Transition-Based Syntactic Linearization with Lookahead Features

Ratish Puduppully †[*], Yue Zhang ‡, Manish Shrivastava †
†Kohli Center on Intelligent Systems (KCIS),
International Institute of Information Technology, Hyderabad (IIIT Hyderabad)
‡Singapore University of Technology and Design
ratish.surendran@research.iiit.ac.in yue_zhang@sutd.edu.sg
m.shrivastava@iiit.ac.in

Abstract

It has been shown that transition-based methods can be used for syntactic word ordering and tree linearization, achieving significantly faster speed compared with traditional best-first methods. State-of-the-art transition-based models give competitive results on abstract word ordering and unlabeled tree linearization, but significantly worse results on labeled tree linearization. We demonstrate that the main cause for the performance bottleneck is the sparsity of SHIFT transition actions rather than heavy pruning. To address this issue, we propose a modification to the standard transition-based feature structure, which reduces feature sparsity and allows lookahead features at a small cost to decoding efficiency. Our model gives the best reported accuracies on all benchmarks, yet still being over 30 times faster compared with best-first-search.

1 Introduction

Word ordering is the abstract language modeling task of making a grammatical sentence by ordering a bag of words (White, 2004; Zhang and Clark, 2015; De Gispert et al., 2014; Bohnet et al., 2010; Filippova and Strube, 2007; He et al., 2009), which is practically relevant to text-to-text applications such as summarization (Wann et al., 2009) and machine translation (Blackwood et al., 2010). Zhang (2013) built a discriminative word ordering model, which takes a bag of words, together with optional POS and dependency arcs on a subset of input words, and yields a sentence together with its dependency parse tree that conforms to input syntactic constraints. The system is flexible with respect to input constraints, performing abstract word ordering when no constraints are given, but gives increasingly confined outputs when more POS and dependency relations are specified. It has been applied to syntactic linearization (Song et al., 2014) and machine translation (Zhang et al., 2014).

One limitation of Zhang (2013) is relatively low time efficiency, due to the use of time-constrained best-first-search (White and Rajkumar, 2009) for decoding. In practice, the system can take seconds to order a bag of words in order to obtain reasonable output quality. Recently, Liu et al. (2015) proposed a transition-based model to address this issue, which uses a sequence of state transitions to build the output. The system of Liu et al. (2015) achieves significant speed improvements without sacrificing accuracies when working with unlabeled dependency trees. With labeled dependency trees as input constraints, however, the system of Liu et al. (2015) gives much lower accuracies compared with Zhang (2013).

While the low accuracy can be attributed to heavy pruning, we show that it can be mitigated by modifying the feature structure of the standard transition-based framework, which scores the output transition sequence by summing the scores of each transition action. Transition actions are treated as an *atomic* output component in each feature instance. This works effectively for most structured prediction tasks, including parsing (Zhang and Clark, 2011a). For word ordering, however, transition actions are significantly more complex and sparse compared

[*]Part of the work was done when the author was a visiting student at Singapore University of Technology and Design.

with parsing, which limits the power of the traditional feature model.

We instead break down complex actions into smaller components, merging some components into configuration features which reduces sparsity in the output action and allows flexible lookahead features to be defined according to the next action to be applied. On the other hand, this change in the feature structure prevents legitimate actions to be scored simultaneously for each configuration state, thereby reducing decoding efficiency. Experiments show that our method is slightly slower compared with Liu et al. (2015), but achieves significantly better accuracies. It gives the best results for all standard benchmarks, being over thirty times faster than Zhang (2013). The new feature structures can be applied to other transition-based systems also.

2 Transition-based linearization

Liu et al. (2015) uses a transition-based model for word ordering, building output sentences using a sequence of state transitions. Instead of scoring output syntax trees directly, it scores the transition action sequence for structural disambiguation. Liu et al.'s transition system extends from transition-based parsers (Nivre and Scholz, 2004; Chen and Manning, 2014), where a *state* consists of a *stack* to hold partially built outputs. Transition-based parsers use a *queue* to maintain input word sequences. However, for word ordering, the input is a set without order. Accordingly, Liu et al. uses a *set* to maintain the input. The transition actions are:

- SHIFT-*Word*-POS, which removes *Word* from the *set*, assigns POS to it and pushes it onto the *stack* as the top word S_0;
- LEFTARC-LABEL, which removes the second top of *stack* S_1 and builds a dependency arc $S_1 \xleftarrow{LABEL} S_0$;
- RIGHTARC-LABEL, which removes the top of *stack* S_0 and builds a dependency arc $S_1 \xrightarrow{LABEL} S_0$.

