Neurocomputing 000 (2018) 1–10

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Lattice-to-sequence attentional Neural Machine Translation models

Zhixing Tan[a], Jinsong Su[b], Boli Wang[a], Yidong Chen[a], Xiaodong Shi[a,*]

[a] School of Information Science and Engineering, Xiamen University, China
[b] Software School, Xiamen University, China

A R T I C L E I N F O

Article history:
Received 27 April 2017
Revised 15 December 2017
Accepted 1 January 2018
Available online xxx

Communicated by Dr. Tie-Yan Liu

Keywords:
Neural Machine Translation
Word lattice
Recurrent Neural Network
Gated Recurrent Unit

A B S T R A C T

The dominant Neural Machine Translation (NMT) models usually resort to word-level modeling to embed input sentences into semantic space. However, it may not be optimal for the encoder modeling of NMT, especially for languages where tokenizations are usually ambiguous: On one hand, there may be tokenization errors which may negatively affect the encoder modeling of NMT. On the other hand, the optimal tokenization granularity is unclear for NMT. In this paper, we propose lattice-to-sequence attentional NMT models, which generalize the standard Recurrent Neural Network (RNN) encoders to lattice topology. Specifically, they take as input a word lattice which compactly encodes many tokenization alternatives, and learn to generate the hidden state for the current step from multiple inputs and hidden states in previous steps. Compared with the standard RNN encoder, the proposed encoders not only alleviate the negative impact of tokenization errors but are more expressive and flexible as well for encoding the meaning of input sentences. Experimental results on both Chinese–English and Japanese–English translations demonstrate the effectiveness of our models.

1. Introduction

Recently, NMT [1–4] has achieved great success and attracted much attention in the field of natural language processing (NLP). Different from the statistical machine translation approach that explicitly models latent structures such as word alignment, phrase segmentation and phrase reordering, NMT aims at training a unified encoder-decoder neural network [2,3] that directly maps a source-language sentence to a target-language sentence, where the encoder embeds the input sentence into a sequence of vectors, and then the decoder with attention mechanism produces a translation from the encoded vectors.

Although recent attention has been paid to character-level NMT [5–7] which learn sentence representations and perform generations at the character level, the dominant NMT systems make use of word boundary information to learn the semantic representations of the source sentence and generate the target sentence word by word, typically employing RNNs in both the encoder and the decoder. The reason is that words can encode semantic information more naturally than characters. The word-based models obtain good results for source languages like English, but it does not work equally well for languages without natural word delimiters such as Chinese. The reasons are two-fold. Firstly, the optimal tokenization granularity is not easy to determine for NMT because coarse granularity causes data sparseness while fine granularity results in the loss of useful information. Secondly, the 1-best tokenization may introduce errors that propagate to the later stage and thus negatively impact the learned source sentence representations. Therefore, we believe that it is necessary to learn from more tokenization alternatives for NMT systems to alleviate the sub-optimal tokenization granularity and tokenization error propagation problems.

In this paper, we propose lattice-to-sequence attentional NMT (L2SNMT) models to deal with the above tokenization issues. Word lattice, which compactly represents many tokenization alternatives, has been widely used in various NLP tasks [8–10]. In this paper, we present two lattice-based RNN encoders to simultaneously exploit multiple tokenizations for input sentence modeling. One is the *Lattice-based RNN Encoder with Pre-composition* which first combines the inputs and hidden states vectors derived from multiple tokenizations, and then feeds these vectors into standard RNN. The other is the *Lattice-based RNN Encoder with Post-composition* which learns and updates tokenization-specific vectors for gates, inputs and hidden states, and then generates hidden state vectors for current units. By extending the standard RNN encoder into lattice-based ones, we expect the proposed L2SNMT models not only reduce the negative impact of tokenization errors but also enhance the expressive power and flexibility of encoder in embedding source sentences.

* Corresponding author.
E-mail address: mandel@xmu.edu.cn (X. Shi).

https://doi.org/10.1016/j.neucom.2018.01.010
0925-2312/© 2018 Elsevier B.V. All rights reserved.

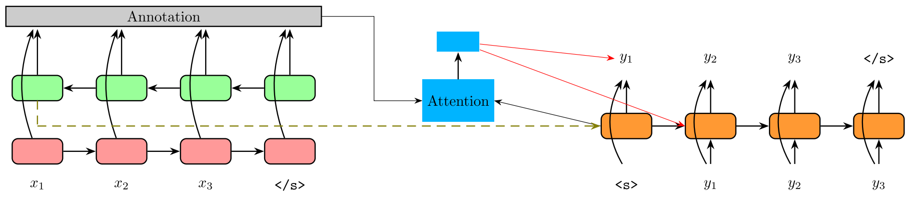


Fig. 1. Overview of attention-based NMT. It has two components: an encoder network and a decoder network with attention mechanism.

We conduct experiments on Chinese–English and Japanese–English translation tasks to investigate the effectiveness of the proposed L2SNMT models. From the experimental results, we conclude that: (1) word boundary information is helpful for NMT to accurately learn the semantics of input Chinese and Japanese sentences. (2) lattice-based RNN encoders are more effective than the standard RNN encoder in NMT. To the best of our knowledge, this is the first attempt to establish NMT encoder based on lattices.

The L2SNMT has been presented in our previous paper [11]. In this article, we make the following significant extensions to our previous work.

1. We integrate lattice posterior score into our proposed NMT models. We also propose a new variant of lattice-based RNN encoder which directly exploits weights associated with lattice input and compare it with other variants.
2. We show that our lattice-based RNN encoders can be further improved with deep RNNs. We also explore the effect of encoder depth that affects the model performance.
3. We investigate different ways to form source annotations from the output of lattice-based RNN.
4. We carry out new experiments on large-scale data to study the robustness of our model on different sizes of training data.
5. We conduct more experiments on Japanese–English translation to investigate the effectiveness of our model on different language pairs.

The remainder of this article is organized as follows: Section 2 briefly describes the conventional sequence-to-sequence attentional NMT, which is the basis of our work. Section 3 gives details of the proposed L2SNMT models. Section 4 reports the experimental results on Chinese–English and Japanese–English translation tasks and studies how the learned encoders improve translation quality. Section 5 summarizes the related work and highlights the differences of our model from previous studies. Finally, we conclude in Section 6 with future directions.

2. Neural Machine Translation

As depicted in Fig. 1, the dominant NMT model is an attention-based NMT [4], which consists of an encoder network and a decoder network with attention mechanism.

Generally, the encoder is modeled as a bidirectional RNN. Given a source sentence $\boldsymbol{x} = x_1, x_2, \ldots, x_S$, the forward RNN reads $\boldsymbol{x}$ word by word in a left-to-right way and uses the recurrent activation function f to learn semantic representation of word sequence $x_{1:i}$ as $\overrightarrow{\mathbf{h}}_i = f(\mathbf{x}_i, \overrightarrow{\mathbf{h}}_{i-1})$, where $\mathbf{x}_i$ is the word embedding of x_i. In a similar way, the backward RNN reversely scans the source sentence and learns the semantic representation $\overleftarrow{\mathbf{h}}_i$ of the word sequence $x_{i:S}$. Then, the hidden states learned by the two RNNs are concatenated to form the *source annotation* $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i^T, \overleftarrow{\mathbf{h}}_i^T]^T$, which encodes all the surrounding words of the i-th word.

