RNN-based Derivation Structure Prediction for SMT

Feifei Zhai, Jiajun Zhang, Yu Zhou and Chengqing Zong
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{ffzhai, jjzhang, yzhou, cqzong}@nlpr.ia.ac.cn

Abstract

In this paper, we propose a novel derivation structure prediction (DSP) model for SMT using recursive neural network (RNN). Within the model, two steps are involved: (1) phrase-pair vector representation, to learn vector representations for phrase pairs; (2) derivation structure prediction, to generate a bilingual RNN that aims to distinguish good derivation structures from bad ones. Final experimental results show that our DSP model can significantly improve the translation quality.

1 Introduction

Derivation structure is important for SMT decoding, especially for the translation model based on nested structures of languages, such as BTG (bracket transduction grammar) model (Wu, 1997; Xiong et al., 2006), hierarchical phrase-based model (Chiang, 2007), and syntax-based model (Galley et al., 2006; Marcu et al., 2006; Liu et al., 2006; Huang et al., 2006; Zhang et al., 2008; Zhang et al., 2011; Zhai et al., 2013). In general, ***derivation structure*** refers to the tuple that records the used translation rules and their compositions during decoding, just as Figure 1 shows.

Intuitively, a good derivation structure usually yields a good translation, while bad derivations always result in bad translations. For example in Figure 1, (a) and (b) are two different derivations for Chinese sentence "布什 与 沙龙 举行 了 会谈". Comparing the two derivations, (a) is more reasonable and yields a better translation. However, (b) wrongly translates phrase "与 沙龙" to "and Sharon" and combines it with [布什;Bush] incorrectly, leading to a bad translation.

To explore the derivation structure's potential on yielding good translations, in this paper, we propose a novel derivation structure prediction (DSP) model for SMT decoding.

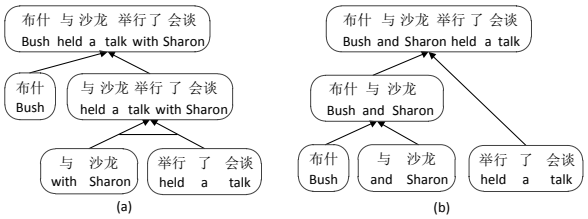


Figure 1: Two different derivation structures of BTG translation model. In the structure, leaf nodes denote the used translation rules. For each node, the first line is the source string, while the second line is its corresponding translation.

The proposed DSP model is built on recursive neural network (RNN). Within the model, two steps are involved: (1) phrase-pair vector representation, to learn vector representations for phrase pairs; (2) derivation structure prediction, to build a bilingual RNN that aims to distinguish good derivation structures from bad ones. Extensive experiments show that the proposed DSP model significantly improves the translation quality, and thus verify the effectiveness of derivation structure on indicating good translations.

We make the following contributions in this work:

- We propose a novel RNN-based model to do derivation structure prediction for SMT decoding. To our best knowledge, this is the first work on this issue in SMT community;
- In current work, RNN has only been verified to be useful on monolingual structure learning (Socher et al., 2011a; Socher et al., 2013). We go a step further, and design a bilingual RNN to represent the derivation structure;
- To train the RNN-based DSP model, we propose a max-margin objective that prefers gold derivations yielded by forced decoding to n-best derivations generated by the conventional BTG translation model.

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers), pages 779–784,
Baltimore, Maryland, USA, June 23-25 2014.

2 The DSP Model

The basic idea of DSP model is to represent the derivation structure by RNN (Figure 2). Here, we build the DSP model for BTG translation model, which is naturally compatible with RNN. We believe that the DSP model is also beneficial to other translation models. We leave them as our future work.

2.1 Phrase-Pair Vector Representation

Phrase pairs, i.e., the used translation rules, are the leaf nodes of derivation structure. Hence, to represent the derivation structure by RNN, we need first to represent the phrase pairs. To do this, we use two unsupervised recursive autoencoders (RAE) (Socher et al., 2011b), one for the source phrase and the other for the target phrase. We call the unit of the two RAEs the ***Leaf Node Network (LNN)***.

