Recurrent Neural Network-Based Semantic Variational Autoencoder for Sequence-to-Sequence Learning

Myeongjun Jang [1]   Seungwan Seo [1]   Pilsung Kang [1]

Abstract

Sequence-to-sequence (Seq2seq) models have played an important role in the recent success of various natural language processing methods, such as machine translation, text summarization, and speech recognition. However, current Seq2seq models have trouble preserving global latent information from a long sequence of words. Variational autoencoder (VAE) alleviates this problem by learning a continuous semantic space of the input sentence. However, it does not solve the problem completely. In this paper, we propose a new recurrent neural network (RNN)-based Seq2seq model, RNN semantic variational autoencoder (RNN–SVAE), to better capture the global latent information of a sequence of words. To reflect the meaning of words in a sentence properly, without regard to its position within the sentence, we construct a document information vector using the attention information between the final state of the encoder and every prior hidden state. Then, the mean and standard deviation of the continuous semantic space are learned by using this vector to take advantage of the variational method. By using the document information vector to find the semantic space of the sentence, it becomes possible to better capture the global latent feature of the sentence. Experimental results of three natural language tasks (i.e., language modeling, missing word imputation, paraphrase identification) confirm that the proposed RNN–SVAE yields higher performance than two benchmark models.

Keywords: *Sequence-to-sequence learning, Recurrent neural network, Auto-encoder, Variational method, Document information vector, Natural language processing*

[1]School of Industrial Management Engineering, Korea University, Seoul, South Korea. Correspondence to: Pilsung Kang <pilsung_kang@korea.ac.kr>.

1. Introduction

Sequence-to-sequence (Seq2seq) models (Cho et al., 2014b; Sutskever et al., 2014), based on recurrent neural networks (RNN), show excellent capability for processing a variable lengths of sequential data. In recent years, these structures have led to the noteworthy development of language models and have played an important role in the development of various tasks of natural language processing (NLP), such as machine translation (Bahdanau et al., 2014; Cho et al., 2014b; Sutskever et al., 2014; Ling et al., 2015; Luong et al., 2015; Zhao & Zhang, 2016; Lee et al., 2016; Ha et al., 2016; Artetxe et al., 2017), machine comprehension (Hermann et al., 2015; Rajpurkar et al., 2016; Yuan et al., 2017), text summarization (Bahdanau et al., 2016; Chan et al., 2016; Nallapati et al., 2016), and speech recognition (Graves & Jaitly, 2014; Huang et al., 2016; Chan et al., 2016; Bahdanau et al., 2016). The simplest Seq2Seq structure is the RNN autoencoder (RNN–AE), which receives a sentence as input and returns itself as output (Dai & Le, 2015). Because this model is an unsupervised method that does not require labeled data, it is very easy to obtain training data. Thus, the RNN–AE can be applied to diverse tasks. It has been used to pre-train parameters of a text classification model, achieving better performance than random parameter initialization (Dai & Le, 2015). It also has been used to generate long-length sentences (Li et al., 2015). Furthermore, it has been applied to not only text data but also to acoustic and video data, which also have sequential information, such as novelty detection of acoustic and video data (Marchi et al., 2017; D'Avino et al., 2017) and representation learning of acoustic data (Amiriparian et al., 2017).

Although the RNN–AE shows good performance in many studies, it has limitations. First, because the compressed information of the input sentence is

S₁: We should build more buildings but we have not enough lumbers

S₂: We need more lumbers to construct more buildings

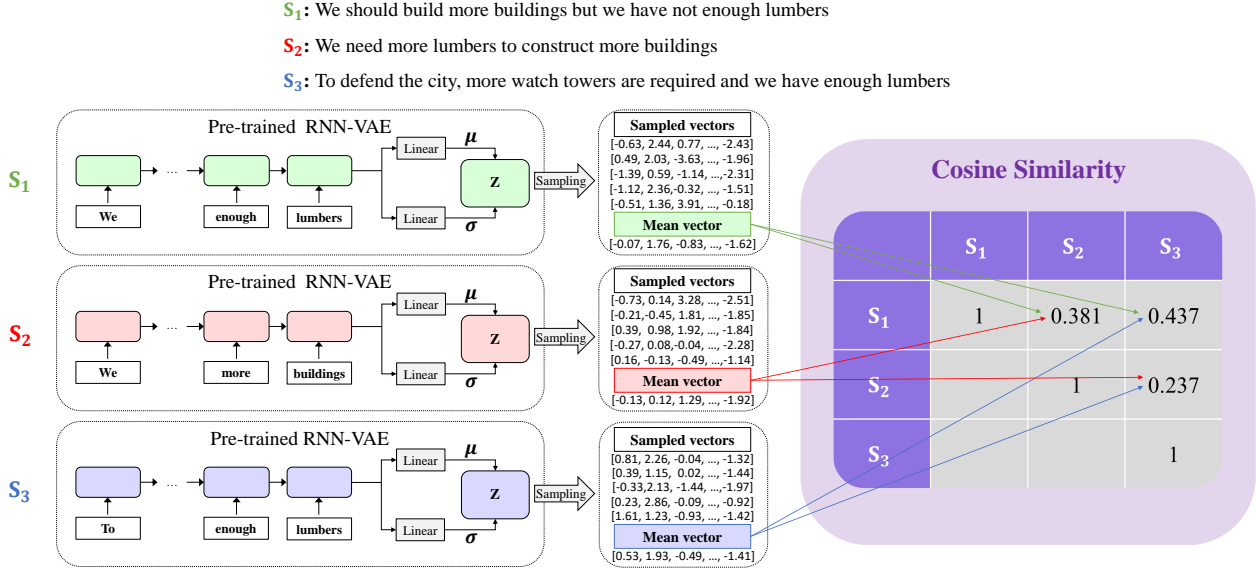**S₃:** To defend the city, more watch towers are required and we have enough lumbers

Figure 1. Cosine similarity of sentence information vectors produced by the VAE.

learned as a fixed vector (i.e., encoder final state), there is a high probability that the output will not be good, even for small changes in the vector value. Second, it is not easy to find the global latent feature of the input sentence, owing to the structural features of the RNN that performs the prediction for the next stage of the series (Bowman et al., 2015). Bowman et al. (2015) proposed the RNN variational autoencoder (RNN–VAE) model to resolve the above issues by applying a variational inference technique (Kingma & Welling, 2013; Rezende et al., 2014) to the RNN–AE. The RNN–VAE model was successful at moderating the high sensitivity of the RNN–AE to small changes in the final state vector values by learning the information of the input sentence as a probabilistic continuous vector, instead of a fixed vector. The RNN–VAE exhibited a better basic structure than RNN–AE for various NLP tasks, including machine translation (Zhang et al., 2016) and text classification (Xu et al., 2017)

However, the RNN–VAE seems unable to completely solve the problem of RNN-AE, because it seems that it does not capture the global latent feature of the input sentence. With RNN–VAE, the mean and standard deviation of the continuous space of the input sentence information are calculated from the final state of the RNN–AE encoder. Because this final state is updated for each step of the word sequence making up the input sentence, it stores much more information about the last or the beginning parts, if bidirectional RNN is used, of the input sentence, rather than all of the sentence information. Therefore, the continuous space of the input sentence information derived from the final encoder state is hardly a semantic space for preserving the global latent feature. Figure 1 shows an example of this RNN–VAE problem with three sentences, S1, S2, and S3. Although S1 and S2 are semantically similar, their syntactic structures are quite different. On the other hand, although S1 and S3 are semantically opposite, they have the same words (enough lumbers) at the end of the sentence. The vector representation of each sentence is the average of sampled vectors from the continuous semantic space of the trained RNN–VAE. We sampled five times for each sentence to reduce bias of the sampled vector and used cosine similarity as the similarity measure between two vector representations. Although S1 is more semantically similar to S2, the cosine similarity between S1 and S3 is higher than that of S1 and S2.