The decoder is a forward RNN which generates the translation $\boldsymbol{y}$ using a nonlinear function $g(\cdot)$:

$$p(y_t|y_{<t}, \boldsymbol{x}) = g(\mathbf{y}_{t-1}, \mathbf{s}_t, \mathbf{c}_t), \quad (1)$$

where $\mathbf{y}_{t-1}$ denotes the word embedding of word y_{t-1}, $\mathbf{s}_t$ and $\mathbf{c}_t$ are the decoding state and the context vector at the time step t, respectively. Formally, $\mathbf{s}_i$ can be computed using a function $f(\cdot)$, such as Long Short-Term Memory (LSTM [12]) or Gated Recurrent Unit (GRU [2]), as follows:

$$\mathbf{s}_t = f(\mathbf{y}_{t-1}, \mathbf{c}_{t-1}, \mathbf{s}_{t-1}), \quad (2)$$

Furthermore, to accurately capture context, the attention mechanism learns the context vector $\mathbf{c}_t$ as the weighted sum of the source annotations $\{\mathbf{h}_i\}$:

$$\mathbf{c}_t = \sum_{i=1}^{S} \alpha_{t,i} \cdot \mathbf{h}_i, \quad (3)$$

where $\alpha_{t,\,i}$ measures how well $\mathbf{s}_t$ and $\mathbf{h}_i$ matches as below:

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i'=1}^{S} \exp(e_{t,i'})}, \quad (4)$$

$$e_{t,i} = \mathbf{v}_a^T \tanh(\mathbf{W}_a \mathbf{s}_t + \mathbf{U}_a \mathbf{h}_i). \quad (5)$$

where $\mathbf{W}_a$, $\mathbf{U}_a$ and $\mathbf{v}_a$ are the weight matrices of attention model. In this way, the relevant source words can be automatically selected to predict target words.

3. The proposed models

In this section, we give a detailed description to the L2SNMT models. We first describe how to build the word lattice of each input sentence. Then we briefly review the standard GRU, which is chosen as the basic unit of our encoder. Next, we describe lattice-based RNN encoders in detail. Finally, we describe the decoder network of our L2SNMT.

3.1. Word lattice

As shown in Fig. 2, a word lattice can be represented as a directed graph $G = \langle V, E \rangle$, where V is the node set and E is the edge set. Given the word lattice of a character sequence $c_{1:N} = c_1, \ldots, c_N$, the node $v_i \in V$ denotes the position between c_i and c_{i+1} and the edge $e_{i:j} \in E$ departs from v_i and arrives at v_j from left to right, covering the subsequence $c_{i+1:j}$ that is recognized as a possible word.

Intuitively, different segmentation results should have different effects when combining inputs and hidden states derived from multiple tokenizations. To do this, we apply *forward–backward algorithm* [13] to calculate the posterior scores of edges in the lattice. Given an edge $e_{i:j}$ in the word lattice, we denote the number of paths from v_0 to v_i with α_i, which is iteratively computed as $\alpha_i = \sum_{k:e_{k,i} \in E} \alpha_k$ and $\alpha_1 = 1$. Similarly, the number of paths from v_N to v_j is denoted as β_j, which is also computed as $\beta_j = \sum_{k:e_{j,k} \in E} \beta_k$ and $\beta_N = 1$.

Finally, we calculate the weight of $e_{i:j}$ as

$$w_{i,j} = \frac{\alpha_i \beta_j}{\sum_{k:e_{k,j} \in E} \alpha_k \beta_j} \quad (6)$$

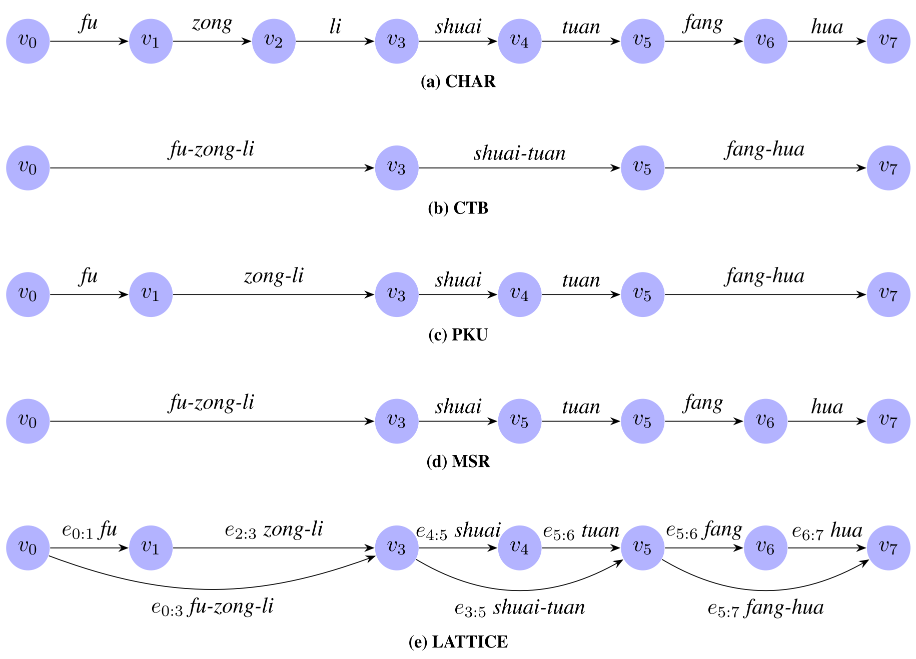


Fig. 2. Three kinds of word segmentations and the resulting word lattice of the Chinese character sequence "*fu zong li shuai tuan fang hua*". We insert "-" between characters in a word just for clarity.

In this work, we train many word segmenters using multiple corpora with different annotation standards, such as *the Penn Chinese Treebank 6.0* (*CTB*), *the Peking University Corpus* (*PKU*) and *the Microsoft Research Corpus* (*MSR*). For each input sentence, we tokenize it using these different segmenters and generate a word lattice by merging the predicted tokenizations that are same in different segmenters. Fig. 2 provides an example. Using *CTB, PKU, MSR* segmenters, we segment the Chinese sentence "*fu zong li shuai tuan fang hua*" into different token sequences, all of which are combined to produce the final word lattice. Specifically, both *CTB* and *MSR* segmenters produce the same tokenization "*fu-zong-li*", which is merged into the edge $e_{0:3}$ in the lattice.

3.2. Gated Recurrent Unit

3.2.1. Recurrent Neural Network

Recurrent Neural Networks are powerful models that have been widely used in various NLP tasks. Formally speaking, RNN is a function that takes two arguments: the current input $\mathbf{x}_t$ and its previous state $\mathbf{h}_{t-1}$, which can be described by the following equation:

$$\mathbf{h}_t = \mathrm{RNN}(\mathbf{x}_t, \mathbf{h}_{t-1}) \quad (7)$$

There are many instances of RNN and the most popular ones are LSTM and GRU. Compared with LSTM, GRU not only alleviates the *exploding or vanishing gradient problem* of the conventional RNN, but also is easier to compute and implement. Therefore, in this work, we mainly focused on GRU to implement the lattice-based encoders. However, other variants of RNN unit is also applicable to our encoders.

