Character n-gram Embeddings to Improve RNN Language Models

Abstract

This paper proposes a novel Recurrent Neural Network (RNN) language model that takes advantage of character information. We focus on character n-grams based on research in the field of word embedding construction (Wieting et al. 2016). Our proposed method constructs word embeddings from character n-gram embeddings and combines them with ordinary word embeddings. We demonstrate that the proposed method achieves the best perplexities on the language modeling datasets: Penn Treebank, WikiText-2, and WikiText-103. Moreover, we conduct experiments on application tasks: machine translation and headline generation. The experimental results indicate that our proposed method also positively affects these tasks.

Introduction

Neural language models have played a crucial role in recent advances of neural network based methods in natural language processing (NLP). For example, neural encoder-decoder models, which are becoming the de facto standard for various natural language generation tasks including machine translation BIBREF1 , summarization BIBREF2 , dialogue BIBREF3 , and caption generation BIBREF4 can be interpreted as conditional neural language models. Moreover, neural language models can be used for rescoring outputs from traditional methods, and they significantly improve the performance of automatic speech recognition BIBREF5 . This implies that better neural language models improve the performance of application tasks.

In general, neural language models require word embeddings as an input BIBREF6 . However, as described by BIBREF7 , this approach cannot make use of the internal structure of words although the internal structure is often an effective clue for considering the meaning of a word. For example, we can comprehend that the word `causal' is related to `cause' immediately because both words include the same character sequence `caus'. Thus, if we incorporate a method that handles the internal structure such as character information, we can improve the quality of neural language models and probably make them robust to infrequent words.

To incorporate the internal structure, BIBREF7 concatenated character embeddings with an input word embedding. They demonstrated that incorporating character embeddings improved the performance of RNN language models. Moreover, BIBREF8 and BIBREF9 applied Convolutional Neural Networks (CNN) to construct word embeddings from character embeddings.

On the other hand, in the field of word embedding construction, some previous researchers found that character INLINEFORM0 -grams are more useful than single characters BIBREF0 , BIBREF10 . In particular, BIBREF0 demonstrated that constructing word embeddings from character INLINEFORM1 -gram embeddings outperformed the methods that construct word embeddings from character embeddings by using CNN or a Long Short-Term Memory (LSTM).

Based on their reports, in this paper, we propose a neural language model that utilizes character INLINEFORM0 -gram embeddings. Our proposed method encodes character INLINEFORM1 -gram embeddings to a word embedding with simplified Multi-dimensional Self-attention (MS) BIBREF11 . We refer to this constructed embedding as char INLINEFORM2 -MS-vec. The proposed method regards char INLINEFORM3 -MS-vec as an input in addition to a word embedding.

We conduct experiments on the well-known benchmark datasets: Penn Treebank, WikiText-2, and WikiText-103. Our experiments indicate that the proposed method outperforms neural language models trained with well-tuned hyperparameters and achieves state-of-the-art scores on each dataset. In addition, we incorporate our proposed method into a standard neural encoder-decoder model and investigate its effect on machine translation and headline generation. We indicate that the proposed method also has a positive effect on such tasks.

RNN Language Model

In this study, we focus on RNN language models, which are widely used in the literature. This section briefly overviews the basic RNN language model.

In language modeling, we compute joint probability by using the product of conditional probabilities. Let INLINEFORM0 be a word sequence with length INLINEFORM1 , namely, INLINEFORM2 . We formally obtain the joint probability of word sequence INLINEFORM3 as follows: DISPLAYFORM0

INLINEFORM0 is generally assumed to be 1 in this literature, i.e., INLINEFORM1 , and thus we can ignore its calculation.

To estimate the conditional probability INLINEFORM0 , RNN language models encode sequence INLINEFORM1 into a fixed-length vector and compute the probability distribution of each word from this fixed-length vector. Let INLINEFORM2 be the vocabulary size and let INLINEFORM3 be the probability distribution of the vocabulary at timestep INLINEFORM4 . Moreover, let INLINEFORM5 be the dimension of the hidden state of an RNN and let INLINEFORM6 be the dimensions of embedding vectors. Then, RNN language models predict the probability distribution INLINEFORM7 by the following equation: DISPLAYFORM0

where INLINEFORM0 is a weight matrix, INLINEFORM1 is a bias term, and INLINEFORM2 is a word embedding matrix. INLINEFORM3 and INLINEFORM4 are a one-hot vector of an input word

INLINEFORM5 and the hidden state of the RNN at timestep INLINEFORM6 , respectively. We define INLINEFORM7 at timestep INLINEFORM8 as a zero vector, that is, INLINEFORM9 . Let INLINEFORM10 represent an abstract function of an RNN, which might be the LSTM, the Quasi-Recurrent Neural Network (QRNN) BIBREF12 , or any other RNN variants.