RNN–AE uses an RNN architecture to construct the encoder and decoder, as shown in Figure 2. The encoder compresses the information of a series of inputs in a sequence (e.g., the words in a sentence), $\mathbf{w} = \{w_1, \ldots, w_T\}$, into a fixed vector, $\mathbf{v}$,

In this paper, we present RNN–SVAE for overcoming RNN–VAE limitations by generating a document information vector to capture the global latent feature of the input sentence. The document information vector consists of the word weights of a linear combination most correctly representing the paragraph vector using word vectors in the embedding space. This document information vector is combined with the final encoder
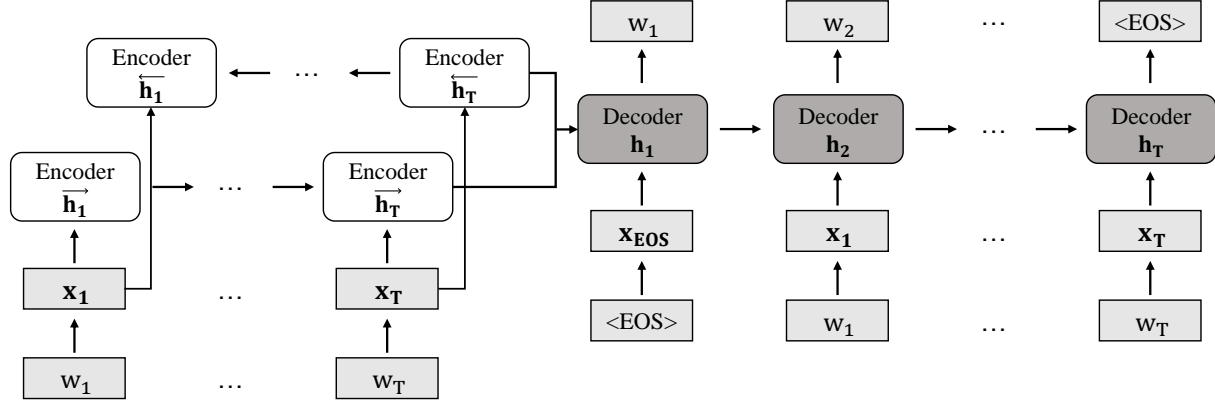
Figure 2. Structure of Seq2Seq AE model with Bi-directional structure.

state. The RNN–SVAE is trained based on the combined vector to find an appropriate continuous space of the input sentence. RNN–SVAE's effectiveness is verified by comparing its performance to that of RNN–AE and RNN-VAE, using three tasks: language modeling, missing word imputation, and paraphrase identification.

The rest of this paper is organized as follows. In Section 2, we briefly review past research on the autoencoder structure and demonstrate the methodologies used in this study. In Section 3, we catalogue the architecture of RNN–SVAE. In Section 4, experimental settings of each task are described, followed by results and discussion. Finally, in Section 5, we conclude our current work with some future research directions.

2. Background

2.1. RNN–AE

The AE, first introduced by Rumelhart et al. (1985), is a neural network-based unsupervised learning algorithm that has been employed for various tasks, including feature representation, anomaly detection, and transfer learning (Baldi, 2012; Bengio et al., 2013; Zhu et al., 2016; Sakurada & Yairi, 2014; Chen et al., 2017; Lyudchik, 2016; Zhuang et al., 2015; Deng et al., 2013). Input and output are the same in the AE structure. Thus, the AE's learning objective is to approximate the output to the input as closely as possible. The preceding part, compressing the information of the input vector to the latent vector, is called the *encoder*, and the following part, reconstructing the information from the latent vector to the output, is called the *decoder*.

The AE, first introduced by Rumelhart et al. (1985), is a neural network-based unsupervised learning algorithm that has been employed for various tasks, in-

cluding feature representation, anomaly detection, and transfer learning (Baldi, 2012; Bengio et al., 2013; Zhu et al., 2016; Sakurada & Yairi, 2014; Chen et al., 2017; Lyudchik, 2016; Zhuang et al., 2015; Deng et al., 2013). Input and output are the same in the AE structure. Thus, the AE's learning objective is to approximate the output to the input as closely as possible. The preceding part, compressing the information of the input vector to the latent vector, is called the *encoder*, and the following part, reconstructing the information from the latent vector to the output, is called the *decoder*.

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1}), \tag{1}$$
$$\mathbf{v} = q(\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}), \tag{2}$$

where w_i is the ith input word, $\mathbf{x}_i$ is the input vector of the w_i, and $\mathbf{h}_i$ is the hidden state of the ith sequence. f and q are nonlinear functions, where $q(\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}) = \mathbf{h}_T$. The decoder is trained to maximize the conditional probability of predicting the next word, $\hat{y}_{t'}$, given a fixed vector, $\mathbf{v}$, and previously predicted words, $\{\hat{y}_1, \ldots, \hat{y}_{t'-1}\}$. Thus, the purpose of the decoder is to maximize the probability of predicting the target sequence, $\mathbf{y} = \{y_1, \cdots, y_{t'}\}$,

$$p(\hat{\mathbf{y}}) = \prod_{t=1}^{T'} p(\hat{y}_t | \{\hat{y}_1, \ldots, \hat{y}_{t-1}\}, \mathbf{v}). \tag{3}$$

Because the objective is to precisely reconstruct the input, $\mathbf{y}$ is identical to $\mathbf{x}$ in the RNN–AE. The conditional probability of RNN structure at time, t, is defined as

$$p(\hat{y}_t | \{\hat{y}_1, \ldots, \hat{y}_{t-1}\}, \mathbf{v}) = g(\hat{y}_{t-1}, \mathbf{s}_t, \mathbf{v}), \tag{4}$$

where $\mathbf{s}_t$ and g denote the hidden state of the decoder at time, t, and a non-linear function, respectively.
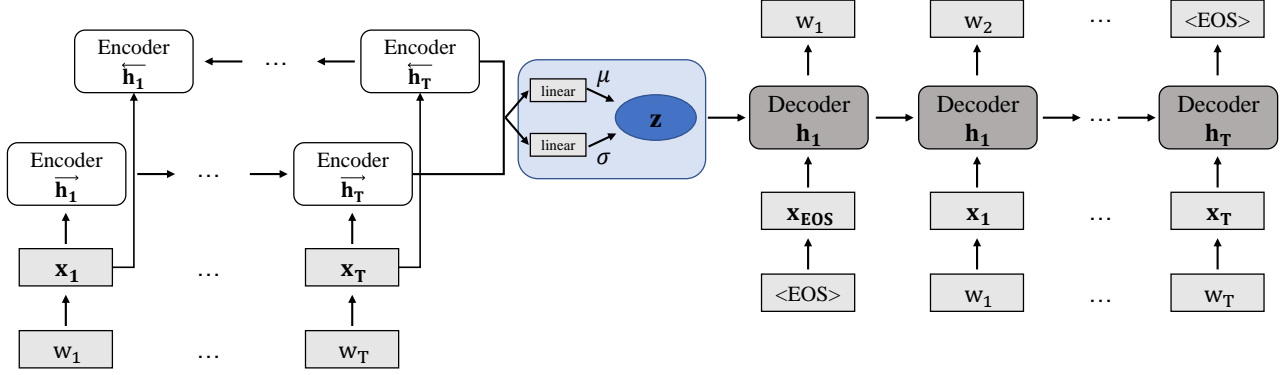
Figure 3. Structure of RNN–VAE model with Bi-directional structure