3.2.2. The standard GRU

GRU is a variant of the standard RNN unit. Fig. 3(a) provides the graphical illustration of GRU. At time step *t*, GRU has two gates: a *reset gate* $\mathbf{r}_t$ that determines how to combine the new input $\mathbf{x}_t$ with the previous memory $\mathbf{h}_{t-1}$, the other is an *update gate* $\mathbf{z}_t$ which defines how much of the previous memory to keep around. Formally, the GRU transition equations are as follows:

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1}), \quad (8)$$

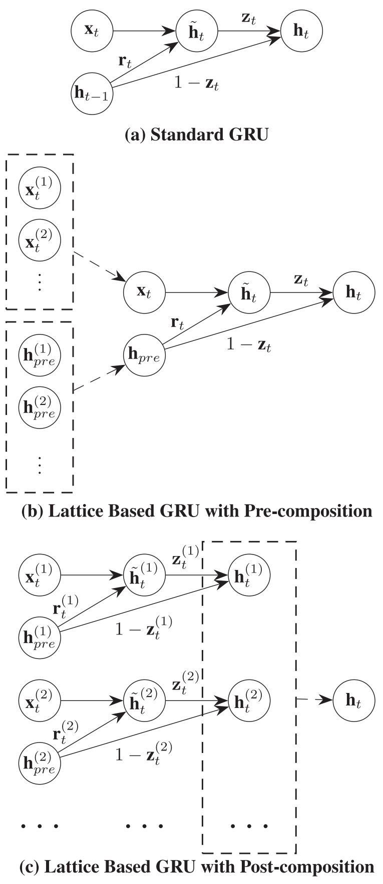

(a) Standard GRU
(b) Lattice Based GRU with Pre-composition
(c) Lattice Based GRU with Post-composition

Fig. 3. The architectures of the standard GRU and lattice-based GRUs. Note that in our GRUs, the preceding hidden state is not always the one at the time step $t-1$. For notational clarity, we use the subscript "*pre*" rather than $t-1$ to index the preceding hidden state. The dashed box represents a composition function.

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1}), \tag{9}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c x_t + \mathbf{U}_c(\mathbf{r}_t \odot \mathbf{h}_{t-1})), \tag{10}$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t. \tag{11}$$

where σ denotes the logistic sigmoid function, $\odot$ is the element-wise multiplication, and $\mathbf{W}_*$ and $\mathbf{U}_*$ are the weight matrices, respectively.

3.2.3. Deep stacked GRU

Recent studies [14–17] proved that deep stacked RNNs can further improve the performance of NMT. Inspired by these works, we investigate the use of *Deep Stacked GRU* in our lattice-based RNN encoders. By using deep stacked GRU, we expect our encoders can learn more expressive and flexible representations.

Specifically, we use stacked GRUs with residual connections [18] between layers, which naturally extends the single layered GRU to multi-layered one. Given the current input $\mathbf{x}_t$ and previous states $\mathbf{h}_{t-1} = (\mathbf{h}^1_{t-1}, \dots, \mathbf{h}^L_{t-1})$, where L is the total number of layers, the computation of the ℓ-th layer is described by the following equation:

$$\mathbf{h}^\ell_t = \mathrm{GRU}^\ell(\mathbf{x}^{\ell-1}_t, \mathbf{h}^\ell_{t-1}) \tag{12}$$

We use GRU^ℓ to represent the computation of Eqs. (8)–(11) involved in the ℓ-th layer. And $\mathbf{x}^{\ell-1}_t$ is the input from the below layer, which has the following definition:

$$\mathbf{x}^\ell_t = \begin{cases} \mathbf{x}_t & \ell = 0 \\ f(\mathbf{h}^\ell, \mathbf{x}^{\ell-1}) & \ell \geq 1 \end{cases} \tag{13}$$

where f is the residual function. If $\mathbf{h}^\ell$ and $\mathbf{x}^{\ell-1}$ have the same dimension, then we have $f(\mathbf{h}^\ell, \mathbf{x}^{\ell-1}) = \mathbf{h}^\ell + \mathbf{x}^{\ell-1}$, otherwise an additional linear projection is employed to match the dimensions and f becomes $\mathbf{h}^\ell + \mathbf{W}\mathbf{x}^{\ell-1}$. We take $\mathbf{h}^L$ as the final output of deep stacked GRU. Note that when $L = 1$, the definition is equivalent to the standard GRU.

3.3. Lattice-based RNN encoders

To deal with the tokenization problems mentioned above, we propose lattice-based RNN encoders for NMT. Similar to the popular NMT model [4], our encoders are bidirectional lattice RNNs. Here we only introduce the forward RNN for clarity, the backward RNN can be extended in a similar way.

Our encoders scan a source sentence character by character. Hidden states are generated from multiple input candidate words and the corresponding preceding hidden states only at potential word boundaries. Specifically, at time step t, we first identify all edges linking to v_t, each of which covers different input words with different preceding hidden states. In particular, for the kth edge,[1] we denote its input word vector and the corresponding preceding hidden state as $\mathbf{x}^{(k)}_t$ and $\mathbf{h}^{(k)}_{pre}$, respectively. As shown in Fig. 2, the word lattice contains two edges $e_{0:3}$ and $e_{1:3}$, both pointing to v_3. Therefore, at the 3rd time step, there exist two input words "*fu-zong-li*" and "*zong-li*" with different preceding hidden states.

Obviously, the network topology of word lattice is richer than that of word sequence. The same character subsequence may contain many tokenization results, therefore, there may exist many inputs and preceding hidden states at each time step, which is beyond the modeling ability of the standard RNN. We then present two lattice-based RNNs to exploit multiple tokenizations simultaneously:

3.3.1. Lattice-based RNN with pre-composition (LRNN-PRE)

Fig. 3(b) describes the architecture of LRNN-PRE with GRU as the recurrent unit, which we denoted as LGRU-PRE.[2] Assume there is potentially a word boundary at time step t, all possible word embeddings $\{\mathbf{x}^*_t\}$ are combined into a compressed vector $\mathbf{x}_t$. In a similar way, the hidden state vectors $\{\mathbf{h}^*_{pre}\}$ of preceding time steps are also compressed into $\mathbf{h}_{pre}$. Once finished, both $\mathbf{x}_t$ and $\mathbf{h}_{pre}$ can be feed into standard GRU to generate the final hidden state vector $\mathbf{h}_t$. Formally, all the operations described above are defined by the following equations:

$$\mathbf{x}_t = g(\mathbf{x}^{(1)}_t, \mathbf{x}^{(2)}_t, \dots, \mathbf{x}^{(K)}_t), \tag{14}$$

$$\mathbf{h}_{pre} = g(\mathbf{h}^{(1)}_{pre}, \mathbf{h}^{(2)}_{pre}, \dots, \mathbf{h}^{(K)}_{pre}), \tag{15}$$

$$\mathbf{r}_t = \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{pre}), \tag{16}$$

$$\mathbf{z}_t = \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{pre}), \tag{17}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c(\mathbf{r}_t \odot \mathbf{h}_{pre})), \tag{18}$$

$$\mathbf{h}_t = (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t, \tag{19}$$

where Eqs. (14) and (15) are used to produce the compressed representations $\mathbf{x}_t$ and $\mathbf{h}_{pre}$, the others are identical to those of the standard GRU, which can be replaced with the computation of deep stacked GRU. $g(*)$ is a composition function, for which we will investigate three definitions later on.

