Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Partial-Tree Linearization: Generalized Word Ordering for Text Synthesis

Yue Zhang
Singapore University of Technology and Design
20 Dover Drive, Singapore 138682
yue_zhang@sutd.edu.sg

Abstract

We present partial-tree linearization, a generalized word ordering (i.e. ordering a set of input words into a grammatical and fluent sentence) task for text-to-text applications. Recent studies of word ordering can be categorized into either abstract word ordering (no input syntax except for POS) or tree linearization (input words are associated with a full unordered syntax tree). Partial-tree linearization covers the whole spectrum of input between these two extremes. By allowing POS and dependency relations to be associated with any subset of input words, partial-tree linearization is more practical for a dependency-based NLG pipeline, such as transfer-based MT and abstractive text summarization. In addition, a partial-tree linearizer can also perform abstract word ordering and full-tree linearization. Our system achieves the best published results on standard PTB evaluations of these tasks.

1 Introduction

Word ordering is a fundamental problem in natural language generation (NLG). It can be performed as a final linearization step in an NLG pipeline [Reiter and Dale, 1997], or as an integrated functionality in an application system that performs NLG. The latter case is most commonly seen in statistical machine translation (SMT) [Koehn *et al.*, 2003; Chiang, 2007], an important reason being the high (NP-hard without constraints) computational complexity of independent word ordering. The current state-of-the-art SMT systems avoid word permutation by treating target word *ordering* as source word *reordering*: a target word order must be derived from the source word order according to reordering rules and constraints that allow practical decoding speed.

On the other hand, performing word ordering separately in a pipeline has many potential advantages. For SMT, it offers better modularity between adequacy (translation) and fluency (linearization), and can potentially improve target grammaticality for syntactically different languages (e.g. Chinese and English). More importantly, a stand-alone word ordering component can be applied to a wide range of NLG tasks, including not only transfer-based machine translation [Chang and Toutanova, 2007], but also abstractive text summarization [Barzilay and McKeown, 2005] and grammar correction [Lee and Seneff, 2006]. All these fields are still growing, to which better NLG is likely to bring new advances.

Some recent work begins to study word ordering as a focused problem. Given an input set of words, the task is to order them into a grammatical and fluent sentence. Wan *et al.* [2009] propose two algorithms to order an input bag of words by building a dependency tree. He *et al.* [2009] study the recovery of word order for a given dependency tree. Zhang and Clark [2011] apply learning-guided search to order an input bag of words using the combinatory categorial grammar (CCG). These methods can be viewed as a step towards an independent linearization system for NLG problems, but with simplifications that are overly strong for practical application. For example, all these methods assume that the gold-standard parts-of-speech (POS) are given for all input words.

We follow Wan *et al.* [2009] and He *et al.* [2009], and choose the dependency grammar for syntax-based word ordering. Being a light-weight grammar formalism that captures lexical semantic relations, the dependency grammar is widely used by NLP applications, and is one of the most commonly used grammar for SMT. The work of Wan *et al.* [2009] and He *et al.* [2009] can be viewed as solving a general-purpose linearization problem with two extreme input conditions. Both find an output word order by building a dependency tree. While the former assumes that no dependency relations are given for any input words, the latter assumes that all gold-standard dependencies are given. In practice, the input information is likely to lie in the spectrum between these two extreme conditions. Take transfer-based Chinese-English translation for example. Many types of dependency relations, including the subject, the object, and the noun modifier, can be mapped directly from the source to the target. However, language-specific relations such as the ba-construction in Chinese cannot be projected onto English. This results in an input set of words from lexical transfer, together with some dependency relations between them: a situation which neither Wan *et al.* [2009] nor He *et al.* [2009] can handle.

In this paper, we demonstrate the possibility of *partial-tree linearization*, a word ordering task that accepts any amount of input dependencies, ranging from none to a full unordered tree. Our system is also flexible on input POS, allowing both POS and dependencies to be defined on any subset of input

words. More input POS and dependencies lead to more specified output. As a result, our system is more practical than previous work for the aforementioned NLG applications. In addition, being a generalized word ordering system, a partial-tree linearizer can perform both abstract word ordering (no input dependencies) and full tree linearization (full unordered tree), given corresponding inputs.