2.2. RNN–VAE

RNN–VAE is a generative model that improves the RNN–AE to capture the global feature of the input sentence. RNN–VAE replaces the deterministic function, $q(\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}) = \mathbf{h}_T$, of RNN—AE with the posterior recognition model, $q(\mathbf{z}|\mathbf{x})$, which compresses the information of input sentence, $\mathbf{x}$, into a probabilistic distribution. The parameters, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, determining $q(\mathbf{z}|\mathbf{x})$, are calculated as a linear transformation of the encoder output. Thus, RNN–VAE is a model that learns the compressed information of the input sentence as a region of latent space, rather than as a single point. The structure of RNN–VAE model is shown in Figure 3

If the RNN–VAE is trained only with the RNN–AE's reconstruction objective, it would encode the input sentence as an isolated point which means that it makes the variance of $q(\mathbf{z}|\mathbf{x})$ very small (Bowman et al., 2015). To deal with this problem, in addition to the reconstruction objective, the RNN–VAE has another objective that approximates the posterior distribution, $q(\mathbf{z}|\mathbf{x})$, to the prior distribution, $p(\mathbf{z})$. This is generally a standard Gaussian distribution, $(\mu = \vec{0}, \ \sigma = \vec{1})$. The Kullback-Leibler divergence (KLD) is used to compute the difference between the two distributions. Thus, the objective of RNN–VAE is defined as

$$\mathcal{L}(\theta : \mathbf{x}) = -KLD(q_\theta(\mathbf{z}|\mathbf{x})|p(\mathbf{z})) + E_{q_\theta(\mathbf{z}|\mathbf{x})}[log p_\theta(\mathbf{x}|\mathbf{z})], \quad (5)$$

where θ is the model parameter (i.e., μ and σ of Gaussian distribution) in the RNN–VAE. This objective allows the RNN–VAE to decode output at every point in the continuous space, having high probability under the prior distribution.

2.3. Paragraph Vector

Paragraph vector (Le & Mikolov, 2014) has been widely used to represent a paragraph using an arbitrary number of words into a fixed low-dimensional continuous vector to overcome the limitations of the bag-of-words (BoW) method. There are two main ways to learn the paragraph vector: the paragraph vector with distributed memory (PV-DM) method and the paragraph vector with distributed BoW (PV-DBOW) method. The PV-DM method, which considers the order of word sequence, has a similar model structure to continuous BoW (CBOW) of the Word2Vec model. This model takes the paragraph token vector, $\mathbf{p}$, and word vectors, $\mathbf{x}_i, \ldots, \mathbf{x}_{i+(t-1)}$, to predict the next word, $\mathbf{x}_{i+t}$, when the sliding window size is set to t. Thus, the paragraph vector is trained to maximize the probability of its appearance with the words contained in the sliding window of the paragraph. In the PV-DBOW method, the words included in the fixed window are arbitrarily sampled from those constituting the paragraph. This model takes the paragraph vector as input and predicts the sampled words. Therefore, it does not consider the order of the paragraph's word sequence. Both methods define the probability that a paragraph token and a word token appear together, using the dot product between the vectors of each token. Therefore, the paragraph vector is located close to the word vectors within the paragraph from the semantic embedding.

2.4. Attention Mechanism

The attention mechanism (Luong et al., 2015; Bahdanau et al., 2014), recently recognized for its effectiveness, is widely used for image captioning (Xu et al., 2015), tree parsing (Vinyals et al., 2015), question answering (Hermann et al., 2015), and machine trans-
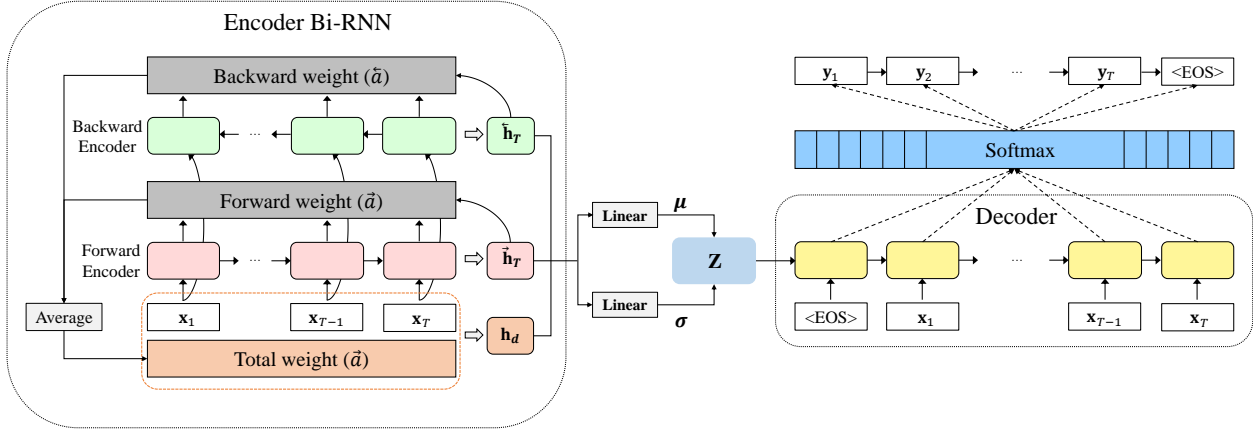
Figure 4. Structure of SVAE model

lation. The main problem of vanilla RNN is that it hardly preserves the information of the words from the front of sentence when the input sentence becomes longer. This is because it only uses the last hidden state of the encoder. Although the long short-term memory (LSTM, (Hochreiter & Schmidhuber, 1997)) or gated recurrent unit (GRU, (Cho et al., 2014a)) tends to alleviate the problem, they still have trouble preserving the well-balanced semantic information of a sentence, regardless of the word appearance sequence. An attention mechanism solves this problem by using a weighted combination of total hidden states (i.e., context vector) and the last hidden state for each decoding step. The weights of the context vector can be regarded as the importance of input sentence words in the corresponding step.

3. Model Structure

In this study, we propose the RNN semantic variational autoencoder (RNN–SVAE) which represents the global latent feature of an input sentence better than RNN–VAE. As shown in Figure 4, RNN–SVAE integrates the final hidden state and the document information vector, based on the attention vectors of bidirectional RNN (bi-RNN) hidden states, before estimating the parameters of Gaussian distribution. Because every word in the input sentence is equally considered in the document information vector, the RNN–SVAE can preserve the global latent feature better than the RNN–VAE, which has highly skewed information toward the latter words. Additionally, because the document information vector is computed by aggregating the attention vectors, model-training is not required to separately learn the document information vector; it is learned simultaneously with the hidden

state during RNN training.

