Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation

Jinsong Su\textsuperscript{1}, Zhixing Tan\textsuperscript{1}, Deyi Xiong\textsuperscript{2}, Rongrong Ji\textsuperscript{1}, Xiaodong Shi\textsuperscript{1}, Yang Liu\textsuperscript{3}\textsuperscript{*}

Xiamen University, Xiamen, China\textsuperscript{1}
Soochow University, Suzhou, China\textsuperscript{2}
Tsinghua University, Beijing, China\textsuperscript{3}

\{jssu, rrji, mandel\}@xmu.edu.cn, playinf@stu.xmu.edu.cn
dyxiong@suda.edu.cn, liuyang2011@tsinghua.edu.cn

Abstract

Neural machine translation (NMT) heavily relies on word-level modelling to learn semantic representations of input sentences. However, for languages without natural word delimiters (e.g., Chinese) where input sentences have to be tokenized first, conventional NMT is confronted with two issues: 1) it is difficult to find an optimal tokenization granularity for source sentence modelling, and 2) errors in 1-best tokenizations may propagate to the encoder of NMT. To handle these issues, we propose word-lattice based Recurrent Neural Network (RNN) encoders for NMT, which generalize the standard RNN to word lattice topology. The proposed encoders take as input a word lattice that compactly encodes multiple tokenizations, and learn to generate new hidden states from arbitrarily many inputs and hidden states in preceding time steps. As such, the word-lattice based encoders not only alleviate the negative impact of tokenization errors but also are more expressive and flexible to embed input sentences. Experiment results on Chinese-English translation demonstrate the superiorities of the proposed encoders over the conventional encoder.

Introduction

In the last two years, NMT (Kalchbrenner and Blunsom 2013; Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015) has obtained state-of-the-art translation performance on some language pairs. In contrast to conventional statistical machine translation that models latent structures and correspondences between the source and target languages in a pipeline, NMT trains a unified encoder-decoder (Cho et al. 2014; Sutskever, Vinyals, and Le 2014) neural network for translation, where an encoder maps the input sentence into a fixed-length vector, and a decoder generates a translation from the encoded vector.

Most NMT systems adopt RNNs such as Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) and Gated Recurrent Unit (GRU) (Cho et al. 2014) to learn vector representations of source sentences and to generate target sentences due to the strong capability of RNNs in capturing long-distance dependencies. Generally, such representations and generations are learned and performed at the word level. Very recently, we have also seen successful approaches in character-level NMT (Ling et al. 2015; Costa-Juss’a and Fonollosa 2016; Chung, Cho, and Bengio 2016). In these methods, each input sentence is first segmented into a sequence of words, after which RNNs are used to learn word representations and perform generations at the character level. Note that all approaches so far exploit word boundary information for learning source representations. Such a modeling philosophy works well for languages like English. However, it is not a desirable choice for languages without natural word delimiters such as Chinese. The underlying reasons are twofold: 1) the optimal tokenization granularity is always a paradox for machine translation since coarse granularity causes data sparseness while fine granularity results in the loss of useful information, and 2) there may exist some errors in 1-best tokenizations, which potentially propagate to source sentence representations. Therefore, it is necessary to offer multiple tokenizations instead of a single tokenized sequence to NMT for better source sentence modelling.

In order to handle these tokenization issues, we propose word-lattice based RNN encoders for NMT. Word lattice is a packed representation of many tokenizations, which is achieved by using a word lattice topology. We expect to not only eliminate the negative effect caused by 1-best tokenization errors but also make the encoder more expressive and flexible to learn semantic representations of source sentences. In this work, we investigate and compare two word-lattice based RNN encoders: 1) a Deep Word-Lattice Based GRU Encoder which builds on the combinations of inputs and hidden states derived from multiple tokenizations without any change to the architecture of the standard GRU, and 2) a Shallow Word-Lattice Based GRU Encoder which learns and updates tokenization-specific vectors for gates, inputs and hidden states, and then generates hidden state vectors for current units. In both encoders, many different tokenizations can be simultaneously exploited for input sentence modeling.

To demonstrate the effectiveness of the proposed encoders, we carry out experiments on Chinese-English translation tasks. Experiment results show that: 1) it is really
necessary to exploit word boundary information for learning accurate embeddings of input Chinese sentences; (2) word-lattice based RNN encoders outperform the standard RNN encoder in NMT. To the best of our knowledge, this is the first attempt to build NMT on word lattices.

