Discriminative Syntax-Based Word Ordering for Text Generation

Yue Zhang*
Singapore University of Technology and Design

Stephen Clark**
University of Cambridge

Word ordering is a fundamental problem in text generation. In this article, we study word ordering using a syntax-based approach and a discriminative model. Two grammar formalisms are considered: Combinatory Categorial Grammar (CCG) and dependency grammar. Given the search for a likely string and syntactic analysis, the search space is massive, making discriminative training challenging. We develop a learning-guided search framework, based on best-first search, and investigate several alternative training algorithms.

The framework we present is flexible in that it allows constraints to be imposed on output word orders. To demonstrate this flexibility, a variety of input conditions are considered. First, we investigate a "pure" word-ordering task in which the input is a multi-set of words, and the task is to order them into a grammatical and fluent sentence. This task has been tackled previously, and we report improved performance over existing systems on a standard Wall Street Journal test set. Second, we tackle the same reordering problem, but with a variety of input conditions, from the bare case with no dependencies or POS tags specified, to the extreme case where all POS tags and unordered, unlabeled dependencies are provided as input (and various conditions in between). When applied to the NLG 2011 shared task, our system gives competitive results compared with the best-performing systems, which provide a further demonstration of the practical utility of our system.

1. Introduction

Word ordering is a fundamental problem in natural language generation (NLG, Reiter and Dale 1997). In this article we focus on text generation: Starting with a bag of words, or lemmas, as input, the task is to generate a fluent and grammatical sentence using

* Singapore University of Technology and Design. 20 Dover Drive, Singapore. E-mail: yue_zhang@sutd.edu.sg.
** University of Cambridge Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge, UK. E-mail: stephen.clark@cl.cam.ac.uk.

Submission received: 17 April 2013; revised version received: 23 April 2014; accepted for publication: 22 June 2014.

doi:10.1162/COLI_a_00229

those words. Additional annotation may also be provided with the input—for example, part-of-speech (POS) tags or syntactic dependencies. Applications that can benefit from better text generation algorithms include machine translation (Koehn 2010), abstractive text summarization (Barzilay and McKeown 2005), and grammar correction (Lee and Seneff 2006). Typically, statistical machine translation (SMT) systems (Chiang 2007; Koehn 2010) perform generation into the target language as part of an integrated system, which avoids the high computational complexity of independent word ordering. 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 standalone word ordering component can in principle be applied to a wide range of text generation tasks, including transfer-based machine translation (Chang and Toutanova 2007).

Most word ordering systems use an n-gram language model, which is effective at controling local fluency. Syntax-based language models, in particular dependency language models (Xu, Chelba, and Jelinek 2002), are sometimes used in an attempt to improve global fluency through the capturing of long-range dependencies. In this article, we take a syntax-based approach and consider two grammar formalisms: Combinatory Categorial Grammar (CCG) and dependency grammar. Our system also employs a discriminative model. Coupled with heuristic search, a strength of the model is that arbitrary features can be defined to capture complex syntactic patterns in output hypotheses. The discriminative model is trained using syntactically annotated data.

From the perspective of search, word ordering is a computationally difficult problem. Finding the best permutation for a set of words according to a bigram language model, for example, is NP-hard, which can be proved by linear reduction from the traveling salesman problem. In practice, exploring the whole search space of permutations is often prevented by adding constraints. In phrase-based machine translation (Koehn, Och, and Marcu 2003; Koehn et al. 2007), a distortion limit is used to constrain the position of output phrases. In syntax-based machine translation systems such as Wu (1997) and Chiang (2007), synchronous grammars limit the search space so that polynomial-time inference is feasible. In fluency improvement (Blackwood, de Gispert, and Byrne 2010), parts of translation hypotheses identified as having high local confidence are held fixed, so that word ordering elsewhere is strictly local.

In this article we begin by proposing a general system to solve the word ordering problem, which does not rely on constraints (which are typically task-specific). In particular, we treat syntax-based word ordering as a structured prediction problem, for which the input is a multi-set (bag) of words and the output is an ordered sentence, together with its syntactic analysis (either CCG derivation or dependency tree, depending on the grammar formalism being used). Given an input, our system searches for the highest-scored output, according to a syntax-based discriminative model. One advantage of this formulation of the reordering problem, which can perhaps be thought of as a "pure" text realization task, is that systems for solving it are easily evaluated, because all that is required is a set of sentences for reordering and a standard evaluation metric such as BLEU (Papineni et al. 2002). However, one potential criticism of the "pure" problem is that it is unclear how it relates to real realization tasks, since in practice (e.g., in statistical machine translation systems) the input does provide constraints on the possible output orderings. Our general formulation still allows task-specific contraints to be added if appropriate. Hence as a test of the flexibility of our system,

and a demonstration of the applicability of the system to more realistic text generation scenarios, we consider two further tasks for the dependency-based realization system.

The first task considers a variety of input conditions for the dependency-based system, determined by two parameters. The first is whether POS information is provided for each word in the input multi-set. The second is whether syntactic dependencies between the words are provided. The extreme case is when all dependencies are provided, in which case the problem reduces to the tree linearization problem (Filippova and Strube 2009; He et al. 2009). However, the input can also lie between the two extremes of no- and full-dependency information.

The second task is the NLG 2011 shared task, which provides a further demonstration of the practical utility of our system. The shared task is closer to a real realization scenario, in that lemmas, rather than inflected words, are provided as input. Hence some modifications are required to our system in order that it can perform some word inflection, as well as deciding on the ordering. The shared task data also uses labeled, rather than unlabeled, syntactic dependencies, and so the system was modified to incorporate labels. The final result is that our system gives competitive BLEU scores, compared to the best-performing systems on the shared task.

