Encoding Source Language with Convolutional Neural Network for Machine Translation

Fandong Meng[1] Zhengdong Lu[2] Mingxuan Wang[1] Hang Li[2] Wenbin Jiang[1] Qun Liu[3,1]
[1]Institute of Computing Technology, Chinese Academy of Sciences
{mengfandong,wangmingxuan,jiangwenbin,liuqun}@ict.ac.cn
[2]Noah's Ark Lab, Huawei Technologies
{Lu.Zhengdong,HangLi.HL}@huawei.com
[3]Centre for Next Generation Localisation, Dublin City University

Abstract

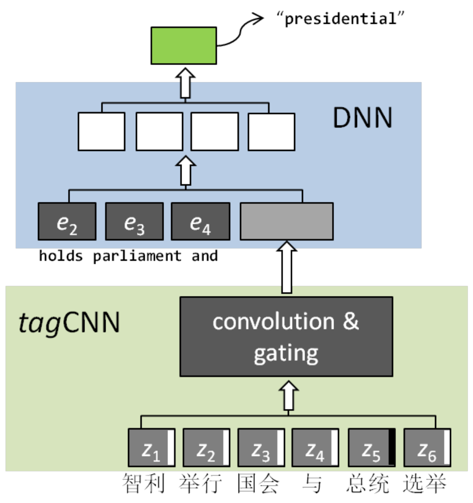
The recently proposed neural network joint model (NNJM) (Devlin et al., 2014) augments the n-gram target language model with a heuristically chosen source context window, achieving state-of-the-art performance in SMT. In this paper, we give a more systematic treatment by summarizing the relevant source information through a convolutional architecture guided by the target information. With different guiding signals during decoding, our specifically designed convolution+gating architectures can pinpoint the parts of a source sentence that are relevant to predicting a target word, and fuse them with the context of entire source sentence to form a unified representation. This representation, together with target language words, are fed to a deep neural network (DNN) to form a stronger NNJM. Experiments on two NIST Chinese-English translation tasks show that the proposed model can achieve significant improvements over the previous NNJM by up to +1.01 BLEU points on average.

1 Introduction

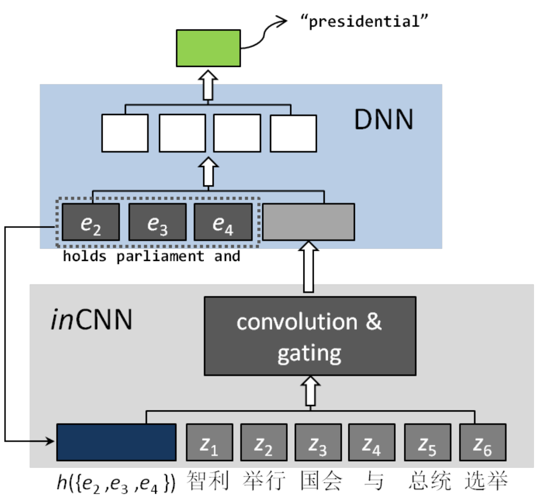
Learning of continuous space representation for source language has attracted much attention in both traditional statistical machine translation (SMT) and neural machine translation (NMT). Various models, mostly neural network-based, have been proposed for representing the source sentence, mainly as the encoder part in an encoder-decoder framework (Bengio et al., 2003; Auli et al., 2013; Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014). There has been some quite recent work on encoding only "relevant" part of source sentence during the decoding process, most notably neural network joint model (NNJM) in (Devlin et al., 2014), which extends the n-grams target language model by additionally taking a fixed-length window of source sentence, achieving state-of-the-art performance in statistical machine translation.

In this paper, we propose novel convolutional architectures to dynamically encode the relevant information in the source language. Our model covers the entire source sentence, but can effectively find and properly summarize the relevant parts, guided by the information from the target language. With the guiding signals during decoding, our specifically designed convolution architectures can pinpoint the parts of a source sentence that are relevant to predicting a target word, and fuse them with the context of entire source sentence to form a unified representation. This representation, together with target words, are fed to a deep neural network (DNN) to form a stronger NNJM. Since our proposed joint model is purely lexicalized, it can be integrated into any SMT decoder as a feature.

