Source Segment Encoding for Neural Machine Translation

Qiang Wang[1,2](✉), Tong Xiao[1,2], and Jingbo Zhu[1,2]

[1] Natural Language Processing Lab, Northeastern University, Shenyang, China
{xiaotong,zhujingbo}@mail.neu.edu.cn
[2] NiuTrans Inc., Shenyang, China
wangqiangneu@gmail.com

Abstract. Sequential word encoding lacks explicit representations of structural dependencies (e.g. tree, segment) over the source words in neural machine translation. Instead of using source syntax, in this paper we propose a source segment encoding (SSE) approach to modeling source segments in encoding process by two methods. One is to encode off-the-shelf n-grams of the source sentence into original source memory. The other is to jointly learn an optimal segmentation model with the translation model in an end-to-end manner without any supervision of segmentation. Experimental results show that the SSE method yields an improvement of 2.1+ BLEU points over the baselines on the Chinese-English translation task.

Keywords: Source segment encoding · Structure learning
Neural machine translation

1 Introduction

Neural machine translation (NMT) exploits an encoder-decoder framework to model the whole translation process in an end-to-end fashion, and has achieved state-of-the-art performance in many language pairs [17,19,22]. For the encoder, a popular way is to treat the source sentence as a sequence of words. In a view point of memory network [18], the encoder reads the source sentence, and then builds a source memory where each memory cell is corresponding to a source word, referred to as a *word-level cell*.

Recent studies suggested that the sequential word encoding lacks explicit representations of the structural dependencies (e.g. tree, segment) among the source words [4,7,8,13]. Many studies resort to source syntax to improve word-level representation by enhancing word embedding [13,16], guiding encoding order [4], or learning latent graph structure of the source-side [7]. Most of these works are required to prepare a good source parser in advance, which is scarce for some languages and may cause the error propagation for the downstream applications. Alternatively, [8] propose to model latent source-side segment in the attention layer of NMT. But their method slows down the system significantly.

In this paper, we develop a *source segment encoding* (SSE) approach to enhancing original word-level representation by using source segment. In form, the source segment consists of a subsequence of consecutive source words[1]. The segment has the advantage of putting more emphasis on local dependencies over the words, which is proved helpful in NMT [8]. In SSE, we propose two methods to incorporate the source segment in encoding. One is to directly encode off-the-shelf n-grams of the source sentence into the source memory, where a n-gram is equivalent to a segment. However, the size of segments explodes as n gets larger. To alleviate this problem, we present the other method which jointly learns a segmentation model with the translation model to capture an optimal segmentation of the source sentence. The segmentation model is trained end-to-end without any supervisions of segmented sentences. Afterwards, the source memory is enhanced by combining the original word-level cells with the representations of all segmentations (referred to as *segment-level cells*). In addition, our model is light and requires almost no modification of the standard decoder network. We evaluate our model on Chinese→English translation task. Experimental results on various test sets show that our model yields an average improvement of 2.1+ BLEU points over the baseline.

2 Attention-Based NMT

Given a source sentence $X = (x_1, \ldots, x_{L_s})$, and a target sentence $Y = (y_1, \ldots, y_{L_t})$, the translation probability $P(Y|X)$ can be decomposed by the chain rule:

$$P(Y|X) = \prod_{t=1}^{L_t} P(y_t|y_{<t}, X) \tag{1}$$

where $y_{<t} = (y_1, \ldots, y_{t-1})$ denotes the previous translated sequence. The NMT directly models the conditional probability as:

$$P(y_t|y_{<t}, X) = \phi(H, z_{t-1}, e'_{y_{t-1}}) \tag{2}$$

where H is the source memory of X, z_{t-1} is the target hidden state at the decoding time step $t-1$, $e'_{y_{t-1}}$ is the target word embedding of the previous generated word, $\phi(\cdot)$ is the function of predicting the next target word.

