Discrete Structural Planning for Neural Machine Translation

Anonymous EMNLP submission

Abstract

Structural planning is important for producing long sentences, which is a missing part in current language generation models. In this work, we add a planning phase in neural machine translation to control the coarse structure of output sentences. The model first generates some planner codes, then predicts real output words conditioned on them. The codes are learned to capture the coarse structure of the target sentence. In order to learn the codes, we design an end-to-end neural network with a discretization bottleneck, which predicts the simplified part-of-speech tags of target sentences. Experiments show that the translation performance are generally improved by planning ahead. We also find that translations with different structures can be obtained by manipulating the planner codes.

1 Introduction

When human speaks, it is difficult to ensure the grammatical or logical correctness without any form of planning. Linguists have found evidence through speech errors or particular behaviors that indicate speakers are planning ahead (Redford, 2015). Such planning can happen in discourse or sentence level, and sometimes we may notice it through inner speech.

In contrast to human, a neural machine translation (NMT) model does not have the planning phase when it is asked to generate a sentence. Although we can argue that the planning is done in the hidden layers, however, such structural information remains uncertain in the continuous vectors until the concrete words are sampled. In tasks such as machine translation, a source sentence can have multiple valid translations with different syntactic structures. As a consequence, in each step of generation, the model is unaware of the "big picture" of the sentence to produce, resulting in uncertainty of word prediction.

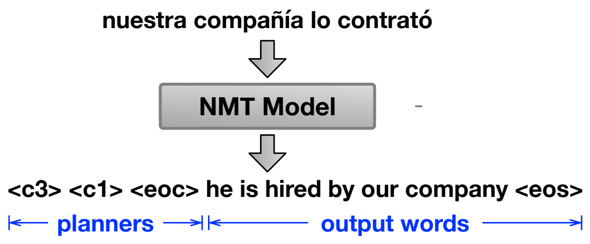
nuestra compañía lo contrató
NMT Model
<c3> <c1> <eoc> he is hired by our company <eos>
planners
output words

Figure 1: Illustration of the proposed sentence generation framework. The model predicts the planner codes before generating real output words.

In this research, we try to let the model plan the coarse structure of the output sentence before decoding real words. As illustrated in Fig. 1, in our proposed framework, we insert some planner codes into the beginning of the output sentences. The sentence structure of the translation is governed by the codes.

An NMT model takes an input sentence X and produce a translation Y. Let S_Y denotes the syntactic structure of the translation. Indeed, the input sentence already provides rich information about the target-side structure S_Y.

For example, given the Spanish sentence in Fig. 1, we can easily know that the translation will have a noun, a pronoun and a verb. Such obvious structural information does not have uncertainty, and thus does not require planning. In this example, the uncertain part is the order of the noun and the pronoun. Thus, we want to learn a set of planner codes C_Y to disambiguate such uncertain information about the sentence structure. By conditioning on the codes, we can potentially increase the effectiveness of beam search as the search space is properly regulated.

In this work, we use simplified POS tags to annotate the structure S_Y. We learn the planner codes by putting a discretization bottleneck in an end-to-end network that reconstructs S_Y with both X and C_Y. The codes are merged with the target sentences in the training data. Thus, no modification to the NMT model is required. Experiments show the translation performance is generally improved with structural planning. More interestingly, we can control the structure of output sentences by manipulating the planner codes.

2 Learning Structural Planners

In this section, we first extract the structural annotation S_Y by simplifying the POS tags. Then we explain the code learning model for obtaining the planner codes.

2.1 Structural Annotation with POS Tags

To reduce uncertainty in the decoding phase, we want a structural annotation that describes the "big picture" of the sentence. For instance, the annotation can tell whether the sentence to generate is in a "NP VP" order. The uncertainty of local structures can be efficiently solved by beam search or the NMT model itself.

In this work, we extract such coarse structural annotations S_Y through a simple two-step process that simplifies the POS tags of the target sentence:

1. Remove all tags other than "**N**", "**V**", "**PRP**", "**,**" and "**.**". Note that all tags begin with "N" (e.g. **NNS**) are mapped to "**N**", and tags begin with "V" (e.g. **VBD**) are mapped to "**V**".