Two variants of the joint model are also proposed, with coined name tagCNN and inCNN, with different guiding signals used from the decoding process. We integrate the proposed joint models into a state-of-the-art dependency-to-string translation system (Xie et al., 2011) to evaluate their effectiveness. Experiments on NIST Chinese-English translation tasks show that our model is able to



(a) tagCNN


"presidential"
DNN
e_2 e_3 e_4
holds parliament and
inCNN
convolution & gating
z_1 z_2 z_3 z_4 z_5 z_6
$h(\{e_2, e_3, e_4\})$ 智利 举行 国会 与 总统 选举

(b) inCNN

Figure 1: Illustration for joint LM based on CNN encoder.

achieve significant improvements of +1.93 BLEU points on average over the baseline. Our model also outperforms Devlin et al. (2014)'s NNJM by up to +1.01 BLEU points.

RoadMap: In the remainder of this paper, we start with a brief overview of joint language model in Section 2, while the convolutional encoders, as the key component of which, will be described in detail in Section 3. Then in Section 4 we discuss the decoding algorithm with the proposed models. The experiment results are reported in Section 5, followed by Section 6 and 7 for related work and conclusion.

2 Joint Language Model

Our joint model with CNN encoders can be illustrated in Figure 1 (a) & (b), which consists 1) a CNN encoder, namely tagCNN or inCNN, to represent the information in the source sentences, and 2) an NN-based model for predicting the next words, with representations from CNN encoders and the history words in target sentence as inputs.

In the joint language model, the probability of the target word $\mathbf{e}_n$, given previous k target words $\{\mathbf{e}_{n-k}, \cdots, \mathbf{e}_{n-1}\}$ and the representations from CNN-encoders for source sentence S are

$$\begin{aligned} tag\text{CNN:} \quad & p(\mathbf{e}_n | \phi_1(S, \text{index of tagged word})) \\ in\text{CNN:} \quad & p(\mathbf{e}_n | \, \phi_2(S, h(\{\mathbf{e}\}_{n-k}^{n-1}), \, \{\mathbf{e}\}_{n-k}^{n-1}), \end{aligned}$$

where $\phi_1(S, \text{index of tagged word})$ stands for the representation given by tagCNN with the set of indexes of source words aligned to the target word $\mathbf{e}_n$, and $\phi_2(S, h(\{\mathbf{e}\}_{n-k}^{n-1}))$ stands for the representation from inCNN with the attention signal $h(\{\mathbf{e}\}_{n-k}^{n-1})$.

Let us use the example in Figure 1, where the task is to translate the Chinese sentence

Chinese:	智利	举行	国会	与	总统	选举
Pinyin:	Zhìlì	Jǔxíng	Guóhuì	Yǔ	Zǒngtǒng	Xuǎnjǔ

into English. In evaluating a target language sequence "`holds parliament and presidential`", with "`holds parliament and`" as the proceeding words (assume 4-gram LM), and the affiliated source word[1] of "`presidential`" being "`Zǒngtǒng`" (determined by word alignment), tagCNN generates $\phi_1(S, \{4\})$ (the index of "`Zǒngtǒng`" is 4), and inCNN generates $\phi_2(S, h(\texttt{holds parliament and}))$. The DNN component then takes "`holds parliament and`" and (ϕ_1 or ϕ_2) as input to give the conditional probability for next word, e.g., $p(\texttt{"presidential"}|\phi_{1|2})$.

3 Convolutional Models

We start with the generic architecture for convolutional encoder, and then proceed to tagCNN and inCNN as two extensions.

3.1 Generic CNN Encoder

The basic architecture is of a generic CNN encoder is illustrated in Figure 2 (a), which has a fixed architecture consisting of six layers:

Layer-0: the input layer, which takes words in the form of embedding vectors. In our work, we set the maximum length of sentences to 40 words. For sentences shorter than that, we put zero padding at the beginning of sentences.

Layer-1: a convolution layer after Layer-0, with window size = 3. As will be discussed in Section 3.2 and 3.3, the guiding signal are injected into this layer for "guided version".

Layer-2: a local gating layer after Layer-1, which simply takes a weighted sum over feature-maps in non-adjacent window with size = 2.

Layer-3: a convolution layer after Layer-2, we perform another convolution with window size = 3.