Following the attention-based model presented in [1], we model H using a bidirectional recurrent neural network (bi-RNN) consisting of a forward RNN and a backward RNN [15] to represent a source sentence as a sequence of memory cells. More formally, $H = (h_1, \ldots, h_{L_s})$, where $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ is a memory cell constructed by the concatenation of the forward annotation vector $\overrightarrow{h}_i$ and the backward annotation vector $\overleftarrow{h}_i$:

$$\begin{aligned} \overrightarrow{h}_i &= \overrightarrow{f}(e_{x_i}, \overrightarrow{h}_{i-1}) \\ \overleftarrow{h}_i &= \overleftarrow{f}(e_{x_i}, \overleftarrow{h}_{i+1}) \end{aligned} \tag{3}$$

[1] Actually, the basic unit can be smaller than word, e.g. subword or character. We use word as the basic unit of source language in this paper.

where $\overrightarrow{f}(\cdot)$ and $\overleftarrow{f}(\cdot)$ are two gated recurrent units (GRUs) [2], e_{x_i} is the source word embedding of word x_i.

At the decoding time step t, the function ϕ generates the distribution of next target word using a conditional GRU[2]:

$$\phi(H, z_{t-1}, e'_{y_{t-1}}) \propto g(e'_{y_{t-1}}, z_t, c_t) \tag{4}$$

where $g(\cdot)$ is a two-layer feedforward neural network, and c_t is the context vector as the source condition linking up the encoder and the decoder. A two-layer GRU is used to calculate z_t. The first GRU layer produces the intermediate state $\tilde{z}_t$ based on the input $(z_{t-1}, e'_{y_{t-1}})$, and the second GRU layer produces the current state z_t with the input $(\tilde{z}_t, c_t)$. c_t is defined as a weighted sum of each cell in H:

$$c_t = \sum_{j=1}^{L_s} a_{t,j} * h_j \tag{5}$$

where $a_{t,j}$ is the alignment weight of the j-th source word and t-th target word. $a_{t,j}$ is normalized over $(x_1, \ldots, x_{L_s})$ with a single-layer feedforward neural network r:

$$a_{t,j} = \frac{exp\{r(\tilde{z}_t, h_j)\}}{\sum_{k=1}^{L_s} exp\{r(\tilde{z}_t, h_k)\}} \tag{6}$$

3 Source Segment Encoding

For the conventional attention model in NMT (as described in Sect. 2), every memory cell h_i makes a contribution to the context vector c_t by matching its own state h_i with the decoding state $\tilde{z}_t$, which is a case of content-based addressing [6]. Intuitively, more cells can produce more substantial context due to various views provided by different cells. However, for the standard model, the source memory is built by word-level cells and the number of memory cells is limited to be equal to the source words count. On the other hand, a segment corresponds to a block of memory cells in a memory network. Introducing segment also makes sense as a segment contains more hierarchical and structural information than an independent cell. As a result, in our approach, we extend the source memory by incorporating the segment-level cells.

Given a source sentence X, the set of all segments in X is $S(X) = \{X_i^j\}$, where $X_i^j = (x_i, \ldots, x_j)$, $1 \leq i \leq j \leq L_s$. Taking computing cost into consideration, we choose a subset $\tilde{S}(X) \subset S(X)$ to delegate the set of whole segments, where $|\tilde{S}(X)| = m$. Then we define a function $\psi(\cdot)$ to encode every segment $S_k \in \tilde{S}(X)$ into a vector s_k, where $|s_k| = |h_i| = 2d$, and d is the dimension of $\overrightarrow{h}_i$. Let $H = (h_1, \ldots, h_{L_s})$ be the baseline word-level cells as described in

[2] We follow the implementation in *dlmt*, referred to https://github.com/nyu-dl/dl4mt-tutorial/blob/master/docs/cgru.pdf.