Incorporating Character nn-gram Embeddings

We incorporate char INLINEFORM0 -MS-vec, which is an embedding constructed from character INLINEFORM1 -gram embeddings, into RNN language models since, as discussed earlier, previous studies revealed that we can construct better word embeddings by using character INLINEFORM2 -gram embeddings BIBREF0 , BIBREF10 . In particular, we expect char INLINEFORM3 -MS-vec to help represent infrequent words by taking advantage of the internal structure.

Figure FIGREF4 is the overview of the proposed method using character 3-gram embeddings (char3-MS-vec). As illustrated in this figure, our proposed method regards the sum of char3-MS-vec and the standard word embedding as an input of an RNN. In other words, let INLINEFORM0 be char INLINEFORM1 -MS-vec and we replace Equation with the following: DISPLAYFORM0

Multi-dimensional Self-attention

To compute INLINEFORM0 , we apply an encoder to character INLINEFORM1 -gram embeddings. Previous studies demonstrated that additive composition, which computes the (weighted) sum of embeddings, is a suitable method for embedding construction BIBREF13 , BIBREF0 . Thus, we adopt (simplified) multi-dimensional self-attention BIBREF11 , which computes weights for each dimension of given embeddings and sums up the weighted embeddings (i.e., element-wise weighted sum) as an encoder. Let INLINEFORM2 be the character INLINEFORM3 -gram embeddings of an input word, let

INLINEFORM4 be the number of character INLINEFORM5 -grams extracted from the word, and let INLINEFORM6 be the matrix whose INLINEFORM7 -th column corresponds to INLINEFORM8 , that is, INLINEFORM9 . The multi-dimensional self-attention constructs the word embedding INLINEFORM10 by the following equations: DISPLAYFORM0

where INLINEFORM0 means element-wise product of vectors, INLINEFORM1 is a weight matrix, INLINEFORM2 is the INLINEFORM3 -th column of a given matrix, and INLINEFORM4 is the INLINEFORM5 -th element of a given vector. In short, Equation applies the softmax function to each row of INLINEFORM6 and extracts the INLINEFORM7 -th column as INLINEFORM8 .

Let us consider the case where an input word is `the' and we use character 3-gram in Figure FIGREF4 . We prepare special characters `' and `$' to represent the beginning and end of the word, respectively. Then, `the' is composed of three character 3-grams: `th', `the', and `he$'. We multiply the embeddings of these 3-grams by transformation matrix INLINEFORM0 and apply the softmax function to each row as in Equation . As a result of the softmax, we obtain a matrix that contains weights for each embedding. The size of the computed matrix is identical to the input embedding matrix: INLINEFORM1 . We then compute Equation EQREF7 , i.e., the weighted sum of the embeddings, and add the resulting vector to the word embedding of `the'. Finally, we input the vector into an RNN to predict the next word.

Word Tying

BIBREF14 and BIBREF15 proposed a word tying method (WT) that shares the word embedding matrix (INLINEFORM0 in Equation) with the weight matrix to compute probability distributions (INLINEFORM1 in Equation EQREF3). They demonstrated that WT significantly improves the performance of RNN language models.

In this study, we adopt char INLINEFORM0 -MS-vec as the weight matrix in language modeling. Concretely, we use INLINEFORM1 instead of INLINEFORM2 in Equation EQREF3 , where INLINEFORM3 contains char INLINEFORM4 -MS-vec for all words in the vocabulary.

Experiments on Language Modeling

We investigate the effect of char INLINEFORM0 -MS-vec on the word-level language modeling task. In detail, we examine the following four research questions;

Datasets

We used the standard benchmark datasets for the word-level language modeling: Penn Treebank (PTB) BIBREF16 , WikiText-2 (WT2), and WikiText-103 (WT103) BIBREF17 . BIBREF18 and BIBREF17 published pre-processed PTB, WT2, and WT103. Following the previous studies, we used these pre-processed datasets for our experiments.

Table TABREF14 describes the statistics of the datasets. Table TABREF14 demonstrates that the vocabulary size of WT103 is too large, and thus it is impractical to compute char INLINEFORM0 -MS-vec for all words at every step. Therefore, we did not use INLINEFORM1 for word tying. In other words, we used only word embeddings INLINEFORM2 as the weight matrix INLINEFORM3 in WT103.

For machine translation, we used two kinds of language pairs: English-French and English-German sentences in the IWSLT 2016 dataset. The dataset contains about 208K English-French pairs and 189K English-German pairs. We conducted four translation tasks: from English to each language (En-Fr and En-De), and their reverses (Fr-En and De-En).