Layer-4: we perform a global gating over feature-maps on Layer-3.

Layer-5: fully connected weights that maps the output of Layer-4 to the this layer as the final representation.

3.1.1 Convolution

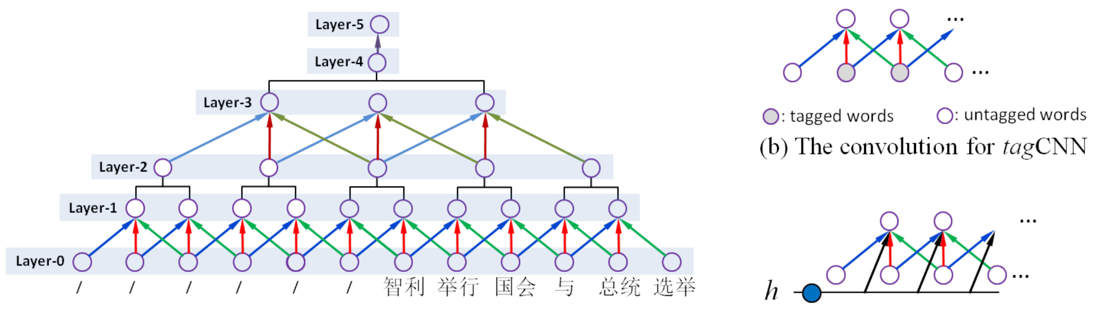
As shown in Figure 2 (a), the convolution in Layer-1 operates on sliding windows of words (width k_1), and the similar definition of windows carries over to higher layers. Formally, for source sentence input $\mathbf{x}=\{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$, the convolution unit for feature map of type-f (among F_ℓ of them) on Layer-ℓ is

$$z_i^{(\ell,f)}(\mathbf{x}) = \sigma(\mathbf{w}^{(\ell,f)}\hat{\mathbf{z}}_i^{(\ell-1)} + b^{(\ell,f)}), \quad \ell = 1,3, \quad f = 1,2,\cdots,F_\ell \quad (1)$$

where

- $z_i^{(\ell,f)}(\mathbf{x})$ gives the output of feature map of type-f for location i in Layer-ℓ;
- $\mathbf{w}^{(\ell,f)}$ is the parameters for f on Layer-ℓ;
- $\sigma(\cdot)$ is the Sigmoid activation function;

[1]For an aligned target word, we take its aligned source words as its affiliated source words. And for an unaligned word, we inherit its affiliation from the closest aligned word, with preference given to the right (Devlin et al., 2014). Since the word alignment is of many-to-many, one target word may has multi affiliated source words.



(a) The generic architecture for CNN encoder

(b) The convolution for *tag*CNN

(c) The convolution for *in*CNN

Figure 2: Illustration for the CNN encoders.

- $\hat{\mathbf{z}}_i^{(\ell-1)}$ denotes the segment of Layer-ℓ−1 for the convolution at location i , while

$$\hat{\mathbf{z}}_i^{(0)} \overset{\text{def}}{=} [\mathbf{x}_i^\top, \ \ \mathbf{x}_{i+1}^\top, \ \ \mathbf{x}_{i+2}^\top]^\top$$

  concatenates the vectors for 3 words from sentence input $\mathbf{x}$.

3.1.2 Gating

Previous CNNs, including those for NLP tasks (Hu et al., 2014; Kalchbrenner et al., 2014), take a straightforward convolution-pooling strategy, in which the "fusion" decisions (e.g., selecting the largest one in max-pooling) are based on the values of feature-maps. This is essentially a soft template matching, which works for tasks like classification, but harmful for keeping the composition functionality of convolution, which is critical for modeling sentences. In this paper, we propose to use separate gating unit to release the score function duty from the convolution, and let it focus on composition.

We take two types of gating: 1) for Layer-2, we take a local gating with non-overlapping windows (size = 2) on the feature-maps of convolutional Layer-1 for representation of segments, and 2) for Layer-4, we take a global gating to fuse all the segments for a global representation. We found that this gating strategy can considerably improve the performance of both *tag*CNN and *in*CNN over pooling.