Sect. 2, and $S = (s_1, \dots, s_m)$ be the segment-level cells generated by $\psi(\cdot)$, we concatenate all encodings together as the final source memory H^*:

$$\begin{aligned} H^* =& [H; S] \\ =& [(h_1, \dots, h_{L_s}); (s_1, \dots, s_m)] \end{aligned} \tag{7}$$

The advantage is that the number of the cells can be arbitrary, and is not restricted to the length of the source sequence. Note that as we do not change the model parameters of the original encoding, we can reuse the baseline model when training the SSE model. Then the attention model computes the alignment weights as usual but with the new source memory $H^* = (h_1^*, \dots, h_{L_s+m}^*)$[3]:

$$a_{t,j}^* = \frac{exp\{r(\tilde{z}_t, h_j^*)\}}{\sum_{k=1}^{L_s+m} exp\{r(\tilde{z}_t * h_k^*)\}} \tag{8}$$

where

$$c_t^* = \sum_{j=1}^{L_s+m} a_{t,j}^* * h_j^* \tag{9}$$

In the following we describe two methods for producing $S = (s_1, \dots, s_m)$.

3.1 N-Gram-Based SSE

In phrase-based Statistical Machine Translation [9], all source phrases limited by a max length are memorized explicitly in a big phrase table. When translating a sentence, the decoder accesses the table and then generates corresponding translations. Inspired by this, a direct way to construct $\tilde{S}(X)$ is to use a variety of off-the-shelf n-grams. More specifically, given the order of n-gram (i.e., n), we can generate all possible segments $\tilde{S}_{ng}(X) = \{x_i^j\}$ subject to $j - i + 1 \leq n$.

Next, the core problem is how to represent a segment. Instead of encoding a segment using the last hidden state of recurrent neural network [2],

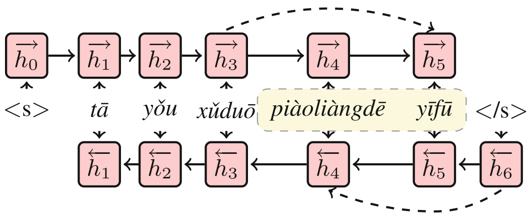

Fig. 1. An example of RNN-MINUS for encoding a segment. The dotted rounded rectangle denotes the encoded segment. <s> and </s> represent the beginning and ending of a sentence respectively, with dummy vectors.

[3] We share the same model parameters for aligning the word-level cells and the segment-level cells. Independent parameters may bring further improvement, which needs our further investigation.

we follow the span encoding in syntactic parsing [3,21]. Here we refer to this method as RNN-MINUS. Given the bidirectional RNN encoding $H = \{[\overrightarrow{h}_1; \overleftarrow{h}_1], \ldots, [\overrightarrow{h}_{L_s}; \overleftarrow{h}_{L_s}]\}$, RNN-MINUS encodes a segment x_i^j as:

$$\psi_{ng}(x_i^j) = [\overrightarrow{h}_j - \overrightarrow{h}_{i-1}; \overleftarrow{h}_i - \overleftarrow{h}_{j+1}] \tag{10}$$

For the beginning and ending of the sequence, we add dummy vectors $\overrightarrow{h}_0 = 0$ and $\overleftarrow{h}_{L_s+1} = 0$ to make Eq. (10) feasible. See Fig. 1 for the RNN-MINUS encoding of an example sequence.

The idea behind RNN-MINUS is simple: assuming that the information before entering a segment is I_s (i.e., information of $\{x_1, \ldots, x_{s-1}\}$), and the information after passing through this segment is I_e (i.e., information of $\{x_1, \ldots, x_e\}$), then we can regard the information offered by this segment as $I_e - I_s$. In NMT, we generally regard the hidden state of bidirectional RNN in each time step as the corresponding encoded information. Therefore, for segment x_i^j, the left-to-right information offered by the forward RNN can be represented as $\overrightarrow{h}_j - \overrightarrow{h}_{i-1}$. Likewise, the right-to-left information for x_i^j is represented as $\overleftarrow{h}_i - \overleftarrow{h}_{j+1}$.