Using n-dimension word embedding, RAE can learn a n-dimension vector for any phrase. Meanwhile, RAE will build a binary tree for the phrase, as Figure 2 (in box) shows, and compute a reconstruction error to evaluate the vector. We use $E(T_{ph})$ to denote the reconstruction error given by RAE, where ph is the phrase and T_{ph} is the corresponding binary tree. In RAE, higher error corresponds to worse vector. More details can be found in (Socher et al., 2011b).

Given a phrase pair (sp, tp), we can use LNN to generate two n-dimension vectors, representing sp and tp respectively. Then, we concatenate the two vectors directly, and get a vector $r \in \mathbb{R}^{2n}$ to represent phrase pair (sp, tp) (shown in Figure 2). The vector r is evaluated by combining the reconstruction error on both sides:

$$E(T_{sp}, T_{tp}) = \frac{1}{2}[E(T_{sp}) + E(T_{tp}) \cdot \frac{N_s}{N_t}] \quad (1)$$

where T_{sp} and T_{tp} are the binary trees for sp and tp. N_s and N_t denote the number of nodes in T_{sp} and T_{tp}. Note that in order to unify the errors on the two sides, we use ratio N_s/N_t to eliminate the influence of phrase length.

Then, according to Equation (1), we compute an LNN score to evaluate the vector of all phrase pairs, i.e., leaf nodes, in derivation d:

$$LNN(d) = -\sum\nolimits_{(sp,tp)} E(T_{sp}, T_{tp}) \quad (2)$$

where (sp, tp) is the used phrase pair in derivation d. Obviously, the derivation with better phrase-pair representations will get a higher LNN score.

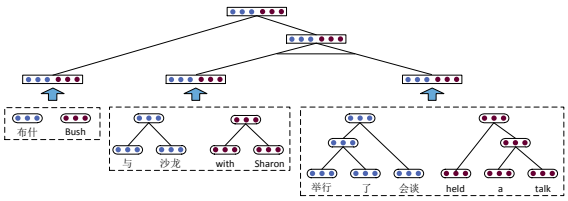


Figure 2: Illustration of DSP model, based on the derivation structure in Figure 1(a).

The LNN score will serve as part of the DSP model for predicting good derivation structures.

2.2 Derivation Structure Prediction

Using the vector representations of phrase pairs, we then build a ***Derivation Structure Network (DSN)*** for prediction (Figure 2).

In DSN, the derivation structure is represented by repeatedly applying ***unit neural network*** (UNN, Figure 3) at each non-leaf node. The UNN receives two node vectors $r_1 \in \mathbb{R}^{2n}$ and $r_2 \in \mathbb{R}^{2n}$ as input, and induces a vector $p \in \mathbb{R}^{2n}$ to represent the parent node.

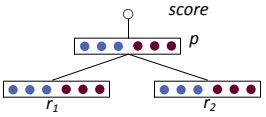


Figure 3: The unit neural network used in DSN.

For example, in Figure 2, node [与 沙龙; with Sharon] serves as the first child with vector r_1, and node [举行 了 会谈; held a talk] as the second child with vector r_2. The parent node vector p, representing [与 沙龙 举行 了 会谈; held a talk with Sharon], is computed by merging r_1 and r_2:

$$p = f(W_{UNN}[r_1; r_2] + b_{UNN}) \quad (3)$$

where $[r_1; r_2] \in \mathbb{R}^{4n \times 1}$ is the concatenation of r_1 and r_2, $W_{UNN} \in \mathbb{R}^{2n \times 4n}$ and $b_{UNN} \in \mathbb{R}^{2n \times 1}$ are the network's parameter weight matrix and bias term respectively. We use $\tanh(\cdot)$ as function f.

