Insertion Position Selection Model for Flexible Non-Terminals in Dependency Tree-to-Tree Machine Translation

Toshiaki Nakazawa
Japan Science and Technology Agency
5-3, Yonbancho, Chiyoda-ku, Tokyo, 102-8666, Japan
nakazawa@pa.jst.jp

John Richardson and **Sadao Kurohashi**
Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
john@nlp.ist.i.kyoto-u.ac.jp kuro@i.kyoto-u.ac.jp

Abstract

Dependency tree-to-tree translation models are powerful because they can naturally handle long range reorderings which is important for distant language pairs. The translation process is easy if it can be accomplished only by replacing non-terminals in translation rules with other rules. However it is sometimes necessary to adjoin translation rules. Flexible non-terminals have been proposed as a promising solution for this problem. A flexible non-terminal provides several insertion position candidates for the rules to be adjoined, but it increases the computational cost of decoding. In this paper we propose a neural network based insertion position selection model to reduce the computational cost by selecting the appropriate insertion positions. The experimental results show the proposed model can select the appropriate insertion position with a high accuracy. It reduces the decoding time and improves the translation quality owing to reduced search space.

1 Introduction

Tree-to-tree machine translation models currently receive limited attention. However, we believe that using target-side syntax is important to achieve high-quality translations between distant language pairs which require long range reorderings. Especially, using dependency trees on both source and target sides is promising for this purpose (Menezes and Quirk, 2007; Nakazawa and Kurohashi, 2010; Richardson et al., 2014). Tree-based translation models naturally realize word reorderings using the non-terminals or anchors for the attachment in the translation rules: therefore they do not need a reordering model which string-based models require. For example, suppose we have a translation rule with the word alignment shown in Figure 1, it is easy to translate a new input sentence which has "図書館 (*library*)" instead of "公園 (*park*)" because we can accomplish it by simply substituting "library" for the target word "park" without considering the reordering. In this case, the source word "公園" and target word "park" work as the non-terminals.

On the other hand, it is problematic when we need to adjoin a subtree which is not presented in training sentences, which we call *floating* subtree in this paper. The floating subtrees are not necessarily adjuncts, but any words or phrases. For example, suppose the Japanese input sentence in Figure 1 has "突然 (*suddenly*)", but the training corpus provides only a translation rule without the word. In this case we cannot directly use the rule for the translation because it does not know where to insert the translation of the floating word in the output. As another example, there is no context information available for the children of the OOV word in the input sentence, so we need some special process to translate them.

Previous work deals with this problem by using glue rules (Chiang, 2005) or limiting the dependency structures to be well-formed (Shen et al., 2008). Richardson et al. (2016) introduces the concept of *flexible* non-terminals. It provides multiple possible insertion positions for the floating subtree rather than fixed insertion positions. A possible insertion position must satisfy the following conditions:

- it must be a child of the aligned word of the parent of the floating subtree

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 2271–2277,
Austin, Texas, November 1-5, 2016.

- it must not violate the projectivity of the dependency tree

For example, possible insertion positions for the floating word "突然" are shown in gray arrows in Figure 1. Since "突然" is a child of "電話する", and the translation of "電話する" is "called", insertion positions must be a child of "called". Also, insertion positions do not violate the projectivity of the target tree. Flexible non-terminals are analogous to the *auxiliary tree* of the tree adjoining grammars (TAG) (Joshi, 1985), which is successfully adopted in machine translation (DeNeefe and Knight, 2009). The difference is that TAG is defined on the constituency trees rather than the dependency trees.

Flexible non-terminals are powerful to handle floating subtrees and it achieve better translation quality. However the computational cost of decoding becomes high even though they are compactly represented in the lattice form (Cromieres and Kurohashi, 2014). In our experiments, using flexible non-terminals causes the decoding to be 3 to 6 times slower than when they are not used. Flexible non-terminals increase the number of translation rules because the insertion positions are selected during the decoding. However, we think it is possible to restrict possible insertion positions or even select only one insertion position by looking at the tree structures on both sides.

In this paper, we propose a method to select the appropriate insertion position before decoding. This can not only reduce the decoding time but also improve the translation quality because of reduced search space.

