Lattice-Based Recurrent Neural Network Encoders for Neural Machine Translation

Jinsong Su[1], **Zhixing Tan**[1], **Deyi Xiong**[2], **Rongrong Ji**[1], **Xiaodong Shi**[1], **Yang Liu**[3*]
Xiamen University, Xiamen, China[1]
Soochow University, Suzhou, China[2]
Tsinghua University, Beijing, China[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 has been successfully used in a variety of NLP tasks (Xu et al. 2005; Dyer, Muresan, and Resnik 2008; Jiang, Mi, and Liu 2008; Wang, Zong, and Xue 2013). Generalizing the standard RNN to 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 *Shallow 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 *Deep 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

[*] Corresponding author.

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 $\mathbf{x}$=$x_1, x_2 ... x_I$ from left to right and applies the recurrent activation function ϕ to learn semantic representation of word sequence $x_{1:i}$ as $\overrightarrow{h}_i$=$\phi(\overrightarrow{h}_{i-1}, \vec{x}_i)$. Similarly, the backward RNN scans the source sentence in a reverse direction and learns the semantic representation $\overleftarrow{h}_i$ of the word sequence $x_{i:I}$. Finally, we concatenate the hidden states of the two RNNs to form the source annotation $h_i = [\overrightarrow{h}_i^T, \overleftarrow{h}_i^T]^T$, 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:

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

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

$$s_j = f(s_{j-1}, y_{j-1}, m_j), \quad (2)$$

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 $\{h_i\}$:

$$m_j = \sum_{i=1}^{I} \alpha_{j,i} \cdot h_i, \quad (3)$$

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

$$\alpha_{j,i} = \frac{exp(e_{j,i})}{\sum_{i'=1}^{I} exp(e_{j,i'})}, \quad (4)$$

$$e_{j,i} = v_a^T tanh(W_a s_{j-1} + U_a h_i), \quad (5)$$

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:

$$r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1}), \quad (6)$$

$$z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1}), \quad (7)$$

$$\tilde{h}_t = \phi(W x_t + U(r_t \odot h_{t-1})), \quad (8)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t, \quad (9)$$

where σ is the logistic sigmoid function, $\odot$ denotes the elementwise multiplication, and W^* and U^* 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 $G = \langle V, E \rangle$, where V is node set and E is edge set. Given the word lattice of a character sequence $c_{1:N}$=$c_1 ... c_N$, node $v_i \in V$ denotes the position between c_i and c_{i+1}, 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. 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 Penn Chinese Treebank 6.0* (*CTB*). For each input sentence, we tokenize it using these 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 "副-总-理", 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

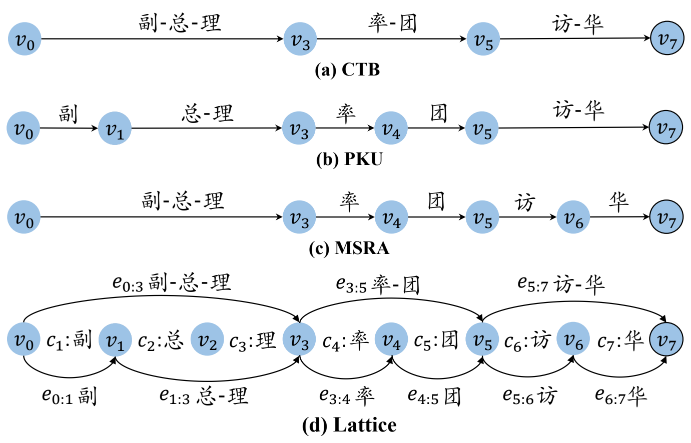
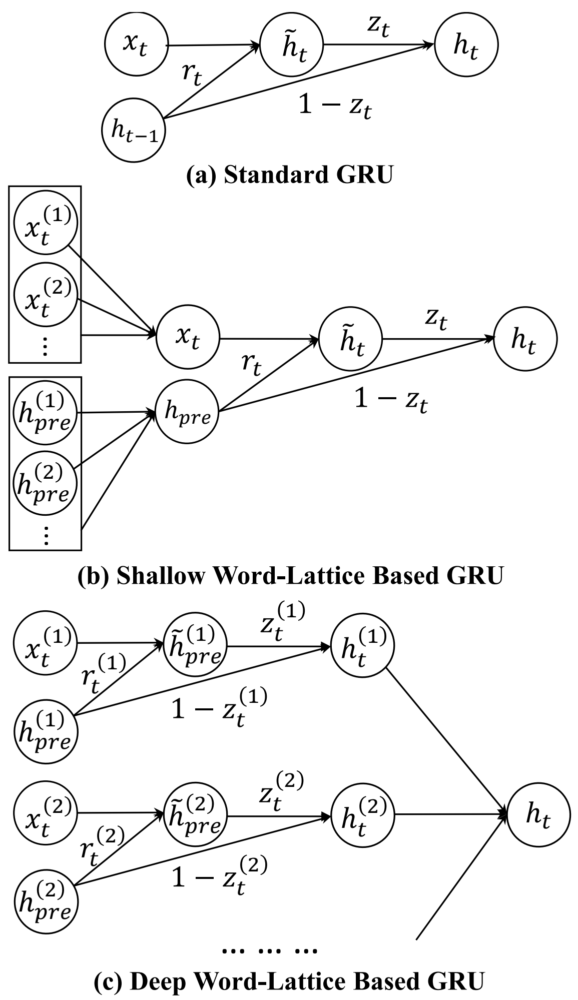