- **Local Gating:** On Layer-1, for every gating window, we first find its original input (before convolution) on Layer-0, and merge them for the input of the gating network. For example, for the two windows: word (3,4,5) and word (4,5,6) on Layer-0, we use concatenated vector consisting of embedding for word (3,4,5,6) as the input of the local gating network (a logistic regression model) to determine the weight for the convolution result of the two windows (on Layer-1), and the weighted sum are the output of Layer-2.

- **Global Gating:** On Layer-3, for feature-maps at each location i, denoted $\mathbf{z}_i^{(3)}$, the global gating network (essentially soft-max,parameterized $\mathbf{w}_g$), assigns a normalized weight

$$\omega(\mathbf{z}_i^{(3)}) = e^{\mathbf{w}_g^\top \mathbf{z}_i^{(3)}} / \sum_j e^{\mathbf{w}_g^\top \mathbf{z}_j^{(3)}},$$

  and the gated representation on Layer-4 is given by the weighted sum $\sum_i \omega(\mathbf{z}_i^{(3)})\mathbf{z}_i^{(3)}$.

3.1.3 Training of CNN encoders

The CNN encoders, including tagCNN and inCNN that will be discussed right below, are trained in a joint language model described in Section 2, along with the following parameters

- the embedding of the words on source and the proceeding words on target;
- the parameters for the DNN of joint language model, include the parameters of soft-max for word probability.

The training procedure is identical to that of neural network language model, except that the parallel corpus is used instead of a monolingual corpus. We seek to maximize the log-likelihood of training samples, with one sample for every target word in the parallel corpus. Optimization is performed with the conventional back-propagation, implemented as stochastic gradient descent (LeCun et al., 1998) with mini-batches.

3.2 tagCNN

tagCNN inherits the convolution and gating from generic CNN (as described in Section 3.1), with the only modification in the input layer. As shown in Figure 2 (b), in tagCNN, we append an extra tagging bit (0 or 1) to the embedding of words in the input layer to indicate whether it is one of affiliated words

$$\mathbf{x}_i^{(\text{AFF})} = [\mathbf{x}_i^\top \; 1]^\top, \quad \mathbf{x}_j^{(\text{NON-AFF})} = [\mathbf{x}_j^\top \; 0]^\top.$$

Those extended word embedding will then be treated as regular word-embedding in the convolutional neural network. This particular encoding strategy can be extended to embed more complicated dependency relation in source language, as will be described in Section 5.4.

This particular "tag" will be activated in a parameterized way during the training for predicting the target words. In other words, the supervised signal from the words to predict will find, through layers of back-propagation, the importance of the tag bit in the "affiliated words" in the source language, and learn to put proper weight on it to make tagged words stand out and adjust other parameters in tagCNN accordingly for the optimal predictive performance. In doing so, the joint model can pinpoint the parts of a source sentence that are relevant to predicting a target word through the already learned word alignment.

3.3 inCNN

Unlike tagCNN, which directly tells the location of affiliated words to the CNN encoder, inCNN sends the information about the proceeding words in target side to the convolutional encoder to help retrieve the information relevant for predicting the next word. This is essentially a particular case of attention model, analogous to the automatic alignment mechanism in (Bahdanau et al., 2014), where the attention signal is from the state of a generative recurrent neural network (RNN) as decoder.

Basically, the information from proceeding words, denoted as $h(\{\mathbf{e}\}_{n-k}^{n-1})$, is injected into every convolution window in the source language sentence, as illustrated in Figure 2 (c). More specifically, for the window indexed by t, the input to convolution is given by the concatenated vector

$$\hat{\mathbf{z}}_t = [h(\{\mathbf{e}\}_{n-k}^{n-1}),\; \mathbf{x}_t^\top,\; \mathbf{x}_{t+1}^\top,\; \mathbf{x}_{t+2}^\top]^\top.$$

In this work, we simply use the vectors concatenated from word-embedding for words $\{\mathbf{e}_{n-k} \cdots, \mathbf{e}_{n-k}\}$ as $h(\{\mathbf{e}\}_{n-k}^{n-1})$, although $h(\cdot)$ can be non-trivially realized with another neural network. Through layers of convolution and gating, inCNN can 1) retrieve the relevant segments of