The structured prediction problem we solve is a very hard problem. Due to the use of syntax, and the search for a sentence together with a single CCG derivation or dependency tree, the search space is exponentially larger than the *n*-gram word permutation problem. No efficient algorithm exists for finding the optimal solution. Kay (1996) recognized the computational difficulty of chart-based generation, which has many similarities to the problem we address in his seminal paper. We tackle the high complexity by using learning-guided best-first search, exploring a small path in the whole search space, which contains the most likely structures according to the discriminative model. One of the contributions of this article is to introduce, and provide a discriminative solution to, this difficult structured prediction problem, which is an interesting machine learning problem in its own right.

This article is based on, and significantly extends, three conference papers (Zhang and Clark 2011; Zhang, Blackwood, and Clark 2012; Zhang 2013). It includes a more detailed description and discussion of our guided-search approach to syntax-based word ordering, bringing together the CCG- and dependency-based systems under one unified framework. In addition, we discuss the limitations of our previous work, and show that a better model can be developed through scaling of the feature vectors. The resulting model allows fair comparison of constituents of different sizes, and enables the learning algorithms to expand negative examples during training, which leads to significantly improved results over our previous work. The competitive results on the NLG 2011 shared task data are new for this article, and demonstrate the applicability of our system to more realistic text realization scenarios.

The contributions of this article can be summarized as follows:

- We address the problem of syntax-based word ordering, drawing attention to this challenging language modeling task and offering a general solution that does not rely on constraints to limit the search space.
- We present a novel method for solving the word ordering problem that gives the best reported accuracies to date on the standard *Wall Street Journal* data.

- We show how our system can be used with two different grammar formalisms: Combinatory Categorial Grammar and dependency grammar.
- We show how syntactic constraints can be easily incorporated into the system, presenting results for the dependency-based system with a range of input conditions.
- We demonstrate the applicability of the system to more realistic text realization scenarios by obtaining competitive results on the NLG 2011 shared task data, including performing some word inflection as part of a joint system that also performs word reordering.
- More generally, we propose a learning-guided, best-first search algorithm for application of discriminative models to extremely large search spaces containing structures of varying sizes. This method could be applied to other complex structured prediction tasks in NLP and machine learning.

2. Overview of the Search and Training Algorithms

In this section, the CCG-based system is used to describe the search and training algorithms. However, the same approach can be used for the dependency-based system, as described in Section 4: Instead of building hypotheses by applying CCG rules in a bottom-up manner, the dependency-based system creates dependency links between words.

Given a bag of words, the goal is to put them into an ordered sentence that has a plausible CCG derivation. The search space of the decoding problem consists of all possible CCG derivations for all possible word permutations, and the search goal is to find the highest-scored derivation in the search space. This is an NP-hard problem, as mentioned in the Introduction. We apply learning-guided search to address the high complexity. The intuition is that, because the whole search space cannot be exhausted in order to find the optimal solution, we choose to explore a small area in the search space. A statistical model is used to guide the search, so that only a small portion of the search space containing the most plausible hypotheses is explored.

One natural choice for the decoding algorithm is best-first search, which uses an agenda to order hypotheses, and expands the highest-scored hypothesis on the agenda at each step. The resulting hypotheses after each hypothesis expansion are put back on the agenda, and the process repeats until a goal hypothesis (a full sentence) is found. This search process is guided by the current scores of the hypotheses, and the search path will contain the most plausible hypotheses if they are scored higher than implausible ones. An alternative to best-first search is A* search, which makes use of a heuristic function to estimate future scores. A* can potentially be more efficient given an effective heuristic function; however, it is not straightforward to define an admissible and accurate estimate of future scores for our problem, and we leave this research question to future work.

In our formulation of the word ordering problem, a hypothesis is a phrase or sentence together with its CCG derivation. Hypotheses are constructed bottom–up: starting from single words, smaller phrases are combined into larger ones according to CCG rules. To allow the combination of hypotheses, we use an additional structure to store a set of hypotheses that have been expanded, which we call **accepted**

hypotheses. When a hypothesis from the agenda is expanded, it is combined with all accepted hypotheses in all possible ways to produce new hypotheses. The data structure for accepted hypotheses is similar to that used for best-first parsing (Caraballo and Charniak 1998), and we adopt the term **chart** for this structure. However, note there are important differences to the parsing problem. First, the parsing problem has a fixed word order and is considerably simpler than the word ordering problem we are tackling. Second, although we use the term *chart*, the structure for accepted hypotheses is not a dynamic programming chart in the same way as for the parsing problem. In our previous papers (Zhang and Clark 2011; Zhang, Blackwood, and Clark 2012), we applied a set of beams to this structure, which makes it similar to the data structure used for phrase-based MT decoding (Koehn 2010). However, we will show later that this structure is unnecessary when the model has more discriminative power, and a conceptually simpler single beam can be used. We will also investigate the possibility of applying dynamic-programming-style pruning to the chart.

We now give an overview of the training algorithm, which is crucial to both the speed and accuracy of the resulting decoder. CCGBank (Hockenmaier and Steedman 2007) is used to train the model. For each training sentence, the corresponding CCGBank derivation together with all its sub-derivations are treated as gold-standard hypotheses. All other hypotheses that can be constructed from the same bag of words are non-gold hypotheses. From the generation perspective this assumption is too strong, because sentences can have multiple orderings (with multiple derivations) that are both grammatical and fluent. Nevertheless, it is the most feasible choice given the training data available.

The efficiency of the decoding algorithm is dependent on the training algorithm because the agenda is ordered according to the hypothesis scores. Hence, a better model will lead to a goal hypothesis being found more quickly. In the ideal situation, all gold-standard hypotheses are scored higher than all non-gold hypotheses, and therefore only gold-standard hypotheses are expanded before the gold-standard goal hypothesis is found. In this case, the minimum number of hypotheses is expanded and the output is correct. The best-first search decoder is optimal not only with respect to accuracy but also speed. This ideal situation can hardly be met in practice, but it determines the goal of the training algorithm: to find a model that scores gold-standard hypotheses higher than non-gold ones.

