Word Embedding Attention Network: Generating Words by Querying Distributed Word Representations for Paraphrase Generation

Shuming Ma[1], Xu Sun[1,2], Wei Li[1], Sujian Li[1], Wenjie Li[3], Xuancheng Ren[1]
[1]MOE Key Lab of Computational Linguistics, School of EECS, Peking University
[2]Deep Learning Lab, Beijing Institute of Big Data Research, Peking University
[3]Department of Computing, The Hong Kong Polytechnic University
{shumingma, xusun, liweitj47, lisujian, renxc}@pku.edu.cn
cswjli@comp.polyu.edu.hk

Abstract

Most recent approaches use the sequence-to-sequence model for paraphrase generation. The existing sequence-to-sequence model tends to memorize the words and the patterns in the training dataset instead of learning the meaning of the words. Therefore, the generated sentences are often grammatically correct but semantically improper. In this work, we introduce a novel model based on the encoder-decoder framework, called Word Embedding Attention Network (WEAN). Our proposed model generates the words by querying distributed word representations (i.e. neural word embeddings), hoping to capturing the meaning of the according words. Following previous work, we evaluate our model on two paraphrase-oriented tasks, namely text simplification and short text abstractive summarization. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on these three benchmark datasets.[1]

1 Introduction

Paraphrase is a restatement of the meaning of a text using other words. Many natural language generation tasks are paraphrase-orientated, such as text simplification and short text summarization. Text simplification is to make the text easier to read and understand, especially for poor readers, while short text summarization is to generate a brief sentence to describe the short texts (e.g. posts on the social media). Most recent approaches use sequence-to-sequence model for paraphrase generation (Prakash et al., 2016; Cao et al., 2017). It compresses the source text information into dense vectors with the neural encoder, and the neural decoder generates the target text using the compressed vectors.

Although neural network models achieve success in paraphrase generation, there are still two major problems. One of the problem is that the existing sequence-to-sequence model tends to memorize the words and the patterns in the training dataset instead of the meaning of the words. The main reason is that the word generator (i.e. the output layer of the decoder) does not model the semantic information. The word generator, which consists of a linear transformation and a softmax operation, converts the Recurrent Neural Network (RNN) output from a small dimension (e.g. 500) to a much larger dimension (e.g. 50,000 words in the vocabulary), where each dimension represents the score of each word. The latent assumption of the word generator is that each word is independent and the score is irrelevant to each other. Therefore, the scores of a word and its synonyms may be of great difference, which means the word generator learns the word itself rather than the relationship between words.

The other problem is that the word generator has a huge number of parameters. Suppose we have a sequence-to-sequence model with a hidden size of 500 and a vocabulary size of 50,000. The word generator has up to 25 million parameters, which is even larger than other parts of the encoder-decoder model in total. The huge size of parameters will result in slow convergence, because there are a lot of parameters to be learned. Moreover, under the distributed framework, the more parameters a model has, the more bandwidth and memory it consumes.

To tackle both of the problems, we propose a novel model called Word Embedding Attention Network (WEAN). The word generator of WEAN

[1]The code is available at https://github.com/lancopku/WEAN

is attention based, instead of the simple linear softmax operation. In our attention based word generator, the RNN output is a query, the candidate words are the values, and the corresponding word representations are the keys. In order to predict the word, the attention mechanism is used to select the value matching the query most, by means of querying the keys. In this way, our model generates the words according to the distributed word representations (i.e. neural word embeddings) in a retrieval style rather than the traditional generative style. Our model is able to capture the semantic meaning of a word by referring to its embedding. Besides, the attention mechanism has a much smaller number of parameters compared with the linear transformation directly from the RNN output space to the vocabulary space. The reduction of the parameters can increase the convergence rate and speed up the training process. Moreover, the word embedding is updated from three sources: the input of the encoder, the input of the decoder, and the query of the output layer.

Following previous work (Cao et al., 2017), we evaluate our model on two paraphrase-oriented tasks, namely text simplification and short text abstractive summarization. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on all of the benchmark datasets.

2 Proposed Model

We propose a novel model based on the encoder-decoder framework, which generates the words by querying distributed word representations with the attention mechanism. In this section, we first present the overview of the model architecture. Then, we explain the details of the word generation, especially the way to query word embeddings.