The flexibilities of input syntax raise challenges to search. We adopt and make improvements to the learning-guided search framework of Zhang *et al.* [2012], which gives the current state-of-the-art results for word ordering with gold-standard input POS. Applying the framework also allows us to make a direct comparison between the dependency grammar and CCG for word ordering. Zhang *et al.* [2012] showed that a system using CCG and learning-guided search outperforms the system of Wan *et al.* [2009], which uses the dependency grammar and different search algorithms. However, the effects of CCG and learning-guided search in the improvement have not been studied separately. By comparison with Zhang *et al.* [2012] and Wan *et al.* [2009], respectively, we found that the learning-guided search framework was the main reason for their improvements, while the dependency grammar is as useful as CCG for word ordering.

We perform a comprehensive set of evaluations to our system. When no input POS or dependencies are given, our system performs abstract word ordering (without input POS), a challenging task for language modeling. We report the first results that we are aware of on this evaluation. When gold-standard input POS are given, we compare our system to the work of Wan *et al.* [2009], Zhang and Clark [2011], and Zhang *et al.* [2012], and our system gives the best published results, 3 BLEU points higher than the previous best results. When both gold-standard input POS and dependencies are given, our system performs the task of tree linearization [He *et al.*, 2009; Filippova and Strube, 2007; Belz *et al.*, 2011], for which we report the best accuracies on the standard Penn Treebank (PTB) data. In addition, we report results with various proportions of randomly selected input POS and dependencies to simulate practical conditions.

2 Task specification

The goal of our work is to develop a general-purpose linearization system for practical text-to-text applications. We assume that a set of words and base phrases (e.g. proper noun phrases), together with some POS and dependency information about these words, are provided by the NLG pipeline prior to linearization. The task of our system is to compose a grammatical and fluent sentence using these available inputs.

The additional syntactic information is important in specifying the target output, without which the word ordering problem becomes a pure language modeling task. Such information is collected from the source text, and carried over to the target text. It serves to bridge the source analysis and target composition components in a text-to-text application. We choose POS and dependency relations due to their wide usage for natural language analysis.

An important difference between our work and previous work is that we treat POS and dependency relations as *optional*, which we find necessary for both machine translation and summarization, because in practice the analysis component is not guaranteed to provide POS and dependencies for each input word. Hence we call our task *partial-tree linearization*, a novel yet practically important task for NLG.

```
Input  : words – a set of words with optional POS
         and dependencies.
Output: An ordered sentence with its syntax tree.
agenda ← NewPriorityQueue();
for word ∈ words do
    if HasPOS(word) then
        EnQueue(agenda, word);
    else
        for word′ ∈ AssignPOS(word) do
            EnQueue(agenda, word′);
        end
    end
end
chart ← NewBeam();
goals ← {};
while not TimeOut() do
    h ← DeQueue(agenda);
    if GoalHypothesis(h) then
        AddToSet(goals, h);
    else
        for h_c ∈ chart do
            if NoCollision(h, h_c) then
                h′ ← Combine(h, h_c, HeadLeft);
                if h′ then EnQueue(agenda, h′);
                ;
                h′ ← Combine(h, h_c, HeadRight);
                if h′ then EnQueue(agenda, h′);
                ;
                h′ ← Combine(h_c, h, HeadLeft);
                if h′ then EnQueue(agenda, h′);
                ;
                h′ ← Combine(h_c, h, HeadRight);
                if h′ then EnQueue(agenda, h′);
                ;
            end
        end
        AddToBeam(chart, h);
    end
end
if Empty(goals) then
    return Default(chart);
else
    return BestInSet(goals);
end
```

Algorithm 1: The search algorithm.

3 Approach

Our task is NP-hard and computationally very challenging. We apply time-constrained best-first search, which

has been used for CCG-based NLG problems [White, 2004; White and Rajkumar, 2009; Zhang *et al.*, 2012]. The intuition is to explore the most confident part of the search space within a timeout limit, and return the best answer.

3.1 Search

For our task, a search hypothesis is an ordered phrase or sentence, together with its dependency analysis. Hypotheses are constructed bottom-up. Given an input set of words, a set of initial hypotheses are constructed by assigning POS-tags to each word. Hypotheses are expanded by combination with each other to form larger hypotheses, and the best-first search strategy is applied to find a hypothesis that meets the goal condition. For all experiments in this paper, we define a goal hypothesis as one that covers all input words.