It is worth noting that although RNN-MINUS generates encodings for each individual segment, it is still context dependent. As $[\overrightarrow{h}_i; \overleftarrow{h}_i]$ encodes the left and right contexts of the position i, the value of the subtraction of these vectors may vary in different contexts. In other words, the same segment can have different representations when the surrounding context changes. In addition, RNN-MINUS is based on the existing encoding representations with no increase in model size.

3.2 Joint-Learning-Based SSE

The n-gram-based SSE is straightforward but the size of used segments scales linearly with the order of n. In consequence, the encoding of segments consumes more memory space (especially for GPU) and slows down the system. As a result, we propose to an end-to-end joint learning of both source segmentation model and translation model, which can learn a *latent and optimal* segmentation of a source sentence rather than accessing all possible segments.

To determine a segment, we define two tags B and M for each position in the source sequence like [14], where B denotes the beginning of a segment, and M denotes the case of the middle. Then we build an identifier layer on top of the bi-RNN encoder to estimate the probability of the identity tag of each position, which can be regarded as a sequence labeling problem with two tags in each position. In this work we model this problem using a uni-directional GRU layer $Layer_{gru}$ followed by a two-layer feedforward neural network $Layer_{fnn}$. Figure 2 illustrates the network architecture of the identifier layer. The final result is a scalar denoting the probability of B for the position j, which is computed as:

$$\begin{aligned} P(B|j) &= sigmoid(W_1 * o_j + b_1) \\ o_j &= tanh(W_2 * v_j + b_2) \\ v_j &= gru(h_j, v_{j-1}) \end{aligned} \tag{11}$$

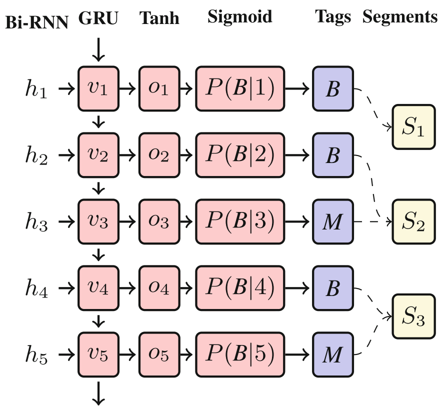
Bi-RNN
GRU
Tanh
Sigmoid
Tags
Segments
h_1 v_1 o_1 $P(B|1)$ B
h_2 v_2 o_2 $P(B|2)$ B
h_3 v_3 o_3 $P(B|3)$ M
h_4 v_4 o_4 $P(B|4)$ B
h_5 v_5 o_5 $P(B|5)$ M
S_1
S_2
S_3

Fig. 2. An example of learning latent segments in the identifier layer. The tag of each position is induced from existing encoding representations by bi-RNN. S denotes the learned segments. In this case, our model compresses the source sequence with 5 words into 3 segments.

where $o_j \in \mathbb{R}^{d_i \times 1}$, $v_j \in \mathbb{R}^{d_i \times 1}$ are the outputs of the first layer of $Layer_{fnn}$ and $Layer_{gru}$ respectively. d_i is the dimension of the hidden states in the identifier layer. $W_1 \in \mathbb{R}^{1 \times d_i}$, $b_1 \in \mathbb{R}^1$, $W_2 \in \mathbb{R}^{d_i \times d_i}$, $b_2 \in \mathbb{R}^{d_i}$ are model parameters in $Layer_{fnn}$. $P(M|j)$ can be obtained by $1 - P(B|j)$. Then, we infer the tag for each position as follows:

$$T(j) = \begin{cases} B \ j = 1 \\ B \ j \neq 1, P(B|j) \geq P(M|j) \\ M \ j \neq 1, P(B|j) < P(M|j) \end{cases} \tag{12}$$

To prevent the sequence of illegal tags, we always assign B to the beginning position. After having the tag for each position, we can take every word sequence between two Bs as an identified segment. For example, the tag sequence $(B_1, B_2, M_3, M_4, B_5)$ contains 3 segments $(1, 1), (2, 4), (5, 5)$. Obviously, the obtained segments essentially define a segmentation of the sentence.

