.1 Datasets

We perform language modeling task using the datasets introduced by (Botha and Blunsom, 2014) on six languages: Czech (Cs), German (De), English (En), Spanish (Es), French (Fr), and Russian (Ru). Each dataset contains roughly one million tokens for training, and we follow the same pre-processing and data splits as reported in (Botha and Blunsom, 2014).

Language	$\lvert V_w \rvert$	word2vec	FastText	Mimick	GWR
Cs	46k	412.7	397.9	401.7	**352.1**
De	36k	263.8	248.2	258.6	**233.2**
En	17k	259.8	256.3	257.2	**240.7**
Es	27k	190.7	186.9	189.2	**174.7**
Fr	25k	205.2	207.7	202.4	**185.6**
Ru	62k	311.5	282.7	293.1	**243.1**

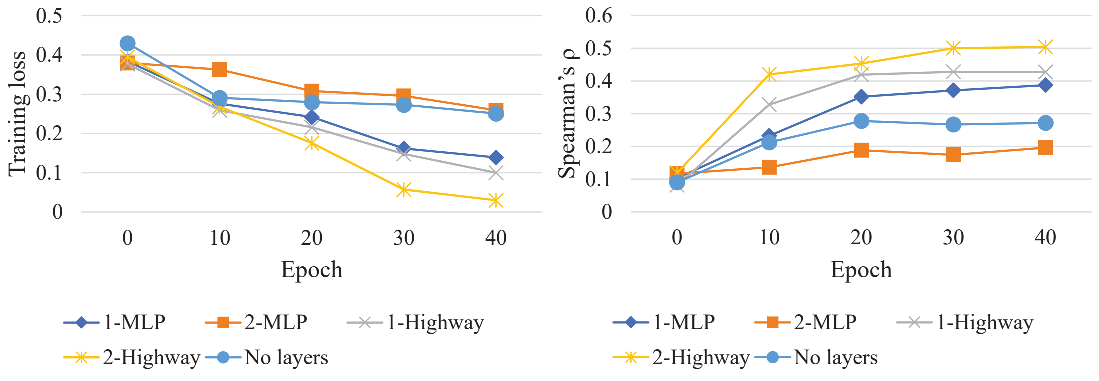
Table 3: Test perplexity on the language modeling for six languages. Best results are in bold.

4.3.2 Results

Table 3 shows the results. We observe a similar trend as with the performance of the word similarity task. The subword-based methods show better performance than the word-based method with most languages. Furthermore, the proposed GWR shows superior performances over these subword-based methods by a large margin. Based on the performance of the word2vec used in training GWR, we achieve notably improved performance in morphologically rich languages. For example, the performance on Slavic languages (Cs, Ru), which are morphologically rich, shows better perplexity reduction (15 and 22% for Czech and Russian, respectively) than other languages that are less significant languages in terms of morphology. This suggests that GWR is more useful and effective in morphologically rich languages.

5 Analysis

5.1 Effect of Highway Networks



(a) Training loss (b) Word similarity performance (Spearman's ρ)

Figure 2: Training loss and word similarity performance with different architectures. The term of k-MLP indicates that k-layered MLP are stacked to the proposed model, and it is a same manner with a highway network.

As a component of the proposed model, we use a highway network rather than a multi-layer perceptron (MLP) that is widely used in CNNs (Kim, 2014; Cao and Lu, 2017). To show the effectiveness of a highway network in our architecture, we assess the training loss and performance of word similarity during training step when each network (highway network, MLP) is stacked to the proposed model. As a representative language in this analysis, we use the WS353 dataset in Spanish, which is a morphologically rich language. The results are shown in Figure 2.

In Figure 2a, we can observe that a highway network basically shows faster convergence than does MLP with respect to training loss. The interesting results occur when the model is deeper with each network (2-highway, 2-MLP). While 2-MLP shows slower convergence than 1-MLP, 2-Highway shows faster convergence than 1-Highway. These results are related to the property of a highway network, which allows the model to train more deeply.

Figure 2b shows the performance of word similarity task. We observe a similar trend with the results of the training loss. The highway network shows superior performance compared to MLP. Furthermore, when 2-Highway is stacked to the proposed model, it shows better performance than that of 1-Highway. In contrast, 2-MLP performs worse than 1-MLP, even when the proposed model does not have a layer (No layers).

Through this analysis, we observe that a highway network enhances the semantic similarity of words, and makes the training faster compared to a simple MLP. Furthermore, stacked highway networks enable us to train more deep model that leads the strong performance.

	In-Vocabulay				**Out-of-Vocabulary**			
	connect	teach	taxicab	computes	bluejacket	vehicals	globalise	computerization
word2vec	connect	teach	taxicab	computes	-	-	-	-
	connecting	learn	taxi	calculates				
	communicate	instruct	minibus	logarithmic				
	interact	enroll	motorbike	satisfies				
	linking	educate	truck	generates				
GWR	connect	teach	taxicab	computes	jacket	vehicles	global	computational
	connecting	instruct	taxi	compute	trouser	bicycles	worldwide	computation
	connects	learn	limousine	calculates	sleeve	cars	globally	computerized
	connected	teaching	minibus	logarithmic	t-shirt	automobiles	globalisation	visualization
	linking	understand	motorbike	computable	pants	trailers	globalization	computations

Table 4: Nearest neighbors of word representations for vocabulary and OOV words. Nearest words are sorted in descending order of similarity that computed by cosine similarity.