2.1 Overview

Word Embedding Attention Network is based on the encoder-decoder framework, which consists of two components: a source text encoder, and a target text decoder. Figure 1 is an illustration of our model. Given the source texts, the encoder compresses the source texts into dense representation vectors, and the decoder generates the paraphrased texts. To predict a word, the decoder uses the hidden output to query the word embeddings. The word embeddings assess all the candidate words, and return the word whose embedding matches the query most. The selected word is emitted as the predicted token, and its embedding is then used as the input of the LSTM at the next time step. After the back propagation, the word embedding is updated from three sources: the input of the encoder, the input of the decoder, and the query of the output layer. We show the details of our WEAN in the following subsection.

2.2 Encoder and Decoder

The goal of the source text encoder is to provide a series of dense representation of complex source texts for the decoder. In our model, the source text encoder is a Long Short-term Memory Network (LSTM), which produces the dense representation $\{h_1, h_2, ..., h_N\}$ from the source text $\{x_1, x_2, ..., x_N\}$:

The goal of the target text decoder is to generate a series of paraphrased words from the dense representation of source texts. Fisrt, the LSTM of the decoder compute the dense representation of generated words s_t. Then, the dense representations are fed into an attention layer (Bahdanau et al., 2014) to generate the context vector c_t, which captures context information of source texts. Attention vector c_t is calculated by the weighted sum of encoder hidden states:

$$c_t = \sum_{i=1}^{N} \alpha_{ti} h_i \tag{1}$$

$$\alpha_{ti} = \frac{e^{g(s_t, h_i)}}{\sum_{j=1}^{N} e^{g(s_t, h_j)}} \tag{2}$$

where $g(s_t, h_i)$ is an attentive score between the decoder hidden state s_t and the encoder hidden state h_i.

In this way, c_t and s_t respectively represent the context information of source texts and the target texts at the t^{th} time step.

2.3 Word Generation by Querying Word Embedding

For the current sequence-to-sequence model, the word generator computes the distribution of output words y_t in a generative style:

$$p(y_t) = softmax(W s_t) \tag{3}$$

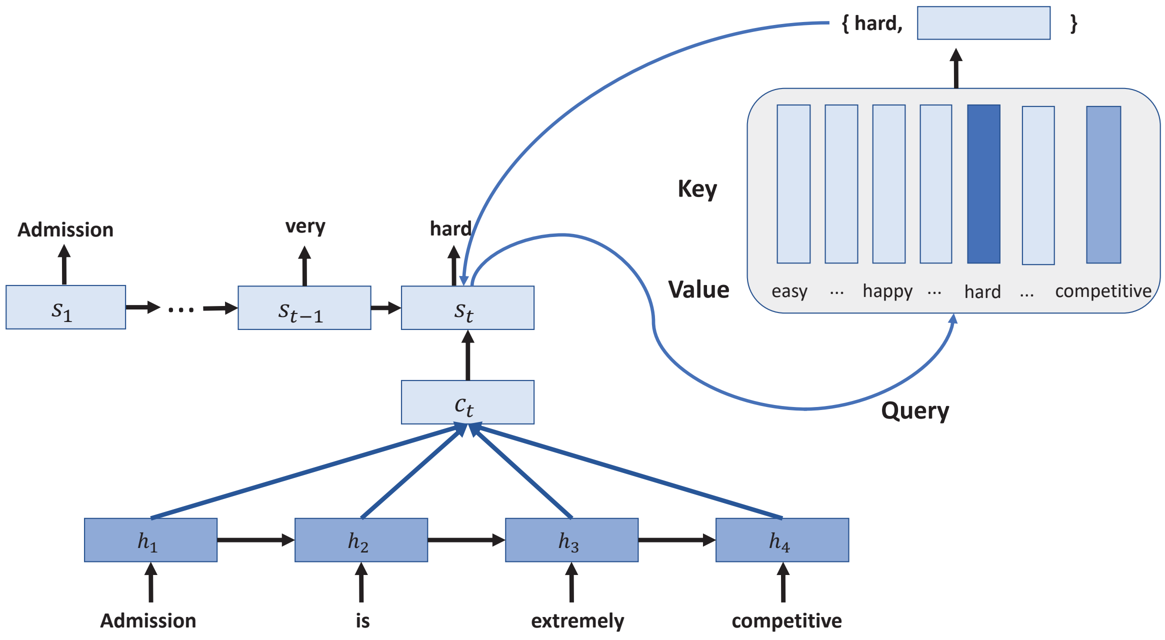

Figure 1: An overview of Word Embedding Attention Network.

where $W \in R^{k \times V}$ is a trainable parameter matrix, k is hidden size, and V is the number of words in the vocabulary. When the vocabulary is large, the number of parameters will be huge.