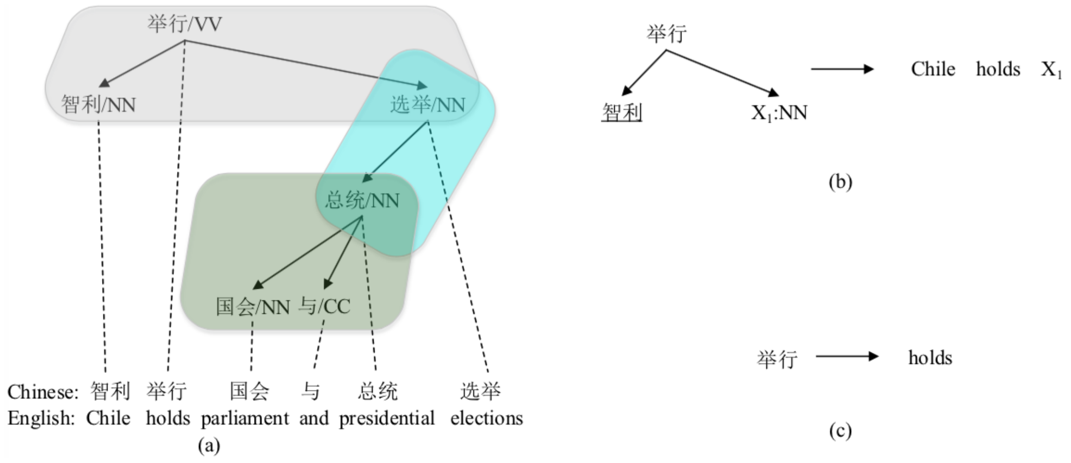

举行/VV
智利/NN
选举/NN
总统/NN
国会/NN
与/CC
Chinese: 智利 举行 国会 与 总统 选举
English: Chile holds parliament and presidential elections
(a)
举行
智利
X1:NN
Chile holds X1
(b)
举行
holds
(c)

Figure 3: Illustration for a dependency tree (a) with three head-dependents relations in shadow, an example of head-dependents relation rule (b) for the top level of (a), and an example of head rule (c). "X_1:NN" indicates a substitution site that can be replaced by a subtree whose root has part-of-speech "NN". The underline denotes a leaf node.

source sentences, and 2) compose and transform the retrieved segments into representation recognizable by the DNN in predicting the words in target language. Different from that of tagCNN, inCNN uses information from proceeding words, hence provides complementary information in the augmented joint language model of tagCNN. This has been empirically verified when using feature based on tagCNN and that based on inCNN in decoding with greater improvement.

4 Decoding with the Joint Model

Our joint model is purely lexicalized, and therefore can be integrated into *any* SMT decoders as a feature. For a hierarchical SMT decoder, we adopt the integrating method proposed by Devlin et al. (2014). As inherited from the n-gram language model for performing hierarchical decoding, the leftmost and rightmost $n-1$ words from each constituent should be stored in the state space. We extend the state space to also include the indexes of the affiliated source words for each of these edge words. For an aligned target word, we take its aligned source words as its affiliated source words. And for an unaligned word, we use the affiliation heuristic adopted by Devlin et al. (2014). In this paper, we integrate the joint model into the state-of-the-art dependency-to-string machine translation decoder as a case study to test the efficacy of our proposed approaches. We will briefly describe the dependency-to-string translation model and then the description of MT system.

4.1 Dependency-to-String Translation

In this paper, we use a state-of-the-art dependency-to-string (Xie et al., 2011) decoder (Dep2Str), which is also a hierarchical decoder. This dependency-to-string model employs rules that represent the source side as head-dependents relations and the target side as strings. A head-dependents relation (HDR) is composed of a head and all its dependents in dependency trees. Figure 3 shows a dependency tree (a) with three HDRs (in shadow), an example of HDR rule (b) for the top level of (a), and an example of head rule (c). HDR rules are constructed from head-dependents relations. HDR rules can act as both translation rules and reordering rules. And head rules are used for translating source words.

We adopt the decoder proposed by Meng et al. (2013) as a variant of Dep2Str translation that is easier to implement with comparable performance. Basically they extract the HDR rules with GHKM (Galley et al., 2004) algorithm. For the decoding procedure, given a source dependency tree T, the decoder transverses T in post-order. The bottom-up chart-based decoding algorithm with cube pruning (Chiang, 2007; Huang and Chiang, 2007) is used to find the k-best items for each node.