3.1. Document Information Vector

For both PV-DM and PV–CBOW methods, a paragraph vector is placed near the word vectors constituting it because a d-dimensional paragraph vector is trained to maximize the dot product of the d-dimensional word vectors in the paragraph. This implies that a linear combination of d linearly independent word vectors can accurately reconstruct the paragraph vector. Furthermore, because the paragraph vector has a high similarity to the vectors of words constituting the paragraph, it is possible to approximate the paragraph vector using the embedding vectors of its words, as follows.

$$\mathbf{v}_d \cong \sum_{i=1}^{T} a_i \mathbf{x}_i, \qquad (6)$$

where $\mathbf{x}_i$ and a_i denote the ith word vector and its linear combination weight, respectively. $\mathbf{v}_d$ is the paragraph vector.

Whereas PV-DM and PV–CBOW explicitly learn the paragraph vector during model training, the proposed document information vector computes it implicitly using information obtained during Seq2seq model training. Because the last hidden state, $(\mathbf{h}_T)$, of the encoder, is a vector containing the sequential information of the input sentence, we compute the weight, a_i, using the relationship between the $\mathbf{h}_T$ and the ith hidden state, $(\mathbf{h}_i)$. Many past studies used their dot product as a similarity measure (Karpathy et al., 2014; Karpathy & Fei-Fei, 2015). We instead use the normalized value of the dot product between $\mathbf{h}_T$ and $\mathbf{h}_i$

as the a_i,

$$a_i = \frac{e_i}{\sum_{k=1}^{T} e_k}, \qquad \text{where} \quad e_i = \mathbf{h}_i \cdot \mathbf{h}_T. \qquad (7)$$

It is possible to use many other alignment models that are proposed by Luong et al. (2015) or Bahdanau et al. (2014). However, we used a simple normalized dot product to focus on the effectiveness of document vector itself.

Using the standard RNN structure tends to give a larger weight to the words at the end of the input sentence. The closer i is to T, the more similar $\mathbf{h}_i$ is to $\mathbf{h}_T$. To solve this problem, we use bi-RNN (Schuster & Paliwal, 1997) to take the average of the forward weight, $(\overrightarrow{a})$, and the backward weight, $(\overleftarrow{a})$,

$$\overleftarrow{a_i} = \frac{\overleftarrow{\mathbf{h}_i} \cdot \overleftarrow{\mathbf{h}_T}}{\Sigma_{k=1}^{T} \overleftarrow{\mathbf{h}_k} \cdot \overleftarrow{\mathbf{h}_T}}, \quad \overrightarrow{a_i} = \frac{\overrightarrow{\mathbf{h}_i} \cdot \overrightarrow{\mathbf{h}_T}}{\Sigma_{\mathbf{k}=1}^{T} \overrightarrow{\mathbf{h}_k} \cdot \overrightarrow{\mathbf{h}_T}}, \quad \bar{a}_i = \frac{\overrightarrow{a_i} + \overleftarrow{a_i}}{2}, \quad (8)$$

where $\overrightarrow{\mathbf{h}_i}$ and $\overleftarrow{\mathbf{h}_i}$ is the forward and backward hidden states at the ith word, respectively. Finally, we compute the document information vector by combining the total weight, $(\bar{a})$, and the word sequence, $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$, of the input sentence,

$$\mathbf{h}_d = \sum_{i=1}^{T} \bar{a}_i \mathbf{x}_i, \qquad (9)$$

where the $\mathbf{h}_d$ is the document information vector of the input sentence. Contrary to the paragraph vector, which should be trained separately from the RNN model to learn the sentence vector, the proposed document information vector can be computed simultaneously using the learned parameters of the RNN model. Hence, unlike the paragraph vector, it is not necessary to learn the sentence vector for each new sentence.

3.2. RNN–SVAE

The structure of RNN–SVAE model is created by adding the document information vector to the RNN–VAE model. The overall structure of the proposed model is summarized in Figure 4. We construct the final state of the encoder by concatenating the forward final state $(\overrightarrow{\mathbf{h}_T})$, backward final state $(\overleftarrow{\mathbf{h}_T})$, and document information vector $(\mathbf{h}_d)$ as follows,

$$\mathbf{h}_L = [\overrightarrow{\mathbf{h}_T} \; ; \; \overleftarrow{\mathbf{h}_T} \; ; \; \mathbf{h}_d] \qquad (10)$$

Next, the mean, $(\boldsymbol{\mu})$, and the standard deviation, $(\boldsymbol{\sigma})$, vectors of the continuous semantic space is calculated from the encoder's last state $(\mathbf{h}_L)$ via linear transformation. These vectors have the same dimension as the

global latent vector, $(\mathbf{z})$. Finally, we sample the global latent vector, which functions as the semantic vector, of the input sentence and is used as an input vector to the decoder, from the continuous semantic space.

$$\boldsymbol{\mu} = \mathbf{W}_\mu \mathbf{h}_L + \mathbf{b}_\mu, \quad \boldsymbol{\sigma} = \mathbf{W}_\sigma \mathbf{h}_L + \mathbf{b}_\sigma, \qquad (11)$$

$$\mathbf{z} \sim Gaussian(\overrightarrow{0}, \overrightarrow{1}), \qquad (12)$$

where $\mathbf{W}_\mu$ and $\mathbf{b}_\mu$ are the weight and bias for $\boldsymbol{\mu}$, respectively, whereas, $\mathbf{W}_\sigma$ and $\mathbf{b}_\sigma$ are the weight and bias, respectively, for $\boldsymbol{\sigma}$.

Similar to RNN–VAE, the RNN–SVAE's cost function reflects two objectives, as shown in Eq. (5). The first objective is to closely approximate the posterior distribution, $q(\mathbf{z}|\mathbf{x})$, with the parameters, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, to the prior distribution, $p(\mathbf{z})$, which is the standard Gaussian distribution. To do so, the KLD, having the standard Gaussian distribution, should be minimized. The second objective is to maximize the conditional probability, $p(y_1, \ldots, y_{T'}|\mathbf{x}_1, \cdots, \mathbf{x}_T)$, like in the general Seq2seq model. $\mathbf{w} = \{w_1, \ldots, w_T\}$ is the word sequence of the input sentence, and $\mathbf{y} = \{y_1, \ldots, y_{T'}\}$ is the output sequence. Because the RNN–SVAE model is an autoencoder structure, $\mathbf{w}$ is identical to $\mathbf{y}$.

4. Experiments

During our experiments, we verified the RNN–SVAE with the three tasks: language modeling, missing word imputation, and paraphrase identification. As baseline models, RNN–AE and RNN–VAE were also used. Evaluation and comparison were both conducted quantitatively with the standard evaluation metrics, and qualitatively by exploiting the output examples of the three models.

4.1. Language Modeling

To evaluate the fundamental ability of RNN-SVAE as an autoencoder, language modeling was tested first.