Our model generates the words in a retrieval style rather than the traditional generative style, by querying the word embeddings. We denote the combination of the source context vector c_t and the target context vector s_t as the query q_t:

$$q_t = \tanh(W_c[s_t; c_t]) \tag{4}$$

The candidate words w_i and their corresponding embeddings e_i are paired as the key-value pairs $\{w_i, e_i\}(i = 1, 2, ..., n)$, where n is the number of candidate words. We give the details of how to determine the set of candidate words in Section 2.4. Our model uses q_t to query the key-value pairs $\{w_i, e_i\}(i = 1, 2, ..., n)$ by evaluating the relevance between the query q_t and each word vector e_i with a score function $f(q_t, e_i)$. The query process can be regarded as the attentive selection of the word embeddings. We borrow the attention energy functions (Luong et al., 2015) as the relevance score function $f(q_t, e_i)$:

$$f(q_t, e_i) = \begin{cases} q_t^T e_i & \text{dot} \\ q_t^T W_a e_i & \text{general} \\ v^T \tanh(W_q q_t + W_e e_i) & \text{concat} \end{cases} \tag{5}$$

where W_q and W_e are two trainable parameter matrices, and v^T is a trainable parameter vector. In implementation, we select the general attention function as the relevance score function, based on the performance on the validation sets. The key-value pair with the highest score $\{w_t, e_t\}$ is selected. At the test stage, the decoder generates the key w_t as the t^{th} predicted word, and inputs the value e_t to the LSTM unit at the $t + 1^{th}$ time step. At the training stage, the scores are normalized as the word probability distribution:

$$p(y_t) = softmax(f(q_t, e_i)) \tag{6}$$

2.4 Selection of Candidate Key-value Pairs

As described in Section 2.3, the model generates the words in a retrieval style, which selects a word according to its embedding from a set of candidate key-value pairs. We now give the details of how to obtain the set of candidate key-value pairs. We extract the vocabulary from the source text in the training set, and select the n most frequent words as the candidate words. We reuse the embeddings of the decoder inputs as the values of the candidate words, which means that the decoder input and the predicted output share the same vocabulary and word embeddings. Besides, we do not use any pretrained word embeddings in our model, so that all of the parameters are learned from scratch.

2.5 Training

Although our generator is a retrieval style, WEAN is as differentiable as the sequence-to-sequence model. The objective of training is to minimize the cross entropy between the predicted word probability distribution and the golden one-hot distribution:

$$L = -\sum_i \hat{y}_i \log p(y_i) \tag{7}$$

We use Adam optimization method to train the model, with the default hyper-parameters: the learning rate $\alpha = 0.001$, and $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$.

3 Experiments

Following the previous work (Cao et al., 2017), we test our model on the following two paraphrase orientated tasks: text simplification and short text abstractive summarization.

3.1 Text Simplification

3.1.1 Datasets

The datasets are both from the alignments between English Wikipedia website[2] and Simple English Wikipedia website.[3] The Simple English Wikipedia is built for "the children and adults who are learning the English language", and the articles are composed with "easy words and short sentences". Therefore, Simple English Wikipedia is a natural public simplified text corpus.

- **Parallel Wikipedia Simplification Corpus (PWKP).** PWKP (Zhu et al., 2010) is a widely used benchmark for evaluating text simplification systems. It consists of aligned complex text from English WikiPedia (as of Aug. 22nd, 2009) and simple text from Simple Wikipedia (as of Aug. 17th, 2009). The dataset contains 108,016 sentence pairs, with 25.01 words on average per complex sentence and 20.87 words per simple sentence. Following the previous work (Zhang and Lapata, 2017), we remove the duplicate sentence pairs, and split the corpus with 89,042 pairs for training, 205 pairs for validation and 100 pairs for test.

- **English Wikipedia and Simple English Wikipedia (EW-SEW).** EW-SEW is a publicly available dataset provided by Hwang et al. (2015). To build the corpus, they first align the complex-simple sentence pairs, score the semantic similarity between the complex sentence and the simple sentence, and classify each sentence pair as a good, good partial, partial, or bad match. Following the previous work (Nisioi et al., 2017), we discard the unclassified matches, and use the good matches and partial matches with a scaled threshold greater than 0.45. The corpus contains about 150K good matches and 130K good partial matches. We use this corpus as the training set, and the dataset provided by Xu et al. (Xu et al., 2016) as the validation set and the test set. The validation set consists of 2,000 sentence pairs, and the test set contains 359 sentence pairs. Besides, each complex sentence is paired with 8 reference simplified sentences provided by Amazon Mechanical Turk workers.