3.3.2. Lattice-based RNN with post-composition (LRNN-POST)

The architecture of LGRU-POST is shown in Fig. 3(c). In this unit, the reset gate $\mathbf{r}^{(k)}_t$, the update gate $\mathbf{z}^{(k)}_t$ and the hidden state vector $\mathbf{h}^{(k)}_t$ are set and updated with respect to the kth edge ending with character c_t specifically, and a composed hidden state vector $\mathbf{h}_t$ is generated from $\{\mathbf{h}^{(*)}_t\}$. LGRU-POST merges the hidden states specific to different tokenizations, whereas LGRU-PRE merges the inputs and the preceding hidden states before the computation of GRU. Formally, the transition equations are defined as follows:

$$\mathbf{r}^{(k)}_t = \sigma(\mathbf{W}_r\mathbf{x}^{(k)}_t + \mathbf{U}_r\mathbf{h}^{(k)}_{pre}), \tag{20}$$

$$\mathbf{z}^{(k)}_t = \sigma(\mathbf{W}_z\mathbf{x}^{(k)}_t + \mathbf{U}_z\mathbf{h}^{(k)}_{pre}), \tag{21}$$

$$\tilde{\mathbf{h}}^{(k)}_t = \tanh(\mathbf{W}_c\mathbf{x}^{(k)}_t + \mathbf{U}_c(\mathbf{r}_{tk} \odot \mathbf{h}^{(k)}_{pre})), \tag{22}$$

$$\mathbf{h}^{(k)}_t = (\mathbf{1} - \mathbf{z}^{(k)}_t) \odot \mathbf{h}^{(k)}_{pre} + \mathbf{z}^{(k)}_t \odot \tilde{\mathbf{h}}^{(k)}_t, \tag{23}$$

$$\mathbf{h}_t = g(\mathbf{h}^{(1)}_t, \mathbf{h}^{(2)}_t, \dots, \mathbf{h}^{(K)}_t). \tag{24}$$

where Eqs. (20)–(23) calculate the gating vectors and the hidden state vectors with respect to different tokenizations and also can be replaced with the corresponding equations of deep stacked GRU. Eq. (24) is a composition function that is used to produce the final hidden state vector at the time step t.

[1] We index the edges from left to right according to the positions of their starting nodes.

[2] We rename Shallow Word-Lattice based GRU and Deep Word-Lattice based GRU in our previous work to Lattice-based GRU with Pre-composition and Lattice-based GRU with Post-composition, respectively.

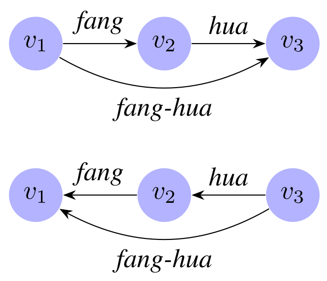


Fig. 4. An example of forward lattice and backward lattice.

3.3.3. Composition functions

For the composition function $g(*)$ involved in the two lattice-based GRUs, we explore the following three functions, of which computation procedures are illustrated by taking $\mathbf{h}_t$ as an example.

1. *Weighting function* which uses the posterior score associated with lattice edges. We first normalize the posterior score $s_t^{(k)}$ of the k-th edge as follows:

$$w_t^{(k)} = \frac{s_t^{(k)}}{\sum_{j=1}^{K} s_t^{(j)}} \tag{25}$$

where K is the total number of identified edges in time step t. Then we define $\mathbf{h}_t$ as the weighted sum of preceding hidden state vectors:

$$\mathbf{h}_t = \sum_{k=1}^{K} w_t^{(k)} \mathbf{h}_t^{(k)} \tag{26}$$

If $\mathbf{h}_t$ is a sequence of states as in the case of deep stacked GRU, then we employ multiple independent composition functions with $\mathbf{h}_t$ in Eq. (26) replaced with $\mathbf{h}_t^{\ell}$.

2. *Pooling function* which enables our encoders to automatically capture the most relevant part for the source sentence modeling. Using this function, we define $\mathbf{h}_t$ as follows:

$$\mathbf{h}_t[i] = \max(\mathbf{h}_t^{(1)}[i], \ldots, \mathbf{h}_t^{(K)}[i]) \tag{27}$$

where $\mathbf{h}[i]$ denotes the i-th element of vector $\mathbf{h}$. The extension to deep stacked GRU is similar to the weighting function.

3. *Gating function* which first automatically learns the weights of components in lattice-based GRUs and then produces $\mathbf{h}_t$ in a similar way to weighting function. For LGRU-PRE and LGRU-POST, w_k in Eq. (26) is calculated as follows:

$$w_k = \frac{\exp(\mathbf{W}_g \mathbf{h}_{pre}^{(k)} + \mathbf{b}_g)}{\sum_{j=1}^{K} \exp(\mathbf{W}_g \mathbf{h}_{pre}^{(j)} + \mathbf{b}_g)} \tag{28}$$

where $\mathbf{W}_g$ and $\mathbf{b}_g$ are the weight matrix and the bias term, respectively. For deep stacked GRU, we concatenate all states in $\mathbf{h}_t$ together to calculate the weight w_k.

3.4. Decoder network

In the specific implementation of encoders, we directly reverse the edge directions of the forward lattice to form the backward lattice, as illustrated in Fig. 4. How- ever, the computations of forward lattice GRU and backward word-lattice GRU are not symmetric. For example, edge "*fang-hua*" is merged with "*hua*" in forward lattice RNN, while it is actually merged with "*fang*" in backward lattice RNN. Due to the asymmetric structure of forward and backward lattice, the same word embedding may merge to different positions in the resulting forward and backward states. Given a sequence of forward outputs $(\overrightarrow{\mathbf{h}}_1, \ldots, \overrightarrow{\mathbf{h}}_S)$ and backward outputs $(\overleftarrow{\mathbf{h}}_1, \ldots, \overleftarrow{\mathbf{h}}_S)$ by the encoder, the decoder explores two ways to obtain source annotations:

1. *Concatenation*: The source annotation $\mathbf{h}_i$ is obtained by concatenation of forward output vector $\overrightarrow{\mathbf{h}}_i$ and backward output vector $\overleftarrow{\mathbf{h}}_i$. This approach is the same as sequence models and was used in our previous work.
2. *Replication*: To address the asymmetric structure of forward and backward lattice, hidden states can be first arranged according to edges and then concatenated together. Formally, given an edge $e_{i:j} \in E$ and k is the unique index representing edge $e_{i:j}$, let $\phi_f[k]$ denotes the time step where edge $e_{i:j}$ is merged into in the forward lattice-based RNN. $\phi_b[k]$ is defined similarly for backward lattice-based RNN. The source annotation $\mathbf{h}_i$ are defined as $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_{\phi_f[i]}^T, \overleftarrow{\mathbf{h}}_{\phi_b[i]}^T]^T$. In this way, a hidden vector $\overrightarrow{\mathbf{h}}_i$ or $\overleftarrow{\mathbf{h}}_i$ may replicate multiple times.

Besides using different ways to form source annotations, the remaining parts of decoder network of our L2SNMT share the same architecture of conventional attention-based NMT. The decoder uses attention mechanism to produce a context vector as calculated by Eqs. (3)–(5). We use GRU as the activation function f in (2). The g function in Eq. (1) is a deep neural network with a single maxout [19] hidden layer, as described below:

$$\mathbf{t} = \text{maxout}(\mathbf{s}_{t-1}, \mathbf{y}_{t-1}, \mathbf{c}_t) \tag{29}$$

And the probability distribution is calculated using softmax function:

$$p(y_t | y_{<t}, \boldsymbol{x}) \propto \exp(\mathbf{y}^T \mathbf{W}_o \mathbf{t}) \tag{30}$$

4. Experiments

4.1. Setup

We evaluated our proposed models on two translation tasks: Chinese–English and Japanese–English. The evaluation metric is BLEU [20] for both task as calculated by the `multi-bleu.perl` script.[3] Moreover, we conducted paired bootstrap sampling [21] to test the significance in BLEU score differences.