Using the state transition system, the bag of words $\{$*John, loves, Mary*$\}$ can be ordered by (SHIFT-*John*-NNP, SHIFT-*loves*-VBZ, LEFTARC-SBJ, SHIFT-*Mary*-NNP, RIGHTARC-OBJ).

Liu et al. (2015) use a discriminative perceptron model with beam search (Zhang and Clark, 2011a), designing decoding algorithms that accommodate flexible constraints. The features include word(w), pos(p) and dependency label(l) information of words on the stack (S_0, S_1, ... from the top). For example, the word on top of stack is S_0w and the POS of the stack top is S_0p. The full set of feature templates can be found in Table 2 of Liu et al. (2015), reproduced here in Table 1. These templates are called *configuration features*. When instantiated, they are combined with each legal output *action* to score the action. Therefore, actions are atomic in feature instances.

Unigrams
S_0w; S_0p; $S_{0,l}w$; $S_{0,l}p$; $S_{0,r}w$; $S_{0,r}p$; $S_{0,l2}w$; $S_{0,l2}p$; $S_{0,r2}w$; $S_{0,r2}p$;
S_1w; S_1p; $S_{1,l}w$; $S_{1,l}p$; $S_{1,r}w$; $S_{1,r}p$; $S_{1,l2}w$; $S_{1,l2}p$; $S_{1,r2}w$; $S_{1,r2}p$;
Bigram
$S_0wS_{0,l}w$; $S_0wS_{0,l}p$; $S_0pS_{0,l}w$; $S_0pS_{0,l}p$; $S_0wS_{0,r}w$; $S_0wS_{0,r}p$; $S_0pS_{0,r}w$; $S_0pS_{0,r}p$;
$S_1wS_{1,l}w$; $S_1wS_{1,l}p$; $S_1pS_{1,l}w$; $S_1pS_{1,l}p$; $S_1wS_{1,r}w$; $S_1wS_{1,r}p$; $S_1pS_{1,r}w$; $S_1pS_{1,r}p$;
S_0wS_1w; S_0wS_1p; S_0pS_1w; S_0pS_1p
Trigram
$S_0wS_0pS_{0,l}w$; $S_0wS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,l}p$; $S_0pS_{0,l}wS_{0,l}p$; $S_0wS_0pS_{0,r}w$; $S_0wS_{0,l}wS_{0,r}p$; $S_0wS_0pS_{0,r}p$; $S_0pS_{0,r}wS_{0,r}p$;
$S_1wS_1pS_{1,l}w$; $S_1wS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,l}p$; $S_1pS_{1,l}wS_{1,l}p$; $S_1wS_1pS_{1,r}w$; $S_1wS_{1,l}wS_{1,r}p$; $S_1wS_1pS_{1,r}p$; $S_1pS_{1,r}wS_{1,r}p$;
Linearization
w_0; p_0; $w_{-1}w_0$; $p_{-1}p_0$; $w_{-2}w_{-1}w_0$; $p_{-2}p_{-1}p_0$; $S_{0,l}wS_{0,l2}w$; $S_{0,l}pS_{0,l2}p$; $S_{0,r2}wS_{0,r}w$; $S_{0,r2}pS_{0,r}p$; $S_{1,l}wS_{1,l2}w$; $S_{1,l}pS_{1,l2}p$; $S_{1,r2}wS_{1,r}w$; $S_{1,r2}pS_{1,r}p$;

Table 1: Base feature templates.

Formally, given a configuration C, the score of a possible action a is calculated as:

$$Score(a) = \vec{\theta} \cdot \Phi(\vec{C}, a),$$

where $\vec{\theta}$ is the model parameter vector of the model and $\Phi(\vec{C}, a)$ denotes a sparse feature vector that consists of features with *configuration* and *action* components i.e $\Phi(\vec{C}, a)$ is sparse. $\vec{\theta}$ has to be loaded for each a.