However, unfortunately, we cannot use RNN-MINUS directly to represent the learned segments as the increased model parameters in identifier layer are not reachable during back-propagation[4]. To learn these parameters, we follow the idea used in the local attention model [12]. We explicitly make these parameters

[4] These parameters control the decisions of segmentation, and are not differentiable with respect to the loss of translation, which is a similar problem to hard attention model [24]. The hard attention model picks up a determined source word to align, whereas our model chooses a determined segmentation of source sentence.

part of the translation model, and define the encoding of x_i^j with its boundary confidence, as follows:

$$\begin{aligned}\psi_{jl}(x_i^j) &= [\beta \overrightarrow{h}_j - \alpha \overrightarrow{h}_{i-1}; \alpha \overleftarrow{h}_i - \beta \overleftarrow{h}_{j+1}] \\ \alpha &= P(B|i) \\ \beta &= P(M|j)\end{aligned} \tag{13}$$

In this model, α and β are the left-boundary confidence and the right-boundary confidence of a segment respectively. By using the encoding method defined in Eq. 13, the increased parameters as part of the model can be learned in the standard back-propagation procedure. The boundary confidence also can be seen as a special case of dropout [10], which can alleviate the segmentation errors to some extent and improve the robustness of our segmentation model.

4 Experiments

4.1 Setup

We evaluated our proposed approach on word-based Chinese→English translation task. We used part bitext provided within NIST12 OpenMT[5] and we chose NIST 2006 (MT06) as the validation set, and 2004 (MT04), 2005 (MT05), 2008 (MT08) as the test sets. All the sentences of more than 50 words were filtered out. Data on both sides was tokenized by an in-house implement, where the Chinese-side data was segmented based on n-gram language model. The resulting training data consisted of 1.85M sentence pairs. We limited the vocabularies to the most frequent 30K words in Chinese and English. All the out-of-vocabulary words were replaced with <*UNK*>.

Table 1. Translation results (BLEU score) on Chinese→English tasks. *WC* denotes the standard encoding of word-level cells, while *SC-ngram* and *SC-joint* denotes segment-level cells using our n-gram-based SSE and joint-learning-based SSE respectively.

		Valid.	Test			
♯	System	MT06	MT04	MT05	MT08	Ave.
1	PBSMT	32.09	36.65	31.30	25.99	31.31
2	NMT baseline (WC)	36.88	43.14	36.02	29.57	36.24
3	+ SC-ngram (n=1)	38.31	44.55	37.20	30.15	37.30
4	+ SC-ngram (n=4)	38.48	44.67	37.58	30.28	37.51
5	+ SC-joint	**39.26**	**45.69**	**38.17**	**31.19**	**38.35**
6	SC-ngram (n=1)	37.24	43.65	35.73	29.56	36.31
7	SC-ngram (n=4)	38.67	44.95	37.81	30.63	37.80
8	SC-joint	20.39	23.32	18.26	14.59	18.72

[5] LDC2000T46, LDC2000T47, LDC2000T50, LDC2003E14, LDC2005T10, LDC2002E18, LDC2007T09, LDC2004T08.