2. Remove duplicated consecutive tags.

The following list gives an example of the process:

```
   Input: He found a fox behind the wall.
POS Tags: PRP VBD DT NN IN DT NN .
  Step 1: PRP V N N .
  Step 2: PRP V N .
```

Note that many other annotations can also be considered to represent the syntactic structure, which is left for future work to explore.

2.2 Code Learning

Next, we learn the planner codes C_Y to remove the uncertainty of the sentence structure S_Y when producing a translation. For simplicity, we use the notion S and C to replace S_Y and C_Y in this section.

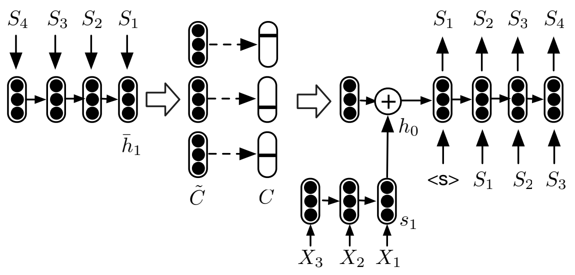

Figure 2: Architecture of the code learning model. The discretization bottleneck is shown as the dashed lines.

We first compute the discrete codes $C_1, .., C_N$ based on simplified POS tags $S_1, ..., S_T$:

$$\bar{h}_t = \mathrm{LSTM}(\mathrm{E}(S_t), \bar{h}_{t+1}; \theta_s) \ , \quad (1)$$

$$[\tilde{C}_1, ..., \tilde{C}_N] = f_{\mathrm{enc}}(\bar{h}_1; \theta_{\mathrm{enc}}) \ , \quad (2)$$

$$C_i = \mathrm{GumbelSoftmax}(\tilde{C}_i) \ , \quad (3)$$

where the tag sequence $S_1, ..., S_T$ is firstly encoded using a *backward* LSTM (Hochreiter and Schmidhuber, 1997). $\mathrm{E}(\cdot)$ denotes the embedding function. Then, we compute a set of vectors $\tilde{C}_1, ..., \tilde{C}_N$, which are latterly discretized in to approximated one-hot vectors $C_1, ..., C_N$ using Gumbel-Softmax trick (Jang et al., 2016; Maddison et al., 2016).

We then combine the information from X and C to initialize a decoder LSTM that sequentially predicts $S_1, ..., S_T$:

$$s_t = \mathrm{LSTM}(\mathrm{E}(X_t), s_{t+1}; \theta_x) \ , \quad (4)$$

$$h_0 = f_{\mathrm{dec}}([C_1, ..., C_N]; \theta_{\mathrm{dec}}) + s_1 \ , \quad (5)$$

$$h_t = \mathrm{LSTM}(\mathrm{E}(S_{t-1}), h_{t-1}; \theta_h) \ , \quad (6)$$

where $[C_1, ..., C_N]$ denotes a concatenation of N one-hot vectors. Note that only h_t is computed with a *forward* LSTM. Both f_{enc} and f_{dec} are affine transformations. Finally, we predict the probability of emitting each tag S_t with

$$P(S_t|S_{1:t-1}, X, C) = \mathrm{softmax}(f_{\mathrm{out}}(h_t; \theta_{\mathrm{out}})) \ . \quad (7)$$

The architecture of the code learning model is depicted in Fig. 2, which can be seen as a sequence auto-encoder with an extra context input X to the decoder. The parameters are optimized with cross-entropy loss.

Once the code learning model is trained, we can obtain the planner codes C for all target sentences in the training data using the encoder part.

3 NMT with Structural Planning

The training data of machine translation dataset is composed of (X, Y) sentence pairs. With the planner codes C_Y we obtained, our training data now becomes a list of $(X, C_Y; Y)$ pairs. As shown in Fig. 1, we connect the planner codes and target sentence with a "$\langle$eoc$\rangle$" token.

With the modified dataset, we train a regular NMT model. We use beam search when decoding sentences, thus the planner codes are searched before emitting real words. The codes are removed from the translation results during evaluation.

4 Related Work

Recently, some methods are proposed to improve the syntactic correctness of the translations. Stahlberg et al. (2016) restricts the search space of the NMT decoder using the lattice produced by a Statistical Machine Translation system. Eriguchi et al. (2017) takes a multi-task approach, letting the NMT model to parse a dependency tree and combine the parsing loss with the original loss.