Learning-guided search places more challenges on the training of a discriminative model than standard structured prediction problems, for example, CKY parsing for CCG (Clark and Curran 2007b). If we take gold-standard hypotheses as positive training examples, and non-gold hypotheses as negative examples, then the training goal is to find a large separating margin between the scores of all positive examples and all negative examples. For CKY parsing, the highest-scored negative example can be found via optimal Viterbi decoding, according to the current model, and this negative example can be used in place of all negative examples during the updating of parameters. In contrast, our best-first search algorithm cannot find an output in reasonable time unless a good model has already been trained, and therefore we cannot run the decoding algorithm in the standard way during training. In our previous papers (Zhang and Clark 2011; Zhang, Blackwood, and Clark 2012), we proposed an approximate online training algorithm, which forces positive examples to be kept in the hypothesis space without being discarded, and prevents the expansion of negative examples during the training process (so that the hypothesis space does not get too large). This algorithm ensures training efficiency, but greatly limits the space of negative examples that is explored during training (and hence fails to replicate the conditions experienced at test

time). In this article, we will show that, with an improved scoring model, it is possible to expand negative examples, which leads to improved performance.

A second and more subtle challenge for our training problem is that we need a stronger model for learning-guided search than for dynamic programming (DP)–based search, such as CKY decoding. For CKY decoding, the model is used to compare hypotheses within each chart cell, which cover the same input words. In contrast, for the best-first search decoder, the model is used to order hypotheses on the agenda, which can cover different numbers of words. It needs much stronger discriminating power, so that it can determine whether a single-word phrase is better than, say, a 40-word sentence. In this article we use scaling of the hypothesis scores by size, so that hypotheses of different sizes can be fairly compared. We also find that, with this new approach, negative examples can be expanded during training and a single beam applied to the chart, resulting in a conceptually simpler and more effective training algorithm and decoder.

3. CCG-Based Word Ordering

3.1 The CCG Grammar

We were motivated to use CCG as one of the grammar formalisms for our syntax-based realization system because of its successful application to a number of related tasks, such as wide-coverage parsing (Hockenmaier 2003; Clark and Curran 2007b; Auli and Lopez 2011), semantic parsing (Zettlemoyer and Collins 2005), wide-coverage semantic analysis (Bos et al. 2004), and generation itself (Espinosa, White, and Mehay 2008). The grammar formalism has been described in detail in those papers, and so here we provide only a short description.

CCG (Steedman 2000) is a lexicalized grammar formalism that associates words with lexical categories. Lexical categories are detailed grammatical labels, typically expressing subcategorization information. During CCG parsing, and during our search procedure, categories are combined using CCG's combinatory rules. For example, a verb phrase in English ($S\backslash NP$) can combine with an NP to its left, in this case using the combinatory rule of (backward) function application:

$$NP \;\; S\backslash NP \Rightarrow S$$

In addition to binary rule instances, such as this one, there are also unary rules that operate on a single category in order to change its type. For example, forward type-raising can change a subject NP into a complex category looking to the right for a verb phrase:

$$NP \Rightarrow S/(S\backslash NP)$$

Such a type-raised category can then combine with a transitive verb type using the rule of forward composition:

$$S/(S\backslash NP) \;\; (S\backslash NP)/NP \Rightarrow S/NP$$

Following Fowler and Penn (2010), we extract the grammar by reading rule instances directly from the derivations in CCGbank (Hockenmaier and Steedman 2007), rather than defining the combinatory rule schema manually as in Clark and Curran

(2007b). Hence the grammar we use can be thought of as a context-free approximation to the mildly content sensitive grammar arising from the use of generalized composition rules (Weir 1988). Hockenmaier (2003) contains a detailed description of the grammar that is obtained in this way, including the various unary type-changing rules, as well as additional rules needed to deal with naturally occurring text, such as punctuation rules.

3.2 The Edge Data Structure

For the rest of this article, the term **edge** is used to refer to a hypothesis in the decoding algorithm. An edge corresponds to a sentence or phrase with a CCG derivation. Edges are built bottom–up, starting from **leaf edges**, which are constructed by assigning possible lexical categories to input words. Each leaf edge corresponds to an input word with a particular lexical category. Two existing edges can be combined if there exists a CCG rule (extracted from CCGbank, as described earlier) that combines their category labels, and if they do not contain the same input word more times than its total count in the input. The resulting edge is assigned a category label according to the CCG rule, and covers the concatenated surface strings of the two sub-edges in their order of combination. New edges can also be built by applying unary rules to a single existing edge. We define a **goal edge** as an edge that covers all input words.

Two edges are **equivalent** if they have the same surface string and identical CCG derivations. Edge equivalence is used for comparison with gold-standard edges. Two edges are **DP-equivalent** when they have the same **DP-signature**. Based on the feature templates in Table 1, we define the DP-signature of an edge as the CCG category at the

Table 1
Feature template definitions, with example instances based on Figure 2.

condition	feature	instance
all edges	category + size	(*S*, *3*)
	category + head word	(*S**NP*, *bought*)
	category + size + head word	(*NP*, *1*, *Lotus*)
	category + head POS	(*S**NP*, *VBD*)
#words > 1	category + leftmost word	(*S*, *IBM*)
	category + rightmost word	(*S**NP*, *Lotus*)
	category + leftmost POS bigram	(*S*, (*NNP*, *VBD*))
	category + rightmost POS bigram	(*S*, (*VBD*, *NNP*))
	category + lmost POS + rmost POS	(*S*, (*NNP*, *NNP*))
binary edges	the binary rule	*NP* (*S**NP*) ⇒ *S*
	the binary rule + head word	(*NP* (*S**NP*) ⇒ *S*, *bought*)
	rule + head word + non-head word	(*NP* (*S**NP*) ⇒ *S*, *bought*, *IBM*)
	bigrams resulting from combination	(*IBM*, *bought*)
	POS bigrams resulting from combination	(*NNP*, *VBD*)
	word trigrams resulting from combination	(*IBM*, *bought*, *Lotus*)
	POS trigrams resulting from combination	(*NNP*, *VBD*, *NNP*)
	resulting lexical category trigrams	(*NP*, (*S**NP*)/*NP*, *NP*)
	resulting word + POS bigrams	(*IBM*, *VBD*)
	resulting POS + word bigrams	(*VBD*, *Lotus*)
	resulting POS + word + POS trigrams	(*NNP*, *bought*, *NNP*)
unary edges	unary rule	
	unary rule + head word	

root of its derivation, the head word associated with the root category, and the multi-set of words it contains, together with the word and POS bigrams on either side of its surface string.