2 Insertion Position Selection

We assume that correct insertion positions can be determined before decoding, using the word to be inserted (I) with the context on the source side and the context of the insertion positions on the target side. On the source side, we use the parent of I (P_s) and the distance of I from P_s (D_s). On the target side, we use the previous (S_p) and next (S_n) sibling of the insertion position, the parent of the insertion position (P_t) and the distance of the insertion position from P_t (D_t). The distances are calculated on the siblings rather than the words in the sentence, and it is a positive or negative value if the insertion position is to the left or to the right of the parent respectively.

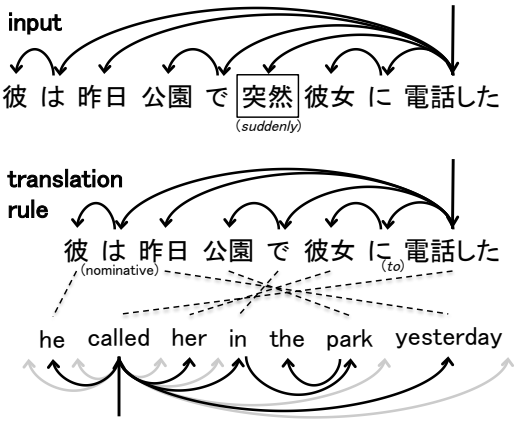

Figure 1: Example of an input sentence and a translation rule. We want to insert "突然 (*suddenly*)" which is not in the translation rule. The possible insertion positions in the target sentence are shown in gray arrows.

Taking the insertion position between "park" and "yesterday" in Figure 1 as an example, I = "突然", P_s = "電話した", D_s = +2, S_p = "park", S_n = "yesterday", P_t = "called" and D_t = -3. In cases where S_p or S_n is empty, we use special words "[[LEFT-START]]", "[[LEFT-END]]", "[[RIGHT-START]]" and "[[RIGHT-END]]". In the case of "yesterday" in Figure 1, S_p = "in" and S_n = "[[RIGHT-END]]". These clues are fed into the neural network model to solve the insertion position selection problem.

2.1 Neural Network Model

Figure 2 shows the neural network model for the insertion position selection. Given an insertion position candidate with an index k, the words (I, P_s, S_p^k, S_n^k, P_t) are first converted into vector representations through the same three embedding layers: surface form embedding (200 dimensions.), part-of-speech embedding (10 dimensions) and dependency type (or phrase category) embedding (10 dimensions), and they are concatenated to create the 220-dimension vectors. The word embedding is a randomly initialized transformation from an one-hot vector to a 200 or 10-dimensional vector, and it is learned during the whole network training.

Using these words and the distances, we create source and target context vectors c_s^k and c_t^k which represent the information of source and target sides, respectively. The distances (integer values) are directly inputted to the network. Then the context vec-

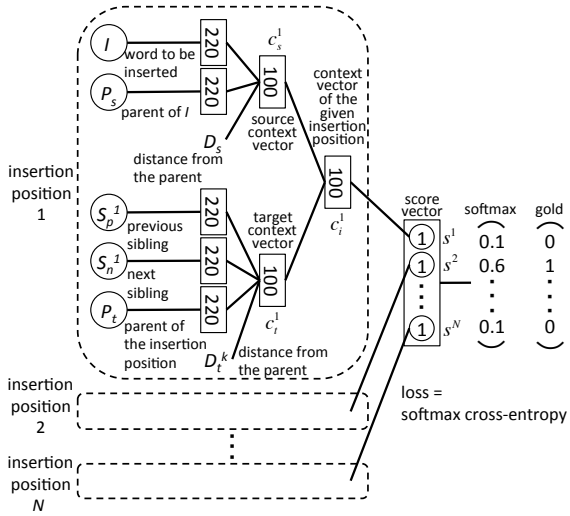

Figure 2: The neural network for the insertion position selection. The numbers inside the boxes show the dimensions of the vectors.

tor of the given insertion position c_i^k is created using c_s^k and c_t^k. Finally we get the score of the given insertion position s^k from c_i^k. These vectors are calculated as follows:

$$\begin{aligned} c_s^k &= \tanh(W_{c_s}[I; P_s; D_s]) \\ c_t^k &= \tanh(W_{c_t}[S_p; S_n; P_t; D_t^k]) \\ c_i^k &= \tanh(W_{c_i}[c_s^k; c_t^k]) \\ s^k &= W_s c_i^k \end{aligned}$$

where ";" means concatenation of the vectors. The size of c_s^k, c_t^k and c_i^k is 100 in our experiments.