[2]http://en.wikipedia.org
[3]http://simple.wikipedia.org

3.1.2 Evaluation Metrics

Following the previous work (Nisioi et al., 2017; Hu et al., 2015), we evaluate our model with different metrics on two tasks.

- **Automatic evaluation.** We use the BLEU score (Papineni et al., 2002) as the automatic evaluation metric. BLEU is a widely used metric for machine translation and text simplification, which measures the agreement between the model outputs and the gold references. The references can be either single or multiple. In our experiments, the references are single on PWKP, and multiple on EW-SEW.

- **Human evaluation.** Human evaluation is essential to evaluate the quality of the model outputs. Following Nisioi et al. (2017) and Zhang et al. (2017), we ask the human raters to rate the simplified text in three dimensions: Fluency, Adequacy and Simplicity. Fluency assesses whether the outputs are grammatically right and well formed. Adequacy represents the meaning preservation of the simplified text. Both the scores of fluency and adequacy range from 1 to 5 (1 is very bad and 5 is very good). Simplicity shows how simpler the model outputs are than the source text, which ranges from 1 to 5.

3.1.3 Settings

Our proposed model is based on the encoder-decoder framework. The encoder is implemented on LSTM, and the decoder is based on LSTM with Luong style attention (Luong et al., 2015). We

PWKP	BLEU
PBMT (Wubben et al., 2012)	46.31
Hybrid (Narayan and Gardent, 2014)	53.94
EncDecA (Zhang and Lapata, 2017)	47.93
DRESS (Zhang and Lapata, 2017)	34.53
DRESS-LS (Zhang and Lapata, 2017)	36.32
Seq2seq (our implementation)	48.26
WEAN (our proposal)	**54.54**

Table 1: Automatic evaluation of our model and other related systems on PWKP datasets. The results are reported on the test sets.

EW-SEW	BLEU
PBMT-R (Wubben et al., 2012)	67.79
Hybrid (Narayan and Gardent, 2014)	48.97
SBMT-SARI (Xu et al., 2016)	73.62
NTS (Nisioi et al., 2017)	84.70
NTS-w2v (Nisioi et al., 2017)	87.50
EncDecA (Zhang and Lapata, 2017)	88.85
DRESS (Zhang and Lapata, 2017)	77.18
DRESS-LS (Zhang and Lapata, 2017)	80.12
Seq2seq (our implementation)	88.97
WEAN (our proposal)	**94.45**

Table 2: Automatic evaluation of our model and other related systems on EW-SEW datasets. The results are reported on the test sets.

tune our hyper-parameter on the development set. The model has two LSTM layers. The hidden size of LSTM is 256, and the embedding size is 256. We use Adam optimizer (Kingma and Ba, 2014) to learn the parameters, and the batch size is set to be 64. We set the dropout rate (Srivastava et al., 2014) to be 0.4. All of the gradients are clipped when the norm exceeds 5.

3.1.4 Baselines

We compare our model with several neural text simplification systems.

- **Seq2seq** is our implementation of the sequence-to-sequence model with attention mechanism, which is the most popular neural model for text generation.
- **NTS** and **NTS-w2v** (Nisioi et al., 2017) are two sequence-to-sequence model with extra mechanism like prediction ranking, and NTS-w2v uses a pretrain word2vec.
- **DRESS** and **DRESS-LS** (Zhang and Lapata, 2017) are two deep reinforcement learning sentence simplification models.
- **EncDecA** is a model based on the encoder-decoder with attention, implemented by Zhang and Lapata (2017).
- **PBMT-R** (Wubben et al., 2012) is a phrase based machine translation model which reranks the outputs.
- **Hybrid** (Narayan and Gardent, 2014) is a hybrid approach which combines deep semantics and mono-lingual machine translation.
- **SBMT-SARI** (Xu et al., 2016) is a syntax-based machine translation model which is trained on PPDB dataset (Ganitkevitch et al., 2013) and tuned with SARI.

PWKP	Fluency	Adequacy	Simplicity	All
NTS-w2v	3.54	3.47	3.38	3.46
DRESS-LS	3.68	3.55	3.50	3.58
WEAN	**3.77**	**3.66**	**3.58**	**3.67**
Reference	3.76	3.60	3.44	3.60

EW-SEW	Fluency	Adequacy	Simplicity	All
PBMT-R	3.36	2.92	3.37	3.22
SBMT-SARI	3.41	**3.63**	3.25	3.43
NTS-w2v	3.56	3.52	3.42	3.50
DRESS-LS	3.59	3.43	**3.65**	3.56
WEAN	**3.61**	3.56	**3.65**	**3.61**
Reference	3.71	3.64	3.45	3.60

Table 3: Human evaluation of our model and other related systems on PWKP and EW-SEW datasets. The results are reported on the test sets.