For headline generation, we used sentence-headline pairs extracted from the annotated English Gigaword corpus BIBREF35 in the same manner as BIBREF2 . The training set contains about 3.8M sentence-headline pairs. For evaluation, we exclude the test set constructed by BIBREF2 because it contains some invalid instances, as reported in BIBREF33 . We instead used the test sets constructed by BIBREF33 and BIBREF34 .

Baseline RNN Language Model

For base RNN language models, we adopted the state-of-the-art LSTM language model BIBREF19 for PTB and WT2, and QRNN for WT103 BIBREF12 . BIBREF20 demonstrated that the standard LSTM trained with appropriate hyperparameters outperformed various architectures such as Recurrent Highway Networks (RHN) BIBREF21 . In addition to several regularizations, BIBREF19 introduced Averaged Stochastic Gradient Descent (ASGD) BIBREF22 to train the 3-layered LSTM language model. As a result, their ASGD Weight-Dropped LSTM (AWD-LSTM) achieved state-of-the-art results on PTB and WT2. For WT103, BIBREF23 achieved the top score with the 4-layered QRNN. Thus, we used AWD-LSTM for PTB and WT2, and QRNN for WT103 as the base language models, respectively. We used their implementations for our experiments.

Results

Table TABREF15 shows perplexities of the baselines and the proposed method. We varied INLINEFORM0 for char INLINEFORM1 -MS-vec from 2 to 4. For the baseline, we also applied two word embeddings to investigate the performance in the case where we use more kinds of word embeddings. In detail, we prepared INLINEFORM2 and used INLINEFORM3 instead of INLINEFORM4 in Equation . Table TABREF15 also shows the number of character INLINEFORM5 -grams in each dataset. This table indicates that char INLINEFORM6 -MS-vec improved the performance of state-of-the-art models except for char4-MS-vec on WT103. These results indicate that char INLINEFORM7 -MS-vec can raise the quality of word-level language models. In particular, Table TABREF15 shows that char3-MS-vec achieved the best scores consistently. In contrast, an additional word embedding did not improve the performance. This fact implies that the improvement of char INLINEFORM8 -MS-vec is caused by using character INLINEFORM9 -grams. Thus, we answer yes to the first research question.

Table TABREF16 shows the training time spent on each epoch. We calculated it on the NVIDIA Tesla P100. Table TABREF16 indicates that the proposed method requires more computational time than the baseline unfortunately. We leave exploring a faster structure for our future work.

Table TABREF17 shows perplexities on the PTB dataset where the frequency of an input word is lower than 2,000 in the training data. This table indicates that the proposed method can improve the performance even if an input word is infrequent. In other words, char INLINEFORM0 -MS-vec helps represent the meanings of infrequent words. Therefore, we answer yes to the second research question in the case of our experimental settings.

We explored the effectiveness of multi-dimensional self-attention for word embedding construction. Table TABREF24 shows perplexities of using several encoders on the PTB dataset. As in BIBREF8 , we applied CNN to construct word embeddings (charCNN in Table TABREF24). Moreover, we applied the summation and standard self-attention, which computes the scalar value as a weight for a character INLINEFORM0 -gram embedding, to construct word embeddings (char INLINEFORM1 -Sum-vec and char INLINEFORM2 -SS-vec, respectively). For CNN, we used hyperparameters identical to BIBREF8 (“Original Settings” in Table TABREF24) but the setting has two differences from other architectures: 1. The dimension of the computed vectors is much larger than the dimension of the baseline word embeddings and 2. The dimension of the input character embeddings is much smaller than the dimension of the baseline word embeddings. Therefore, we added two configurations: assigning the dimension of the computed vectors and input character embeddings a value identical to the baseline word embeddings (in Table TABREF24 , “Small CNN result dims” and “Large embedding dims”, respectively).

Table TABREF24 shows that the proposed char INLINEFORM0 -MS-vec outperformed charCNN even though the original settings of charCNN had much larger parameters. Moreover, we trained charCNN with two additional settings but CNN did not improve the baseline performance. This result implies that char INLINEFORM1 -MS-vec is better embeddings than ones constructed by applying CNN to character embeddings. Table TABREF24 also indicates that char INLINEFORM2 -Sum-vec was harmful to the performance. Moreover, char INLINEFORM3 -SS-vec did not have a positive effect on the baseline. These results answer yes to the third research question; our use of multi-dimensional self-attention is more appropriate for constructing word embeddings from character INLINEFORM4 -gram embeddings.

Table TABREF24 also shows that excluding INLINEFORM0 from word tying (“Exclude INLINEFORM1 from word tying”) achieved almost the same score as the baseline. Moreover, this table indicates that performance fails as the the number of parameters is increased. Thus, we need to assign INLINEFORM2 to word tying to prevent over-fitting for the PTB dataset. In addition, this result implies that the performance of WT103 in Table TABREF15 might be raised if we can apply word tying to WT103.