Figure 1: Three kinds of word segmentations and the resulting word lattice of the Chinese character sequence "c_1副c_2总c_3理c_4率c_5团c_6访c_7华". We insert "-" between characters in a word just for clarity.



(a) Standard GRU
(b) Shallow Word-Lattice Based GRU
(c) Deep Word-Lattice Based GRU

Figure 2: The architectures of the standard GRU and word-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.

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 kth edge[1], we denote its input word vector and the corresponding preceding hidden state as $x_t^{(k)}$ and $h_{pre}^{(k)}$, 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 tth time step, we combine all possible word embeddings $\{x_t^*\}$ into a compressed x_t. Similarly, the hidden state vectors $\{h_{pre}^*\}$ 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:

$$x_t = g(x_t^{(1)}, x_t^{(2)}, ...), \tag{10}$$
$$h_{pre} = g(h_{pre}^{(1)}, h_{pre}^{(2)}, ...), \tag{11}$$
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{pre}), \tag{12}$$
$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{pre}), \tag{13}$$
$$\tilde{h}_t = \phi(Wx_t + U(r_t \odot h_{pre})), \tag{14}$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t, \tag{15}$$

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 kth edge ending with c_t, and then generate a composed hidden state vector h_t from $\{h_t^{(*)}\}$. 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:

$$r_t^{(k)} = \sigma(W^{(r)}x_t^{(k)} + U^{(r)}h_{pre}^{(k)}), \tag{16}$$
$$z_t^{(k)} = \sigma(W^{(z)}x_t^{(k)} + U^{(z)}h_{pre}^{(k)}), \tag{17}$$
$$\tilde{h}_t^{(k)} = \phi(Wx_t^{(k)} + U(r_{tk} \odot h_{pre}^{(k)}), \tag{18}$$

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

$$h_t^{(k)} = z_t^{(k)} \odot h_{pre}^{(k)} + (1 - z_t^{(k)}) \odot \tilde{h}_t^{(k)}, \quad (19)$$

$$h_t = g(h_t^{(1)}, h_t^{(2)}, ...), \quad (20)$$

where Eqs. (16)-(19) calculate the gating vectors and the hidden state vectors depending on different tokenizations, and Eq. (20) is used to produce the final hidden state vector at the time step t.

For the composition function $g(*)$ involved in the two word-lattice based GRUs, we explore the following two functions: 1) **Pooling Operation Function** which enables our encoders to automatically capture the most important part for the source sentence modeling, and 2) **Gating Operation Function** which is able to automatically learn the weights of components in word-lattice based GRUs. Taking h_t as an example, we define it as follows

$$h_t = \sum_{k=1}^{K} \frac{\sigma(h_{tk} U^{(g)} + b^{(g)})}{\sum\limits_{k=1}^{K} \sigma(h_{tk} U^{(g)} + b^{(g)})} h_{tk} \quad (21)$$

where $U^{(g)}$ and $b^{(g)}$ are the matrix and the bias term, respectively.

Experiments

Setup

We evaluated the proposed encoders on NIST Chinese-English translation tasks. Our training data consists of 1.25M sentence pairs extracted from LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06, with 27.9M Chinese words and 34.5M English words. We chosed the NIST 2005 dataset as the validation set and the NIST 2002, 2003, 2004, 2006, and 2008 datasets as test sets. The evaluation metric is case-insensitive BLEU (Papineni et al. 2002) as calculated by the `multi-bleu.perl` script. To alleviate the impact of the instability of NMT training, we trained NMT systems five times for each experiment and reported the average BLEU scores. Furthermore, we conducted paired bootstrap sampling (Koehn 2004) to test the significance in BLEU score differences. To obtain lattices of Chinese sentences, we used the toolkit[2] released by Stanford to train word segmenters on *CTB*, *PKU*, and *MSR* corpora.