4.2 MT Decoder

Following Och and Ney (2002), we use a general loglinear framework. Let d be a derivation that convert a source dependency tree into a target string e. The probability of d is defined as:

$$P(d) \propto \prod_i \phi_i(d)^{\lambda_i} \tag{2}$$

where ϕ_i are features defined on derivations and λ_i are the corresponding weights. Our decoder contains the following features:

Baseline Features:

- translation probabilities $P(t|s)$ and $P(s|t)$ of HDR rules;
- lexical translation probabilities $P_{\text{LEX}}(t|s)$ and $P_{\text{LEX}}(s|t)$ of HDR rules;
- rule penalty $\exp(-1)$;
- pseudo translation rule penalty $\exp(-1)$;
- target word penalty $\exp(|e|)$;
- n-gram language model $P_{\text{LM}}(e)$;

Proposed Features:

- n-gram tagCNN joint language model $P_{\text{TLM}}(e)$;
- n-gram inCNN joint language model $P_{\text{ILM}}(e)$.

Our baseline decoder contains the first eight features. The pseudo translation rule (constructed according to the word order of a HDR) is to ensure the complete translation when no matched rules is found during decoding. The weights of all these features are tuned via minimum error rate training (MERT) (Och, 2003). For the dependency-to-string decoder, we set rule-threshold and stack-threshold to 10^{-3}, rule-limit to 100, stack-limit to 200.

5 Experiments

The experiments in this Section are designed to answer the following questions:

1. Are our tagCNN and inCNN joint language models able to improve translation quality, and are they complementary to each other?
2. Do inCNN and tagCNN benefit from their guiding signal, compared to a generic CNN?
3. For tagCNN, is it helpful to embed more dependency structure, e.g., dependency head of each affiliated word, as additional information?

5.1 Setup

Data: Our training data are extracted from LDC data[2]. We only keep the sentence pairs that the length of source part no longer than 40 words, which covers over 90% of the sentence. The bilingual training data consist of 221K sentence pairs, containing 5.0 million Chinese words and 6.8 million English words. The development set is NIST MT03 (795 sentences) and test sets are MT04 (1499 sentences) and MT05 (917 sentences) after filtering with length limit.

Preprocessing: The word alignments are obtained with GIZA++ (Och and Ney, 2003) on the corpora in both directions, using the "grow-diag-final-and" balance strategy (Koehn et al., 2003). We adopt SRI Language Modeling Toolkit (Stolcke and others, 2002) to train a 4-gram language model with modified Kneser-Ney smoothing on the Xinhua portion of the English Gigaword corpus (306 million words). We parse the Chinese sentences with Stanford Parser into projective dependency trees.

Optimization of NN: In training the neural network, we limit the source and target vocabulary to the most frequent 20K words for both Chinese and English, covering approximately 97% and 99% of two corpus respectively. All the out-of-vocabulary words are mapped to a special token UNK. We used stochastic gradient descent to train the joint model, setting the size of minibatch to 500. All joint models used a 3-word target history (i.e., 4-gram LM). The dimension of word embedding is 100.

Metric: We use the case-insensitive 4-gram NIST BLEU[3] as our evaluation metric, with statistical significance test with *sign-test* (Collins et al., 2005) between the proposed models and two baselines.

5.2 Setting for Model Comparisons

We use the tagCNN and inCNN joint language models as additional decoding features to a dependency-to-string baseline system (Dep2Str), and compare them to the neural network joint model with 11 source context words (Devlin et al., 2014). We use the implementation of an open source toolkit[4] with default configuration except the global settings described in Section 5.1. Since our tagCNN and inCNN models are source-to-target and left-to-right (on target side), we only take the source-to-target and left-to-right type NNJM in (Devlin et al., 2014) in comparison. We call this type NNJM as BBN-JM hereafter. Although the BBN-JM in (Devlin et al., 2014) is originally tested in the hierarchical phrase-based (Chiang, 2007) SMT and string-to-dependency (Shen et al., 2008) SMT, it is fairly versatile and can be readily integrated into Dep2Str.