3.1.5 Results

We compare WEAN with state-of-the-art models for text simplification. Table 1 and Table 2 summarize the results of the automatic evaluation. On PWKP dataset, we compare WEAN with PBMT, Hybrid, EncDecA, DRESS and DRESS-LS. WEAN achieves a BLEU score of 54.54, outperforming all of the previous systems. On EW-SEW dataset, we compare WEAN with PBMT-R, Hybrid, SBMT-SARI, and the neural models described above. We do not find any public release code of PBMT-R and SBMT-SARI. Fortunately, Xu et al. (2016) provides the predictions of PBMT-R and SBMT-SARI on EW-SEW test set, so that we can compare our model with these systems.

LCSTS	R-1	R-2	R-L
RNN-W(Hu et al., 2015)	17.7	8.5	15.8
RNN(Hu et al., 2015)	21.5	8.9	18.6
RNN-cont-W(Hu et al., 2015)	26.8	16.1	24.1
RNN-cont(Hu et al., 2015)	29.9	17.4	27.2
SRB(Ma et al., 2017)	33.3	20.0	30.1
CopyNet-W(Gu et al., 2016)	35.0	22.3	32.0
CopyNet(Gu et al., 2016)	34.4	21.6	31.3
RNN-dist(Chen et al., 2016)	35.2	22.6	32.5
DRGD(Li et al., 2017)	37.0	24.2	34.2
Seq2seq	32.1	19.9	29.2
WEAN	**37.8**	**25.6**	**35.2**

Table 4: ROUGE F1 score on the LCSTS test set. R-1, R-2, and R-L denote ROUGE-1, ROUGE-2, and ROUGE-L, respectively. The models with a suffix of 'W' in the table are word-based, while the rest of models are character-based.

It shows that the neural models have better performance in BLEU, and WEAN achieves the best BLEU score with 94.45.

We perform the human evaluation of WEAN and other related systems, and the results are shown in Table 3. DRESS-LS is based on the reinforcement learning, and it encourages the fluency, simplicity and relevance of the outputs. Therefore, it achieves a high score in our human evaluation. WEAN gains a even better score than DRESS-LS. Besides, WEAN generates more adequate and simpler outputs than the reference on PWKP. The predictions of SBMT-SARI are the most adequate among the compared systems on EW-SEW. In general, WEAN outperforms all of the other systems, considering the balance of fluency, adequate and simplicity. We conduct significance tests based on t-test. The significance tests suggest that WEAN has a very significant improvement over baseline, with $p \leq 0.001$ over DRESS-LS in all of the dimension on PWKP, $p \leq 0.05$ over DRESS-LS in the dimension of fluency, $p \leq 0.005$ over NTS-w2v in the dimension of simplicity and $p \leq 0.005$ over DRESS-LS in the dimension of all.

3.2 Large Scale Text Summarization

3.2.1 Dataset

Large Scale Chinese Social Media Short Text Summarization Dataset (LCSTS): LCSTS is constructed by Hu et al. (2015). The dataset consists of more than 2,400,000 text-summary pairs, constructed from a famous Chinese social media website called Sina Weibo.[4] It is split into three parts, with 2,400,591 pairs in PART I, 10,666 pairs in PART II and 1,106 pairs in PART III. All the text-summary pairs in PART II and PART III are manually annotated with relevant scores ranged from 1 to 5. We only reserve pairs with scores no less than 3, leaving 8,685 pairs in PART II and 725 pairs in PART III. Following the previous work (Hu et al., 2015), we use PART I as training set, PART II as validation set, and PART III as test set.

3.2.2 Evaluation Metrics

Our evaluation metric is ROUGE score (Lin and Hovy, 2003), which is popular for summarization evaluation. The metrics compare an automatically produced summary against the reference summaries, by computing overlapping lexical units, including unigram, bi-gram, trigram, and longest common subsequence (LCS). Following previous work (Rush et al., 2015; Hu et al., 2015), we use ROUGE-1 (unigram), ROUGE-2 (bi-gram) and ROUGE-L (LCS) as the evaluation metrics in the reported experimental results.