Several works further incorporate the target-side syntactic structures explicitly. Nadejde et al. (2017) interleaves CCG supertags with normal output words in the target side. Instead of predicting words, Aharoni and Goldberg (2017) trains a NMT model to generate linearized constituent parse trees. Wu et al. (2017) proposed a model to generate words and parse actions simultaneously. The word prediction and action prediction are conditioned on each other. However, none of the these methods plan the structure before translation.

Similar to our code learning approach, some works also learn the discrete codes for different purposes. Shu and Nakayama (2018) compresses the word embeddings by learning the concept codes to represent each word. Kaiser et al. (2018) breaks down the dependency among words with shorter code sequences. The decoding can be faster by predicting the shorter artificial codes.

5 Experiments

We evaluate our models on IWSLT 2014 German-to-English task (Cettolo et al., 2014) and ASPEC Japanese-to-English task (Nakazawa et al., 2016), containing 178K and 3M bilingual pairs respectively. We use Kytea (Neubig et al., 2011) to tokenize Japanese texts and moses toolkit (Koehn et al., 2007) for other languages. Using byte-pair encoding (Sennrich et al., 2016), we force the vocabulary size of each language to be 20K for IWSLT dataset and 40K for ASPEC dataset.

Code Setting	Capacity	S_Y acc.	C_Y acc.
N=1, K=4	2 bits	27%	63%
N=2, K=2	2 bits	23%	67%
N=2, K=4	4 bits	35%	41%
N=4, K=2	4 bits	22%	44%
N=4, K=4	8 bits	44%	27%

Table 1: A comparison of different code settings on IWSLT 2014 dataset. The accuracy of reconstructing S_Y in the code model, and the accuracy of predict C_Y in the NMT model are reported.

For IWSLT 2014 dataset, we concatenate all five TED/TEDx development and test corpus to form a test set containing 6750 pairs. For evaluation, we report *tokenized BLEU* with moses tool.

5.1 Evaluation of Planner Codes

In the code learning model, all hidden layers have 256 hidden units. The model is trained using Nesterov's accelerated gradient (NAG) (Nesterov, 1983) for maximum 50 epochs with a learning rate of 0.25. We test different settings of code length N and the number of code types K. The information capacity of the codes will be $N \log K$ bits. In Table 1, we evaluate the learned codes for different settings. S_y accuracy evaluates the accuracy of correctly reconstructing S_y with the source sentence X and the code C_y. C_y accuracy reflects the chance of guessing the correct code C_y given X.

We can see a clear trade-off between S_Y accuracy and C_Y accuracy. When the code has more capacity, it can recover S_Y more accurately, however, resulting in a lower probability for the NMT model to guess the correct code. We found the setting of $N = 2, K = 4$ has a balanced trade-off.

5.2 Evaluation of NMT Models

To make a strong baseline, we use 2 layers of bi-directional LSTM encoders with 2 layers of LSTM decoders in the NMT model. The hidden layers have 256 units for IWSLT De-En task and 1000 units for ASPEC Ja-En task. We apply Key-Value Attention (Miller et al., 2016) in the first decoder layer. Residual connection (He et al., 2016) is used to combine the hidden states in two decoder layers. Dropout is applied everywhere outside of the recurrent function with a drop rate of 0.2 . To train the NMT models, we also use the NAG optimizer

Dataset	Model	BLEU(%)		
		BS=1	**BS=3**	**BS=5**
De-En	baseline	27.90	29.26	29.52
	plan (N=2, K=4)	**28.35**	**29.59**	**29.78**
Ja-En	baseline	**23.92**	25.08	25.26
	plan (N=2, K=4)	22.79	**25.53**	**25.69**

Table 2: A comparison of translation performance with different beam sizes (BS).

with a learning rate of 0.25, which is annealed by a factor of 10 if no improvement of loss value is observed in 20K iterations. Best parameters are chosen on a validation set.

As shown in Table 2, by conditioning the word prediction on the generated planner codes, the translation performance is generally improved over a strong baseline. The improvement may be the result of properly regulating the search space.

However, when we apply greedy search on Ja-En dataset, the BLEU score is much lower compared to the baseline. We also tried to beam search the planner codes then switch to greedy search, but the results are not significantly changed. We hypothesize that it is important to simultaneously explore multiple candidates with drastically different structures on Ja-En task. By planning ahead, more diverse candidates can be explored, which improves beam search but not greedy search. If so, the results are in line with a recent study (Li et al., 2016) that shows the performance of beam search depends on the diversity of candidates.