4.1.1. DATA SET AND PREPROCESSING

In this study, we used the News Crawl data of WMT' 17[1], English monolingual corpus, and TED Talk data of WIT3[2] (Cettolo et al., 2012). The News Crawl'13 dataset was used to train the RNN models, whereas all datasets were used to test the models. For the language modeling task, it is common to exclude very long sentences (i.e., longer than 30 to 50 words) to accelerate training (Bahdanau et al., 2014; Artetxe et al., 2017). Therefore, we only used sentences shorter than

---

[1] http://www.statmt.org/wmt17/translation-task.html
[2] https://wit3.fbk.eu/

Table 1. Number of sentences for each data set

	News Crawl'13	News Crawl'14	News Crawl'15	News Crawl'16	TED Talk
Train	2,500,000	-	-	-	-
Test	15,000	15,000	15,000	15,000	15,000

40 words for computational efficiency. For the training dataset, we randomly sampled 2,500,000 sentences from the News Crawl'13 dataset. As test dataset, News Crawl'13, News Crawl'14, News Crawl'15, and News Crawl'16 datasets of WMT' 17 English monolingual corpus and the TED Talk dataset were used. For each dataset, 15,000 sentences were randomly sampled. For the test data of News Crawl'13 dataset, the sampled sentences in the training dataset were excluded when sampling the test dataset. The number of training and test sentences in each dataset is summarized in Table 1.

Prior to training the model, we performed tokenization[3] after removing punctuation marks and converting uppercase letters to lowercase letters for all sentences. Following tokenization, we pre-trained the word vectors by using the skip-gram model (Mikolov et al., 2013). Words that appeared fewer than seven times in the training dataset were replaced with the "UNK" token (i.e., unknown word). We set the dimension of word vector to 100, the window size of the skip-gram model to 5, and the negative sampling parameter, k, to 5. Word vector training was done for 10 epochs. Thus, including "UNK" and "EOS" (i.e., end-of-sentence) token, 91,897 unique words were trained.

4.1.2. Model Training and Inference

Because the RNN–SVAE model is rooted on vanilla RNN, it is possible to use any type of RNN cell (e.g., basic RNN cell, LSTM cell, GRU cell) (Cho et al., 2014a). We used the GRU cell that solves the gradient vanishing problem of the basic RNN cell. It also has fewer parameters than the LSTM cell. The RNN–SVAE encoder has a bi-directional RNN structure. The forward and backward RNNs of the encoder each consist of 300 hidden units. The global latent vector and hidden states of the decoder also consist of 300 units.

For fair comparison, the baseline models are designed with the same structure as the RNN-SVAE model. The RNN–VAE model also used the GRU cell and had a bi-RNN structure with 300 hidden units. Its global latent vector and decoder hidden units were composed

---

[3]We used RegExpTokenizer from the NLTK package. (http://www.nltk.org/api/nltk.html)

of 300 units, as in the RNN–SVAE model. The RNN–AE model also used the GRU cell with bi-RNN structure and had the encoder and decoder with 300 hidden units, as in the RNN–VAE and RNN–SVAE models.

The three models were all trained under the same condition. We initialized their parameters using the Xavier initialization (Glorot & Bengio, 2010). We used the Adam optimizer (Kingma & Ba, 2014) for training. We trained the models for 30 epochs. Gradient computation and weight update were done with the mini-batch size of 512. The learning rate was set to 0.001 for the first 10 epochs and to 0.0001 for the remaining 20 epochs.

After model training, beam search was used to obtain the output maximizing conditional probability at the inference phase. We set the beam size to 7 and the maximum length of output to 40. For generative models, such as RNN–VAE and RNN–SVAE, an average of five samples was used as the input vector of the decoder to reduce the bias of sampled global latent vector.

4.1.3. Results

As a performance measure for language modeling, the BLEU score, commonly used for machine translation, was used (Papineni et al., 2002). Although there exists other "teacher forcing" metrics, such as negative log likelihood and perplexity, these metrics are insufficient in evaluating whether the semantic space or vector, i.e. the output of the encoder, reflects the global latent feature of the input sentence because the target token at every time step is provided for those "teacher forcing" metrics. As a result, we used BLEU score as a performance measure that target tokens are not provided in each decoding step. The results of each model are summarized in Table 2. For all five test data sets, the proposed RNN–SVAE significantly outperformed the benchmark models. The BLEU scores of RNN–SVAE were almost as twice those of the RNN–AE. Compared to RNN–VAE, RNN–SVAE improved the BLEU score by at least 3.27 (News Crawl'14) and at most 4.17 (News Crawl'13). The relative BLEU improvements of RNN–SVAE against RNN–VAE were between 8.42% and 11.12%.

Table 3 shows examples of the language modeling

Table 2. BLEU score of each model for the language modeling task

Model \ Data	News Crawl'13	News Crawl'14	News Crawl'15	News Crawl'16	TED Talk
RNN–AE	17.55	20.21	19.98	20.55	24.84
RNN–VAE	37.51	38.62	39.41	38.98	45.49
RNN–SVAE	**41.68**	**41.89**	**43.33**	**43.07**	**49.32**

(a) News Crawl'13

(b) News Crawl'14

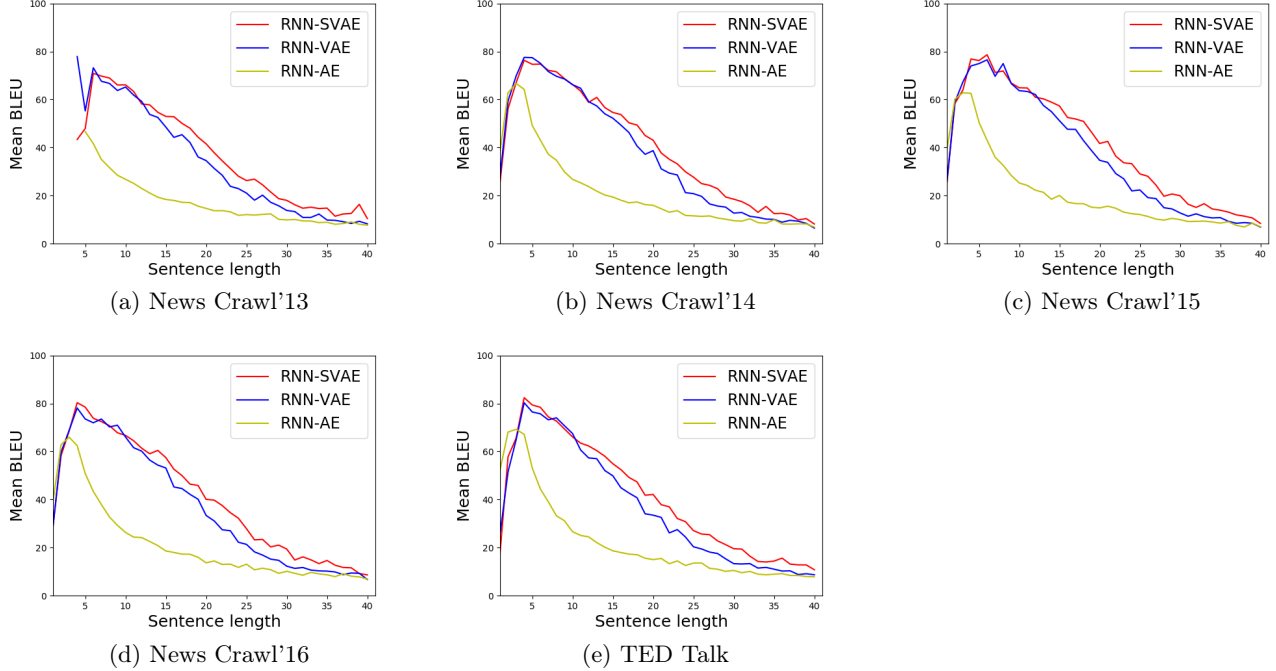(c) News Crawl'15

(d) News Crawl'16

(e) TED Talk

Figure 5. BLEU scores for each model according to the sentence length.

task[4]. Examples of RNN–AE are not given, because its language modeling performance was significantly worse than those of RNN–VAE and RNN–SVAE. Highlighted parts, in gray, show the words exactly matched with the ground truth. In the case of RNN–VAE, both the beginning and the end of sentences fit well with the ground truth. However, it seemed to have difficulty generating the parts in the middle of sentences correctly. However, RNN–SVAE succeeded in generating the entire sentences.