3.2.3 Settings

The vocabularies are extracted from the training sets, and the source contents and the summaries share the same vocabularies. We tune the hyper-parameters based on the ROUGE scores on the validation sets. In order to alleviate the risk of word segmentation mistakes, we split the Chinese sentences into characters. We prune the vocabulary size to 4,000, which covers most of the common characters. We set the word embedding size and the hidden size to 512, the number of LSTM layers of the encoder is 2, and the number of LSTM layers of the decoder is 1. The batch size is 64, and we do not use dropout (Srivastava et al., 2014) on this dataset. Following the previous work (Li et al., 2017), we implement a beam search optimization, and set the beam size to 5.

3.2.4 Baselines

We compare our model with the state-of-the-art baselines.

- **RNN** and **RNN-cont** are two sequence-to-sequence baseline with GRU encoder and decoder, provided by Hu et al. (2015).

[4]`http://weibo.com`

#Param	PWKP	EWSEW	LCSTS
Seq2seq	12.80M	12.80M	2.05M
WEAN	0.13M	0.13M	0.52M

Table 5: The number of the parameters in the output layer. The numbers of rest parameters between Seq2seq and WEAN are the same.

- **RNN-dist** (Chen et al., 2016) is a distraction-based neural model, which the attention mechanism focuses on the different parts of the source content.
- **CopyNet** (Gu et al., 2016) incorporates a copy mechanism to allow part of the generated summary is copied from the source content.
- **SRB** (Ma et al., 2017) is a sequence-to-sequence based neural model with improving the semantic relevance between the input text and the output summary.
- **DRGD** (Li et al., 2017) is a deep recurrent generative decoder model, combining the decoder with a variational autoencoder.
- **Seq2seq** is our implementation of the sequence-to-sequence model with the attention mechanism.

3.2.5 Results

We report the ROUGE F1 score of our model and the baseline models on the test sets. Table 4 summarizes the comparison between our model and the baselines. Our model achieves the score of 37.8 ROUGE-1, 25.6 ROUGE-2, and 35.2 ROUGE-L, outperforming all of the previous models. First, we compare our model with the sequence-to-sequence model. It shows that our model significant outperforms the sequence-to-sequence baseline with a large margin of 5.7 ROUGE-1, 5.7 ROUGE-2, and 6.0 ROUGE-L. Then, we compare our model with other related models. The state-of-the-art model is DRGD (Li et al., 2017), which obtains the score of 37.0 ROUGE-1, 24.2 ROUGE-2, and 34.2 ROUGE-L. Our model has a relative gain of 0.8 ROUGE-1, 1.4 ROUGE-2 and 1.0 ROUGE-L over the state-of-the-art models.

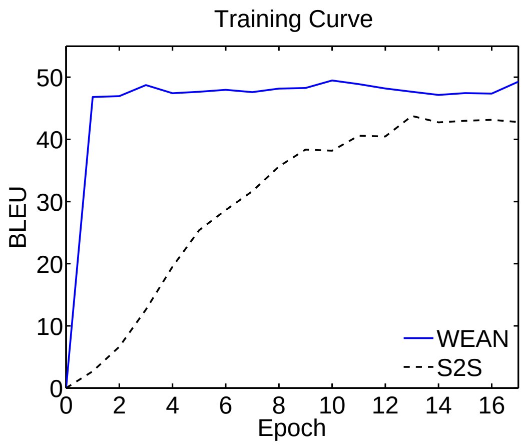

Figure 2: The training curve of WEAN and Seq2seq on the PWKP validation set.

4 Analysis and Discussion

4.1 Reducing Parameters

Our WEAN reduces a large number of the parameters in the output layer. To analyze the parameter reduction, we compare our WEAN model with the sequence-to-sequence model. Table 5 lists the number of the parameters in the output layers of two models. Both PWKP and EWSEWhave the vocabulary size of 50000 words and the hidden size of 256, resulting $50000 \times 256 = 12,800,000$ parameters. LCSTS has a vocabulary size of 4000 and the hidden size of 512, so the seq2seq has $4000 \times 512 = 2,048,000$ parameters in the output layers. WEAN only has two parameter matrices and one parameter vector at most in Equation 5, without regard to the vocabulary size. It has $256 \times 256 \times 2 + 256 = 131,328$ parameters on PWKP and EWSEW, and $512 \times 512 \times 2 + 512 = 524,800$ parameters on LCSTS. Besides, WEAN does not have any extra parameters in the other part of the model.