5.2 Nearest Neighbor of Words

To explore the quality of GWR, we perform a qualitative analysis by finding the nearest neighbors of word representations. Table 4 shows the nearest neighbors of word representations learned from the word2vec and GWR in English.

From nearest neighbors of OOV words, we discover the following: (i) GWR represents word-variants and compound words well (*computerization, bluejacket*); (ii) GWR captures different word styles (*globalise*) and syntactically related words (*global-globally-globalization*); (iii) GWR shows robustness on misspellings (*vehic**a**ls*); (iv) GWR captures semantically related words in most OOV words (*bluejacket-pants, vehicals-bicycles*).

We also derive some interesting results for nearest neighbors of vocabulary words. We observe that semantically or syntactically related words are located in nearest neighbors (*taxicab-minibus, teach-teaching*). Moreover, the proposed model renders lexically related word representation more similar. For example, the neighbors of "*connect*" show that the word-variants (*connects-connected-connecting*) are more closely located than are pre-trained word embeddings in vector space. This rendering satisfies the characteristic of words about *lexical relatedness* (Williams, 1981). Other words also show a similar trend (*teach-teaching*, *computes-compute-computable*). This is probably because the lexically related words share many parts of character sequences.

5.3 Training time

We measure the training time for GWR and corpus-based models. For a fair comparison, we use the same hardware resources, and record the training time when each model achieves the best performance on English data. Word2vec and FastText, which are corpus-based models, take approximately 10 and 14 hours, respectively. In contrast, GWR takes 52 minutes for learning 100,000 words in pre-trained word embeddings (*i.e.,* 16x faster than FastText). This result demonstrates that GWR produces word representations efficiently.

6 Conclusion

In this paper, we have proposed a character-based approach, denoted as GWR, to generate word representations. To this end, we have used a CNN and a highway network to extract salient subword features, and then learned features from pre-trained word embeddings. Notably, the proposed model allows us to produce high quality representations of OOV words by considering subword information. Experimental results demonstrate the efficacy of our methodology and clearly show that the proposed model works well for NLP tasks. We plan to apply the proposed model to different domains such as text classification and named entity recognition.

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (number 2015R1A2A1A10052665).

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation*, pages 265–283.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, pages 1137–1155.

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

Jan Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *Proceedings of the International Conference on Machine Learning*, pages 1899–1907.

Shaosheng Cao and Wei Lu. 2017. Improving word embeddings with convolutional feature learning and subword information. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 3144–3151.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the International Conference on Machine Learning*, pages 160–167.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of the International Conference on World Wide Web*, pages 406–414.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1192–1201.

Felix Hill, Kyunghyun Cho, Sébastien Jean, Coline Devin, and Yoshua Bengio. 2014. Embedding word similarity with neural machine translation. *CoRR*, abs/1412.6448.

Colette Joubarne and Diana Inkpen. 2011. Comparison of semantic similarity for different languages using the google n-gram corpus and second-order co-occurrence measures. In *Proceedings of the Canadian Conference on Artificial Intelligence*, pages 216–221.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the Association for the Advancement of Artificial Intelligence*, pages 2741–2749.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1781.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the International Conference on Machine Learning*, pages 1378–1387.

Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. 2016. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, pages 2278–2324.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1054–1063.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. *Proceedings of the Annual Conference on Natural Language Learning*, pages 104–113.

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 International Conference on Neural Information Processing Systems*, pages 3111–3119.

Amr El-Desoky Mousa, Hong-Kwang Jeff Kuo, Lidia Mangu, and Hagen Soltau. 2013. Morpheme-based feature-rich language models using deep neural networks for lvcsr of egyptian arabic. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 8435–8439.

Alexander Panchenko, Dmitry Ustalov, Nikolay Arefyev, Denis Paperno, Natalia Konstantinova, Natalia Loukachevitch, and Chris Biemann. 2016. Human and machine judgements for russian semantic relatedness. In *Proceedings of the International Conference on Analysis of Images, Social Networks and Texts*, pages 221–235.

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.

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.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Co-learning of word representations and morpheme representations. In *Proceedings of the International Conference on Computational Linguistics*, pages 141–150.

Marta Ruiz Costa-Jussà and José Adrián Rodríguez Fonollosa. 2016. Character-based neural machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 357–361.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the International Conference on Machine Learning*, pages 1818–1826.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 2377–2385.

Joseph P Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1504–1515.

Edwin Williams. 1981. On the notions "lexically related" and" head of a word". *Linguistic Inquiry*, pages 245–274.

Torsten Zesch and Iryna Gurevych. 2006. Automatically creating datasets for measures of semantic relatedness. In *Proceedings of the Workshop on Linguistic Distances*, pages 16–24.