Then, we compute a local score using a simple inner product with a row vector $W^{score}_{UNN} \in \mathbb{R}^{1 \times 2n}$:

$$s(p) = W^{score}_{UNN} \cdot p \quad (4)$$

The score measures how well the two child nodes r_1 and r_2 are merged into the parent node p.

As we all know, in BTG derivations, we have two different ways to merge translation candidates, *monotone* or *inverted*, meaning that we

merge two candidates in a monotone or inverted order. We believe that different merging order (monotone or inverted) needs different UNN. Hence, we keep two different ones in DSN, one for monotone order (with parameter W_{mono}, b_{mono}, and W_{mono}^{score}), and the other for inverted (with parameter W_{inv}, b_{inv}, and W_{inv}^{score}). The idea is that the merging order of the two candidates will determine which UNN will be used to generate their parent's vector and compute the score in Equation (4). Using a set of gold derivations, we can train the network so that correct order will receive a high score by Equation (4) and incorrect one will receive a low score.

Thus, when we merge the candidates of two adjacent spans during BTG-based decoding, the local score in Equation (4) is useful in two aspects: (1) for the same merging order, it evaluates how well the two candidates are merged; (2) for the different order, it compares the candidates generated by monotone order and inverted order.

Further, to assess the entire derivation structure, we apply UNN to each node recursively, until the root node. The final score utilized for derivation structure prediction is the sum of all local scores:

$$DSN(d) = \sum\nolimits_{p} s(p) \tag{5}$$

where d denotes the derivation structure and p is the non-leaf node in d. Obviously, by this score, we can easily assess different derivations. Good derivations will get higher scores while bad ones will get lower scores.

Li et al. (2013) presented a network to predict how to merge translation candidates, in monotone or inverted order. Our DSN differs from Li's work in two points. For one thing, DSN can not only predict how to merge candidates, but also evaluate whether two candidates should be merged. For another, DSN focuses on the entire derivation structure, rather than only the two candidates for merging. Therefore, the translation decoder will pursue good derivation structures via DSN. Actually, Li's work can be easily integrated into our work. We leave it as our future work.

3 Training

In this section, we present the method of training the DSP model. The parameters involved in this process include: word embedding, parameters of the two unsupervised RAEs in LNN, and parameters in DSN.

3.1 Max-Margin Framework

In DSP model, our goal is to assign higher scores to gold derivations, and lower scores to bad ones. To reach this goal, we adopt a max-margin framework (Socher et al., 2010; Socher et al., 2011a; Socher et al., 2013) for training.

Specifically, suppose we have a training data like $(u_i, \mathcal{G}(u_i), \mathcal{A}(u_i))$, where u_i is the input source sentence, $\mathcal{G}(u_i)$ is the ***gold derivation set*** containing all gold derivations of u_i[1], and $\mathcal{A}(u_i)$ is the ***possible derivation set*** that contains all possible derivations of u_i. We want to minimize the following regularized risk function:

$$\begin{aligned} J(\theta) &= \frac{1}{N}\sum_{i=1}^{N} R_i(\theta) + \frac{\lambda}{2} \parallel \theta \parallel^2, \; where \\ R_i(\theta) &= \max_{\hat{d} \in \mathcal{A}(u_i)} \Big(s\big(\theta, u_i, \hat{d}\big) + \Delta\big(\hat{d}, \mathcal{G}(u_i)\big) \Big) \\ &\quad - \max_{d \in \mathcal{G}(u_i)} \Big(s\big(\theta, u_i, d\big) \Big) \end{aligned} \tag{6}$$