Pseudocode of the search algorithm is shown in Algorithm 1, where agenda is a priority queue for best-first search, and chart is a beam to record the k-best accepted hypotheses. The size of the beam k controls the efficiency of hypothesis expansion, and we set $k = 16n$ in our experiments, with n being the number of input words. Lines 1 – 12 perform initialization. For each input word, we check whether it has been associated with an input POS. If so, we put it directly onto agenda as an initial hypothesis; otherwise we assign all possible POS-tags to the word to obtain a set of initial hypotheses. Following the literature on POS-tagging [Collins, 2002], we compile a tag-dictionary for words occurring five or more times in the training data, so that only dictionary POS are assigned to these words. For words out of the tag-dictionary, all POS in the tagset are assigned.

Lines 13 – 32 are the main loop for best-first search. At each step, the highest-scored hypothesis h is taken from agenda. If h meets the goal condition, it is added to the set goals. Otherwise h is expanded by combination with every accepted hypotheses h_c in chart. `NoCollision` checks if h and h_c do not contain overlapping words (i.e. they do not contain a word more than the number of times the word occurs in the input). If there is no collision, the two hypotheses are combined by concatenating their surface strings, and building a dependency arc between their root words. There are four different ways to combine h and h_c: they can be placed in two different orders, and in each case the newly added dependency arc can take two directions. In case the input contains dependency relations that specify the head of the root word of h or h_c, combinations that disagree with the input dependencies are disallowed (by setting $h' = \mathsf{NULL}$). After being expanded, h is put onto chart as an accepted hypothesis.

The main loop is executed until the timeout limit is reached. Then in lines 33 – 37, the highest-scored hypothesis in goals is taken as the output. In case goals is empty, a default output is constructed by greedy combination of accepted hypotheses in the chart in order of decreasing sizes.

3.2 Model and training

We adopt the online learning framework of Zhang *et al.* [2012], but use a different scoring function. The score of a hypothesis h is:

$$score(h) = \frac{\Phi(h) \cdot \vec{\theta}}{|h|},$$

dependency syntax
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{NORM}(size)$,
$\text{WORD}(h) \cdot \text{NORM}(size)$, $\text{POS}(h) \cdot \text{NORM}(size)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot dir$, $\text{WORD}(h) \cdot dir$, $\text{POS}(h) \cdot dir$,
$\text{WORD}(m) \cdot \text{POS}(m) \cdot dir$, $\text{WORD}(m) \cdot dir$, $\text{POS}(m) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot dist$, $\text{WORD}(h) \cdot dist$, $\text{POS}(h) \cdot dist$,
$\text{WORD}(m) \cdot \text{POS}(m) \cdot dist$, $\text{WORD}(m) \cdot dist$, $\text{POS}(m) \cdot dist$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{WORD}(m) \cdot \text{POS}(m) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{WORD}(m) \cdot \text{POS}(m) \cdot dist$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{POS}(m) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{POS}(m) \cdot dist$,
$\text{POS}(h) \cdot \text{WORD}(m) \cdot \text{POS}(m) \cdot dir$,
$\text{POS}(h) \cdot \text{WORD}(m) \cdot \text{POS}(m) \cdot dist$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot dir$, $\text{POS}(h) \cdot \text{POS}(m) \cdot dist$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(b) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(h-1) \cdot \text{POS}(m) \cdot \text{POS}(m+1) \cdot dir$ $(h > m)$,
$\text{POS}(h) \cdot \text{POS}(h+1) \cdot \text{POS}(m) \cdot \text{POS}(m-1) \cdot dir$ $(h < m)$,
$\text{WORD}(h) \cdot \text{POS}(m) \cdot \text{POS}(m_l) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(m) \cdot \text{POS}(m_r) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(m_l) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(m_r) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(s) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(s) \cdot dir$, $\text{POS}(m) \cdot \text{POS}(s) \cdot dir$,
$\text{WORD}(h) \cdot \text{WORD}(s) \cdot dir$, $\text{WORD}(m) \cdot \text{WORD}(s) \cdot dir$,
$\text{POS}(h) \cdot \text{WORD}(s) \cdot dir$, $\text{POS}(m) \cdot \text{WORD}(s) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(s) \cdot dir$, $\text{WORD}(m) \cdot \text{POS}(s) \cdot dir$,
$\text{WORD}(h) \cdot \text{POS}(m) \cdot \text{POS}(s) \cdot \text{POS}(s_2) \cdot dir$,
$\text{POS}(h) \cdot \text{POS}(m) \cdot \text{POS}(s) \cdot \text{POS}(s_2) \cdot dir$,
dependency syntax for completed words
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{WORD}(h_l) \cdot \text{POS}(h_l)$, $\text{POS}(h) \cdot \text{POS}(h_l)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{POS}(h_l)$, $\text{POS}(h) \cdot \text{WORD}(h_l) \cdot \text{POS}(h_l)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{WORD}(h_r) \cdot \text{POS}(h_r)$, $\text{POS}(h) \cdot \text{POS}(h_r)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{POS}(h_r)$, $\text{POS}(h) \cdot \text{WORD}(h_r) \cdot \text{POS}(h_r)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{LVAL}(h)$, $\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{RVAL}(h)$,
$\text{WORD}(h) \cdot \text{POS}(h) \cdot \text{LVAL}(h) \cdot \text{RVAL}(h)$,
$\text{POS}(h) \cdot \text{LVAL}(h)$, $\text{POS}(h) \cdot \text{RVAL}(h)$,
$\text{POS}(h) \cdot \text{LVAL}(h) \cdot \text{RVAL}(h)$,
surface string patterns
$\text{WORD}(B-1) \cdot \text{WORD}(B)$, $\text{POS}(B-1) \cdot \text{POS}(B)$,
$\text{WORD}(B-1) \cdot \text{POS}(B)$, $\text{POS}(B-1) \cdot \text{WORD}(B)$,
$\text{WORD}(B-1) \cdot \text{WORD}(B) \cdot \text{WORD}(B+1)$,
$\text{WORD}(B-2) \cdot \text{WORD}(B-1) \cdot \text{WORD}(B)$,
$\text{POS}(B-1) \cdot \text{POS}(B) \cdot \text{POS}(B+1)$,
$\text{POS}(B-2) \cdot \text{POS}(B-1) \cdot \text{POS}(B)$,
$\text{POS}(B-1) \cdot \text{WORD}(B) \cdot \text{POS}(B+1)$,
$\text{POS}(B-2) \cdot \text{WORD}(B-1) \cdot \text{POS}(B)$,
surface string patterns for complete sentences
$\text{WORD}(0)$, $\text{WORD}(0) \cdot \text{WORD}(1)$,
$\text{WORD}(size-1)$, $\text{WORD}(size-1) \cdot \text{WORD}(size-2)$,
$\text{POS}(0)$, $\text{POS}(0) \cdot \text{POS}(1)$, $\text{POS}(0) \cdot \text{POS}(1) \cdot \text{POS}(2)$,
$\text{POS}(size-1)$, $\text{POS}(size-1) \cdot \text{POS}(size-2)$,
$\text{POS}(size-1) \cdot \text{POS}(size-2) \cdot \text{POS}(size-3)$,

Table 1: Feature templates. Indices on the surface string: h – head on newly added arc; m – modifier on arc; s – nearest sibling of m; b – any index between h and m; h_l, h_r – left/rightmost modifier of h; m_l, m_r – left/rightmost modifier of m; s_2 – nearest sibling of s towards h; B – boundary between the conjoined phrases (index of the first word of the right phrase). Variables: *dir* – direction of the arc, normalized by NORM; *dist* – distance (h-m), normalized; *size* – number of words in the dependency tree. Functions: WORD – word at index; POS – POS at index; NORM – normalize absolute value into 1, 2, 3, 4, 5, $(5, 10]$, $(10, 20]$, $(20, 40]$, 40+.

	Sections	Sentences	Words
Training	2–21	39,832	950,028
Development	22	1,700	40,117
Test	23	2,416	56,684

Table 2: Training, development and test data from PTB.

where $\Phi(h)$ is the feature vector of h and $\vec{\theta}$ represents the parameters of our model. In the equation, the nominator $\Phi(h) \cdot \vec{\theta}$ is a standard linear model, used by Zhang *et al.* [2012]. The denominator $|h| = 2 \times \#\texttt{word}(h) - 1$ represents the size of h in terms of the number of actions in h. Intuitively, both POS-assignment and hypothesis-combination can be treated as hypothesis-building actions. Each action brings a set of new features into $\Phi(h)$. Hence a single-word hypothesis will have fewer features than a multi-word sentence. Our scaled linear model can increase linear separability of hypotheses by eliminating the impact of size differences.