**Neural Machine Translation**

The dominant NMT model is an attention-based one (Bahdanau, Cho, and Bengio 2015), which includes an encoder network and a decoder network with attention mechanism.

The encoder is generally a bidirectional RNN. The forward RNN reads a source sentence $x_{1:T}$ from left to right and applies the recurrent activation function $\phi$ to learn semantic representation of word sequence $x_{1:i}$ as $h_i = \phi(h_{i-1}, \bar{x}_i)$. Similarly, the backward RNN scans the source sentence in a reverse direction and learns the semantic representation $\bar{h}_i$ of the word sequence $x_{i:T}$. Finally, we concatenate the hidden states of the two RNNs to form the source annotation $h_i = [h_i^T, \bar{h}_i^T]^T$, which encodes information about the $i$-th word with respect to all the other surrounding words in the source sentence.

The decoder is a forward RNN producing the translation $y$ in the following way:

$$p(y_j|y_{<j}, x) = g(y_{j-1}, s_j, m_j),$$

where $g(\cdot)$ is a non-linear function, and $s_j$ and $m_j$ denote the decoding state and the source context at the $j$th time step, respectively. In particular, $s_j$ is computed as follows:

$$s_j = f(s_{j-1}, y_{j-1}, m_j),$$

where $f(\cdot)$ is an activation function such as GRU function. According to the attention mechanism, we define $m_j$ as the weighted sum of the source annotations $\{h_i\}$:

$$m_j = \sum_{i=1}^{I} \alpha_{j,i} \cdot h_i,$$

where $\alpha_{j,i}$ measures how well $y_j$ and $h_i$ match as below:

$$\alpha_{j,i} = \frac{exp(e_{j,i})}{\sum_{j'=1}^{T} exp(e_{j,j'})},$$

and

$$e_{j,i} = v_a^T \tanh(W_a s_j - 1 + U_a h_i),$$

where $W_a$, $U_a$ and $v_a$ are the weight matrices of attention model. With this model, the decoder automatically selects words in the source sentence that are relevant to target words being generated.

**Word-Lattice based RNN Encoders**

In this section, we study how to incorporate word lattice into the RNN encoder of NMT. We first briefly review the standard GRU, which is the basis of our encoder units. Then, we describe how to generate the word lattice of each input sentence. Finally, we describe word-lattice based GRU encoders in detail.

**The Standard GRU**

GRU is a new type of hidden unit motivated by LSTM. As illustrated in Figure 2(a), at time step $t$, GRU has two gates: 1) a reset gate $r_t$ which determines how to combine the new input $x_t$ with the previous memory $h_{t-1}$, and 2) an update gate $z_t$ that defines how much of the previous memory to keep around. Formally, the GRU transition equations are as follows:

$$r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1}),$$

$$z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}),$$

$$\tilde{h}_t = \phi(W x_t + U (r_t \odot h_{t-1})),$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t,$$

where $\sigma$ is the logistic sigmoid function, $\odot$ denotes the elementwise multiplication, and $W^r$ and $U^r$ are the weight matrices, respectively.

Similar to LSTM, GRU overcomes exploding or vanishing gradient problem (Hochreiter and Schmidhuber 1997) of the conventional RNN. However, GRU is easier to compute and implement than LSTM. Therefore, in this work, we adopt GRU to build our word lattice based encoders. However, our method can be applied on other RNNs as well.

**Word Lattice**

As shown in Figure 1, a word lattice is a directed graph $G = (V, E)$, where $V$ is node set and $E$ is edge set. Given the word lattice of a character sequence $c_{1:N} = c_{1} \ldots c_{N}$, node $v_t \in V$ denotes the position between $c_t$ and $c_{t+1}$, edge $e_{t,j} \in E$ departs from $v_t$ and arrives at $v_{t+j}$ from left to right, covering the subsequence $c_{t:t+j}$ that is recognized as a possible word. To generate word lattices, we train many word segmenters using multiple corpora with different annotation standards, such as the Peking University Corpus (PKU), the Microsoft Research Corpus (MSR) and the Pennsylvania Chinese Treebank 6.0 (CTB). For each input sentence, we tokenize it using different segmenters and generate a word lattice by merging the predicted tokenizations that are shared in different segmenters. For example, in Figure 1, both CTB and MSR segmenters produce the same tokenization “词...”，which is merged into the edge $e_{0:3}$ in the lattice. Obviously, word lattice has richer network topology than a single word sequence. It encodes many tokenizations for the same character subsequence, and thus there may exist multiple inputs and preceding hidden states at each time step, which can not be simultaneously modelled in the standard RNN.