3.3 The Scoring of Edges

Edges are built bottom–up from input words or existing edges. If we treat the assignment of lexical categories to input words and the application of unary and binary CCG rules to existing edges as edge-building actions, the structure of an edge can be defined recursively as the sub-structure resulting from its *top action* plus the structure of its sub-edges (if any), as shown in Figure 1. Here the top action of an edge refers to the most recent action that has been applied to build the edge.

In our previous papers we used a global linear model to score edges, where the score of an edge e is defined as:

$$f(e) = \Phi(e) \cdot \vec{\theta}$$

$\Phi(e)$ represents the feature vector of e and $\vec{\theta}$ is the parameter vector of the model.

Similar to the structure of e, the feature vector $\Phi(e)$ can be defined recursively:

$$\begin{aligned}\Phi(e) &= \big(\sum_{e_s \in e} \Phi(e_s)\big) + \phi(e) \\ &= \big(\sum_{e_s^r \in^r e} \Phi(e_s^r)\big) + \phi(e)\end{aligned}$$

In this equation, $e_s \in e$ represents a sub-edge of e. Leaf edges do not have any sub-edges. Unary-branching edges have one sub-edge, and binary-branching edges have two sub-edges. $e_s^r \in^r e$ represents a (strictly) recursive sub-edge of e. The feature vector $\phi(e)$ represents the structure of the top action of e; it is extracted according to the feature templates in Table 1. Example instances of the feature templates are given according to the example string and CCG derivation in Figure 2. For leaf edges, $\phi(e)$ includes information about the lexical category label; for unary-branching edges, $\phi(e)$ includes information from the unary rule; for binary-branching edges, $\phi(e)$ includes information from the binary rule, and additionally the token, POS, and lexical category bigrams and trigrams that result from the surface string concatenation of its sub-edges.

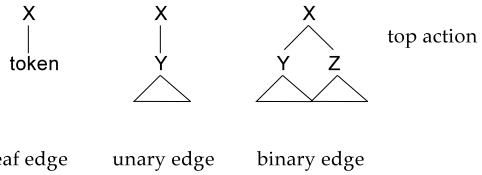

Figure 1
The structure of edges shown recursively.

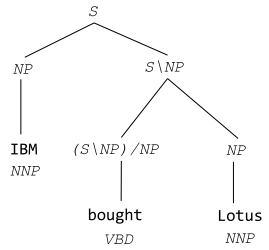

Figure 2
Example string with its CCG derivation, used to give example features in Table 1.

By the given definition of $\Phi(e)$, $f(e)$, the score of edge e, can be computed recursively as e is built during the decoding process:

$$\begin{aligned} f(e) &= \Phi(e) \cdot \vec{\theta} \\ &= \Big(\big(\sum_{e_s \in e} \Phi(e_s) \big) + \phi(e) \Big) \cdot \vec{\theta} \\ &= \big(\sum_{e_s \in e} \Phi(e_s) \cdot \vec{\theta} \big) + \phi(e) \cdot \vec{\theta} \\ &= \big(\sum_{e_s \in e} f(e_s) \big) + \phi(e) \cdot \vec{\theta} \end{aligned}$$

When the top action is applied, the score of $f(e)$ is computed as the sum of $f(e_s)$ (for all $e_s \in e$) plus $\phi(e) \cdot \vec{\theta}$.

An important aspect of the scoring model is that it is used to compare edges with different sizes during decoding. The size of an edge can be measured in terms of the number of words it contains, or the number of syntax rules in its structure. We define the size of an edge as the number of recursive sub-edges in the edge plus one (e.g., the size of a leaf edge is 1), which is equivalent to the number of actions (i.e., lexical category assignment for leaf edges, and rule application for unary/binary edges) that have been applied to construct the edge. Edges with different sizes can have significantly different numbers of features, which can make the training of a discriminative linear model more difficult. Note that it is common in structured prediction problems for feature vectors to have slightly different sizes because of variant feature instantiation conditions. In CKY parsing, for example, constituents with different numbers of unary rules can be kept in the same chart cell and compared with each other, provided that they cover the same span in the input. In our case, however, the sizes of two feature vectors under comparison can be very different indeed, since a leaf edge with one word can be compared with an edge over the entire input sentence.

In our previous papers we observed empirical convergence of online learning using this linear model, and obtained competitive results. However, as explained in Section 2, only positive examples were expanded during training, and the expansion of negative examples led to non-convergence and made online training infeasible. In this article, in order to increase the discriminating power of the model and to make use of negative examples during training, we apply length normalization to the scoring function, so that the score of an edge is independent of its size. To achieve this, we scale the original

linear model score by the number of recursive sub-edges in the edge plus one. For a given edge e, the new score $\hat{f}(e)$ is defined as:

$$\begin{aligned}\hat{f}(e) &= \frac{f(e)}{|e|} \\ &= \frac{\Phi(e) \cdot \vec{\theta}}{|e|} \\ &= \frac{\sum_{e_s^r \in^r e} \phi(e_s^r) \cdot \vec{\theta} + \phi(e) \cdot \vec{\theta}}{|\{e_s^r \in^r e\}| + 1|}\end{aligned}$$

In the equation, $|e|$ represents the size of e, which is equal to the number of actions that have been applied when e is constructed. By dividing the score $f(e)$ by the size of e, the score $\hat{f}(e)$ represents an averaged value of $\phi(e_s^r) \cdot \vec{\theta}$ ($e_s^r \in^r e$) and $\phi(e) \cdot \vec{\theta}$, averaged by the number of recursive sub-edges plus one (i.e., the total actions), and is independent of the size of e. Given normalized feature vectors, the training of the parameter vector $\vec{\theta}$ needs to be adjusted correspondingly, which will be discussed subsequently.