Here, θ is the model parameter. $s(\theta, u_i, d)$ is the DSP score for sentence u_i's derivation d. It is computed by summing LNN score (Equation (2)) and DSN score (Equation (5)):

$$s(\theta, u, d) = LNN_\theta(d) + DSN_\theta(d) \tag{7}$$

$\Delta(\hat{d}, \mathcal{G}(u_i))$ is the structure loss margin, which penalizes derivation $\hat{d}$ more if it deviates more from gold derivations. It is formulated as:

$$\begin{aligned} &\Delta\big(\hat{d}, \mathcal{G}(u_i)\big) \\ &= \sum_{\pi \in \hat{d}} \alpha_s \delta\{\pi \notin \mathcal{G}(u_i)\} + \alpha_t Dist(y(\hat{d}), ref) \end{aligned} \tag{8}$$

The margin includes two parts. For the first part, π is the source span in derivation $\hat{d}$, $\delta\{\cdot\}$ is an indicator function. We use the first part to count the number of source spans in derivation $\hat{d}$, but not in gold derivations. The second part is for target side. $Dist(y(\hat{d}), ref)$ computes the edit-distance between the translation result $y(\hat{d})$ defined by derivation $\hat{d}$ and the reference translation ref. Obviously, this margin can effectively estimate the difference between derivation $\hat{d}$ and gold derivations, both on source side and target side. Note that α_s and α_t are only two hyperparameters for scaling. They are independent of each other, and we set $\alpha_s = 0.1$ and $\alpha_t = 0.1$ respectively.

[1]We investigate the general case here and suppose that one sentence could have several different gold derivations.In the experiment, we only use one gold derivation for simple implementation.

3.2 Learning

As the risk function, Equation (6) is not differentiable. We train the model via the subgradient method (Ratliff et al., 2007; Socher et al., 2013). For parameter θ, the subgriadient of $J(\theta)$ is:

$$\frac{\partial J}{\partial \theta} = \frac{1}{N}\sum_{i}\frac{\partial s(\theta, u_i, \hat{d}_m)}{\partial \theta} - \frac{\partial s(\theta, u_i, d_m)}{\partial \theta} + \lambda\theta$$

where $\hat{d}_m$ is the derivation with the highest DSP score, and d_m denotes the gold derivation with the highest DSP score. We adopt the diagonal variant of AdaGrad (Duchi et al., 2011; Socher et al., 2013) to minimize the risk function for training.

3.3 Training Instances Collection

In order to train the model, we need to collect the gold derivation set $\mathcal{G}(u_i)$ and possible derivation set $\mathcal{A}(u_i)$ for input sentence u_i.

For $\mathcal{G}(u_i)$, we define it by ***force decoding derivation (FDD)***. Basically, FDD refers to the derivation that produces the exact reference translation (single reference in our training data). For example, since "Bush held a talk with Sharon" is the reference of test sentence "布什 与 沙龙 举行 了 会谈", then Figure 1(a) is one of the FDDs. As FDD can produce reference translation, we believe that FDD is of high quality, and take them as gold derivations for training.

For $\mathcal{A}(u_i)$, it should contain all possible derivations of u_i. However, it is too difficult to obtain all derivations. Thus, we use n-best derivations of SMT decoding to simulate the complete derivation space, and take them as the derivations in $\mathcal{A}(u_i)$.

4 Integrating the DSP Model into SMT

To integrate the DSP model into decoding, we take it (named DSP feature) as one of the features in the log-linear framework of SMT. During decoding, the DSP feature is distributed to each node in the derivation structure. For the leaf node, the score in Equation (2), i.e., LNN score, serves as the feature. For the non-leaf node, Equation (4) plays the role. In order to give positive feature value to the log-linear framework (for logarithm), we normalize the DSP scores to [0,1] during decoding. Due to the length limit, we ignore the specific normalization methods here. We just preform some simple transformations (such as adding a constant, computing reciprocal), and convert the scores proportionally to [0,1] at last.