The same network is applied to all the other insertion positions and get their scores. Finally the scores are normalized by the softmax function, and the loss is calculated by the softmax cross-entropy as the loss function. All the links between layers are fully-connected. We use dropout (50%) to avoid overfitting.

2.2 Training Data Generation

The data for training the neural network model can be automatically generated from the word-aligned parallel corpus with dependency parses in both sides by Algorithm 1. The ALIGNMENT function returns the aligned word in the target tree for the given source word[1], and the ISPARENTCHILD function returns TRUE if P_t is the parent of C_t.

[1]In case of the multiple word alignment, we only use the root word of them in both source and target sides.

Algorithm 1 Training Data Generation for NN

```
for all P_s ∈ words in source tree do
    P_t = ALIGNMENT(P_s)
    for all C_s ∈ CHILDREN(P_s) do
        C_t = ALIGNMENT(C_s)
        if ISPARENTCHILD(P_t, C_t) then
            GENERATEDATA(P_s, C_s, P_t, C_t)
        end if
    end for
end for
```

	Ja ↔ En	Ja ↔ Zh
Training	2,020,106	667,520
Development	1,789	2,115
Test	1,812	2,174

Table 1: The number of sentences in ASPEC used in our experiments.

The GENERATEDATA function generates one instance of training data to predict the position of C_t from P_s, C_s and P_t with their contexts by removing C_t in the target tree. The position where C_t exists is regarded as the correct insertion position, and others as incorrect insertion positions. Note that C_s corresponds to I in Figure 2.

2.3 Insertion Position Selection in Translation

Once the neural network model is trained, it can be applied to select the most appropriate insertion positions in the translation rules for the given floating subtree by looking at the score of each insertion position. Translation rules only contain part of the original parallel sentence in most of the cases. This means that the context used for selecting the insertion position is different from that in the training data for the neural network. For example, if the input sentence does not have "公園 で (*in the park*)" in Figure 1, the number of possible insertion positions is 6 and we do not use "in" as the context. However, this is not so problematic because similar or same context may appear in the different part of the corpus.

3 Experiments

We conducted two kinds of experiments: the insertion position selection and translation. We used ASPEC (Nakazawa et al., 2016) as the dataset and the numbers of the sentences of the corpus are shown in Table 1. The Japanese morphological analyzer (Kurohashi et al., 1994) and dependency parser (Kurohashi and Nagao, 1994) are used for Japanese

	Ja → En	En → Ja	Ja → Zh	Zh → Ja
Training	15.7M		5.7M	
Development	160K		58K	
Test	160K		58K	
ave. # IP	3.39	3.15	3.72	3.41
best epoch	89	71	61	79
mean loss	0.089	0.058	0.105	0.056
Accuracy (%)	97.08	97.72	96.51	97.99
Logit Accuracy (%)	55.00	89.03	68.04	83.16

Table 2: The statistics of the data and results of the insertion position selection experiments.

sentences. English sentences are first parsed by nlparser (Charniak and Johnson, 2005) and then converted into word dependency trees using Collins' head percolation table (Collins, 1999). We used Chinese word segmenter KKN (Shen et al., 2014) and dependency parser SKP (Shen et al., 2012) for Chinese sentences. The supervised word alignment Nile (Riesa et al., 2011) was used.

We used a state-of-the-art dependency tree-to-tree decoder (Richardson et al., 2014) with the default settings. The neural network is constructed and trained using the Chainer (Tokui et al., 2015).

3.1 Insertion Position Selection

The training, development and test data for the neural network is automatically generated by the procedure explained in Section 2.2. The size of the generated data from the ASPEC and the average number of insertion positions for each floating subtree are shown in Table 2. We trained the model for 100 epochs and used the best model on the development data for testing. The vocabulary size for the surface form was 50,000.