4.2 Speeding up Convergence

Figure 2 shows the training curve of WEAN and Seq2seq on the PWKP validation set. WEAN achieve near the optimal score in only 2-3 epochs, while Seq2seq takes more than 15 epochs to achieve the optimal score. Therefore, WEAN has much faster convergence rate, compared with Seq2seq. With the much faster training speed, WEAN does not suffer loss in BLEU, and even improve the BLEU score.

Source	*Yoghurt or yogurt is a dairy product produced by bacterial fermentation of milk .*
Reference	*Yoghurt or yogurt is a dairy product* ***made*** *by bacterial fermentation of milk .*
NTS	***. or yoghurt*** *is a dairy product* ***produced*** *by bacterial fermentation of milk .*
NTS-w2v	***It is made*** *by bacterial fermentation of milk .*
PBMT-R	*Yoghurt or yogurt is a dairy product* ***produced*** *by bacterial fermentation* ***of .***
SBMT-SARI	*Yogurt or yogurt is a dairy product* ***drawn up*** *by bacterial fermentation of milk .*
WEAN	*Yoghurt or yogurt is a dairy product* ***made*** *by bacterial fermentation of milk .*
Source	*Depending on the context, another closely-related meaning of constituent is that of a citizen residing in the area governed, represented, or otherwise served by a politician; sometimes this is restricted to citizens who elected the politician.*
Reference	***The word constituent can also be used to refer to*** *a citizen* ***who lives*** *in the area that is governed, represented, or otherwise served by a politician; sometimes* ***the word*** *is restricted to citizens who elected the politician.*
NTS	*Depending on the context, another closely-related meaning of constituent is that of a citizen* ***living*** *in the area governed, represented, or otherwise served by a politician; sometimes this is restricted to citizens who elected the politician.*
NTS-w2v	*This is restricted to citizens who elected the politician.*
PBMT-R	*Depending on the context and meaning of closely-related* ***siemens-martin -rrb- is a*** *citizen living in the area, or otherwise,* ***was governed by a 1924-1930 shurba****; this is restricted to people who elected* ***it****.*
SBMT-SARI	***In terms of*** *the context, another closely-related* ***sense of the component*** *is that of a citizen* ***living*** *in the area* ***covered, make up, or if not, served by a policy****; sometimes this is* ***limited*** *to the people who elected the* ***policy****.*
WEAN	*Depending on the context, another closely-related meaning of constituent is that of a citizen* ***who lives*** *in the area governed, represented, or otherwise served by a politician; sometimes* ***the word*** *is restricted to citizens who elected the politician.*

Table 6: Two examples of different text simplification system outputs in EW-SEW dataset. Differences from the source texts are shown in bold.

4.3 Case Study

Table 6 shows two examples of different text simplification system outputs on EW-SEW. For the first example, NTS, NTS-w2v and PBMT-R miss some essential constituents, so that the sentences are incomplete and not fluent. SBMT-SARI generates a fluent sentence, but the output does not preserve the original meaning. The predicted sentence of WEAN is fluent, simple, and the same as the reference. For the second example, NTS-w2v omits so many words that it lacks a lot of information. PBMT-R generates some irrelevant words, like 'siemens-martin', '-rrb-', and '-shurba', which hurts the fluency and adequacy of the generated sentence. SBMT-SARI is able to generate a fluent sentence, but the meaning is different from the source text, and even more difficult to understand. Compared with the statistic model, WEAN generates a more fluent sentence. Besides, WEAN can capture the semantic meaning of the word by querying the word embeddings, so the generated sentence is semantically correct, and very close to the original meaning.

5 Related Work

Our work is related to the encoder-decoder framework (Cho et al., 2014) and the attention mechanism (Bahdanau et al., 2014). Encoder-decoder framework, like sequence-to-sequence model, has achieved success in machine translation (Sutskever et al., 2014; Jean et al., 2015; Luong et al., 2015; Lin et al., 2018), text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Wang et al., 2017; Ma and Sun, 2017), and other natural language processing tasks (Liu et al., 2017). There are many other methods to improve neural attention model (Jean et al., 2015; Luong et al., 2015).

Zhu et al. (2010) constructs a wikipedia dataset, and proposes a tree-based simplification model. Woodsend and Lapata (2011) introduces a data-driven model based on quasi-synchronous grammar, which captures structural mismatches and complex rewrite operations. Wubben et al.