For Chinese–English, our training data contains 2.5M sentence pairs with 117.8M Chinese characters and 80.8M English words, respectively. We chose the NIST 2005 dataset as the validation set and the NIST 2002, 2003, 2004, 2006, and 2008 datasets as test sets. To obtain lattices of Chinese sentences, we used the toolkit[4] released by Stanford to train word segmenters on *CTB, PKU*, and *MSR* corpora.

For Japanese–English translation, we used the top 1.5M sentence pairs from ASPEC corpus [22] as our training data, with 75.6M Japanese characters and 39.0M English words, respectively. We tuned the model parameters and then tested models using the validation and test set provided by Workshop of Asian Translation.[5] We applied three word segmenters: *Juman, Kytea* [23] and *Mecab* [24] to build lattices of Japanese sentences.

We refer to the lattice-to-sequence attentional NMT system as L2SNMT, which has six variants with different network units and composition operations. We compared them against the following state-of-the-art SMT and NMT systems:

- Moses[6]: an open source phrase-based translation system. We also compare its variant which takes the lattice of source sentence as input.

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl.
[4] http://nlp.stanford.edu/software/segmenter.html#Download.
[5] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/automatic_evaluation_systems/tools.html.
[6] http://www.statmt.org/moses/.

Table 1
Comparison of different ways to form source annotations on Chinese–English translation.

System	Unit	Annotation	MT05	MT02	MT03	MT04	MT06	MT08	ALL
L2SNMT	LGRU-PRE + Gating	Concatenation	38.28	39.95	39.15	40.97	38.44	29,86	37.83
	LGRU-PRE + Gating	Replication	38.04	39.83	38.54	40.58	37.48	29.30	37.29
	LGRU-POST + Gating	Concatenation	38.36	40.47	39.05	40.91	38.22	30.30	37.95
	LGRU-POST + Gating	Replication	38.35	40.00	38.91	40.09	37.85	30.33	37.51

- RNNSEARCH [4]: a state-of-the-art attention-based NMT. In addition to the RNNSEARCH with word-based segmentations, we also compared to that with character-based segmentations to study whether word boundary information is helpful for NMT.
- MSNMT (Multi-source NMT): an attention-based NMT system which also uses many tokenizations of each source sentence as input. Significantly different from our L2SNMT models, it first encodes different tokenizations separately and uses attention mechanism to combine different representation of tokenizations. As a result, it only has two variants with *gating* and *pooling* composition functions.

For MOSES, we used all the training corpus to train the phrase-based translation model and the target-side part of parallel corpus to train a 4-gram language model using the SRILM toolkit [25]. We used the default configuration for both training and decoding.

For RNNSEARCH, MSNMT and L2SNMT, we selected the most frequent 50K words in source and target language as our vocabularies. For Chinese–English translation, such vocabularies cover 98.5%, 98.6%, 99.3% and 97.3% Chinese words in *CTB, PKU, MSR* and lattice corpora, and 99.7% English words, respectively. For Japanese–English translation, such vocabularies cover 99.1%, 98.8%, 99.0% and 98.3% Japanese words in *Juman, Kytea, Mecab* and lattice corpora, and 97.0% English words, respectively.

In addition, all the out-of-vocabulary words are mapped into a special token UNK. We only kept the sentence pairs that are not longer than 70 source characters and 50 target words. In this way, about 89.9% of the Chinese–English parallel sentences and 82.83% of the Japanese–English parallel sentences are covered. We initialized model parameters uniformly between $[-0.08, 0.08]$. We applied Adam [26] ($\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 1 \times 10^{-8}$) to train models until there is no BLEU improvement on the validation set. The learning rate is initialized to 5×10^{-4} and is reduced by half at every epoch. During this procedure, we set the following hyperparameters: word embedding dimension as 320, hidden layer size as 512, batch size as 80, gradient norm as 5.0. Unless otherwise noted, the encoder depth of L2SNMT is set to 1. All the other settings are the same as in [4].

4.2. Parameters and speed

The word- and character-based RNNSEARCH systems have 55.5M and 44.2M parameters, respectively. Only NMT systems with gating operation, introduce no more than 2.7K parameters over those parameters of RNNSEARCH.

We used a single GPU device TitanX to train all NMT models. It takes one hour to train 9000 and 4200 mini-batches for word- and character-based RNNSEARCH systems, respectively. The training speeds of MSNMT and L2SNMT systems are slower than that of word-based RNNSEARCH: about 4800–6000 mini-batches are processed in one hour.

4.3. Comparison of source annotation computation

We first compare the two ways to compute source annotations described in Section 3 on the test set for Chinese–English translation. As shown in Table 1, the concatenation method generally performs better than replication method. We conjecture that replicating hidden states makes the model hard to train due to the complex interactions between encoder and decoder. As a result, in the following experiments, we use the concatenation method to form source annotations.

4.4. Results on Chinese–English translation

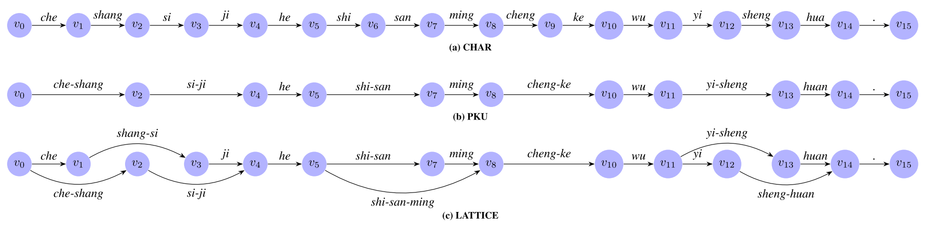
Table 2 reports the experimental results. Similar to the experimental results on small-scale training data that have been reported in our previous work [11],[7] L2SNMT achieves significant improvements over non-lattice NMT systems in all cases. When using LGRU-POST with the *gating* composition function, the L2SNMT system significantly outperforms MOSES, RNNSEARCH and MSNMT by at least gains of 6.18, 1.11, and 0.80 BLEU points, respectively. Besides, we further have the following findings:

1. No matter which word-segmented corpus we used, RNNSEARCH outperforms the character-based NMT system. These results demonstrate that word boundary information is very useful for the embedding of source sentence, especially for language which is not morphologically rich, such as Chinese. This is because the word-level encoding has two advantages over the character-level encoding: First, for the same sentence, its length significantly grows when it is represented as a sequence of characters rather than words, and thus it becomes more difficult to encode long sequences by the RNN-based NMT [27]. Second, the word-level interactions are totally ignored by the character-level encoder when learning the semantics of entire sentences.
2. Multiple inputs with different word segmentation standards are useful for the modeling of NMT encoder. In particular, multiple inputs are more effective for NMT when introducing word lattices. For this, we speculate that rather than only at final time step, all hidden states of lattice-based encoders at all potential word boundaries are influenced by many different tokenizations. To this end, different tokenizations are able to play complementary roles in source sentence representations.
3. No matter which composition function is used, LGRU-POST is slightly more effective than LGRU-PRE. These results accord with our intuition since LGRU-POST exploits multiple tokenizations at a more fine-grained level than LGRU-PRE.
4. Among the three composition functions, *gating* is the most effective one. The gating composition function performs slightly better than weighting function, although the latter directly uses lattice posterior score. Note that our input lattice is obtained by combining three segmentations and all weights of incoming edges are summed to one. As a result, the entropy of weight distribution derived by posterior lattice score is relatively high. However, the weights in gating function are learned by neural networks and thus do not have this limitation.