3.4 The Decoding Algorithm

The decoding algorithm takes a multi-set of input words, turns them into a set of leaf edges, and searches for a goal edge by repeated expansion of existing edges. For best-first decoding, an **agenda** and a **chart** are used. The agenda is a priority queue on which edges to be expanded are ordered according to their current scores. The chart is a fixed-size beam used to record a limited number of accepted edges. During initialization, leaf edges are generated by assigning all possible lexical categories to each input word, before they are put on the agenda. During each step in the decoding process, the highest-scored edge on the agenda is popped off and expanded. If it is a goal edge, it is returned as the output, and the decoding finishes. Otherwise it is extended with unary rules, and combined with existing edges in the chart, using binary rules to produce new edges. The resulting edges are scored and put on the agenda, and the original edge is put into the chart. The process repeats until a goal edge is found, or a timeout limit is reached.

For the timeout case, a default output is produced by greedily combining existing edges in the chart in descending order of size. In particular, edges in the chart are sorted by size, and the largest is taken as the current default output. Then the sorted list is traversed, with an attempt to greedily concatenate the current edges in the list to the right of the current default output. If the combination is not allowed (i.e., the two edges contain some input words more times than its count in the input), the current edge is discarded. Otherwise, the current default output is updated.

In our previous papers we used a set of beams for the chart, each storing a certain number of highest-scored edges that cover a particular number of words. This structure is similar to the chart used for phrase-based SMT decoding. The main reason for the multiple beams is the non-comparability of edges in different beams, which can have feature vectors of significantly different sizes. In this article, however, our chart is a single beam structure containing the top-scored accepted edges. This simple data structure is enabled by the use of the scaled linear model, and leads to comparable accuracies to the multiple-beam chart. In addition to its simplicity, it also fits well with the use of agenda-based search, because edges of different sizes will ultimately be compared with each other on the agenda.

We apply DP-style pruning to the chart, keeping only the highest-scored edge among those that have the same DP-signature. During decoding, before a newly constructed edge e is put into the chart, the chart is examined to check whether it contains an existing edge e_0 with the same DP-signature as e. If such an edge exists, it is popped off the chart and compared with the newly constructed edge e, with the higher scored edge $\tilde{e}$ being put into the chart and the lower scored edge e' being discarded. If the newly constructed edge e is not discarded, then we expand e to generate new edges.

It is worth noting that, in this case, a new edge that results from the expansion of e can have DP-equivalent edges in the agenda or the chart, which had been generated by expansion of its DP-equivalent predecessor $e' = e_0$. Putting such new edges on the agenda will result in the system keeping multiple edges with the same signature. However, because applying DP-style pruning to the agenda requires updating the whole agenda, and is computationally expensive, we choose to tolerate such DP-equivalent duplications in the agenda.

Pseudocode for the decoder is shown as Algorithm 1. INITAGENDA returns an initialized agenda with all leaf edges. INITCHART returns a cleared chart. TIMEOUT

Algorithm 1 The decoding algorithm.

```
a ← INITAGENDA( )
c ← INITCHART( )
while not TIMEOUT( ) do
    new ← []
    e ← POPBEST(a)
    if GOALTEST(e) then
        return e
    end if
    (e′, ẽ) ← DPCHARTPRUNE(c, e)
    if e′ is e then
        continue
    end if
    for e′ ∈ UNARY(e, grammar) do
        APPEND(new, e′)
    end for
    for ẽ ∈ c do
        if CANCOMBINE(e, ẽ, grammar) then
            e′ ← BINARY(e, ẽ, grammar)
            APPEND(new, e′)
        end if
        if CANCOMBINE(ẽ, e, grammar) then
            e′ ← BINARY(ẽ, e, grammar)
            APPEND(new, e′)
        end if
    end for
    for e′ ∈ new do
        ADD(a, e′)
    end for
    ADD(c, e)
end while
```

returns **true** if the timeout limit has been reached, and **false** otherwise. POPBEST pops the top edge from the agenda and returns the edge. GOALTEST takes an edge and returns **true** if and only if the edge is a goal edge. DPCHARTPRUNE takes an edge e and checks whether there exists in the chart an edge e_0 that is DP-equivalent to e. In case e_0 exists, it is popped off the chart and compared with e, with the lower scored edge e' being discarded, and the higher scored edge $\tilde{e}$ being put into the chart. The function returns the pair e' and $\tilde{e}$. CANCOMBINE checks whether two edges can be combined in a given order. Two edges can be combined if they do not contain an overlapping word (i.e., they do not contain a word more times than its count in the input), and their categories can be combined according to the CCG grammar. ADD inserts an edge into the agenda or the chart. In the former case, it is placed into the priority queue according to its score, and, in the latter case, the lowest scored edge in the beam is pruned when the chart is full.

3.5 The Learning Algorithm

We begin by introducing the training algorithm of our previous papers, shown in Algorithm 2, which has the same fundamental structure as the training algorithm of this article but is simpler. The algorithm is based on the decoder, where an agenda is used as a priority queue of edges to be expanded, and a set of accepted edges is kept in a fixed-size chart. The functions INITAGENDA, INITCHART, TIMEOUT, POPBEST, GOALTEST, DPCHARTPRUNE, UNARY, CANCOMBINE, and BINARY are identical to those used in the decoding algorithm. GOLDSTANDARD takes an edge and returns **true** if and only if it is a gold-standard edge. MINGOLD returns the lowest scored gold-standard edge in the agenda. UPDATEPARAMETERS represents the parameter update algorithm. RECOMPUTESCORES updates the scores of edges in the agenda and chart after the model is updated.