5.3 The Main Results

The main results of different models are given in Table 1. Before proceeding to more detailed comparison, we first observe that

- the baseline Dep2Str system gives BLEU 0.5+ higher than the open-source phrase-based system Moses (Koehn et al., 2007);
- BBN-JM can give about +0.92 BLEU score over Dep2Str, a result similar as reported in (Devlin et al., 2014).

Clearly from Table 1, tagCNN and inCNN improve upon the Dep2Str baseline by +1.28 and +1.63 BLEU, outperforming BBN-JM in the same setting by respectively +0.36 and +0.71 BLEU, averaged on NIST MT04 and MT05. These indicate that tagCNN and inCNN can individually provide discriminative information in decoding. It is worth noting that inCNN appears to be more informative

[2]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, LDC2005T06.

[3]`ftp://jaguar.ncsl.nist.gov/mt/resources/mteval-v11b.pl`

[4]http://nlg.isi.edu/software/nplm/

Systems	**MT04**	**MT05**	**Average**
Moses	34.33	31.75	33.04
Dep2Str	34.89	32.24	33.57
+ BBN-JM (Devlin et al., 2014)	36.11	32.86	34.49
+ CNN (generic)	36.13	32.95	34.54
+ *tag*CNN	36.33*	**33.37***	34.85
+ *in*CNN	**36.72***	**33.67***	35.20
+ *tag*CNN + *in*CNN	**36.97***	**34.02***	35.50

Table 1: BLEU-4 scores (%) on NIST MT04-test and MT05-test, of Moses (default settings), dependency-to-string baseline system (Dep2Str), and different features on top of Dep2Str: neural network joint model (BBN-JM), generic CNN, *tag*CNN, *in*CNN and the combination of *tag*CNN and *in*CNN. The boldface numbers and superscript * indicate that the results are significantly better ($p<0.01$) than those of the BBN-JM and the Dep2Str baseline respectively. "+" stands for adding the corresponding feature to Dep2Str.

than the affiliated words suggested by the word alignment (GIZA++). We conjecture that this is due to the following two facts

- *in*CNN avoids the propagation of mistakes and artifacts in the already learned word alignment;

- the guiding signal in *in*CNN provides complementary information to evaluate the translation.

Moreover, when *tag*CNN and *in*CNN are both used in decoding, it can further increase its winning margin over BBN-JM to +1.01 BLEU points (in the last row of Table 1), indicating that the two models with different guiding signals are complementary to each other.

The Role of Guiding Signal It is slight surprising that the generic CNN can also achieve the gain on BLEU similar to that of BBN-JM, since intuitively generic CNN encodes the entire sentence and the representations should in general far from optimal representation for joint language model. The reason, as we conjecture, is CNN yields fairly informative summarization of the sentence (thanks to its sophisticated convolution and gating architecture), which makes up some of its loss on resolution and relevant parts of the source senescence. That said, the guiding signal in both *tag*CNN and *in*CNN are crucial to the power of CNN-based encoder, as can be easily seen from the difference between the BLEU scores achieved by generic CNN, *tag*CNN, and *in*CNN. Indeed, with the signal from the already learned word alignment, *tag*CNN can gain +0.31 BLEU over its generic counterpart, while for *in*CNN with the guiding signal from the proceeding words in target, the gain is more saliently +0.66 BLEU.

5.4 Dependency Head in *tag*CNN

In this section, we study whether *tag*CNN can further benefit from encoding richer dependency structure in source language in the input. More specifically, we dependency head words can be used to further improve *tag*CNN model. As described in Section 3.2, in *tag*CNN, we append a tagging bit (0 or 1) to the embedding of words in the input layer as tags on whether they are affiliated source words. To incorporate dependency head information, we extend the tagging rule in Section 3.2 to add another tagging bit (0 or 1) to the word-embedding for original *tag*CNN to indicate whether it is part of dependency heads of the affiliated words. For example, if $\mathbf{x}_i$ is the embedding of an affiliated

Systems	MT04	MT05	Average
Dep2str	34.89	32.24	33.57
+tagCNN	36.33	33.37	34.85
+tagCNN_dep	36.54	33.61	35.08

Table 2: BLEU-4 scores (%) of tagCNN model with dependency head words as additional tags (tagCNN_dep).