The sizes of both source and target word embedding were set to 512. We set $d = 1024$ and $d_i = 100$ respectively. We followed [17] and used the same dropout mask at each time step, with the dropout probability of 0.1 for full words and 0.2 for other layers. We trained all the NMT models using stochastic gradient algorithm Adadelta [25] with mini-batch size of 80. The baseline NMT models were tuned for 10 epochs and then finetuned by fixing the both source and target embeddings for 10 epochs. Our models were further tuned and finetuned based on the well-tuned baseline NMT model. At test time, we employed beam search with the beam size of 12. All the translation results were evaluated by case-insensitive BLEU-4 metric using *mteval-v13a.pl*. In our experiments, we compared our method with two baselines learned on the same bilingual training data. One is the phrase-based system provided within the NiuTrans open-source SMT toolkit [23]. The other is a standard attention-based NMT system using bidirectional RNN as encoder.

5 Results and Analysis

5.1 Evaluation of Translations

Table 1 shows the BLEU scores in different settings. We can see that all the NMT systems benefit from the combination of our segment-level cells (♯3–5). It confirms that explicitly incorporating bigger linguistic units in encoding helps. This result also agrees with the findings in [4].

In particular, the best result is obtained by combining the word-level cells with the joint-learning-based SSE (♯5), which yields an average improvement of 2.1+ BLEU points than the NMT baseline (♯2). It suggests that learning the segmentation model along with the translation process jointly is effective, even without any supervisions of segmentation. The segmentation errors caused by our segmentation model do not present heavy hurt for translation performance.

Compared row 4 with row 5, the joint-learning-based SSE is more effective than simply arranging all the possible n-grams. More interestingly, we find that if we only use the segment-level cells in joint-learning-based SSE (♯8), the translation performance will decrease dramatically. The reason could be that the segment-level cells are sketchy representations of the source sentence, while the more concrete representations are contained in word-level cells.

Consider n-gram-based SSE, when $n = 1$, using the combination of word-level cells with segment-level cells (♯3) outperforms approximate 1.0 BLEU point than using segment-level cells alone (♯6). But it is worth noting that using independent segment-level cells of $n = 1$ (♯6) obtains an almost identical performance compared to the baseline (♯2). A possible explanation is that the encoding of segment with length 1 is different from conventional word-level cells. That is, the segment is represented by the subtraction of adjacent states and the resulting can put more emphasis on the meaning of the independent word. However, the explicit meaning of word is ambiguous in standard encoding procedures as the bidirectional RNN gives a global meaning in each position.

To our surprise, the superiority of combination disappears when $n = 4$ (♯4 vs. ♯7). It seems that the word-level cells do not play their part and are drowned when mixed with bigger segments. It indicates that using context-sensitive local encoding is comparable to global encoding based on RNN. This finding is consistent with the result in [5]. [5] use a convolution neural network with position embedding, which can be seen as a case of context-sensitive local encoding.

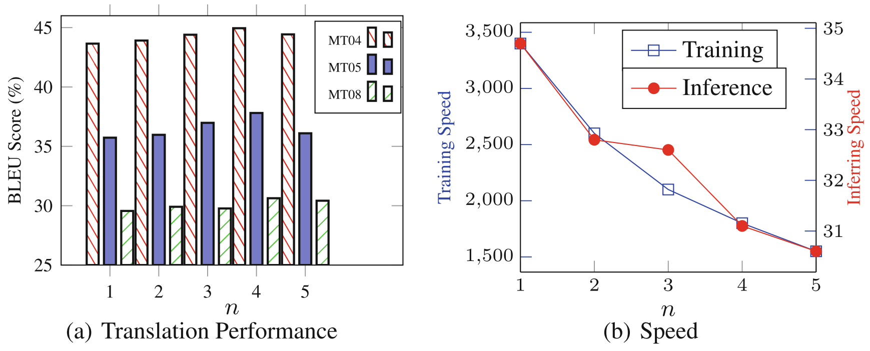

Fig. 3. Translation performance and Speed (words/second) with different n using n-gram-based SSE in Chinese→English task. Average length of source sentence in test set for inference is 24 words.