5 Experiments

5.1 Experimental Setup

To verify the effectiveness of our DSP model, we perform experiments on Chinese-to-English translation. The training data contains about 2.1M sentence pairs with about 27.7M Chinese words and 31.9M English words[2]. We train a 5-gram language model by the Xinhua portion of Gigaword corpus and the English part of the training data. We obtain word alignment by GIZA++, and adopt the grow-diag-final-and strategy to generate the symmetric alignment. We use NIST MT 2003 data as the development set, and NIST MT04-08[3] as the test set. We use MERT (Och, 2004) to tune parameters. The translation quality is evaluated by case-insensitive BLEU-4 (Papineni et al., 2002). The statistical significance test is performed by the re-sampling approach (Koehn, 2004). The baseline system is our in-house BTG system (Wu, 1997; Xiong et al., 2006; Zhang and Zong, 2009).

To train the DSP model, we first use Word2Vec[4] toolkit to pre-train the word embedding on large-scale monolingual data. The used monolingual data contains about 1.06B words for Chinese and 1.12B words for English. The dimensionality of our vectors is 50. The detiled training process is as follows:

(1) Using the BTG system to perform force decoding on FBIS part of the bilingual training data[5], and collect the sentences succeeded in force decoding (86,902 sentences in total)[6]. We then collect the corresponding force decoding derivations as gold derivations. Here, we only use the best force decoding derivation for simple implementation. In future, we will try to use multiple force decoding derivations for training.

(2) Collecting the bilingual phrases in the leaf nodes of gold derivations. We train LNN by these phrases via L-BFGS algorithm. Finally, we get 351,448 source phrases to train the source side RAE and 370,948 target phrases to train the target side RAE.

[2]LDC category number : LDC2000T50, LDC2002E18, LDC2003E07, LDC2004T07, LDC2005T06, LDC2002L27, LDC2005T10 and LDC2005T34.

[3]For MT06 and MT08, we only use the part of news data.

[4]https://code.google.com/p/word2vec/

[5]Here we only use the high quality corpus FBIS to guarantee the quality of force decoding derivation.

[6]Many sentence pairs fail in forced decoding due to many reasons, such as reordering limit, noisy alignment, and phrase length limit (Yu et al., 2013).

(3) Decoding the 86902 sentences by the BTG system to get n-best translations and corresponding derivations. The n-best derivations are used to simulate the entire derivation space. We retain at most 200-best derivations for each sentence.

(4) Leveraging force decoding derivations and n-best derivations to train the DSP model. Note that all parameters, including word embedding and parameters in LNN and DSN, are tuned together in this step. It takes about 15 hours to train the entire network using a 16-core, 2.9 GHz Xeon machine.

5.2 Experimental Results

We compare baseline BTG system and the DSP-augmented BTG system in this section. The final translation results are shown in Table 1.

After integrating the DSP model into BTG system, we get significant improvement on all test sets, about 1.0 BLEU points over BTG system on average. This comparison strongly demonstrates that our DSP model is useful and will be a good complement to current translation models.

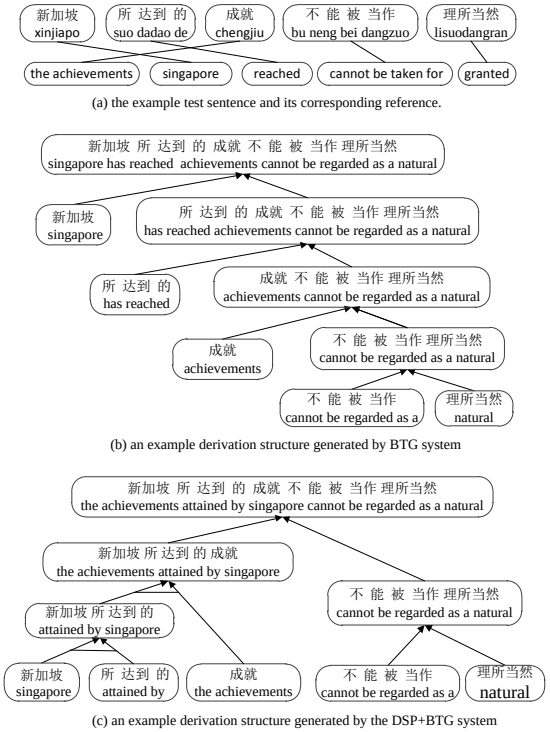
Systems	BLEU(%)				
	MT04	MT05	MT06	MT08	Aver
BTG	36.91	34.69	33.83	27.17	33.15
BTG+DSP	**37.41**	**35.77**	**35.08**	**28.42**	34.17