For comparison, we also tried the logistic regression to predict the correct insertion positions. Because our training data is huge, we used Multi-core LIBLINEAR[2] with L2-regularized logistic regression (primal) solver. The format of training instances are: one-hot (binary) vectors for surface form, POS and dependency type, and distances scaled to [0, 1]. We first find the best value for the C parameter, then train the model. The best insertion position is selected using the estimated probabilities for each insertion position.

The experimental results are also shown in Table 2. We evaluated the results by the mean loss of the model and the accuracy on the test data. The result shows that our model can select the correct insertion position with very high accuracy for every language pair while the classical logistic regression model cannot. This supports our claim stated in the beginning of Section 2.

X → Ja is easier and achieved slightly better accuracy than the reverse direction because Japanese is a head-final language and all children are generally put on the left of their parents. There are some cases judged as incorrect but acceptable insertion positions, and hence the true accuracies are higher than the ones reported above. We also investigated the top-2 accuracy and found that it is above 99.0% for Ja → X and 99.5% for X → Ja.

Table 3 shows the detailed result of Ja $rightarrow$ En experiment. The number of insertion-position is at least 2 (left/right of the parent) and it is easy to solve (more than 99% accuracy). 3 is a situation where the parent has one child, and it is still not so difficult (97-98% accuracy). About 70% of the test data have only 2 or 3 insertion-positions. Difficult cases are the sentences which have many adjuncts as in Figure 1, but we used the scientific paper corpora, where not so many adjuncts appear.

3.2 Translation

We conducted translation experiment using the ASPEC in 3 settings:

- No Flexible: not using the flexible non-terminals and using simple glue rules as in the baseline model of (Richardson et al., 2016) [3]
- Baseline: using the flexible non-terminals without the insertion position selection
- Proposed: using only the most appropriate insertion position for the flexible non-terminals

We also report the translation quality of conventional models for comparison: phrase-based SMT (PBSMT) and hierarchical phrase-based SMT (Hiero). We used the default settings of Moses except -distortion-limit=20 for PBSMT.

The translation quality is evaluated by the automatic evaluation measures BLEU (Papineni et al.,

[2]https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/multicore-liblinear/

[3]52.5% of all the translation rules require glue rule, but it is applied to 22.6% of the rules actually used in the translation.

# of ins. pos. and ratio	Top N accuracy (%)								
	1	2	3	4	5	6	7	8	9
2 (49.39%)	99.22								
3 (18.15%)	96.37	99.60							
4 (11.77%)	95.76	99.06	99.82						
5 (6.30%)	94.24	98.39	99.38	99.85					
6 (4.99%)	93.50	97.82	99.11	99.63	99.98				
7 (3.96%)	93.13	97.40	98.87	99.56	99.89	99.98			
8 (2.42%)	92.77	97.07	98.43	99.23	99.61	99.77	99.97		
9 (1.50%)	92.14	96.19	97.85	99.01	99.50	99.67	99.92	100.00	
10 (0.77%)	92.04	96.62	97.75	98.47	99.03	99.52	99.92	99.92	100.00

Table 3: Detailed Ja → En insertion position selection experimental result.

	Ja → En			En → Ja			Ja → Zh			Zh → Ja		
	BLEU	RIBES	Time	BLEU	RIBES	Time	BLEU	RIBES	Time	BLEU	RIBES	Time
PBSMT	18.45	64.51	-	27.48	68.37	-	27.96	78.90	-	34.65	77.25	-
Hiero	18.72	65.11	-	30.19	73.47	-	27.71	**80.91**	-	**35.43**	**81.04**	-
No Flexible	20.28	65.08	1.00	28.77	75.21	1.00	24.85	66.60	1.00	30.51	73.08	1.00
Baseline	21.61	69.82	6.28	**30.57**	76.13	3.30	28.79	78.11	5.16	34.32	77.82	5.28
Proposed	**22.07**†	**70.49**†	2.25	30.50	**76.69**†	1.27	**29.83**†	79.73†	2.21	34.71†	79.25†	1.89

Table 4: The results of the translation experiments. † means the Proposed method achieved significantly better score than the Baseline ($p < 0.01$).