For efficiency considerations and following transition-based models, Liu et al. (2015) scores all possible actions given a configuration simultaneously. This is effectively the same as formulating the score into

$$Score(a) = \vec{\theta_a} \cdot \Phi(\vec{C}), a \in A.$$

Here A is the full set of actions and $\Phi(\vec{C})$ is fixed, and $\vec{\theta_a}$ for all a can be loaded simultaneously. In a hash-based parameter model, it significantly improves the time efficiency.

3 Feature structure modification

3.1 Two limitations of the baseline model

There are two major limitations in the feature structure of Liu et al. (2015). First, the SHIFT actions, which consist of the word to shift and its POS, are highly sparse. Since the action is combined with all configuration features, there will be no active feature for disambiguating the shift actions for OOV words. This issue does not exist in transition-based parsers because words are not a part of their transition actions. Second, input constraints are not leveraged by the feature model. Although the dependency relations of the word to shift can be given as inputs, they are used only as constraints to the decoder, but not as features to guide the shift action. Such lookahead information on the to-be-shifted word can be highly useful for disambiguation.

For example, consider the bag of words {*John*, *loves*, *Mary*}. Without constraints, both '*John loves Mary*' and '*Mary loves John*' are valid word ordering results. However, given the constraint (*John*, SBJ, *loves*), the correct answer is reduced to the former. The first action to build the two examples are (SHIFT-*John*-NNP) and (SHIFT-*Mary*-NNP), respectively. According to Liu et al.'s feature model, there is no feature to disambiguate the first SHIFT action if both '*John*' and '*Mary*' are OOV words. The system has to maintain both hypotheses and rely on the search algorithm to disambiguate them after the dependency arcs (*John*, SBJ, *loves*) and (*Mary*, OBJ, *loves*) are built. However, given the syntactic constraint that '*John*' is the subject, the disambiguation can be done right when performing the first SHIFT action. This requires the dependency arc label to be extracted for the word to shift e.g.(*John, Mary*), which is a lookahead feature. In addition, the OOV word '*John*' must be excluded from the feature instance, which implies that the SHIFT-*John*-NNP action must be simplified.

set of label and POS of child nodes of L
$L_{cls}; L_{clns}; L_{cps}; L_{cpns};$ $S_0wL_{cls}; S_0pL_{cls}; S_1wL_{cls}; S_1pL_{cls};$ $S_0wL_{clns}; S_0pL_{clns}; S_1wL_{clns}; S_1pL_{clns};$ $S_0wL_{cps}; S_0pL_{cps}; S_1wL_{cps}; S_1pL_{cps};$ $S_0wL_{cpns}; S_0pL_{cpns}; S_1wL_{cpns}; S_1pL_{cpns};$
set of label and POS of siblings of L
$L_{sls}; L_{slns}; L_{sps}; L_{spns};$ $S_0wL_{sls}; S_0pL_{sls}; S_1wL_{sls}; S_1pL_{sls};$ $S_0wL_{slns}; S_0pL_{slns}; S_1wL_{slns}; S_1pL_{slns};$ $S_0wL_{sps}; S_0pL_{sps}; S_1wL_{sps}; S_1pL_{sps};$ $S_0wL_{spns}; S_0pL_{spns}; S_1wL_{spns}; S_1pL_{spns};$
parent label, POS and word of L
$L_{ps}L_{lp}; L_{ps}L_{pp}; L_{ps}L_{wp};$ $S_0wL_{ps}L_{lp}; S_0pL_{ps}L_{lp}; S_1wL_{ps}L_{lp}; S_1pL_{ps}L_{lp};$ $S_0wL_{ps}L_{pp}; S_0pL_{ps}L_{pp}; S_1wL_{ps}L_{pp}; S_1pL_{ps}L_{pp};$ $S_0wL_{ps}L_{wp}; S_0pL_{ps}L_{wp}; S_1wL_{ps}L_{wp}; S_1pL_{ps}L_{wp};$
set of label and POS of child nodes of S_0
$S_{cls}; S_{clns}; S_{cps}; S_{cpns};$ $S_0wS_{cls}; S_0pS_{cls}; S_1wS_{cls}; S_1pS_{cls};$ $S_0wS_{clns}; S_0pS_{clns}; S_1wS_{clns}; S_1pS_{clns};$ $S_0wS_{cps}; S_0pS_{cps}; S_1wS_{cps}; S_1pS_{cps};$ $S_0wS_{cpns}; S_0pS_{cpns}; S_1wS_{cpns}; S_1pS_{cpns};$
set of label and POS of siblings of S_0
$S_{sls}; S_{slns}; S_{sps}; S_{spns};$ $S_0wS_{sls}; S_0pS_{sls}; S_1wS_{sls}; S_1pS_{sls};$ $S_0wS_{slns}; S_0pS_{slns}; S_1wS_{slns}; S_1pS_{slns};$ $S_0wS_{sps}; S_0pS_{sps}; S_1wS_{sps}; S_1pS_{sps};$ $S_0wS_{spns}; S_0pS_{spns}; S_1wS_{spns}; S_1pS_{spns};$
parent label and POS of S_0
$S_{ps}S_{lp}; S_{ps}S_{pp};$ $S_0wS_{ps}S_{lp}; S_0pS_{ps}S_{lp}; S_1wS_{ps}S_{lp}; S_1pS_{ps}S_{lp};$ $S_0wS_{ps}S_{pp}; S_0pS_{ps}S_{pp}; S_1wS_{ps}S_{pp}; S_1pS_{ps}S_{pp};$