Table 1 shows our feature templates, which include both syntax-based features and surface string patterns. Whilst the dependency syntax features are inspired by recent work on higher-order dependency parsing [Koo and Collins, 2010], surface features include mixed word and POS n-grams.

The online learning algorithm is based on the decoding process. At each step, the hypothesis h newly dequeued from agenda is examined. If it is a gold-standard hypothesis (i.e. a sub-tree in the gold-standard goal hypothesis), it is expanded in the same way as in Algorithm 1 to complete the decoding step. Otherwise h is treated as a negative example, the lowest-scored gold-standard hypothesis g_{min} on agenda is taken as a positive example, and the rest of the decoding step is replaced with a parameter update:

$$\vec{\theta} \leftarrow \vec{\theta} + \frac{score(h) - score(g_{min}) + 1}{||\frac{\Phi(g_{min})}{|g_{min}|} - \frac{\Phi(h)}{|h|}||^2} \left(\frac{\Phi(g_{min})}{|g_{min}|} - \frac{\Phi(h)}{|h|} \right)$$

The above equation is a minimal update to $\vec{\theta}$ which ensures that $score(g_{min})$ is greater than $score(h)$ by a margin of 1. The intuition is to adjust the model towards the situation where gold-standard hypotheses have higher scores than non-gold hypotheses, resulting in both better outputs and faster search [Daumé III and Marcu, 2005]. There are no time constraints during training, and the processing of one training example finishes when the gold-standard goal hypothesis is expanded (i.e. dequeued from agenda as the top hypothesis).

4 Experiments

An extensive set of experiments was conducted to evaluate the performance of our system under different settings. The standard Penn Treebank data were used for direct comparison with previous work. The Wall Street Journal data from PTB3 were split into training, development test and final test sets, shown in Table 2. Gold-standard dependency trees were derived from bracketed sentences in the treebank using the Penn2Malt tool[1]. Following Wan *et al.* [2009], Zhang and Clark [2011], and Zhang *et al.* [2012], we treat base noun phrases in PTB (e.g. "Piere Vinken") as a single word for all our experiments, and use the BLEU metric [Papineni *et al.*, 2002] to evaluate the outputs. Our system was implemented in Python, and all the experiments were performed on an Intel Core i7 3770 CPU with Fedora 17 and Python 2.7.

[1] http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html

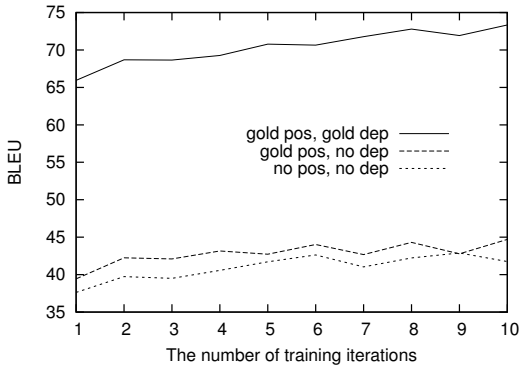

Figure 1: The convergence of the training algorithm.

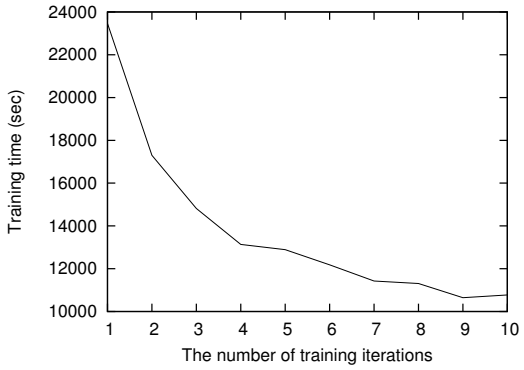

Figure 2: Training time (in seconds) by training iteration.