To train neural networks efficiently, we used the most frequent 50K words in Chinese and English as our vocabularies. 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. 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, which cover 89.9% of the parallel sentences. We apply *Rmsprop* (Graves 2013) (momentum = 0, ρ = 0.99, and ϵ = 1×10^{-4}) to train models until there is no BLEU improvement for 5 epochs on the validation set. During this procedure, we set the following hyper-parameters: word embedding dimension as 320, hidden layer size as 512, learning rate as 5×10^{-4}, batch size as 80, gradient norm as 1.0. All the other settings are the same as in (Bahdanau, Cho, and Bengio 2015).

[2]http://nlp.stanford.edu/software/segmenter.html#Download

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***[3]: 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*.[4] 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 *MultiSoucceNMT* 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

System	Unit	Input	MT05	MT02	MT03	MT04	MT06	MT08	ALL
Moses	——	*CTB*	31.70	33.61	32.63	34.36	31.00	23.96	31.25
		PKU	31.47	33.19	32.43	34.14	30.81	23.85	31.04
		MSR	31.08	32.44	32.08	33.78	30.28	23.60	30.56
		Lattice	31.96	33.44	32.54	34.61	31.36	24.13	31.50
RNNSearch	*GRU*	*Char*	30.30	33.51	31.98	34.34	30.50	22.65	30.74
		CTB	31.38	34.95	32.85	35.44	31.75	23.33	31.78
		PKU	31.42	34.68	33.08	35.32	31.61	23.58	31.76
		MSR	29.92	34.49	32.06	35.10	31.23	23.12	31.35
MultiSourceNMT	*GRU + Pooling*	*CTB+PKU+MSR*	30.27	34.73	31.76	34.79	31.36	23.54	31.34
	GRU + Gating		31.95	35.14	33.36	35.84	32.09	23.60	32.13
LatticeNMT	*SWL-GRU + Pooling*	*Lattice*	32.27	35.35	34.20	36.49	32.27	24.52	$32.69^{\dagger,\ddagger}$
	SWL-GRU + Gating		32.07	**35.94**	34.01	36.48	32.51	24.44	$32.79^{\dagger,\ddagger,\uparrow}$
	DWL-GRU + Pooling		32.18	35.53	33.94	36.42	32.61	24.34	$32.71^{\dagger,\ddagger,\uparrow}$
	DWL-GRU + Gating		**32.40**	35.75	**34.32**	**36.50**	**32.77**	**24.84**	$\mathbf{32.95}^{\dagger,\ddagger,\uparrow}$

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.

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

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

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 "c_1起c_2诉c_3书c_4称c_5，c_6两c_7被c_8告...", the character subsequence $c_{7:8}$"被 告" is incorrectly segmented into two words "被" and "告" by the *CTB* segmenter. As a result, the word sequence "两 被 告" is incorrectly translated into "*the two sides*". In contrast, *LatticeNMT*(*DG*) automatically chooses the right tokenization "被-告" 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-告".

	CTB	PKU	MSR	Lattice
Percentage	93.53%	94.20%	96.36%	**97.14%**

Table 3: The percentages of character spans covered by in-vocabulary words.

(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 *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 the 1-best segmentation corpora. 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

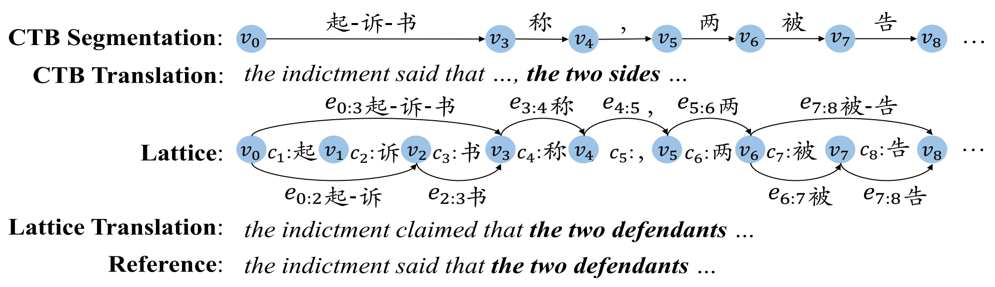

CTB Segmentation:
起-诉-书
称
，
两
被
告
v_0 v_3 v_4 v_5 v_6 v_7 v_8 …
CTB Translation:
the indictment said that …, ***the two sides*** *…*
$e_{0:3}$起-诉-书
$e_{3:4}$称
$e_{4:5}$，
$e_{5:6}$两
$e_{7:8}$被-告
Lattice:
v_0 c_1:起 v_1 c_2:诉 v_2 c_3:书 v_3 c_4:称 v_4 c_5:， v_5 c_6:两 v_6 c_7:被 v_7 c_8:告 v_8 …
$e_{0:2}$起-诉
$e_{2:3}$书
$e_{6:7}$被
$e_{7:8}$告
Lattice Translation:
the indictment claimed that ***the two defendants*** *…*
Reference:
the indictment said that ***the two defendants*** *…*

Figure 3: Translations of *RNNSearch*(*CTB*) and *LatticeNMT*(*DG*).