Whereas the word sequences generated by the RNN–SVAE are not the same as the ground truth, the chosen words are semantically very similar to those in the ground truth. As shown in Table 4. When the word "nodes" is replaced by "fibres" in the first example sentence, it is still comprehensible and does not undermine the meaning of the original sentence. Similarly,

"Norwich" and "Southampton" are both the name of cities in the England in the second sample, and "frost" and "snow" are semantically very similar words in the third example.

Figure 5 shows the average BLEU score of each model per the sentence length of each dataset. When the sentence length is relatively short, sentence generation performance was similar. Even the RNN–AE worked well with very short sentences (i.e., less than five words). Additionally, there was no significant difference between the RNN–VAE and RNN–SVAE. When the sentence length was moderate, RNN–AE tended to fail to generate the original sentence; its BLEU scores were much lower than those of the other two methods. When comparing RNN–VAE and RNN–SVAE, the RNN–SVAE worked better than RNN–VAE in most cases. When the sentence length was long, all three models had trouble generating the original sentence. This is still an open research topic in the field of machine translation.

---

[4]We used smoothing function in NLTK package, because the BLEU score was too optimistic

Table 3. Examples of language modeling outputs generated by RNN–VAE and RNN–SVAE

Type	Sentence
Truth	What is the name of the pension plan
RNN–VAE	What is the name and the name expires
RNN–SVAE	What is the name of the pension plan
Truth	The grandmother of three who whishes to remain anonymous said the experience was so traumatic she will never be able to eat popcorn again
RNN–VAE	The grandmother of the kitchens of how serving ladies advised were the added experience so you so fully flowers to likely to eat never desire
RNN–SVAE	The grandmother of three who whish to remain anonymous said the experience was so traumatic she will never be able to eat before pets again
Truth	The authorities arrested two people but failed to investigate reports that they were part of a large private militia
RNN–VAE	The authorities raped some people but failed to investigate reports that they were part of a large private militia
RNN–SVAE	The authorities arrested two people but failed to investigate reports that they were part of a large private Kuwaiti

Table 4. Examples of wrong words but have high similarity with the ground truth

Type	Sentence
Truth	These <u>nodes</u> range from opening and closing tags to character data and processing instructions
RNN–SVAE	These <u>fibres</u> range from opening and closing tags to character data and processing instructions
Truth	From there Jennings took the controls and flew to <u>Norwich</u>
RNN–SVAE	From there Jennings took the controls and flew to <u>Southampton</u>
Truth	Kevin Walker head of science at the <u>BSBI</u> said the trend was down to the mild winter and a lack of <u>frost</u>
RNN–SVAE	Kevin Walker head of science at the <u>UNK</u> said the trend was down to the mild winter and a lack of <u>snow</u>

4.2. Missing Word Imputation

Missing word imputation is the process of completing a sentence by filling it in with appropriate words (Mani, 2015). We performed this task to evaluate how well the proposed RNN–SVAE reflects the global latent feature of the input sentences. In this task, an incomplete sentence with some words erased was provided as an input to the encoder of seq2seq models. The models were trained to guess the erased words or the sequence of words correctly through the decoder. We tested the missing word imputation performance under three different scenarios. The detailed description of each scenario is summarized below.

- **Scenario 1**: Imputation for the last word of the sentence.

- **Scenario 2**: Imputation for one randomly selected word among the last 20% of the sentence.

- **Scenario 3**: Imputation for the sequence of words corresponding to the last 20% of the sentence.

Scenario 1 was the easiest level and Scenario 3 was the most difficult level. Scenario 1 and 2 can be regarded as a multi-class classification task, whereas Scenario 3 can be regarded as a sequence generation task.

4.2.1. DATA SET

For model training, the training dataset used in the language modelling task (i.e., the randomly sampled 2,500,000 sentences from the News Crawl'13 dataset) was modified. Likewise, we modified the News Crawl'13 test dataset and used it to evaluate performance. For Scenarios 1 and 3, we erased the last word and the last 20% of word sequences from each sentence, respectively. For Scenario 2, we replaced a randomly selected word among the last 20% of the

Table 5. Performance of missing word imputation task.

Scenario Model	Scenario 1 (Accuracy)	Scenario 2 (Accuracy)	Scenario 3 (BLEU)
RNN–AE	15.71	5.76	34.08
RNN–VAE	**16.94**	6.23	34.17
RNN–SVAE	15.05	**6.37**	**34.37**

Table 6. Examples of missing word imputation.

Level 1	Q: Inventories increased across divisions, but were compensated by advance payments received and a better operational ____.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"performance"	"performance"	"performance"	"service"
	Q: Click here for instructions on how to enable javascript in your ____.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"browser"	"browser"	"browser"	"system"
Level 2	Q: David Rhodes and Robert Hendricks (Montreal process technical advisory group tac) described tac's work on a framework of criteria and indicators that provide a common of ____ management of temperate and boreal forests.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"sustainable"	"the"	"the"	"sustainable"
	Q: Such distinctive homes can attract interest from far beyond your ____ market.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"local"	"local"	"local"	"own"
	Q: The list is sorted by country so you shouldn't have a problem to find a ____ near you.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"vendor"	"destination"	"few"	"hotel"
Level 3	Q: If you have text in any page of your site that contain any of the keywords below, you can add your contextual listing there. It's free and your listing will appear online in ____.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"real time containing hyperlink to your page"	"real time http www 'UNK' com au account"	"real time containing your account"	"real time containing hyperlink to your page"
	Q: Energy star is a registered trademark of the US environmental ____.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"protection agency"	"protection agency"	"protection agency"	"insurance program"
	Q: Encrypt within the veritas net backup policy eliminating a seperate process or an extra dedicated ____.			
	Truth	**RNN–AE**	**RNN–VAE**	**RNN–SVAE**
	"device to manage"	"to the application"	"to the enviroment"	"device to manage"

sentence with 0 vector.

4.2.2. MODEL TRAINING AND INFERENCE

Three imputation models were trained under the same condition. We used the Xavier initialization for parameter initialization and the Adam optimizer. The models were trained for 15 epochs with a learning rate 0.001 for the first five epochs and 0.0001 for remaining 10 epochs. Gradient computation and weight updates were done with a mini-batch size of 512. Like the language modeling task, the outputs of RNN–VAE and RNN–SVAE were decoded from the mean vector of five sampled values to reduce the bias of the global latent vector.