Table 2: Lookahead feature templates

As a second example, information about dependents can also be useful for disambiguating SHIFT actions. In the above case, the fact that the *subject* has not been shifted onto the stack can be a useful indicator for not shifting the verb '*loves*' onto the stack in the beginning. Inspired by the above, we exploit a range of lookahead features from syntactic constraints.

3.2 New feature structure for SHIFT actions

We modify the feature structure of Liu et al. (2015) by breaking down the SHIFT-*Word*-*POS* action into three components, namely SHIFT, *Word* and *POS*, using only the action type SHIFT as the output action component in feature instances, while combin-

	no pos no dep		50% pos no dep		all pos no dep		no pos 50% dep		50% pos 50% dep		all pos 50% dep		no pos all dep		50% pos all dep		all pos all dep	
	BL	SP	BL	SP	BL	SP	BL	SP	BL	SP	BL	SP	BL	SP	BL	SP	BL	SP
Z13	42.9	4872	43.4	4856	44.7	4826	50.5	4790	51.4	4737	52.2	4720	73.3	4600	74.7	4431	76.3	4218
L15	47.5	155	47.9	119	48.8	74	54.8	132	55.2	91	56.2	41	77.8	40	79.1	28	81.1	22
Ours	48.0	175	49.0	156	51.5	148	59.0	144	62.0	160	67.1	171	82.8	62	86.2	68	89.9	70

Table 3: Development partial-tree linearization results. *BL* – BLEU score; *SP* – number of milliseconds per sentence. Z13 – best-first system of Zhang (2013) and L15 – transition-based system of Liu et al. (2015).

ing *Word* and *POS* with other configuration features to form a set of lookahead features.

For example, consider the configuration feature S_0w, which captures the word on the top of the stack. Under the feature structure of Liu et al., it is combined with each possible action to form features for scoring the action. As a result, for scoring the action SHIFT-Lw-Lp, S_0w is instantiated into S_0w-SHIFT-Lw-Lp, where Lw is the word to shift and Lp is its POS. Under our new feature structure, the action component is reduced to SHIFT only, while Lw and Lp should be used in lookahead features. Now a effectively equivalent configuration feature to Liu et al.'s S_0w is S_0w-Lw-Lp, with the lookahead Lw and Lp. It gives S_0w-Lw-Lp-SHIFT when combined with the action SHIFT.

This new feature structure reformulates the SHIFT action features only. The LEFTARC/ RIGHTARC actions remain LEFTARC/ RIGHTARC-LABEL since they are not sparse. Note that the change is in the *action features* rather than the *actions* themselves. Given the bag of words $\{$*John*, *loves*, *Mary*$\}$, the action SHIFT-*John*-NNP is still different from the action SHIFT-*Mary*-NNP. However, the *action component* of the features becomes SHIFT only, and the words *John*/ *Mary* must be used as lookahead *configuration features* for their disambiguation.