[7] Note that we adopt Adam instead of RMSprop as the optimizer in this paper, which improves the NMT performance.

Table 2
Case-insensitive BLEU score on Chinese–English translation. †, ‡ and ↑ indicate statistically significantly better than ($p < 0.01$) the best results of Moses, RNNsearch and MSNMT system, respectively. We highlight the highest BLEU score in bold for each set.

System	Unit	Input	MT05	MT02	MT03	MT04	MT06	MT08	ALL
Moses		*CTB*	30.23	31.65	31.43	33.60	32.70	25.76	31.69
	–	*PKU*	30.38	31.61	31.41	33.35	31.95	25.20	31.40
		MSR	29.74	31.58	30.77	33.20	31.65	25.38	31.08
		Lattice	32.47	33.59	33.06	34.05	32.03	25.25	31.76
RNNsearch	GRU	*Char*	34.20	35.86	34.84	37.17	34.08	25.99	33.35
		CTB	36.92	38.85	37.09	39.50	36.31	28.38	36.18
		PKU	37.08	39.73	37.83	40.35	36.95	29.05	36.84
		MSR	36.67	39.58	36.82	39.45	35.82	29.46	36.32
MSNMT	GRU + Pooling	*CTB+PKU+MSR*	35.82	38.30	36.69	38.93	36.06	27.35	35.63
	GRU + Gating		37.60	39.49	38.75	40.20	37.32	29.72	37.15
L2SNMT	LGRU-PRE + Weighting	*Lattice*	38.25	39.94	38.60	40.48	37.68	30.24	37.53†‡
	LGRU-PRE + Pooling		37.84	39.95	38.31	40.55	37.65	30.09	37.46†‡
	LGRU-PRE + Gating		38.28	39.95	39.15	40.97	**38.44**	29.86	37.83$^{†‡↑}$
	LGRU-POST + Weighting		38.11	40.06	**39.25**	40.68	38.07	30.20	37.79$^{†‡↑}$
	LGRU-POST + Pooling		37.66	39.55	38.93	**41.04**	37.93	30.29	37.74†‡
	LGRU-POST + Gating		**38.36**	**40.47**	39.05	40.91	38.22	**30.30**	**37.95**$^{†‡↑}$



PKU translation:	the driver and 13 passengers were not in a lifetime.
Lattice translation:	the driver and 13 passengers were not alive.
Reference:	the bus driver and thirteen passengers were all killed.

Fig. 5. Translations of RNNsearch(*PKU*) and L2SNMT(*PTG*).

4.5. Analysis

In order to take a deep look into the reasons why our encoders are more effective than the conventional encoder, we study the 1-best translations produced by two systems. The first system is the RNNsearch using *PKU* segmentations, denoted by RNNsearch(*PKU*). It yields the best performance among all non-lattice NMT systems. The other is L2SNMT(*PTG*) using LGRU-POST with the *gating* composition operation, which performs better than other L2SNMT systems. We find that the utilization of word lattice brings the following advantages:

1. Word lattices alleviate 1-best tokenization errors that further cause translation errors. As illustrated in Fig. 5, in the source sentence "*che shang si ji he shi san ming cheng ke wu yi sheng huan.*" the character subsequence "*wu yi sheng huan*" is incorrectly segmented into three words "*wu*", "*yi sheng*" and "*huan*" by the *PKU* segmenter. As a result, the wrongly tokenized word "*yi sheng*" is translated into "*lifetime*". In contrast, L2SNMT(*PTG*) automatically chooses the right tokenization "*sheng huan*" due to its higher bidirectional gating weights (0.975 and 0.951). This allows the NMT system to choose the right translation "*alive*" for the word sequence "*sheng huan*".
2. Word lattices endow NMT encoders with highly expressible and flexible capabilities to learn the semantic representations of input sentences. To test this, we considered 1-best segmentations as a special kind of word lattice and used the trained L2SNMT(*PTG*) system to decode the test sets with 1-best *CTB, PKU, MSR* segmentations, respectively. Intuitively, if L2SNMT(*PTG*) mainly depends on 1-best segmentations, it will have similar performances on lattice and the 1-best segmentation corpora. From Table 3, we find that when decoding paths are constrained by 1-best segmentations, the performance of L2SNMT(*PTG*) system degrades significantly. Therefore, our L2SNMT system is able to explore tokenizations with different annotation standards.
3. We also find that the lattice-based method reduces the number of UNK words to a certain extent. We calculated the percentages of the maximal character spans of inputs covered by in-vocabulary words, and compared the *CTB, PKU, MSR* and lattice corpora. Results are shown in Table 4. We observe that the lattice corpus has the highest percentage of character spans covered by in-vocabulary words.

4.6. Effects of encoder depth

Table 5 shows the results of L2SNMT with different encoder depth. We only experiment the LGRU-POST variant with gating composition function, which is the best performing model in our previous experiments. We found that increasing encoder depth can improve the performance of L2SNMT. This finding is consistent with previous works [14–17]. We believe our L2SNMT can be further improved with deep decoders and more dedicated recurrent units, which we will leave it to our future work.

4.7. Results on Japanese–English translation

Table 6 reports the experimental results on Japanese–English translation. L2SNMT also significantly outperforms non-lattice NMT systems in all cases. When using LGRU-POST with the *gating* composition operation, the L2SNMT system outperforms Moses, RNNsearch, MSNMT by at least gains of 6.61, 0.94, and 0.65 BLEU points, respectively. Note that the system performance improvement on Japanese–English translation is slightly smaller than that

Table 3
Experimental results of L2SNMT decoding the test sets with 1-best segmentations.

System	Unit	Lattice	MT02	MT03	MT04	MT06	MT08	ALL
L2SNMT	LGRU-POST + Gating	*CTB+PKU+MSR*	40.47	39.05	40.91	38.22	30.30	37.95
		CTB	36.54	35.12	38.39	35.03	27.90	34.87
		PKU	36.72	35.25	37.84	34.45	27.94	34.63
		MSR	35.44	33.54	36.38	32.76	27.13	33.22

Table 4
The percentages of character spans covered by in-vocabulary words.

	CTB	PKU	MSR	**Lattice**
Percentage	93.31%	93.90%	96.33%	**97.06%**

Table 6
Case-sensitive BLEU score on Japanese–English translation. †, ‡ and ↑ indicate statistically significantly better than ($p < 0.01$) the best results of Moses, RNNsearch and MSNMT system, respectively. We highlight the highest BLEU score in bold for each set.