**Word-Lattice based RNN with GRU Encoders**

To deal with the above problem, we propose word-lattice based GRU encoders for NMT. Similar to the dominant NMT model (Bahdanau, Cho, and Bengio 2015), our encoders are bidirectional RNNs with GRU. Here we only introduce the forward RNN. The backward RNN can be extended in a similar way.

Our encoders scan a source sentence character by character. Only at potential word boundaries, hidden states are generated from many input candidate words and the corresponding preceding hidden states. Specifically, at time step
For $t$, we first identify all edges pointing to $v_t$, each of which covers different input words with preceding hidden states. In particular, for the $k$th edge, we denote its input word vector and the corresponding preceding hidden state as $x_t^{(k)}$ and $h_{t|k}^{pre}$, respectively. As shown in Figure 1, the word lattice contains two edges $e_{0:3}$ and $e_{1:3}$, both of which link to $v_3$. Therefore, there exist two input words “副-总-理” and “总-理” with different preceding hidden states at the 3rd time step. We then propose two word-lattice based GRUs to exploit multiple tokenizations simultaneously:

1. **Shallow Word-Lattice Based GRU (SWL-GRU)**. The architecture of SWL-GRU is shown in Figure 2(b). At the potential word boundary of the $t$th step, we combine all possible word embeddings $\{x_t^i\}$ into a compressed $x_t$. Similarly, the hidden state vectors $\{h_{t|k}^{pre}\}$ of preceding time steps are also compressed into $h_{pre}$. Then, both $x_t$ and $h_{pre}$ are fed into the standard GRU to generate the final hidden state vector $h_t$. Obviously, here we do not change the inner architecture of the standard GRU. The combination of multiple word embeddings into one compressed vector, the combination of all preceding hidden states and the corresponding GRU are defined as follows:

$$x_t = g(x_t^{(1)}, x_t^{(2)}, \ldots),$$

$$h_{pre} = g(h_{pre}^{(1)}, h_{pre}^{(2)}, \ldots),$$

$$r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{pre}),$$

$$z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{pre}),$$

$$\tilde{h}_t = \phi(W x_t + U (r_t \odot h_{pre})),$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t,$$

where Eqs. (10)-(11) are used to generate the compressed representations $x_t$ and $h_{pre}$, the others are the same as those of the standard GRU, and $g(*)$ is a composition function, for which we will investigate two definitions later on.

2. **Deep Word-Lattice Based GRU (DWL-GRU)**. The architecture of DWL-GRU is illustrated in Figure 2(c). In this unit, we set and update the reset gate $r_t^{(k)}$, the update gate $z_t^{(k)}$ and the hidden state vector $h_t^{(k)}$ that are specific to the $k$th edge ending with $v_t$, and then generate a composed hidden state vector $h_t$ from $\{h_t^{(k)}\}$. Different from SWL-GRU, DWL-GRU merges the hidden states specific to different tokenizations rather than the inputs and the preceding hidden states. Formally, the transition equations are defined as follows:

$$r_t^{(k)} = \sigma(W^{(r)} x_t^{(k)} + U^{(r)} h_{pre}^{(k)}),$$

$$z_t^{(k)} = \sigma(W^{(z)} x_t^{(k)} + U^{(z)} h_{pre}^{(k)}),$$

$$\tilde{h}_t^{(k)} = \phi(W x_t^{(k)} + U (r_t^{(k)} \odot h_{pre}^{(k)}).$$

We index the edges from left to right according to the positions of their starting nodes.
hidden layer size as 512, learning rate as $5 \times 10^{-4}$, batch size as 80, gradient norm as 1.0. All the other settings are the same as in (Bahdanau, Cho, and Bengio 2015).

**Baselines**

We refer to the attention-based NMT system with the proposed encoders as **LatticeNMT**, which has four variants with different network units and composition operations. We compared them against the following state-of-the-art SMT and NMT systems:

- **Moses**: an open source phrase-based translation system with default configurations and a 4-gram language model trained on the target portion of training data.
- **RNNSearch** (Bahdanau, Cho, and Bengio 2015): an in-house attention-based NMT system, which has slightly better performance and faster speed than **GroundHog**.\(^3\)

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.