The new feature structure can reduce feature sparsity by allowing lookahead features without word information. For example, a configuration feature S_0w-Lp, which contains only the stack top word and the POS of the lookahead word, can still fire even if the word to shift is OOV, thereby disambiguating OOV words of different POS. In addition, the lookahead Lw and Lp do not have to be combined with every other configuration feature, as with Liu et al. (2015), thereby allowing more flexible feature combination and a leaner model.

3.3 The new features

The new feature structure includes two types of features. The first is the same feature set as Liu et al. (2015), but with the SHIFT action component not having *Word* and *POS* information. We call this type of features as *base features*. The second is a set of *lookahead features*, which are shown in Table 2. Here L_{cls} represents set of arc labels on child nodes (of the word L to shift) that have been shifted on to the stack, L_{clns} represents set of labels on child nodes that have not been shifted, L_{sls} the label set of shifted sibling nodes, L_{slns} the label set of unshifted sibling nodes, L_{cps} the POS set of shifted child nodes, L_{cpns} the POS set of unshifted child nodes, L_{sps} the POS set of shifted sibling nodes and L_{spns} the POS set of unshifted sibling nodes. L_{ps} is a binary feature indicating if the parent has been shifted. L_{lp} represents label on the parent, L_{pp} POS of parent and L_{wp} the parent word form. We define similar lookahead features for S_0. These features are instantiated only for SHIFT actions.

The new feature structure prevents all possible actions from being scored simultaneously, because the lookahead *Word* and *POS* are now in configuration features, rather than output actions, making it necessary to score the shifting of different words or POS separately. This leads to reduced search speed. Nevertheless, our experiments show that they give a desirable tradeoff between efficiency and accuracy.

Note that the new features are much less than a full Cartesian product of lookahead features and the original features. This is a result of manual feature engineering, which allows similar accuracies to be achieved using a much smaller model, thereby increasing the time efficiency.

	unlabeled			labeled
	no pos no dep	all pos no dep	all pos all dep	all pos all dep
W09	-	33.7	-	-
Z11	-	40.1	-	-
Z13	44.7	46.8	76.2	89.3
L15	49.4	50.8	82.3	82.9
This paper	**50.5**	**53.0**	**91.0**	**91.8**

Table 4: Final results. W09 – Wann et al. (2009), Z11 – Zhang and Clark (2011b)

	Fully constrained output
ref.	The spinoff also will compete with Fujitsu
L15	The spinoff with Fujitsu compete also will
Ours	The spinoff also will compete with Fujitsu
ref.	Dr. Talcott led a team of researchers from the National Cancer Institute .
L15	a team of researchers from the National Cancer Institute led Dr. Talcott .
Ours	Dr. Talcott led a team of researchers from the National Cancer Institute .

Table 5: Example outputs.

4 Experiments

Following previous work we conduct experiments on the Penn TreeBank (PTB), using Wall Street Journal sections 2-21 for training, 22 for development and 23 for testing. Gold-standard dependency trees are derived from bracketed sentences using Penn2Malt, and base noun phrases are treated as a single word. The BLEU score is used to evaluate the performance of linearization.

Table 4 shows a difference in scores between transition-based linearization system of Liu et al. (2015) (L15) and best-first system of Zhang (2013) (Z13). L15 performs better for word ordering with unlabeled dependency arcs, but poorly for the task of labeled syntactic linearization.

Table 3 shows a series of development experiments comparing our system with Z13 and L15. We vary the amount of input syntactic constraints by randomly sampling from POS and dependency labels of the development set. Our system gives consistently higher accuracies when compared with both Z13 and L15. Compared to L15, the increase in scores for unconstrained word ordering is due to the introduction of reduced feature sparsity. The improvements on tree linearization tasks involving partial to full dependency constraints are also due to lookahead features that leverage tree information to reduce ambiguity early. Though slower than L15, our system is over 30 times faster compared to Z13.