Similar to the decoding algorithm, the agenda is intialized using all possible leaf edges. During each step, the edge e on top of the agenda is popped off. If it is a gold-standard edge, it is expanded in exactly the same way as in the decoder, with the newly generated edges being put on the agenda, and e being inserted into the chart. If e is not a gold-standard edge, we take it as a negative example e_-, and take the lowest scored gold-standard edge on the agenda e_+ as a positive example, in order to make an update to the parameter vector $\vec{\theta}$. Note that there must exist a gold-standard edge in the agenda, which can be proved by contradiction.[1]

The two edges e_+ and e_- used to perform a model update can be radically different. For example, they may not cover the same words, or even the same number of words. This is different from online training for CKY parsing, for which both positive and negative examples used to adjust parameter vectors reside in the same chart cell, and cover the same sequence of words. The training goal of a typical CKY parser (Clark and Curran 2007a, 2007b) is to find a large separation margin between feature vectors of different derivations of the same sentence, which have comparable sizes. Our goal is to score all gold-standard edges higher than all non-gold edges regardless of their size, which is a more challenging goal. After updating the parameters, the scores of the agenda edges above and including e_-, together with all chart edges, are updated, and e_- is discarded before the start of the next processing step.

1 An example proof can be based on induction, where the basis is that the agenda contains all gold leaf edges at first, and the induction step is based on edge combination.

Algorithm 2 The training algorithm of our previous papers.

```
1: a ← INITAGENDA( )
2: c ← INITCHART( )
3: while not TIMEOUT( ) do
4:     new ← []
5:     e ← POPBEST(a)
6:     if GOLDSTANDARD(e) and GOALTEST(e) then
7:         return e
8:     end if
9:     if not GOLDSTANDARD(e) then
10:        e_− ← e
11:        e_+ ← MINGOLD(a)
12:        UPDATEPARAMETERS(e_+, e_−)
13:        RECOMPUTESCORES(a, c)
14:        continue
15:    end if
16:    (e′, ẽ) ← DPCHARTPRUNE(c, e)
17:    if e′ is e then
18:        continue
19:    end if
20:    for e′ ∈ UNARY(e, grammar) do
21:        APPEND(new, e′)
22:    end for
23:    for ẽ ∈ c do
24:        if CANCOMBINE(e, ẽ, grammar) then
25:            e′ ← BINARY(e, ẽ, grammar)
26:            APPEND(new, e′)
27:        end if
28:        if CANCOMBINE(ẽ, e, grammar) then
29:            e′ ← BINARY(ẽ, e, grammar)
30:            APPEND(new, e′)
31:        end if
32:    end for
33:    for e′ ∈ new do
34:        ADD(a, e′)
35:    end for
36:    ADD(c, e)
37: end while
```

One way of viewing the training process is that it pushes gold-standard edges towards the top of the agenda, and, crucially, pushes them above non-gold edges (Zhang and Clark 2011). Given a positive example e_+ and a negative example e_-, a perceptron-style update is used to penalize the score for $\phi(e_-)$ and reward the score of $\phi(e_+)$:

$$\vec{\theta} \leftarrow \vec{\theta}_0 + \phi(e_+) - \phi(e_-)$$

Here $\vec{\theta}_0$ and $\vec{\theta}$ denote the parameter vectors before and after the update, respectively. This method proved effective empirically (Zhang and Clark 2011), but it did not

converge well when an n-gram language model was integrated into the system (Zhang, Blackwood, and Clark 2012).

Hence we applied an alternative method for score updates that proved more effective than the perceptron update and enabled the incorporation of a large-scale language model (Zhang, Blackwood, and Clark 2012). This method treats parameter update as finding a separation between gold-standard and non-gold edges. Given a positive example e_+ and a negative example e_-, we make a minimum update to the parameters so that the score of e_+ is higher than that of e_- by a margin of 1:

$$\vec{\theta} \leftarrow \arg\min_{\vec{\theta}'} \| \vec{\theta}' - \vec{\theta}_0 \|, \text{ such that } \Phi(e_+)\vec{\theta}' - \Phi(e_-)\vec{\theta}' \geq 1$$

The update is similar to the parameter update of online large-margin learning algorithms, such as 1-best MIRA (Crammer et al. 2006), and has a closed-form solution:

$$\vec{\theta} \leftarrow \vec{\theta}_0 + \frac{f(e_-) - f(e_+) + 1}{\| \Phi(e_+) - \Phi(e_-) \|^2} \big(\Phi(e_+) - \Phi(e_-)\big)$$

This online learning method proved more effective than the perceptron algorithm empirically, but still has an important shortcoming in that it did not provide competitive results when allowing the expansion of negative examples during training, which can potentially improve the discriminative model (since expanding negative examples can result in a more representative sample of the search space). We address this issue by introducing a scaled linear model in this article, which, when combined with the expansion of negative examples, significantly improves performance. We apply the same online large-margin training principle; however, the parameter update has to be adjusted for the scaled linear model. In particular, the new goal is to find a separation between $\hat{f}(e_+)$ and $\hat{f}(e_-)$ instead of $f(e_+)$ and $f(e_-)$, for which the optimization corresponding to the parameter update becomes:

$$\vec{\theta} \leftarrow \arg\min_{\vec{\theta}'} \| \vec{\theta}' - \vec{\theta}_0 \|, \quad \text{such that } \frac{\Phi(e_+)\vec{\theta}'}{|e_+|} - \frac{\Phi(e_-)\vec{\theta}'}{|e_-|} \geq 1$$

where $\vec{\theta}_0$ and $\vec{\theta}$ represent the parameter vectors before and after the update, respectively. The equation has a closed-form solution:

$$\vec{\theta} \leftarrow \vec{\theta}_0 + \frac{\hat{f}(e_-) - \hat{f}(e_+) + 1}{\| \frac{\Phi(e_+)}{|e_+|} - \frac{\Phi(e_-)}{|e_-|} \|^2} \big(\frac{\Phi(e_+)}{|e_+|} - \frac{\Phi(e_-)}{|e_-|}\big)$$