5.2 Impact of n

Figure 3(a) shows the BLEU scores on all the test sets along with different settings of n. We only compare the segment-level cells in order to eliminate the effect of the standard word-level cells. It is obvious that BLEU is improved as n increases. The best result is achieved when $n = 4$, with sharp decreases as n grows bigger. The largest gap is 1.49 BLEU points in all test tests averagely ($n = 4$ vs. $n = 1$). It is an evidence that the translation model can generate better translation results by observing more source contexts. It is also consistent with our intuition that learning and memorizing more source segments are important in the translation process. However, when n is too large ($n > 4$ in our experiments), BLEU starts to drop sharply meaning that current model is saturated and can not benefit from more cells. We also plot the system speed as a function of n (Fig. 3(b)). As expected, a choice of larger n slows down the system. Together with the BLEU results in Fig. 3(a), it suggests that choosing n around 3 is optimal for trade-off of BLEU improvement and speed decrease.

5.3 Samples of Learned Segments

Table 2 presents three samples of the learned source segments by our joint-learning-based SSE in Chinese→English task. An interesting finding is that our

Table 2. Samples of learned source segments in Chinese→English translation task. A red rectangle denotes a segment.

♯	Source Sentences		
1	*zhōngguó*	pàituán fùměi cǎigòu	*èrshíduōyì měiyuán gāokējì shèbèi*
2	*míngnián*	rìběn jīngjì zēngzhǎng sùdù	*kěnéng fàng huǎn*
3	*wèishēngbù : quánguó cānyǐnyè*	*jiāng zhúbù shíxíng*	*wèishēng jiāndū gōngshì zhìdù*

model appears capable of capturing the positions of subject, verb and temporal adverbial. For example, "*zhōngguó*" ("China" in English), "*rìběn jīngjì zēngzhǎng sùdù*" ("the growth rate of Japanese economic" in English) is the subjects of the 1st and 2nd samples respectively; "*cǎigòu*" ("purchase" in English), "*shīxíng*" ("implement" in English) is the verbs of the 1st and 3rd samples respectively; "*míngnián*" ("next year" in English) is the temporal adverbial of the 2nd sample. Note that the segmentation model is trained without any supervisions of segmented sentences, and learned absolutely from the translation procedures.

6 Related Work

The essence of our model is to improve the original encoder network. Apart from those syntax-enhancement methods as introduced in Sect. 1 [4,7,13,16], multi-task learning and deeper network can also enhance the representational power of the encoder.

In multi-task learning framework, the encoder network is shared in different tasks, which can benefit from joint objective function. The translation model can be trained with the source syntax parsing task in [11], or with the source reorder task in [26]. Both these methods require other external resources, such as human-annotated treebanks or source-side monolingual data, whereas our model only needs bitext data for translation model.

Deeper encoder network models have also been successfully employed in NMT. [27] introduce the fast-forward connections method to train a deep Long Short-Term Memory (LSTM) network (18 LSTM layers) as the encoder. Also, [5] propose a deep convolutional encoder with source position embedding. These models have a high cost for training and inference, whereas our model is light and easy to be implemented. It is worth noting that [5] also can be seen as a case of modeling segment in some sense due to the local filters in convolution neural network. However, our model is still based on recurrent neural network.

Another related work is [20], which apply a pre-prepared phrase table to label the source phrases and can directly generate a target phrase at one step. By contrast, our approach learns all segments without any supervision.