We compare final test scores with previous methods in the literature in Table 4. Our system improves upon the previous best scores by 8.7 BLEU points for the task of unlabeled syntactic linearization. For the task of labeled syntactic linearization, we achieve the score of 91.8 BLEU points, the highest results reported so far.

Table 5 contains examples of fully constrained output . In the first example '*will*' is the ROOT node with two child nodes '*also*' and '*compete*'. Lookahead feature for child dependency labels L_{cls}, L_{clns} on the node '*will*' can help order the segment '*also will compete*' correctly in our system. Without such features, the system of L15 yields an output that starts with '*The spinoff with Fujitsu*' which is locally fluent, but leaving the words '*also*' and '*will*' difficult to handle. In the second example, '*Dr. Talcott*' is OOV. Hence system of L15 is not able to score it and thus order it correctly. Our system makes use of both POS and dependency label of '*Dr. Talcott*' to order it correctly.

5 Conclusion

We identified a feature sparsity issue in state-of-the-art transition-based word ordering, proposing a solution by redefining the feature structure and introducing lookahead features. The new method gives the best accuracies on a set of benchmarks, which show that transition-based methods are a fast and accurate choice for syntactic linearization. Future work include the testing of this model in a linearization shared task (Belz et al., 2011) and investigating the integration of large scale training data (Zhang et al., 2012; Liu and Zhang, 2015).

We release our source code under GPL at `https://github.com/SUTDNLP/ZGen/releases/tag/v0.2`.

Acknowledgments

We thank Yijia Liu for helpful discussions and for sharing the Latex templates and the anonymous reviewers for their constructive comments. This work was supported by the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301.

References

Anja Belz, Michael White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. 2011. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the 13th European workshop on natural language generation*, pages 217–226. Association for Computational Linguistics.

Graeme Blackwood, Adrià De Gispert, and William Byrne. 2010. Fluency constraints for minimum bayes-risk decoding of statistical machine translation lattices. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 71–79. Association for Computational Linguistics.

Bernd Bohnet, Leo Wanner, Simon Mille, and Alicia Burga. 2010. Broad coverage multilingual deep sentence generation with a stochastic multi-level realizer. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 98–106. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1:740–750.

A De Gispert, M Tomalin, and W Byrne. 2014. Word ordering with phrase-based grammars. *14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014*, pages 259–268.

Katja Filippova and Michael Strube. 2007. Generating constituent order in german clauses. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 320.

Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. 2009. Dependency based chinese sentence realization. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2*, pages 809–816.

Jiangming Liu and Yue Zhang. 2015. An empirical comparison between n-gram and syntactic language models for word ordering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 369–378, Lisbon, Portugal, September. Association for Computational Linguistics.

Yijia Liu, Yue Zhang, Wanxiang Che, and Bing Qin. 2015. Transition-based syntactic linearization. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 113–122.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of english text. In *Proceedings of the 20th international conference on Computational Linguistics*, page 64. Association for Computational Linguistics.

Linfeng Song, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1522–1528.

Stephen Wann, Mark Dras, Robert Dale, and Cécile Paris. 2009. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 852–860.

Michael White and Rajakrishnan Rajkumar. 2009. Perceptron reranking for ccg realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 410–419. Association for Computational Linguistics.

Michael White. 2004. Reining in ccg chart realization. In *Natural Language Generation*, pages 182–191. Springer Berlin Heidelberg.

Yue Zhang and Stephen Clark. 2011a. Syntactic processing using the generalized perceptron and beam search. *Computational linguistics*, 37(1):105–151.

Yue Zhang and Stephen Clark. 2011b. Syntax-based grammaticality improvement using ccg and guided search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2015. Discriminative syntax-based word ordering for text generation. *Computational Linguistics*, 41(3):503–538.

Yue Zhang, Graeme Blackwood, and Stephen Clark. 2012. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 736–746. Association for Computational Linguistics.

Yue Zhang, Kai Song, Linfeng Song, Jingbo Zhu, and Qun Liu. 2014. Syntactic smt using a discriminative text generation model. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 177–182, Doha, Qatar, October. Association for Computational Linguistics.

Yue Zhang. 2013. Partial-tree linearization: generalized word ordering for text synthesis. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2232–2238. AAAI Press.