Moreover, to investigate the effect of only char INLINEFORM0 -MS-vec, we ignore INLINEFORM1 in Equation EQREF5 . We refer to this setting as “Remove word embeddings INLINEFORM2 ” in Table TABREF24 . Table TABREF24 shows cahr3-MS-vec and char4-MS-vec are superior to char2-MS-vec. In the view of perplexity, char3-MS-vec and char4-MS-vec achieved comparable scores to each other. On the other hand, char3-MS-vec is composed of much smaller parameters. Furthermore, we decreased the embedding size INLINEFORM3 to adjust the number of parameters to the same size as the baseline (“Same #Params as baseline” in Table TABREF24). In this setting, char3-MS-vec achieved the best perplexity. Therefore, we consider that char3-MS-vec is more useful than char4-MS-vec, which is the answer to the fourth research question. We use the combination of the char3-MS-vec INLINEFORM4 and word embedding INLINEFORM5 in the following experiments.

Finally, we compare the proposed method with the published scores reported in previous studies. Tables TABREF25 , TABREF26 , and TABREF27 , respectively, show perplexities of the proposed method and previous studies on PTB, WT2, and WT103. Since AWD-LSTM-MoS BIBREF28 and AWD-LSTM-DOC BIBREF29 achieved the state-of-the-art scores on PTB and WT2, we combined char3-MS-vec with them. These tables show that the proposed method improved the performance of the base model and outperformed the state-of-the-art scores on all datasets. In particular, char3-MS-vec improved perplexity by at least 1 point from current best scores on the WT103 dataset.

Tables TABREF31 and TABREF32 show the results of machine translation and headline generation, respectively. These tables show that EncDec+char3-MS-vec outperformed EncDec in all test data. In other words, these results indicate that our proposed method also has a positive effect on the neural encoder-decoder model. Moreover, it is noteworthy that char3-MS-vec improved the performance of EncDec even though the vocabulary set constructed by BPE contains subwords. This implies that character INLINEFORM0 -gram embeddings improve the quality of not only word embeddings but also subword embeddings.

In addition to the results of our implementations, the lower portion of Table TABREF32 contains results reported in previous studies. Table TABREF32 shows that EncDec+char3-MS-vec also outperformed the methods proposed in previous studies. Therefore, EncDec+char3-MS-vec achieved the top scores in the test sets constructed by BIBREF33 and BIBREF34 even though it does not have a task-specific architecture such as the selective gate proposed by BIBREF33 .

In these experiments, we only applied char3-MS-vec to EncDec but BIBREF38 indicated that combining multiple kinds of subword units can improve the performance. We will investigate the effect of combining several character INLINEFORM0 -gram embeddings in future work.

Experiments on Applications

As described in Section SECREF1 , neural encoder-decoder models can be interpreted as conditional neural language models. Therefore, to investigate if the proposed method contributes to encoder-decoder models, we conduct experiments on machine translation and headline generation tasks.

Experimental Settings

We employed the neural encoder-decoder with attention mechanism described in BIBREF34 as the base model. Its encoder consists of a 2-layer bidirectional LSTM and its decoder consists of a 2-layer LSTM with attention mechanism proposed by BIBREF36 . We refer to this neural encoder-decoder as EncDec. To investigate the effect of the proposed method, we introduced char3-MS-vec into EncDec. Here, we applied char3-MS-vec to both the encoder and decoder. Moreover, we did not apply word tying technique to EncDec because it is default setting in the widely-used encoder-decoder implementation.

We set the embedding size and dimension of the LSTM hidden state to 500 for machine translation and 400 for headline generation. The mini-batch size is 64 for machine translation and 256 for headline generation. For other hyperparameters, we followed the configurations described in BIBREF34 . We constructed the vocabulary set by using Byte-Pair-Encoding (BPE) BIBREF37 because BPE is a currently widely-used technique for vocabulary construction. We set the number of BPE merge operations to 16K for machine translation and 5K for headline generation.

Conclusion

In this paper, we incorporated character information with RNN language models. Based on the research in the field of word embedding construction BIBREF0 , we focused on character INLINEFORM0 -gram embeddings to construct word embeddings. We used multi-dimensional self-attention BIBREF11 to encode character INLINEFORM1 -gram embeddings. Our proposed char INLINEFORM2 -MS-vec improved the performance of state-of-the-art RNN language models and achieved the best perplexities on Penn Treebank, WikiText-2, and WikiText-103. Moreover, we investigated the effect of char INLINEFORM3 -MS-vec on application tasks, specifically, machine translation and headline generation. Our experiments show that char INLINEFORM4 -MS-vec also improved the performance of a neural encoder-decoder on both tasks.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP18K18119. We would like to thank the anonymous reviewers for their helpful suggestions and comments.