4.2.3. RESULTS

As a quantitative evaluation metric, the simple accuracy (i.e., the proportion of the correctly predicted
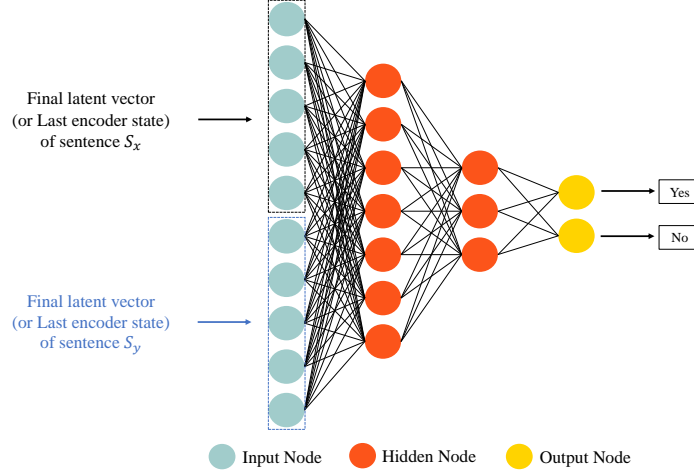
Figure 6. Structure of paraphrase identification model

words to the number of total missing words) was used for Scenarios 1 and 2 (i.e., predicting a single word), whereas the BLEU score was used for Scenario 3 (i.e., predicting a sequence of words).

Table 5 shows the performance of each model for missing word imputation. For Scenario 1, RNN–VAE yielded the highest accuracy, whereas RNN–SVAE resulted in the lowest accuracy. Because imputation for the last word requires more information about the end of the sentence than the global information of the whole sentence, RNN–VAE and RNN–AE, which preserve more information of the end of sentences, showed good performances. Although RNN–SVAE resulted in the worst performance, we found that its imputation results were semantically quite similar to the target word in many examples, as shown in Table 6.

For more difficult tasks, such as Scenario 2 and 3, on the other hand, the RNN–SVAE outperformed the other methods. As shown in Table 6, not only did RNN–SVAE achieve higher accuracy, or BLEU score, it also predicted semantically similar words to the correct answers.

4.3. Paraphrase Identification

Paraphrase identification is a task that determines whether two different sentences have the same meaning (Rus et al., 2008; Hu et al., 2014). In this study, we constructed a binary classification model to determine whether two sentences are paraphrased when global latent vectors, or the last hidden state for RNN—AE, of each sentence are used as input, as shown in Figure 6. Like the previous tasks, the mean vector of five sampled vectors is used as input to the paraphrase iden-

tification model for RNN–VAE and RNN–SVAE. The model is constructed with a feed-forward multi-layer perceptron consisting of two hidden layers. The number of hidden units of the first and the second layer are set to 100 and 50, respectively.

4.3.1. DATA SET

We used the MS Paraphrase Corpus dataset (Dolan et al., 2004; Quirk et al., 2004) to perform the paraphrase identification task. This dataset consists of 5,801 pairs of sentences with 4,076 pairs for training and 1,725 pairs for test. The training dataset consists of 2,753 "equivalent" sentence pairs and 1,323 "not equivalent" sentence pairs, as judged by human raters. The test set consists of 1,147 and 578 "equivalent" and "not equivalent" sentence pairs, respectively.

Dolan et al. (2004) noted that, although the collected paraphrase sentences were judged "not equivalent" by the human raters, it was not desirable to use "not equivalent" sentence pairs as negative class data, because they have significant overlaps between them, in terms of information content and wording. Therefore, we used "equivalent" sentences of the MS Paraphrase Corpus as the positive class dataset and modified one side of the sentence pair to use as the non-paraphrase dataset. The non-paraphrase dataset is generated by replacing 20% of randomly selected words in a paired sentence with other words in the pre-trained word vector dictionary used in language modeling and missing word imputation tasks. For the training data, we used 2,753 pairs of sentences as the positive class and generated 2,753 pairs of negative class sentences by using the method described above. Similarly, 1,147 pairs of sentences for the test were used as the positive class,

Table 7. Result of paraphrase identification

Metric Model	Error rate (1 - Accuracy)	False alarm rate (1 - Precision)	Miss rate (1 - Recall)
RNN–AE	5.10 ± 0.56	5.09 ± 0.80	**5.59 ± 0.92**
RNN–VAE	6.05 ± 0.43	5.64 ± 0.78	6.02 ± 0.88
RNN–SVAE	**4.65 ± 0.33**	**3.68 ± 0.65**	5.86 ± 0.71

Table 8. Result of paraphrase sentence similarity

Model	RNN-VAE	RNN–SVAE
Cosine similarity	0.598	**0.631**

and 1,147 pairs of negative class sentences were generated for the test data. Thus, a total of 5,506 training pairs and 2,294 test pairs were constructed. The ratio of paraphrase pairs to non-paraphrase pairs was the same.

4.3.2. TRAINING DETAILS

The paraphrase identification models for RNN–AE, RNN–VAE, and RNN–SVAE were trained under the same conditions. The parameters of all models were initialized by using Xavier initialization. Gradient computation and weight updates were done with a mini-batch size of 512. The models were trained for 100 epochs using the Adam optimizer with a learning rate of 0.001. To prevent overfitting, dropout (Srivastava et al., 2014) is used for each layer. The dropout rate is set to 0.3. We repeated training 30 times for each model to obtain the statistical significance of the results.

4.3.3. RESULTS

We used three evaluation metrics: (1) the overall error rate, (2) false alarm rate (i.e., the proportion of incorrectly classified paraphrases as "equivalent" among the paraphrases classified as "equivalent" by the model) and (3) miss rate (i.e., the proportion of incorrectly classified paraphrases that were actually "equivalent" among the actual "equivalent" paraphrases). The results and standard deviations of paraphrase identification task of each model are summarized in Table 7. The RNN–SVAE resulted in better performance than RNN–AE and RNN—VAE in terms of error rate and false alarm rate. These performance improvements are also supported by the statistical hypothesis testing at a significant level of 0.01. Although RNN–AE showed the best performance in terms of miss rate, there is no statistically significant difference between the performance of RNN–AE and that of RNN–VAE

or RNN–SVAE at a significant level of 0.01. Compared to RNN–VAE, RNN–SVAE reduced the error rate by 23.1% ∼ 34.8%, which strongly supports the notion that the RNN–SVAE can better capture the global latent context over RNN–VAE.

In addition to paraphrase identification, which was evaluated by the binary decision, we also compared the similarity between latent vectors of two "equivalent" sentences judged by human raters. This evaluation was conducted only with RNN–VAE and RNN–SVAE to exploit the effect of adding document information vector to variational-based RNN models. Table 8 shows that not only did the RNN–SVAE model achieve higher identification accuracy, it also generated more similar latent vectors for two similar sentences than RNN–VAE.

5. Conclusion

For RNN–based autoencoder models (e.g., RNN–AE and RNN—VAE) the final hidden state of the encoder does not contain sufficient information about the entire sentence. In this paper, we proposed RNN–SVAE to overcome this limitation. To consider the information of words in the sentence, we constructed a document information vector by a linear combination of word vectors of input sentence. The weights of individual words are computed using the attention information between the final state of the encoder and every prior hidden state. We then combined this document information vector with the final hidden state of the bi-directional RNN encoder to construct the global latent vector as the output of the encoder part. Then, the mean and standard deviation of the continuous semantic space were learned to take advantage of variational method.