4.1 Search and training

Figure 1 shows three sets of development experiments under different input assumptions. For each input setting, we draw the BLEU scores against the number of training iterations. The top curve represents our system with input words, POS, and dependencies — the same input setting as He *et al.* [2009]; the middle curve represents our system with input words and POS, but no dependencies — the same input setting as Wan *et al.* [2009]; the bottom curve represents our system with only input words, which presents a more challenging search problem.

The curves demonstrate the convergence of the training algorithm: as the number of training iteration increases, the BLEU score increases until it reaches its peak at a particular iteration. For the final tests, we apply the optimal number of training iterations in the corresponding development experiments. For the rest of the development tests we only report results under the optimal training iteration.

A comparison between the curves shows that the availabil-

no POS, no dependencies
the board as a nonexecutive director will join Pierre Vinken , 61 years old , Nov. 29 .
Mr. Vinken , the Dutch publishing group of Elsevier N.V. is chairman .
and Consolidated Gold Fields PLC , former chairman of Rudolph Agnew , 55 years old was named a nonexecutive director of this British industrial conglomerate .
no POS, 50% random dependencies
Pierre Vinken , 61 years old , will join the board Nov. 29 as a nonexecutive director .
chairman of Elsevier N.V. , the Dutch publishing group is Mr. Vinken .
, Rudolph Agnew , was 55 years old and former chairman of Consolidated Gold Fields PLC named a nonexecutive director of this British industrial conglomerate .
50% random POS, all dependencies
Pierre Vinken , 61 years old , will join the board Nov. 29 as a nonexecutive director .
Mr. Vinken is chairman of Elsevier N.V. , the Dutch publishing group .
Rudolph Agnew , 55 years old and former chairman of Consolidated Gold Fields PLC , was named a nonexecutive director of this British industrial conglomerate .

Table 4: Outputs for the first three development test sentences with various input information.

ity of POS and dependencies in the input results in higher output fluency. An intuitive reason is that the search problem is simpler with more structural information specified by the input. The availability of dependency relations leads to more significant improvements than POS, demonstrating their importance in determining the surface word order.

Figure 2 shows the time taken for each training iteration with no input POS or dependencies, which is a more direct measure of convergence for leaning-guided search. As the number of training iteration increases, the training time decreases. Convergence of training is demonstrated since when the trained model gets better, fewer incorrect hypotheses are expanded before the gold goal hypotheses are found.

To measure the effect of our scaled linear model, we performed a development test using the standard linear model of Zhang *et al.* [2012], with no input POS or dependencies. The standard linear model gave a BLEU score of 40.61, over 2 points lower than our scaled model (42.89).

An additional set of development experiments were performed under the same input setting to measure the effects of the timeout limit, where the BLEU scores of the system was 42.89, 43.42, 43.58, and 43.72 when the timeout limit was 5s, 10s, 30s, and 60s, respectively.

4.2 Partial-tree linearization experiments

In the previous section, we made artificial assumptions on the forms of input. As discussed previously, in practice we are unlikely to anticipate the types of input information precisely. In this section, we simulate practical situations in dependency-based pipelines by measuring the performance of our system using randomized input POS and dependency relations. For the maximum flexibility, our system is trained without input POS or dependencies. During testing, if additional POS and dependencies are available, they will be used to further specify the output. From the perspective of search (Algorithm 1), input POS and dependencies are used as optional constraints to the search space, which reduce the amount of ambiguities.

Table 3 shows a set of results with various amounts of POS and dependencies in the input. For each test, we randomly sampled a percentage of words for which the gold-standard POS or dependencies are given for the input. As can be seen from the table, increased amounts of POS and dependency input led to higher BLEU scores, while dependencies were more effective than POS in specifying the word orders in the outputs. When all POS and dependencies are given, our adaptive system gave a BLEU score of 76.28, higher than the artificial setting in the previous section (74.79), which assumes that all POS and dependencies are given in both training and testing. This comparison is important since it demonstrates the effectiveness of our adaptive system in utilizing additional input POS and dependencies.

Table 4 shows the outputs of our system for the first three development test sentences with different input settings. These examples illustrate the effect of input dependencies in specifying the outputs, as utilized by our linearization system. Take the second sentence for example. When only input words are given, the output of the system reads grammatical but does not make sense. With increasing amounts of dependency relations given, the relative orders between the noun phrases become increasingly sensible. When all dependencies are given, the output is exactly the same as the reference.