- **MultiSourceNMT**: an attention-based NMT system which also uses many tokenizations of each source sentence as input. However, it performs combination at the final time step, significantly different from the proposed word-lattice based GRUs. Similarly, it has two variants with different composition functions.

**Overall Performance**

Table 1 reports the experiment results. Obviously, **LatticeNMT** significantly outperforms non-lattice NMT systems in all cases. When using DWL-GRU with the gating composition operation, the **LatticeNMT** system outperforms **Moses**, **RNNSearch**, **MultiSourceNMT** by at least gains of 1.45, 1.17, and 0.82 BLEU points, respectively.

**Parameters.** 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**.

**Speed.** We used a single GPU device TitanX to train models. It takes one hour to train 9,000 and 4,200 mini-batches for word- and character-based **RNNSearch** systems, respectively. The training speeds of **MultiSourceNMT** and **LatticeNMT** systems are slower than that of word-based **RNNSearch**: about 4,800–6,000 mini-batches are processed in one hour.

From the table, we further have the following findings:

1. On all word-segmented corpora (e.g., **CTB**, **PKU**, **MSR**), **RNNSearch** performs better than character-based NMT system, which demonstrates that for Chinese which is not morphologically rich, word boundary information is very useful for source sentence encoding in NMT. For this, we speculate that the character-level encoding has two disadvantages in comparison with the word-level encoding. First, when a source sentence is represented as a sequence of

\[^3\]http://www.statmt.org/moses/

\[^4\]https://github.com/lisa-groundhog/GroundHog

\[h_t = \sum_{k=1}^{K} \sigma(h_{tk} U^{(g)} + b^{(g)}) h_{tk} \]

where $U^{(g)}$ and $b^{(g)}$ are the matrix and the bias term, respectively.
characters rather than words, the sequence length grows significantly. However, translating long sequences still remains a great challenge for the attention-based NMT (Bentivogli et al. 2016). Second, the character-level encoder is unable to capture word-level interactions for learning the representations of entire sentences.

2) Multiple inputs with different word segmentation standards are useful for NMT. Furthermore, multiple inputs based on word lattices achieve better results. The underlying reason may be that rather than only at final time step, all hidden states of word-lattice based encoders at potential word boundaries are influenced by many different tokenizations. This enables different tokenizations to fully play complementary roles in learning source sentence representations.

(3) No matter which composition function is used, DWL-GRU is slightly more effective than SWL-GRU. These results accord with our intuition since DWL-GRU exploits multiple tokenizations at a more fine-grained level than SWL-GRU.

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 CTB segmentations, denoted by RNNSearch(CTB). It yields the best performance among all non-lattice NMT systems. The other is LatticeNMT(DG) using DWL-GRU with the gating composition operation, which performs better than other LatticeNMT 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 Figure 3, in the source sentence “c1起c2诉c3邮c4诉c5, c6两c7被c8告...”, the character subsequence c7-s“被 吹” is incorrectly segmented into two words “被” and “吹” by the CTB segmenter. As a result, the word sequence “两 被 吹” is incorrectly translated into “the two sides”. In contrast, LatticeNMT(DG) automatically chooses the right tokenization “被 吹” due to its higher bidirectional gating weights (0.982 and 0.962). This allows the NMT system to choose the right translation “the two defendants” for the word sequence “两 被 吹”.

2) Word lattices endow NMT encoders with highly expressive 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 LatticeNMT(DG) system to decode the test sets with 1-best CTB, PKU, MSR segmentations, respectively. Intuitively, if LatticeNMT(DG) mainly depends on 1-best segmentations, it will have similar performances on lattice and non-lattice NMT systems. From Table 3, we find that when decoding paths are constrained by 1-best segmentations, the performance of LatticeNMT(DG) system degrades significantly. Therefore, our LatticeNMT system is able to explore tokenizations with different annotation standards.

3) We also find that the lattice-based method reduces the

Table 1: Evaluation of translation quality. †, ‡ and †† indicate statistically significantly better than (p<0.01) the best results of Moses, RNNSearch and MultiSourceNMT system, respectively. We highlight the highest BLEU score in bold for each set.