System	Unit	Input	Dev	Test
Moses	–	*Juman*	18.00	17.10
		Kytea	17.31	16.51
		Mecab	17.64	17.23
		Lattice	18.10	17.50
RNNsearch	GRU	*Char*	22.17	21.91
		Juman	23.10	23.15
		Kytea	23.40	23.17
		Mecab	23.36	23.00
MSNMT	GRU + Pooling	*Juman+Kytea+Mecab*	23.90	23.13
	GRU + Gating		23.73	23.46
L2SNMT	LGRU-PRE + Weighting	*Lattice*	23.93	$23.77^{\dagger,\ddagger}$
	LGRU-PRE + Pooling		23.76	23.58†
	LGRU-PRE + Gating		24.12	$23.71^{\dagger,\ddagger}$
	LGRU-POST + Weighting		**24.12**	$23.91^{\dagger,\ddagger}$
	LGRU-POST + Pooling		23.81	$23.71^{\dagger,\ddagger}$
	LGRU-POST + Gating		24.05	$\mathbf{24.11}^{\dagger,\ddagger,\uparrow}$

of Chinese–English translation. For this result, we speculate that the Chinese–English datasets contain four reference translations for each sentence while Japanese–English dataset only have single references. Even so, all these improvements are still significant at different levels.

5. Related work

There have been many works that exploit deep learning to learn sentence representations. The most straightforward method is the neural *Bag of Words* model. However, it has the drawback of ignoring the important information in word order. Hence, many researchers focus on order-sensitive models, which can be classified into the following two categories: (1) *sequence models* and (2) *topological models*. The most representative sequence models are RNNs [28] along with LSTM [3,12,29–31] or GRU [2,32]. Moreover, some researchers replace standard RNNs with non-sequential ones, including *multi-dimensional RNNs* [33], *grid RNNs* [34] and *higher order RNNs* [35]. Sentence representations in topological models are built based on the topological structures over words [36–41]. Besides these models, convolutional neural networks [42–45] have been widely used to model sentences.

Specific to NMT, the dominant NMT almost solely depends on word-level source sentence modeling accompanied with explicit tokenizations [2–4], which can easily be affected/interrupted by unknown words. In order to solve this problem, researchers have proposed an alternative approach: character-based modeling. For example, Costa-Jussà and Fonollosa [6] combined character based embeddings with convolutional and highway layers [46] to produce word embeddings. Similarly, Ling et al. [5] applied a bidirectional LSTM to learn semantic representations of words from character embeddings. Lee et al. [47] proposed a slightly different method by explicitly marking each character with its relative location in a word. Recently, a new approach was presented by Chung et al. [7], where they evaluated a character-level decoder without using explicit segmentations for NMT. But they still used a subword-level encoder. Subword-level segmentations such as Byte Pair Encoding (BPE [48]) and Word-piece [15] have proved to be very effective in handling rare words. The BPE approach also utilizes word boundaries and splits rare words into subwords. We believe that the BPE approach is complementary to our approach. In general, word boundary information is useful for the encoder modeling in NMT.

Different from the approaches mentioned above, we incorporate word lattice into RNN encoders, which, to our best knowledge, has never been investigated before in NMT. Existing models that are most related to ours are proposed by Tai et al. [40], Le and Zuidema [41], Kalchbrenner et al. [34], Soltani and Jiang [35], Ladhak et al. [49]: Tai et al. [40] presented *tree-structured LSTMs*; Le and Zuidema [41] further introduced the forest convolutional network for sentence modeling; Kalchbrenner et al. [34] presented *grid RNNs* which arranged the inputs in a multi-dimensional grid. Soltani and Jiang [35] proposed higher order RNNs where more memory units are used so as to record more preceding states, and then they are all recurrently fed to the hidden layers as feedbacks through different weighted paths. Ladhak et al. [49] proposed a method which is similar to our LGRU-POST to allow RNNs to encode word lattices, and showed its effectiveness on automatic speech recognition. Sperber et al. [50] proposed to use Tree LSTM to encode lattices in NMT, which is different from our RNN based approach. Their studies also confirmed the usefulness of lattice in NMT.

Compared with these models, our work is significantly different in the following way. We adopt word lattices rather than parse trees and forests to improve sentence modeling, thus our network structures rely on the input word lattices, remarkably different from the prefixed structures of *grid RNNs* or *high order RNNs*.

Table 5
Comparison of our L2SNMT with different encoder depth.

System	Unit	Layers	Parameters	MT05	MT02	MT03	MT04	MT06	MT08	ALL
L2SNMT	LGRU-POST + Gating	1	55.5M	38.36	40.47	39.05	40.91	38.22	30.30	37.95
	LGRU-POST + Gating	2	59.0M	38.51	40.40	39.17	41.15	38.69	30.34	38.19
	LGRU-POST + Gating	3	62.2M	38.74	40.94	39.40	41.13	38.88	30.94	38.37
	LGRU-POST + Gating	4	65.3M	39.09	40.94	39.90	41.26	38.94	31.17	38.55

Moreover, our encoders have the ability to simultaneously deal with multiple tokenization-specific input vectors, while tree-structured LSTMs, grid RNNs and high order RNNs can only process one input vector at each time step.

6. Conclusions and future work

In this paper, we presented lattice-to-sequence attentional NMT models. Different from the standard RNN encoders, the lattice-based RNN encoders of our L2SNMT simultaneously exploit the inputs and preceding hidden states specific to different tokenizations for source sentence modeling. Thus, they reduce error propagations of 1-best tokenizations, and at the same time, are more expressive and flexible than the standard encoder. Experimental results on Chinese–English and Japanese–English translation show that our models outperform a variety of baselines.

Future work can be pursued in the following directions. In this work, our network structures depend on the word lattices of source sentences. They can be extended to incorporate the segmentation model into source sentence representation learning. In this case, we allow tokenization and translation to collaborate with each other. Furthermore, we have interest in exploring better combination strategies and other recurrent units to further improve our models.

Acknowledgments

This work was supported by the Natural Science Foundation of China (Grant nos. 61573294, 61672440); the Ph.D. Programs Foundation of Ministry of Education of China (Grant no. 20130121110040); the Foundation of the State Language Commission of China (Grant nos. WT135-10, YB135-49); the Natural Science Foundation of Fujian Province (Grant no. 2016J05161); the Fund of Research Project of Tibet Autonomous Region of China (Grant no. Z2014A18G2-13); and the National Key Technology R&D Program (Grant no. 2012BAH14F03).