2002) and RIBES (Isozaki et al., 2010) with the significance testing by bootstrap resampling (Koehn, 2004). RIBES is more sensitive to word order than BLEU, so we expect an improvement in RIBES. We also investigated relative decoding time compared to the No Flexible setting. Note that we used the word "decoding" for only exploring the search space, and it does not include constructing the search space (as the table lookup in Phrase-based SMT). Our whole translation process is:

1. translation rule extraction
2. insertion-position selection
3. decoding

At the time of the second step, we have all the translation rules applicable to the input sentence. The computation time for each step is $3 \gg 1 \gg 2$ so we only focus on the time for step 3 in the experiments (the computation time for step 2 is negligibly small).

The results are shown in Table 4. The Proposed method achieved significantly better automatic evaluation scores than the Baseline for all the language pairs except the BLEU score of En → Ja direction. Also, the decoding time is reduced by about 60% relative to that of the Baseline.

Our tree-based model is better than the conventional models except C → J, where the accuracy of Chinese parsing for the input sentences has a bad effect.

4 Conclusion

In this paper we have proposed a neural network based insertion position selection model to reduce the computational cost of the decoding for dependency tree-to-tree translation with flexible non-terminals. The model successfully finds the appropriate insertion position from the candidates and it leads to faster translation speed and better translation quality due to the reduced search space.

Currently, we use only words as the context but it seems promising to use subtrees as well. For example, using the information of the subtree "in the park" is more informative than using only "in" in Figure 1. This is especially important for Japanese as the target language because children of verbs are often case markers and they do not provide enough information when selecting the appropriate insertion position. It is possible to adopt existing models of creating vector representation of dependency subtrees such as the model using recursive neural networks (Liu et al., 2015) and convolutional neural networks (Mou et al., 2015).

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 173–180.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270. Association for Computational Linguistics.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Fabien Cromieres and Sadao Kurohashi. 2014. Translation rules with right-hand side lattices. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 577–588. Association for Computational Linguistics.

Steve DeNeefe and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 727–736. Association for Computational Linguistics.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 206–250. Cambridge University Press, Cambridge.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July. Association for Computational Linguistics.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Sadao Kurohashi, Toshihisa Nakamura, Yuji Matsumoto, and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of The International Workshop on Sharable Natural Language*, pages 22–28.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 285–290. Association for Computational Linguistics.

Arul Menezes and Chris Quirk, 2007. *Proceedings of the Second Workshop on Statistical Machine Translation*, chapter Using Dependency Order Templates to Improve Generality in Translation, pages 1–8. Association for Computational Linguistics.

Lili Mou, Hao Peng, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015. Discriminative neural sentence modeling by tree-based convolution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2315–2325, Lisbon, Portugal, September. Association for Computational Linguistics.

Toshiaki Nakazawa and Sadao Kurohashi. 2010. Fully syntactic ebmt system of kyoto team in ntcir-8. In *In Proceedings of the 8th NTCIR Workshop Meeting on Evaluation of Information Access Technologies (NTCIR-8)*, pages 403–410.

Toshiaki Nakazawa, Manabu Yaguchi, Kiyotaka Uchimoto, Masao Utiyama, Eiichiro Sumita, Sadao Kurohashi, and Hitoshi Isahara. 2016. ASPEC: Asian scientific paper excerpt corpus. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2204–2208, Portorož, Slovenia, May.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

John Richardson, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2014. Kyotoebmt: An example-based dependency-to-dependency translation framework. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84. Association for Computational Linguistics.

John Richardson, Fabien Cromierès, Toshiaki Nakazawa, and Sadao Kurohashi. 2016. Flexible non-terminals for dependency tree-to-tree reordering. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 11–19. Association for Computational Linguistics.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the 2011 Conference on Empirical*

Methods in Natural Language Processing, pages 497–507. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Association for Computational Linguistics*.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2012. A reranking approach for dependency parsing with variable-sized subtree features. In *Proceedings of 26th Pacific Asia Conference on Language Information and Computing*, pages 308–317.

Mo Shen, Hongxiao Liu, Daisuke Kawahara, and Sadao Kurohashi. 2014. Chinese morphological analysis with character-level pos tagging (short paper). In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL2014)*, Baltimore, USA.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.