5.3 Qualitative Analysis

Instead of letting the beam search to decide the planner codes, we can also choose the codes manually. Table 3 gives an example of the candidate translations produced by the model when conditioning on different planner codes.

input	AP no katei ni tsuite nobeta. (Japanese)
code 1	`<c4> <c1> <eoc>` `the process of AP is described .`
code 2	`<c1> <c1> <eoc>` `this paper describes the process of AP .`
code 3	`<c3> <c1> <eoc>` `here was described on process of AP .`
code 4	`<c2> <c1> <eoc>` `they described the process of AP .`

Table 3: Example of translation results conditioned on different planner codes in Ja-En task

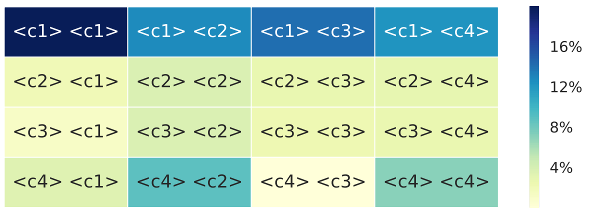

Figure 3: Distribution of assigned planner codes for English sentences in ASPEC Ja-En dataset

As shown in Table 3, we can obtain translations with drastically different structures by manipulating the codes. The results show that the proposed method can be useful for sampling paraphrased translations with high diversity.

The distribution of the codes learned for 3M English sentences in ASPEC Ja-En dataset is shown in Fig. 3. We found the code "`<c1> <c1>`" is assigned to 20% of the sentences, whereas "`<c4> <c3>`" is not assigned to any sentence. The skewed distribution may indicate that the capacity of the codes is not fully exploited, and thus leaves room for further improvement.

6 Discussion

Instead of learning discrete codes, we can also directly predict the structural annotations (e.g. POS tags), then translate based on the predicted structure. However, as the simplified POS tags are also long sequences, the error of predicting the tags will be propagated to word generation. In our experiments, doing so degrades the performance by around 8 BLEU points on IWSLT dataset.

7 Conclusion

In this paper, we add a planning phase in neural machine translation, which generates some planner codes to control the structure of the output sentence. To learn the codes, we design an end-to-end neural network with a discretization bottleneck to predict the simplified POS tags of target sentences. Experiments show that the proposed method generally improves the translation performance. We also confirm the effect of the planner codes, by being able to sample translations with drastically different structures using different planner codes.

The planning phase helps the decoding algorithm by removing the uncertainty of the sentence structure. The framework described in this paper can be extended to plan other latent factors, such as the sentiment or topic of the sentence.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *ACL*.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*.

Akiko Eriguchi, Yoshimasa Tsuruoka, and Kyunghyun Cho. 2017. Learning to parse and translate improves neural machine translation. In *ACL*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *CoRR*, abs/1611.01144.

Lukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *CoRR*, abs/1803.03382.

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 *ACL*.

Jiwei Li, Will Monroe, and Daniel Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *CoRR*, abs/1611.08562.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, abs/1611.00712.

Alexander H. Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *EMNLP*.

Maria Nadejde, Siva Reddy, Rico Sennrich, Tomasz Dwojak, Marcin Junczys-Dowmunt, Philipp Koehn, and Alexandra Birch. 2017. Predicting target language ccg supertags improves neural machine translation. In *WMT*.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. Aspec: Asian scientific paper excerpt corpus. In *LREC*.

Yurii Nesterov. 1983. A method for unconstrained convex minimization problem with the rate of convergence o (1/k2). In *Doklady an SSSR*, volume 269, pages 543–547.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable japanese morphological analysis. In *ACL*, pages 529–533.

Melissa A Redford. 2015. The handbook of speech production. pages 420–423.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909.

Raphael Shu and Hideki Nakayama. 2018. Compressing word embeddings via deep compositional code learning. In *ICLR*.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. *CoRR*, abs/1605.04569.

Shuangzhi Wu, Dongdong Zhang, Nan Yang, Mu Li, and Ming Zhou. 2017. Sequence-to-dependency neural machine translation. In *ACL*.