4.3 Test results

We set the timeout threshold to 5s for all final tests. Table 5 presents three sets of test results according to different input settings. The column "no POS, no dep" shows our results with input words, but no POS or dependencies. This is an abstract language modeling task, with no additional information given to specify the output. Hardly any previous results have been reported on this task. The column "gold POS, no dep" shows the results of various systems with input words and gold-standard POS, and we make direct comparisons with previous work in this evaluation.

Both our system and the system of Wan *et al.* [2009] perform dependency-based word ordering, and our higher performance demonstrates the advantage of learning-guided search. A comparison with Zhang *et al.* [2012] shows that the light-weight dependency grammar can be as useful as the lexicalized CCG in word ordering, under a similar search framework. This result is of practical importance, given the wide use of dependency grammars in NLG applications.

Input	no POS no dep	50% POS no dep	all POS no dep	all POS 50% dep	50% POS 50% dep	50% POS all dep	no POS all dep	no POS 50% dep	all POS all dep
BLEU	42.89	43.41	44.71	52.18	51.40	74.68	73.34	50.46	76.28

Table 3: Partial-tree linearization, with different proportions of input POS and dep. randomly selected from full trees.

	no POS no dep	gold POS no dep	gold POS gold dep
Wan *et al.* [2009]	—	33.7	—
Zhang and Clark [2011]	—	40.1	—
Zhang *et al.* [2012]	—	43.8	—
this work	44.7	46.8	76.2

Table 5: Test results using PTB data.

The column "gold POS, gold dep" shows our results with input words, POS and dependencies. This corresponds to a unlabeled tree linearization task. A recent shared task has been organized for labeled tree linearization [Belz *et al.*, 2011], where the top three participating systems gave BLEU scores of 89.1, 85.8, and 79.4, respectively. To evaluate the performance of our system on labeled tree linearization, we extended Algorithm 1 by constructing arc labels during hypothesis combination (lines 20–29), and extending the feature set with label information. We omit more details due to space limitations. Our system gave a BLEU score of 89.3, comparable to the best-performing system of the shared task, although our results are not directly comparable because we used the PTB POS set and the Penn2Malt dependency labels, which are less fine-grained than the shared task data.

5 Discussions

In this section we discuss three questions concerning the practicality of our system. The first question is the availability of large-scale training data. In this paper, our system is trained using a manually annotated news corpus, which consists of about 40 thousand sentences. One cost-effective way of obtaining large-scale training data for our system is to annotate raw text automatically using a statistical dependency parser. The current state-of-the-art parsers achieve reasonable accuracy and efficiency for parsing news text [Zhang and Nivre, 2011]. For text in a different domain, parsing accuracy can drop significantly, leading to lower-quality syntax in automatically obtained training data. However, it does not necessarily result in lower output quality of our system trained using such data: since the gold-standard surface strings are still manually written, our system may output parse trees that resemble the output of the parser, but with reasonable word orders. Empirical verification is an interesting research question beyond the scope of this paper, which we leave for future work.

Second, Zhang *et al.* [2012] applied a large-scale n-gram language model to improve local fluency, which can also be employed by our system. We did not include n-gram language models in our experiments for two main reasons. First, we want to train our system solely using the standard PTB data, so that we provide closed-domain results for future comparisons. In addition, our results without n-gram language models are better than those of Zhang *et al.* [2012]. Second, our discriminative model does contain n-gram features, plus combined word and POS n-gram features, which are trained consistently in an integrated model. Hence we expect the effect of large-scale n-gram language models to level off with increasing amounts of training data for the integrated model. We leave verifications to future work. In situations with constrained memory resources, incorporating n-gram language models allows the training of the syntax model with smaller data, and therefore is very useful for a balance between the overall model size and the scale of local order information.

Third, our system can be made more flexible by allowing word selection, for which there are two motivating examples. The first example is the insertion of function words: although content words are likely to be available from an NLG pipeline, function words are less likely to be precisely obtained prior to linearization. The second example is multiple choices for the same word. For example, different morphological forms of a content word can be provided in a mutually exclusive form so that the correct morphology is decided by the linearization system. Our system naturally supports both situations without substantial changes to the search framework. To support the insertion of function words, the only change that is needed is a new goal condition: a goal hypothesis is one that contains all content words and any subset of a set of additional function words provided by the input. The support of word choices is also straightforward, requiring only the definition of mutual-exclusiveness between input words. An intuitive implementation is to give the same ID to mutually-exclusive words, so that the existing mechanism of word-count control can be applied without change. Empirical studies of these problems require adjustments to the standard training data, which we leave for future work.