The proposed RNN–SVAE was verified through three NLP tasks: language modeling, missing word imputa-

tion, and paraphrase identification. Despite the simple structure of RNN–SVAE combining the document information vector with the RNN–VAE model, experimental results showed that RNN–SVAE achieved higher performance than RNN–AE and RNN–AE for all tasks requiring global latent meaning of the input sentence. The only exception is missing word imputation for a very short sentence, which does not significantly depend on the global semantic information.

Although the experimental results are very favorable for RNN–SVAE, there are some limitations of the current study. This provides some future research directions. First, the prior distribution is assumed to be a specific distribution, such as standard Gaussian. To improve the performance of RNN–SVAE, it will be worth attempting to find an appropriate prior distribution of data. Additionally, there is the risk of learning a model that is far from the actual data distribution. Thus, as in adversarial autoencoder (Makhzani et al., 2015) for image data, further research is needed to map prior distribution to data distribution in language modeling. Second, we should use the Bi-RNN structure to find the weight of a word that is not biased on one side of the sentence. To apply RNN–SVAE to one-directional RNN structures, it is necessary to study a method of re-adjusting weight properly, so that the weight of words is not biased to one side.

References

Amiriparian, Shahin, Freitag, Michael, Cummins, Nicholas, and Schuller, Björn. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *Proc. of the DCASE 2017 Workshop*, 2017.

Artetxe, Mikel, Labaka, Gorka, Agirre, Eneko, and Cho, Kyunghyun. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bahdanau, Dzmitry, Chorowski, Jan, Serdyuk, Dmitriy, Brakel, Philemon, and Bengio, Yoshua. End-to-end attention-based large vocabulary speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 4945–4949. IEEE, 2016.

Baldi, Pierre. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of ICML*

Workshop on Unsupervised and Transfer Learning, pp. 37–49, 2012.

Bengio, Yoshua, Courville, Aaron, and Vincent, Pascal. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Bowman, Samuel R, Vilnis, Luke, Vinyals, Oriol, Dai, Andrew M, Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Cettolo, Mauro, Girardi, Christian, and Federico, Marcello. Wit3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, volume 261, pp. 268, 2012.

Chan, William, Jaitly, Navdeep, Le, Quoc, and Vinyals, Oriol. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 4960–4964. IEEE, 2016.

Chen, Jinghui, Sathe, Saket, Aggarwal, Charu, and Turaga, Deepak. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pp. 90–98. SIAM, 2017.

Cho, Kyunghyun, Van Merriënboer, Bart, Bahdanau, Dzmitry, and Bengio, Yoshua. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014a.

Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014b.

Dai, Andrew M and Le, Quoc V. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pp. 3079–3087, 2015.

D'Avino, Dario, Cozzolino, Davide, Poggi, Giovanni, and Verdoliva, Luisa. Autoencoder with recurrent neural networks for video forgery detection. In *IS&T International Symposium on Electronic Imaging: Media Watermarking, Security, and Forensics*, 2017.

Deng, Jun, Zhang, Zixing, Marchi, Erik, and Schuller, Bjorn. Sparse autoencoder-based feature transfer learning for speech emotion recognition. In *Affective*

Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on, pp. 511–516. IEEE, 2013.

Dolan, Bill, Quirk, Chris, and Brockett, Chris. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of the 20th international conference on Computational Linguistics*, pp. 350. Association for Computational Linguistics, 2004.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, 2010.

Graves, Alex and Jaitly, Navdeep. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1764–1772, 2014.

Ha, Thanh-Le, Niehues, Jan, and Waibel, Alexander. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*, 2016.

Hermann, Karl Moritz, Kocisky, Tomas, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hu, Baotian, Lu, Zhengdong, Li, Hang, and Chen, Qingcai. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pp. 2042–2050, 2014.

Huang, Zhiying, Tang, Jian, Xue, Shaofei, and Dai, Lirong. Speaker adaptation of rnn-blstm for speech recognition based on speaker code. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pp. 5305–5309. IEEE, 2016.

Karpathy, Andrej and Fei-Fei, Li. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3128–3137, 2015.

Karpathy, Andrej, Joulin, Armand, and Fei-Fei, Li F. Deep fragment embeddings for bidirectional image sentence mapping. In *Advances in neural information processing systems*, pp. 1889–1897, 2014.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Le, Quoc and Mikolov, Tomas. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196, 2014.

Lee, Jason, Cho, Kyunghyun, and Hofmann, Thomas. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*, 2016.

Li, Jiwei, Luong, Minh-Thang, and Jurafsky, Dan. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057*, 2015.

Ling, Wang, Trancoso, Isabel, Dyer, Chris, and Black, Alan W. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Lyudchik, Olga. Outlier detection using autoencoders. Technical report, 2016.

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, Goodfellow, Ian, and Frey, Brendan. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Mani, Arathi. Solving text imputation using recurrent neural networks. 2015.

Marchi, Erik, Vesperini, Fabio, Squartini, Stefano, and Schuller, Björn. Deep recurrent neural network-based autoencoders for acoustic novelty detection. *Computational intelligence and neuroscience*, 2017, 2017.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Nallapati, Ramesh, Xiang, Bing, and Zhou, Bowen. Sequence-to-sequence rnns for text summarization. 2016.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and Zhu, Wei-Jing. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Quirk, Chris, Brockett, Chris, and Dolan, Bill. Monolingual machine translation for paraphrase generation. 2004.

Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Rus, Vasile, McCarthy, Philip M, Lintean, Mihai C, McNamara, Danielle S, and Graesser, Arthur C. Paraphrase identification with lexico-syntactic graph subsumption. In *FLAIRS conference*, pp. 201–206, 2008.

Sakurada, Mayu and Yairi, Takehisa. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pp. 4. ACM, 2014.

Schuster, Mike and Paliwal, Kuldip K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

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

Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Vinyals, Oriol, Kaiser, Łukasz, Koo, Terry, Petrov, Slav, Sutskever, Ilya, and Hinton, Geoffrey. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, pp. 2773–2781, 2015.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhudinov, Ruslan, Zemel, Rich, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pp. 2048–2057, 2015.

Xu, Weidi, Sun, Haoze, Deng, Chao, and Tan, Ying. Variational autoencoder for semi-supervised text classification. In *AAAI*, pp. 3358–3364, 2017.

Yuan, Xingdi, Wang, Tong, Gulcehre, Caglar, Sordoni, Alessandro, Bachman, Philip, Subramanian, Sandeep, Zhang, Saizheng, and Trischler, Adam. Machine comprehension by text-to-text neural question generation. *arXiv preprint arXiv:1705.02012*, 2017.

Zhang, Biao, Xiong, Deyi, Su, Jinsong, Duan, Hong, and Zhang, Min. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*, 2016.

Zhao, Shenjian and Zhang, Zhihua. Deep character-level neural machine translation by learning morphology. 2016.

Zhu, Zhuotun, Wang, Xinggang, Bai, Song, Yao, Cong, and Bai, Xiang. Deep learning representation using autoencoder for 3d shape retrieval. *Neurocomputing*, 204:41–50, 2016.

Zhuang, Fuzhen, Cheng, Xiaohu, Luo, Ping, Pan, Sinno Jialin, and He, Qing. Supervised representation learning: Transfer learning with deep autoencoders. In *IJCAI*, pp. 4119–4125, 2015.