source word and $\mathbf{x}_j$ the dependency head of word $\mathbf{x}_i$, the extended input of tagCNN would contain

$$\begin{aligned}\mathbf{x}_i^{(\text{AFF, NON-HEAD})} &= [\mathbf{x}_i^\top \;\; 1 \;\; 0]^\top \\ \mathbf{x}_j^{(\text{NON-AFF, HEAD})} &= [\mathbf{x}_j^\top \;\; 0 \;\; 1]^\top\end{aligned}$$

If the affiliated source word is the root of a sentence, we only append 0 as the second tagging bit since the root has no dependency head. From Table 2, with the help of dependency head information, we can improve tagCNN by +0.23 BLEU points averagely on two test sets.

6 Related Work

The seminal work of neural network language model (NNLM) can be traced to Bengio et al. (2003) on monolingual text. It is recently extended by Devlin et al. (2014) to include additional source context (11 source words) in modeling the target sentence, which is clearly most related to our work, with however two important differences: 1) instead of the ad hoc way of selecting a context window in (Devlin et al., 2014), our model covers the entire source sentence and automatically distill the context relevant for target modeling; 2) our convolutional architecture can effectively leverage guiding signals of vastly different forms and nature from the target.

Prior to our model there is also work on representing source sentences with neural networks, including RNN (Cho et al., 2014; Sutskever et al., 2014) and CNN (Kalchbrenner and Blunsom, 2013). These work typically aim to map the entire sentence to a vector, which will be used later by RNN/LSTM-based decoder to generate the target sentence. As demonstrated in Section 5, the representation learnt this way cannot pinpoint the relevant parts of the source sentences (e.g., words or phrases level) and therefore is inferior to be directly integrated into traditional SMT decoders.

Our model, especially inCNN, is inspired by is the automatic alignment model proposed in (Bahdanau et al., 2014). As the first effort to apply attention model to machine translation, it sends the state of a decoding RNN as attentional signal to the source end to obtain a weighted sum of embedding of source words as the summary of relevant context. In contrast, inCNN uses 1) a different attention signal extracted from proceeding words in partial translations, and 2) more importantly, a convolutional architecture and therefore a highly nonlinear way to retrieve and summarize the relevant information in source.

7 Conclusion and Future Work

We proposed convolutional architectures for obtaining a guided representation of the entire source sentence, which can be used to augment the n-gram target language model. With different guiding signals from target side, we devise tagCNN and inCNN, both of which are tested in enhancing a dependency-to-string SMT with +1.93 BLEU points over baseline and +1.01 BLEU points over the state-of-the-art in (Devlin et al., 2014). For future work, we will consider encoding more complex linguistic structures to further enhance the joint model.

References

[Auli et al.2013] Michael Auli, Michel Galley, Chris Quirk, and Geoffrey Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1044–1054, Seattle, Washington, USA, October.

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

[Bengio et al.2003] Yoshua Bengio, Rjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal OF Machine Learning Research*, 3:1137–1155.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. 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)*, pages 1724–1734, Doha, Qatar, October.

[Collins et al.2005] Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 531–540.

[Devlin et al.2014] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June.

[Galley et al.2004] Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule. In *Proceedings of HLT/NAACL*, volume 4, pages 273–280. Boston.

[Hu et al.2014] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

[Huang and Chiang2007] Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Annual Meeting-Association For Computational Linguistics*, volume 45, pages 144–151.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL*.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. 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*, pages 48–54.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June.

[LeCun et al.1998] Y. LeCun, L. Bottou, G. Orr, and K. Muller. 1998. Efficient backprop. In *Neural Networks: Tricks of the trade*. Springer.

[Meng et al.2013] Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü, and Qun Liu. 2013. Translation with source constituency and dependency trees. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1076, Seattle, Washington, USA, October.

[Och and Ney2002] Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302.

[Och and Ney2003] Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational linguistics*, 29(1):19–51.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167.

[Shen et al.2008] Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585.

[Stolcke and others2002] Andreas Stolcke et al. 2002. Srilm-an extensible language modeling toolkit. In *Proceedings of the international conference on spoken language processing*, volume 2, pages 901–904.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

[Xie et al.2011] Jun Xie, Haitao Mi, and Qun Liu. 2011. A novel dependency-to-string model for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226.