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, 61622209 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

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR2015*.

Bentivogli, L.; Bisazza, A.; Cettolo, M.; and Federico, M. 2016. Neural versus phrase-based machine translation quality: a case study. In *arXiv:1608.04631*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proc. of EMNLP2014*, 1724–1734.

Chung, J.; Cho, K.; and Bengio, Y. 2016. A character-level decoder without explicit segmentation for neural machine translation. In *Proc. of ACL2016*, 1693–1703.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Costa-Juss'a, M. R., and Fonollosa, J. A. R. 2016. Character-based neural machine translation. In *Proc. of ACL2016*, 357–361.

Dyer, C.; Muresan, S.; and Resnik, P. 2008. Generalizing word lattice translation. In *Proc. of ACL2008*, 1012–1020.

Graves, A.; Fernandez, S.; and Schmidhuber, J. 2007. Multi-dimensional recurrent neural networks. In *Proc. of ICANN2007*, 549–558.

Graves, A. 2013. Generating sequences with recurrent neural networks. In *arXiv:1308.0850v5*.

Hermann, K. M., and Blunsom, P. 2013. The role of syntax in vector space models of compositional semantics. In *Proc. of ACL2013*, 894–904.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation* 1735–1780.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proc. of NIPS2014*, 2042–2050.

Iyyer, M.; Graber, J. B.; Claudino, L.; Socher, R.; and Daum'e III, H. 2014. A neural network for factoid question answering over paragraphs. In *Proc. of EMNLP2014*, 633–644.

Jiang, W.; Mi, H.; and Liu, Q. 2008. Word lattice reranking for chinese word segmentation and part-of-speech tagging. In *Proc. of COLING2008*, 385–392.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent continuous translation models. In *Proc. of EMNLP2013*, 1700–1709.

Kalchbrenner, N.; Danihelka, I.; and Graves, A. 2015. Grid long short-term memory. In *arXiv:1507.01526*.

Kalchbrenner, N.; Grefenstette, E.; and Blunsom, P. 2014. A convolutional neural network for modelling sentences. In *Proc. of ACL2014*, 655–665.

Kim, Y. 2014. Convolutional neural networks for sentence classification. In *Proc. of EMNLP2014*, 1746–1751.

Koehn, P. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP2004*, 388–395.

Le, P., and Zuidema, W. 2015. The forest convolutional network-compositional distributional semantics with a neural chart and without binarization. In *Proc. of EMNLP2015*, 1155–1164.

Lee, H.-G.; Lee, J.; Kim, J.-S.; and Lee, C.-K. 2015. Naver machine translation system for wat 2015. In *Proc. of WAT2015*, 69–73.

Ling, W.; Trancoso, I.; Dyer, C.; and Black, A. W. 2015. Character-based neural machine translation. In *arXiv:1511.04586*.

Liu, P.; Qiu, X.; Chen, X.; Wu, S.; and Huang, X. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proc. of EMNLP2015*, 2326–2335.

Mou, L.; Peng, H.; Li, G.; Xu, Y.; Zhang, L.; and Jin, Z. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proc. of EMNLP2015*, 2315–2325.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proc. of ACL2002*, 311–318.

Socher, R.; Lin, C. C.-Y.; Ng, A. Y.; and Manning, C. D. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proc. of ICML2011*, 129–136.

Soltani, R., and Jiang, H. 2016. Higher order recurrent neural networks. In *arXiv:1605.00064*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS2014*, 3104–3112.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proc. of ACL2015*, 1556–1566.

Wang, Z.; Zong, C.; and Xue, N. 2013. A lattice-based framework for joint chinese word segmentation, pos tagging and parsing. In *Proc. of ACL2013*, 623–627.

Xu, J.; Matusov, E.; Zens, R.; and Ney, H. 2005. Integrated chinese word segmentation in statistical machine translation. In *Proc. of IWSLT 2005*.

Zhu, X.; Sobhani, P.; and Guo, H. 2015. Long short-term memory over tree structures. In *Proc. of ICML2015*, 1604–1612.