Table 1: Final translation results. **Bold numbers** denote that the result is significantly better than baseline BTG system ($p < 0.05$). Column "Aver" gives the average BLEU points of the 4 test sets.

To have a better intuition for the effectiveness of our DSP model, we give a case study in Figure 4. It depicts two derivations built by BTG system and BTG+DSP system respectively.

From Figure 4(b), we can see that BTG system yields a bad translation due to the bad derivation structure. In the figure, BTG system makes three mistakes. It attaches candidates [成就; achievements], [所 达到 的; has reached] and [新加坡; singapore] to the big candidate [不能 被 当做 理所当然; cannot be regarded as a natural]. Consequently, the noun phrase "新加坡 所 达到 的 成就" is translated separately, rather than as a whole, leading to a bad translation.

Differently, the DSP model is designed for predicting good derivations. In Figure 4(c), the used translation rules are actually similar to Figure 4(b). However, under a better guidance to build good derivation structure, BTG+DSP system generates a much better translation result than BTG system.



(a) the example test sentence and its corresponding reference.

(b) an example derivation structure generated by BTG system

(c) an example derivation structure generated by the DSP+BTG system

Figure 4: Different derivation structures.

6 Conclusion

In this paper, we explored the method of derivation structure prediction for SMT. To fulfill this task, we have made several major efforts as follows:

(1) We propose a novel derivation structure prediction model based on RNN, including two close and interactive parts: LNN and DSN.

(2) We extend monolingual RNN to bilingual RNN to represent the derivation structure.

(3) We train LNN and DSN by derivations from force decoding. In this way, the DSP model learns a preference to good derivation structures.

Experimental results show that the proposed DSP model improves the translation performance significantly. By this, we verify the effectiveness of derivation structure on indicating good translations. We believe that our work will shed new lights to SMT decoding.

Acknowledgement

We would like to thank the three anonymous reviewers for their valuable comments and suggestions. The research work has been partially funded by the Natural Science Foundation of China under Grant No. 61333018 and 61303181, and the Key Project of Knowledge Innovation Program of Chinese Academy of Sciences.

References

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 961–968, Sydney, Australia, July. Association for Computational Linguistics.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A syntax-directed translator with extended domain of locality. In *Proceedings of AMTA*.

Peng Li, Yang Liu, and Maosong Sun. 2013. Recursive autoencoders for itg-based translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. Spmt: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction. In *Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. 2011a. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of ACL-COLING*, pages 505–512.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1123, Seattle, Washington, USA, October. Association for Computational Linguistics.

Feifei Zhai, Jiajun Zhang, Yu Zhou, and Chengqing Zong. 2013. Unsupervised tree induction for tree-based translation. *Transactions of Association for Computational Linguistics(TACL)*, pages 291–300.

Jiajun Zhang and Chengqing Zong. 2009. A framework for effectively integrating hard and soft syntactic rules into phrase based translation. In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation*, pages 579–588, Hong Kong, December. City University of Hong Kong.

Min Zhang, Hongfei Jiang, Aiti Aw, Haizhou Li, Chew Lim Tan, and Sheng Li. 2008. A tree sequence alignment-based tree-to-tree translation model. In *Proceedings of ACL-08: HLT*, pages 559–567, Columbus, Ohio, June. Association for Computational Linguistics.

Jiajun Zhang, Feifei Zhai, and Chengqing Zong. 2011. Augmenting string-to-tree translation models with fuzzy use of source-side syntax. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 204–215, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.