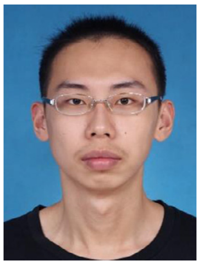
References

[1] N. Kalchbrenner, P. Blunsom, Recurrent continuous translation models, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2013, pp. 1700–1709.
[2] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using encoder–decoder for statistical machine translation, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1724–1734.
[3] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Proceedings of the International Conference on Neural Information Processing Systems, NIPS, 2014, pp. 3104–3112.
[4] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, Proceedings of the International Conference on Learning Representations, ICLR, 2015.
[5] W. Ling, I. Trancoso, C. Dyer, A.W. Black, Character-based neural machine translation, arXiv:1511.04586 [cs], 2015.
[6] M.R. Costa-jussà, J.A.R. Fonollosa, Character-based neural machine translation, Proceedings of the Association for Computational Linguistics, ACL, 2016, pp. 357–361.
[7] J. Chung, K. Cho, Y. Bengio, A character-level decoder without explicit segmentation for neural machine translation, Proceedings of the Association for Computational Linguistics, ACL, 2016, pp. 1693–1703.
[8] C. Dyer, S. Muresan, P. Resnik, Generalizing word lattice translation, Proceedings of the Association for Computational Linguistics, ACL, 2008, pp. 1012–1020.
[9] W. Jiang, H. Mi, Q. Liu, Word lattice reranking for Chinese word segmentation and part-of-speech tagging, Proceedings of the International Conference on Computational Linguistics, COLING, 2008, pp. 385–392.
[10] Z. Wang, C. Zong, N. Xue, A lattice-based framework for joint Chinese word segmentation, POS tagging and parsing, Proceedings of the Association for Computational Linguistics, ACL, 2013, pp. 623–627.
[11] J. Su, Z. Tan, D. Xiong, R. Ji, X. Shi, Y. Liu, Lattice-based recurrent neural network encoders for neural machine translation, Proceedings of the Conference on Artificial Intelligence, AAAI, 2017, pp. 3302–3308.
[12] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[13] L.R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[14] J. Zhou, Y. Cao, X. Wang, P. Li, W. Xu, Deep recurrent models with fast-forward connections for neural machine translation, Trans. Assoc. Comput. Linguist. 4 (2016) 371–383.
[15] Y. Wu, M. Schuster, Z. Chen, Q.V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al., Google's neural machine translation system: Bridging the gap between human and machine translation, arXiv:1609.08144 [cs], 2016.
[16] M. Wang, Z. Lu, J. Zhou, Q. Liu, Deep Neural Machine Translation with Linear Associative Unit, in: Proc. of ACL, 2017, pp. 136–145.
[17] A.V.M. Barone, J. Helcl, R. Sennrich, B. Haddow, A. Birch, Deep architectures for neural machine translation, in: Proc. of WMT, 2017, pp. 99–107.
[18] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[19] I.J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, in: Proc. of ICML, 2013, pp. 1319–1327.
[20] K. Papineni, S. Roukos, T. Ward, W. Zhu, Bleu: a method for automatic evaluation of machine translation, Proceedings of the Association for Computational Linguistics, ACL, 2002, pp. 311–318.
[21] P. Koehn, Statistical significance tests for machine translation evaluation, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2004, pp. 388–395.
[22] T. Nakazawa, M. Yaguchi, K. Uchimoto, M. Utiyama, E. Sumita, S. Kurohashi, H. Isahara, Aspec: asian scientific paper excerpt corpus, Proceedings of the International Conference on Language Resources and Evaluation, LREC, 2016, pp. 2204–2208.
[23] G. Neubig, Y. Nakata, S. Mori, Pointwise prediction for robust, adaptable Japanese morphological analysis, Proceedings of the Association for Computational Linguistics, ACL, 2011, pp. 529–533.
[24] T. Kudo, K. Yamamoto, Y. Matsumoto, Applying conditional random fields to Japanese morphological analysis, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2004, pp. 230–237.
[25] A. Stolcke, Srilm - an extensible language modeling toolkit, Proceedings of the International Conference on Spoken Language Processing, ICSLP, 2002, pp. 901–904.
[26] D. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proc. of ICLR, 2015.
[27] L. Bentivogli, A. Bisazza, M. Cettolo, M. Federico, Neural versus phrase-based machine translation quality: a case study, in: Proc. of EMNLP, 2016, pp. 257–267.
[28] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, S. Khudanpur, Recurrent neural network based language model, Proceedings of the INTERSPEECH, 2010, pp. 1045–1048.
[29] P. Le, W. Zuidema, Compositional distributional semantics with long short term memory, Proceedings of the Joint Conference on Lexical and Computational Semantics, SEM, 2015, pp. 10–19.
[30] X. Zhu, P. Sobhani, H. Guo, Long short-term memory over tree structures, Proceedings of the International Conference on International Conference on Machine Learning, ICML, 2015, pp. 1604–1612.
[31] P. Liu, X. Qiu, X. Chen, S. Wu, X. Huang, Multi-timescale long short-term memory neural network for modelling sentences and documents, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 2326–2335.
[32] X. Chen, X. Qiu, C. Zhu, S. Wu, X. Huang, Sentence modeling with gated recursive neural network, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 655–665.
[33] A. Graves, S. Fernandez, J. Schmidhuber, Multi-dimensional recurrent neural networks, Proceedings of the International Conference on Artificial Neural Networks, ICANN, 2007, pp. 549–558.
[34] N. Kalchbrenner, I. Danihelka, A. Graves, Grid long short-term memory, arXiv:1507.0152 [cs], 2015.
[35] R. Soltani, H. Jiang, Higher order recurrent neural networks, arXiv:1605.00064 [cs], 2016.
[36] R. Socher, C.C.-Y. Lin, A.Y. Ng, C.D. Manning, Parsing natural scenes and natural language with recursive neural networks, Proceedings of the International Conference on International Conference on Machine Learning, ICML, 2011, pp. 129–136.
[37] K.M. Hermann, P. Blunsom, The role of syntax in vector space models of compositional semantics, Proceedings of the Association for Computational Linguistics, ACL, 2013, pp. 894–904.
[38] M. Iyyer, J.B. Graber, L. Claudino, R. Socher, H. Daum'e III, A neural network for factoid question answering over paragraphs, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 633–644.
[39] L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, Z. Jin, Discriminative neural sentence modeling by tree-based convolution, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 2315–2325.
[40] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks, Proceedings of the Association for Computational Linguistics, ACL, 2015, pp. 1556–1566.
[41] P. Le, W. Zuidema, The forest convolutional network-compositional distributional semantics with a neural chart and without binarization, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2015, pp. 1155–1164.

[42] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, J. Mach. Learn. Res. 12 (2011) 2493–2537.
[43] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, Proceedings of the Association for Computational Linguistics, ACL, 2014, pp. 655–665.
[44] Y. Kim, Convolutional neural networks for sentence classification, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, 2014, pp. 1746–1751.
[45] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, Proceedings of the International Conference on Neural Information Processing Systems, NIPS, 2014, pp. 2042–2050.
[46] R.K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, Proceedings of the International Conference on Neural Information Processing Systems, NIPS, 2015, pp. 2368–2376.
[47] H.-G. Lee, J. Lee, J.-S. Kim, C.-K. Lee, Naver machine translation system for wat 2015, Proceedings of the Workshop on Asian Translation, WAT, 2015, pp. 69–73.
[48] R. Sennrich, B. Haddow, M. Birch, Neural machine translation of rare words with subword units, in: Proc. of ACL, 2016, pp. 1715–1725.
[49] F. Ladhak, A. Gandhe, M. Dreyer, L. Mathias, A. Rastrow, B. Hoffmeister, Latticernn: recurrent neural networks over lattices, Proceedings of the INTERSPEECH, 2016, pp. 695–699.
[50] M. Sperber, G. Neubig, J. Niehues, A. Waibel, Neural lattice-to-sequence models for uncertain inputs, Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP, Copenhagen, Denmark, 2017, pp. 1380–1389.

Zhixing Tan is currently a Ph.D. candidate at School of Information Science and Engineering, Xiamen University. His research interests include natural language processing, machine translation and deep learning.

Jinsong Su received his Ph.D. degree in computer sicence from Chinese Academy of Science, Beijing, China, in 2011. He is now an associate professor in the Software School of Xiamen University. His research interests include natural language processing and machine learning.



Boli Wang is currently a Ph.D. candidate at School of Information Science and Engineering, Xiamen University. His research interests include natural language processing and deep learning.

Yidong Chen received his Ph.D. degree in mathematics from Xiamen University, Xiamen, China, in 2008. He is now an associate professor in the Cognitive Science Department of Xiamen University. His research interests include machine translation and semantic analysis.

Xiaodong Shi received his Ph.D. degree in computer software from National University of Defense Technology, Changsha, China, in 1994. He is now a professor in the Cognitive Science Department of Xiamen University. His research interests include natural language processing and artificial intelligence.