6 Conclusions

We studied partial-tree linearization by building a word ordering system that accepts an input set of words with optional POS and dependencies. Compared to previous work on word ordering, the input flexibility makes our system more useful for practical text-to-text applications, such as summarization, grammar correction and machine translation. In addition,

- we applied learning-guided search to dependency-based word ordering, showing that the framework is more effective than previous methods for the same task, and directly allows partial-tree linearization;
- we reported the best published results for a word ordering task on PTB, showing that the light-weight dependency grammar is as useful as the lexicalized CCG grammar for ordering a bag of words;

Finally, we provided various evaluation results for abstract word ordering and tree linearization using standard PTB data,

including an abstract word ordering task without input POS, which has not been reported before, but is a very import metric for language modeling evaluation. Our results can serve as a baseline for future comparison on these tasks.

References

[Barzilay and McKeown, 2005] Regina Barzilay and Kathleen McKeown. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328, 2005.

[Belz *et al.*, 2011] Anja Belz, Mike White, Dominic Espinosa, Eric Kow, Deirdre Hogan, and Amanda Stent. The first surface realisation shared task: Overview and evaluation results. In *Proceedings of the Generation Challenges Session at EWNLG*, pages 217–226, Nancy, France, September 2011. Association for Computational Linguistics.

[Chang and Toutanova, 2007] Pi-Chuan Chang and Kristina Toutanova. A discriminative syntactic word order model for machine translation. In *Proceedings of ACL*, pages 9–16, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

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

[Collins, 2002] Michael Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*, pages 1–8, Philadelphia, USA, 2002.

[Daumé III and Marcu, 2005] Hal Daumé III and Daniel Marcu. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176, 2005.

[Filippova and Strube, 2007] Katja Filippova and Michael Strube. Generating constituent order in german clauses. In *Proceedings of ACL*, pages 320–327, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[He *et al.*, 2009] Wei He, Haifeng Wang, Yuqing Guo, and Ting Liu. Dependency based chinese sentence realization. In *Proceedings of ACL/AFNLP*, pages 809–816, Suntec, Singapore, August 2009.

[Koehn *et al.*, 2003] Philip Koehn, Franz Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of NAACL/HLT*, Edmonton, Canada, May 2003.

[Koo and Collins, 2010] Terry Koo and Michael Collins. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11, Uppsala, Sweden, July 2010.

[Lee and Seneff, 2006] J. Lee and S. Seneff. Automatic grammar correction for second-language learners. In *Proc. Interspeech*, pages 1978–1981, 2006.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002.

[Reiter and Dale, 1997] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, March 1997.

[Wan *et al.*, 2009] Stephen Wan, Mark Dras, Robert Dale, and Cécile Paris. Improving grammaticality in statistical sentence generation: Introducing a dependency spanning tree algorithm with an argument satisfaction model. In *Proceedings of EACL*, pages 852–860, Athens, Greece, March 2009.

[White and Rajkumar, 2009] Michael White and Rajakrishnan Rajkumar. Perceptron reranking for CCG realization. In *Proceedings of EMNLP*, pages 410–419, Singapore, August 2009. Association for Computational Linguistics.

[White, 2004] Michael White. Reining in CCG chart realization. In *Proc. INLG-04*, pages 182–191, 2004.

[Zhang and Clark, 2011] Yue Zhang and Stephen Clark. Syntax-based grammaticality improvement using CCG and guided search. In *Proceedings of EMNLP*, pages 1147–1157, Edinburgh, Scotland, UK., July 2011.

[Zhang and Nivre, 2011] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings ACL/HLT, short papers*, pages 188–193, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[Zhang *et al.*, 2012] Yue Zhang, Graeme Blackwood, and Stephen Clark. Syntax-based word ordering incorporating a large-scale language model. In *Proceedings of EACL*, pages 736–746, Avignon, France, April 2012.