Pseudocode for the new training algorithm of this article is shown in Algorithm 3, where MAXNONGOLD returns the highest-scored non-gold edge in the chart. In addition to the aforementioned difference in parameter updates, new code is added to perform additional updates when gold-standard edges are removed from the chart. In our previous work, parameter updates happen only when the top edge from the agenda is not a gold-standard edge. In this article, the expansion of negative training examples will lead to negative examples being put into the chart during training, and hence the possibility of gold-standard edges being removed from the chart. There are two situations when this can happen. First, if a non-gold edge is inserted into the chart,

Algorithm 3 The training algorithm of this article.

```
1: a ← INITAGENDA( )
2: c ← INITCHART( )
3: while not TIMEOUT( ) do
4:     new ← []
5:     e ← POPBEST(a)
6:     if GOLDSTANDARD(e) and GOALTEST(e) then
7:         return e
8:     end if
9:     if not GOLDSTANDARD(e) then
10:        e_ ← e
11:        e+ ← MINGOLD(a)
12:        UPDATEPARAMETERS(e+, e_)
13:        RECOMPUTESCORES(a, c)
14:    end if
15:    (e′, ẽ) ← DPCHARTPRUNE(c, e)
16:    if GOLDSTANDARD(e′) then
17:        UPDATEPARAMETERS(e′, ẽ)
18:        REMOVE(c, ẽ)
19:        ADD(c, e′)
20:    else
21:        if e′ is e then
22:            continue
23:        end if
24:    end if
25:    for e′ ∈ UNARY(e, grammar) do
26:        APPEND(new, e′)
27:    end for
28:    for ẽ ∈ c do
29:        if CANCOMBINE(e, ẽ, grammar) then
30:            e′ ← BINARY(e, ẽ, grammar)
31:            APPEND(new, e′)
32:        end if
33:        if CANCOMBINE(ẽ, e, grammar) then
34:            e′ ← BINARY(ẽ, e, grammar)
35:            APPEND(new, e′)
36:        end if
37:    end for
38:    for e′ ∈ new do
39:        ADD(a, e′)
40:    end for
41:    e′ ← ADD(c, e)
42:    if GOLDSTANDARD(e′) then
43:        ẽ ← MAXNONGOLD(c)
44:        UPDATEPARAMETERS(e′, ẽ)
45:        ADD(c, e′)
46:    end if
47: end while
```

and there exists a gold-standard edge in the chart with the same DP-signature but a lower score, the gold-standard edge will be removed from the chart because of DP-style pruning (since only the highest-scored edge with the same DP-signature is kept in the chart).

Second, if the chart is full when a non-gold edge is put into the chart (recall that the chart is a fixed-size beam), then the lowest scored edge on the chart will be removed. This edge can be a gold-standard edge. In both the first and second case, a gold-standard edge is pruned as the result of the expansion of a negative example. On the other hand, in order for the gold-standard goal edge to be constructed, all gold-standard edges that have been expanded must remain in the chart. As a result, our training algorithm triggers a parameter update whenever a gold-standard edge is removed from the chart, the scores of all chart edges are updated, and the original pruned gold edge is returned to the chart. The original pruned gold-standard edge is treated as the positive example for the update. For the first situation, the newly inserted non-gold edge with the same DP-signature is taken as the negative example, and will be discarded after the parameter update (with a new score that is lower than the new score of the corresponding gold-standard). In the second situation, the highest-scored non-gold edge in the chart is taken as the negative example, and removed from the chart after the update.

In summary, there are two main differences between Algorithms 2 and 3. First, line 14 in Algorithm 2, which skips the expansion of negative examples, is removed in Algorithm 3. Second, lines 16–20 and 42–46 are added in Algorithm 3, which correspond to the updating of parameters when a gold-standard edge is removed from the chart. In addition, the definitions of UPDATEPARAMETERS are different for the perceptron training algorithm (Zhang and Clark 2011), the large-margin training algorithm (Zhang, Blackwood, and Clark 2012), and the large-margin algorithm of this article, as explained earlier.

4. Dependency-Based Word Ordering and Tree Linearization

As well as CCG, the same approach can be applied to the word ordering problem using other grammar formalisms. In this section, we present a dependency-based word ordering system, where the input is again a multi-set of words with gold-standard POS, and the output is an ordered sentence together with its *dependency* parse. Except for necessary changes to the edge data structure and edge expansion, the same algorithm can be applied to this task.

In addition to abstract word ordering, our framework can be used to solve a more informed, dependency-based word ordering task: tree linearization (Filippova and Strube 2009; He et al. 2009), a task that is very similar to abstract word ordering from a computational perspective. Both tasks involve the permutation of a set of input words, and are NP-hard. The only difference is that, for tree linearization, full unordered dependency trees are given as input. As a result, the output word permutations are more constrained (under the projectivity assumption), and more information is available for search disambiguation.

Tree linearization can be treated as a special case of word ordering, where a grammar constraint is applied such that the output sentence has to be consistent with the input tree. There is a spectrum of grammar constraints between abstract word ordering (no constraints) and tree linearization (full tree constraints). For example, one constraint might consist of a set of dependency relations between input words, but which do not form a complete unordered spanning tree. We call this word ordering task the **partial-tree linearization** problem, a task that is perhaps closer to NLP applications

than both the abstract word ordering task and the full tree linearization problem, in the sense that NLG pipelines might provide some syntactic relations between words for the linearization step, but not the full spanning tree.

The main content of this section is based on a conference paper (Zhang 2013), which we extend by using the technique of expanding negative training examples (one of the overall contributions of this article).

4.1 Full- and Partial-Tree Linearization

Given a multi-set of input words W and a set of head-dependent relations H between the words in W, the task is to find an ordered sentence consisting of all the words in W and a dependency tree that contains all the relations in H. If each word in W is given a POS tag and H covers all words in W, then the task is (full-)tree linearization; if not then the task is partial-tree linearization. For partial-tree linearization, a subset of W is given fixed POS tags. In all cases, a word either has exactly one (gold) POS tag, or no POS tags.