<table>
<thead>
<tr>
<th>System</th>
<th>Unit</th>
<th>Input</th>
<th>MT05</th>
<th>MT02</th>
<th>MT03</th>
<th>MT04</th>
<th>MT06</th>
<th>MT08</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moses</td>
<td>—</td>
<td>PKU</td>
<td>31.70</td>
<td>33.61</td>
<td>32.63</td>
<td>34.36</td>
<td>31.00</td>
<td>23.96</td>
<td>31.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSR</td>
<td>31.47</td>
<td>33.19</td>
<td>32.43</td>
<td>34.14</td>
<td>30.81</td>
<td>23.85</td>
<td>31.04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lattice</td>
<td>31.08</td>
<td>32.44</td>
<td>32.08</td>
<td>33.78</td>
<td>30.28</td>
<td>23.60</td>
<td>30.56</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Char</td>
<td>30.30</td>
<td>33.51</td>
<td>31.98</td>
<td>34.34</td>
<td>30.50</td>
<td>22.65</td>
<td>30.74</td>
</tr>
<tr>
<td>RNNSearch</td>
<td>GRU</td>
<td>CTB</td>
<td>31.38</td>
<td>34.95</td>
<td>32.85</td>
<td>35.44</td>
<td>31.75</td>
<td>23.33</td>
<td>31.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PKU</td>
<td>31.42</td>
<td>34.68</td>
<td>33.08</td>
<td>35.32</td>
<td>31.61</td>
<td>23.58</td>
<td>31.76</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSR</td>
<td>32.92</td>
<td>34.49</td>
<td>32.06</td>
<td>35.10</td>
<td>31.23</td>
<td>23.12</td>
<td>31.35</td>
</tr>
<tr>
<td>MultiSourceNMT</td>
<td>GRU+Pooling/Gating</td>
<td>CTB+PKU+MSR</td>
<td>30.27</td>
<td>34.73</td>
<td>31.76</td>
<td>34.79</td>
<td>31.36</td>
<td>23.54</td>
<td>31.34</td>
</tr>
<tr>
<td>LatticeNMT</td>
<td>GRU+Pooling/Gating</td>
<td>Lattice</td>
<td>31.95</td>
<td>35.14</td>
<td>33.36</td>
<td>35.84</td>
<td>32.09</td>
<td>23.60</td>
<td>32.13</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SWL-GRU+Pooling</td>
<td>32.27</td>
<td>35.35</td>
<td>34.20</td>
<td>36.49</td>
<td>32.27</td>
<td>24.52</td>
<td>32.69</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SWL-GRU+Gating</td>
<td>32.07</td>
<td>35.94</td>
<td>34.01</td>
<td>36.48</td>
<td>32.51</td>
<td>24.44</td>
<td>32.79</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DWL-GRU+Pooling</td>
<td>32.18</td>
<td>35.53</td>
<td>33.94</td>
<td>36.42</td>
<td>32.61</td>
<td>24.34</td>
<td>32.71</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DWL-GRU+Gating</td>
<td>32.40</td>
<td>35.75</td>
<td>34.32</td>
<td>36.50</td>
<td>32.77</td>
<td>24.84</td>
<td>32.95</td>
</tr>
</tbody>
</table>

Table 2: Experiment results of LatticeNMT decoding the test sets with 1-best segmentations.

<table>
<thead>
<tr>
<th>System</th>
<th>Unit</th>
<th>Input</th>
<th>MT02</th>
<th>MT03</th>
<th>MT04</th>
<th>MT06</th>
<th>MT08</th>
<th>ALL</th>
</tr>
</thead>
<tbody>
<tr>
<td>LatticeNMT</td>
<td>DWL-GRU+Gating</td>
<td>Lattice</td>
<td>35.75</td>
<td>34.32</td>
<td>36.50</td>
<td>32.77</td>
<td>24.84</td>
<td>32.95</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTB</td>
<td>33.31</td>
<td>31.44</td>
<td>34.05</td>
<td>30.09</td>
<td>22.59</td>
<td>29.97</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PKU</td>
<td>32.58</td>
<td>31.22</td>
<td>33.62</td>
<td>29.72</td>
<td>22.68</td>
<td>29.68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MSR</td>
<td>30.53</td>
<td>28.82</td>
<td>32.37</td>
<td>28.08</td>
<td>21.83</td>
<td>28.14</td>
</tr>
</tbody>
</table>

Table 3: The percentages of character spans covered by in-vocabulary words.
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 3. We observe that the lattice corpus has the highest percentage of character spans covered by in-vocabulary words.