7 Conclusion

In this paper, we propose two simple yet effective methods to explicitly model the source segments in the encoder of attention-based NMT. In the first method, we directly encode off-the-shelf n-grams of the source sentence into source memory. In the second method, we jointly learn a segmentation model with translation model in the end-to-end manner. Both of the methods require no external resources (e.g. segmented sentences). Experimental results on the word-based Chinese-to-English translation task show that our method outperforms the baseline significantly. It is observed that using larger linguistic unit helps and gives further improvements on top of the word-based NMT system. In addition, we give an evidence that context-sensitive local encoding is comparable to global encoding based on recurrent neural network. Also, we present that the automatically learned segmentation model is sensitive to some key constituents of a sentence (e.g. subject, verb, temporal adverbial) in some cases.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the 3rd International Conference on Learning Representations (2015)
2. Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734. Association for Computational Linguistics (2014)
3. Cross, J., Huang, L.: Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1–11. Association for Computational Linguistics (2016)
4. Eriguchi, A., Hashimoto, K., Tsuruoka, Y.: Tree-to-sequence attentional neural machine translation. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 823–833. Association for Computational Linguistics (2016)
5. Gehring, J., Auli, M., Grangier, D., Dauphin, Y.N.: A convolutional encoder model for neural machine translation. arXiv preprint arXiv:1611.02344 (2016)
6. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
7. Hashimoto, K., Tsuruoka, Y.: Neural machine translation with source-side latent graph parsing. arXiv preprint arXiv:1702.02265 (2017)
8. Kim, Y., Denton, C., Hoang, L., Rush, A.M.: Structured attention networks. arXiv preprint arXiv:1702.00887 (2017)
9. Koehn, P., Och, F.J., Marcu, D.: Statistical phrase-based translation. In: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1, pp. 48–54. Association for Computational Linguistics (2003)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)

11. Luong, M.T., Le, Q.V., Sutskever, I., Vinyals, O., Kaiser, L.: Multi-task sequence to sequence learning. arXiv preprint arXiv:1511.06114 (2015)
12. Luong, T., Pham, H., Manning, D.C.: Effective approaches to attention-based neural machine translation. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412–1421. Association for Computational Linguistics (2015)
13. Nadejde, M., Reddy, S., Sennrich, R., Dwojak, T., Junczys-Dowmunt, M., Koehn, P., Birch, A.: Syntax-aware neural machine translation using CCG. arXiv preprint arXiv:1702.01147 (2017)
14. Peng, F., Feng, F., McCallum, A.: Chinese segmentation and new word detection using conditional random fields. In: COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics (2004)
15. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Sig. Process. **45**(11), 2673–2681 (1997)
16. Sennrich, R., Haddow, B.: Linguistic input features improve neural machine translation. In: Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers, pp. 83–91. Association for Computational Linguistics (2016)
17. Sennrich, R., Haddow, B., Birch, A.: Edinburgh neural machine translation systems for WMT 16. In: Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers, pp. 371–376. Association for Computational Linguistics (2016)
18. Sukhbaatar, S., Weston, J., Fergus, R., et al.: End-to-end memory networks. In: Advances in Neural Information Processing Systems, pp. 2440–2448 (2015)
19. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
20. Tang, Y., Meng, F., Lu, Z., Li, H., Yu, P.L.: Neural machine translation with external phrase memory. arXiv preprint arXiv:1606.01792 (2016)
21. Wang, W., Chang, B.: Graph-based dependency parsing with bidirectional LSTM. In: Proceedings of ACL, vol 1, pp. 2306–2315 (2016)
22. Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al.: Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144 (2016)
23. Xiao, T., Zhu, J., Zhang, H., Li, Q.: NiuTrans: an open source toolkit for phrase-based and syntax-based machine translation. In: Proceedings of the ACL 2012 System Demonstrations, pp. 19–24. Association for Computational Linguistics (2012)
24. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A.C., Salakhutdinov, R., Zemel, R.S., Bengio, Y.: Show, attend and tell: neural image caption generation with visual attention. In: ICML, vol. 14, pp. 77–81 (2015)
25. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)
26. Zhang, J., Zong, C.: Exploiting source-side monolingual data in neural machine translation. In: Proceedings of EMNLP (2016)
27. Zhou, J., Cao, Y., Wang, X., Li, P., Xu, W.: Deep recurrent models with fast-forward connections for neural machine translation. Trans. Assoc. Comput. Linguist. **4**, 371–383 (2016)