(2012) presents a method for text simplification using phrase based machine translation with re-ranking the outputs. Kauchak (2013) proposes a text simplification corpus, and evaluates language modeling for text simplification on the proposed corpus. Narayan and Gardent (2014) propose a hybrid approach to sentence simplification which combines deep semantics and monolingual machine translation. Hwang et al. (2015) introduces a parallel simplification corpus by evaluating the similarity between the source text and the simplified text based on WordNet. Glavaš and Štajner (2015) propose an unsupervised approach to lexical simplification that makes use of word vectors and require only regular corpora. Xu et al. (2016) design automatic metrics for text simplification. Recently, most works focus on the neural sequence-to-sequence model. Nisioi et al. (2017) present a sequence-to-sequence model, and re-ranks the predictions with BLEU and SARI. Zhang and Lapata (2017) propose a deep reinforcement learning model to improve the simplicity, fluency and adequacy of the simplified texts. Cao et al. (2017) introduce a novel sequence-to-sequence model to join copying and restricted generation for text simplification.

Rush et al. (2015) first used an attention-based encoder to compress texts and a neural network language decoder to generate summaries. Following this work, recurrent encoder was introduced to text summarization, and gained better performance (Lopyrev, 2015; Chopra et al., 2016). Towards Chinese texts, Hu et al. (2015) built a large corpus of Chinese short text summarization. To deal with unknown word problem, Nallapati et al. (2016) proposed a generator-pointer model so that the decoder is able to generate words in source texts. Gu et al. (2016) also solved this issue by incorporating copying mechanism.

6 Conclusion

We propose a novel model based on the encoder-decoder framework, which generates the words by querying distributed word representations. Experimental results show that our model outperforms the sequence-to-sequence baseline by the BLEU score of 6.3 and 5.5 on two English text simplification datasets, and the ROUGE-2 F1 score of 5.7 on a Chinese summarization dataset. Moreover, our model achieves state-of-the-art performances on these three benchmark datasets.

Acknowledgements

This work was supported in part by National Natural Science Foundation of China (No. 61673028), and the National Thousand Young Talents Program. Xu Sun is the corresponding author of this paper.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. pages 3152–3158.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2015)*. AAAI, New York, NY.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*. pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M. Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. pages 93–98.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings*. pages 758–764.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL*. pages 63–68.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016*.

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. LCSTS: A large scale chinese short text summarization dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1967–1972.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL HLT 2015*. pages 211–217.

Sébastien Jean, KyungHyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, ACL 2015*. pages 1–10.

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL*. pages 1537–1546.

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

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 2091–2100.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics, HLT-NAACL 2003*.

Junyang Lin, Shuming Ma, Qi Su, and Xu Sun. 2018. Decoding-history-based adaptive control of attention for neural machine translation. *CoRR* abs/1802.01812.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2017. Table-to-text generation by structure-aware seq2seq learning. *CoRR* abs/1711.09724.

Konstantin Lopyrev. 2015. Generating news headlines with recurrent neural networks. *CoRR* abs/1512.01712.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. pages 1412–1421.

Shuming Ma and Xu Sun. 2017. A semantic relevance based neural network for text summarization and text simplification. *CoRR* abs/1710.02318.

Shuming Ma, Xu Sun, Jingjing Xu, Houfeng Wang, Wenjie Li, and Qi Su. 2017. Improving semantic relevance for sequence-to-sequence learning of chinese social media text summarization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*. pages 635–640.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*. pages 280–290.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL*. pages 435–445.

Sergiu Nisioi, Sanja Stajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL*. pages 85–91.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. pages 311–318.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek V. Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, 2016, Osaka, Japan*. pages 2923–2934.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. 2017a. meprop: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. pages 3299–3308.

Xu Sun, Xuancheng Ren, Shuming Ma, Bingzhen Wei, Wei Li, and Houfeng Wang. 2017b. Training simplification and model simplification for deep learning: A minimal effort back propagation method. *CoRR* abs/1711.06528.

Xu Sun, Bingzhen Wei, Xuancheng Ren, and Shuming Ma. 2017c. Label embedding network: Learning label representation for soft training of deep networks. *CoRR* abs/1710.10393.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*. pages 3104–3112.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1054–1059.

Kexiang Wang, Tianyu Liu, Zhifang Sui, and Baobao Chang. 2017. Affinity-preserving random walk for multi-document summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 210–220.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*. pages 1015–1024.

Jingjing Xu, Xu Sun, Xuancheng Ren, Junyang Lin, Binzhen Wei, and Wei Li. 2018. Dp-gan: Diversity-promoting generative adversarial network for generating informative and diversified text. *CoRR* abs/1802.01345.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *TACL* 4:401–415.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*. pages 584–594.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *COLING 2010*. pages 1353–1361.