Related Work

Our work is related to previous studies that learn sentence representations with deep neural networks. Among them, the most straightforward method is the neural Bag-of-Words model, which, however, neglects the important information of word order. Therefore, many researchers resort to order-sensitive models, falling into one of two classes: (1) sequence models and (2) topological models. The most typical sequence models are RNNs with LSTM (Hochreiter and Schmidhuber 1997; Sutskever, Vinyals, and Le 2014; Zhu, Sobhani, and Guo 2015; Liu et al. 2015) or GRU (Cho et al. 2014). Further, some researchers extend standard RNNs to non-sequential ones, such as multi-dimensional RNNs (Graves, Fernandez, and Schmidhuber 2007), grid RNNs (Kalchbrenner, Danihelka, and Graves 2015) and higher order RNNs (Soltani and Jiang 2016). In topological models, sentence representations are composed following given topological structures over words (Socher et al. 2011; Hermann and Blunsom 2013; Iyyer et al. 2014; Mou et al. 2015; Tai, Socher, and Manning 2015; Le and Zuidema 2015). In addition to the aforementioned models, convolutional neural networks are also widely used to model sentences (Collobert et al. 2011; Kalchbrenner, Grefenstette, and Blunsom 2014; Kim 2014; Hu et al. 2014).

Concerning NMT, the conventional NMT relies almost exclusively on word-level source sentence modelling with explicit tokenizations (Sutskever, Vinyals, and Le 2014; Cho et al. 2014; Bahdanau, Cho, and Bengio 2015), which tends to suffer from the problem of unknown words. To address this problem, researchers have proposed alternative character-based modelling. In this respect, Costa-Jussà and Fonollosa (2016) applied character-based embeddings in combination with convolutional and highway layers to produce word embeddings. Similarly, Ling et al. (2015) used a bidirectional LSTM to generate semantic representations of words based on character embeddings. A slightly different approach was proposed by Lee et al. (2015), where they explicitly marked each character with its relative location in a word. Recently, Chung et al. (2016) evaluated a character-level decoder without explicit segmentations for NMT. However, their encoder is still a subword-level one. Overall, word boundary information is very important for the encoder modelling of NMT.

In contrast to the above approaches, we incorporate word lattice into RNN encoders, which, to the best of our knowledge, has never been investigated before in NMT. The most related models to ours are those proposed by Tai et al. (2015), Le et al. (2015), Kalchbrenner et al. (2015), Soltani and Jiang (2016). Tai et al. (2015) presented tree-structured LSTMs, while Le and Zuidema (2015) further introduced the forest convolutional network for sentence modelling. Kalchbrenner et al. (2015) studied grid RNNs where the inputs are arranged in a multi-dimensional grid. In higher order RNNs proposed by Soltani and Jiang (2016), more memory units are used to record more preceding states, which are all recurrently fed to the hidden layers as feedbacks through different weighted paths. Our work is also significantly different from these models. We introduce word lattices rather than parse trees and forests to improve sentence modelling, and thus our network structures depend on the input word lattices, significantly different from the prefixed structures of grid RNNs or high order RNNs. More importantly, our encoders are able to simultaneously process multiple input vectors specific to different tokenizations, while tree-structured LSTMs, grid RNNs and high order RNNs deal with only one input vector at each time step.

Conclusions and Future Work

This paper has presented word-lattice based RNN encoders for NMT. Compared with the standard RNN encoders, our encoders simultaneously exploit the inputs and preceding hidden states depending on multiple tokenizations for source sentence modelling. Thus, they not only alleviate error propagations of 1-best tokenizations, but also are more expressive and flexible than the standard encoder. Experiment results on Chinese-English translation show that our encoders lead to significant improvements over a variety of baselines.

In the future, we plan to continue our work in the following directions. In this work, our network structures depend on the word lattices of source sentences. We will extend our models to incorporate the segmentation model into source sentence representation learning. In this way, tokenization and translation are allowed to collaborate with each other. Additionally, we are interested in exploring better combination strategies to improve our encoders.
Acknowledgments

The authors were supported by National Natural Science Foundation of China (Nos. 61303082, 61403269, 61422210, 6162209 and 61672440), Natural Science Foundation of Fujian Province (No. 2016J05161), Natural Science Foundation of Jiangsu Province (No. BK20140355), the Research Fund for the Doctoral Program of Higher Education of China (No. 20130121110040), the Special Fund Project of Ministry of Education of China (No. WT135-10), and National 863 program of China (No. 2015AA015407). We also thank the reviewers for their insightful comments.

References