4.2 The Edge Data Structure

Similar to the CCG case, **edge** refers to the data structure for a hypothesis in the decoding algorithm. Here a **leaf edge** refers to an input word with a POS tag, and a **non-leaf edge** refers to a phrase or sentence with its dependency tree. Edges are constructed bottom–up, by recursively joining two existing edges and adding an unlabeled dependency link between their head words.

As for the CCG system, edges are scored by a global linear model:

$$f(e) = \frac{\Phi(e) \cdot \vec{\theta}}{|e|}$$

where $\Phi(e)$ represents the feature vector of e and $\vec{\theta}$ is the parameter vector of the model. Table 2 shows the feature templates we use, which are inspired by the rich feature templates used for dependency parsing (Koo and Collins 2010; Zhang and Nivre 2011). In the table, h, m, s, h_l, h_r, m_l, and m_r are the indices of words in the newly constructed edge, where h and m refer to the head and dependent of the newly constructed arc, s refers to the nearest sibling of m (on the same side of h), and h_l, h_r, m_l, and m_r refer to the left and rightmost dependents of h and m, respectively. WORD, POS, LVAL, and RVAL are maps from indices to word forms, POS, left valencies, and right valencies of words, respectively. Example feature instances extracted from the sentence in Figure 3 are shown in the example column. Because of the non-local nature of some of the feature templates we define, we do not apply DP-style pruning for dependency-based tree-linearization.

4.3 The Decoding Algorithm

The decoding algorithm is similar to that of the CCG system, where an **agenda** is a priority queue for edges to expand, and **chart** is a fixed-size beam for a list of accepted edges. During initialization, input words are assigned possible POS tags, resulting in a set of leaf edges that are put onto the agenda. For words with POS constraints, only

Table 2
Feature templates. Indices on the surface string: h = head on newly added arc; m = dependent on arc; s = nearest sibling of m; b = any index between h and m; h_l, h_r = left/rightmost dependent of h; m_l, m_r = left/rightmost dependent 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+.

dependency syntax	**example**
WORD(h) · POS(h) · NORM(*size*) , WORD(h) · NORM(*size*), POS(h) · NORM(*size*) , WORD(h) · POS(h) · *dir*, WORD(h) · *dir*, POS(h) · *dir* , WORD(m) · POS(m) · *dir*, WORD(m) · *dir*, POS(m) · *dir* , WORD(h) · POS(h) · *dist*, WORD(h) · *dist*, POS(h) · *dist* , WORD(m) · POS(m) · *dist*, WORD(m) · *dist*, POS(m) · *dist* , WORD(h) · POS(h) · WORD(m) · POS(m) · *dir*, WORD(h) · POS(h) · WORD(m) · POS(m) · *dist* , WORD(h) · POS(h) · POS(m) · *dir*, WORD(h) · POS(h) · POS(m) · *dist* , POS(h) · WORD(m) · POS(m) · *dir*, POS(h) · WORD(m) · POS(m) · *dist*, POS(h) · POS(m) · *dir*, POS(h) · POS(m) · *dist*,	(*bought*, *VBD*, *4*)
POS(h) · POS(m) · POS(b) · *dir*,	(*VBD*, *NN*, *NNP*, *right*)
POS(h) · POS($h-1$) · POS(m) · POS($m+1$) · *dir* ($h > m$), POS(h) · POS($h+1$) · POS(m) · POS($m-1$) · *dir* ($h < m$),	(*VBD*, *NNP*, *NN*, *– END –* , *right*)
WORD(h) · POS(m) · POS(m_l) · *dir*, WORD(h) · POS(m) · POS(m_r) · *dir*, POS(h) · POS(m) · POS(m_l) · *dir*, POS(h) · POS(m) · POS(m_r) · *dir*,	
POS(h) · POS(m) · POS(s) · *dir*, POS(h) · POS(s) · *dir*, POS(m) · POS(s) · *dir*, WORD(h) · WORD(s) · *dir*, WORD(m) · WORD(s) · *dir*, POS(h) · WORD(s) · *dir*, POS(m) · WORD(s) · *dir*, WORD(h) · POS(s) · *dir*, WORD(m) · POS(s) · *dir*,	(*VBD*, *NN*, *NNP*, *right*)
WORD(h) · POS(m) · POS(s) · POS(s_2) · *dir*, POS(h) · POS(m) · POS(s) · POS(s_2) · *dir*,	
dependency syntax for completed words	
WORD(h) · POS(h) · WORD(h_l) · POS(h_l), POS(h) · POS(h_l), WORD(h) · POS(h) · POS(h_l), POS(h) · WORD(h_l) · POS(h_l), WORD(h) · POS(h) · WORD(h_r) · POS(h_r), POS(h) · POS(h_r), WORD(h) · POS(h) · POS(h_r), POS(h) · WORD(h_r) · POS(h_r), WORD(h) · POS(h) · LVAL(h), WORD(h) · POS(h) · RVAL(h), WORD(h) · POS(h) · LVAL(h) · RVAL(h), POS(h) · LVAL(h), POS(h) · RVAL(h), POS(h) · LVAL(h) · RVAL(h),	(*VBD*, *1*, *2*)
surface string patterns	
WORD($B-1$) · WORD(B), POS($B-1$) · POS(B), WORD($B-1$) · POS(B), POS($B-1$) · WORD(B), WORD($B-1$) · WORD(B) · WORD($B+1$), WORD($B-2$) · WORD($B-1$) · WORD(B), POS($B-1$) · POS(B) · POS($B+1$), POS($B-2$) · POS($B-1$) · POS(B), POS($B-1$) · WORD(B) · POS($B+1$), POS($B-2$) · WORD($B-1$) · POS(B),	(*NNP*, *bought*, *NNP*)

Table 2
(*continued*)

dependency syntax	example
surface string patterns for complete sentences	
WORD(0), WORD(0) · WORD(1),	
WORD($size-1$), WORD($size-1$) · WORD($size-2$),	
POS(0), POS(0) · POS(1), POS(0) · POS(1) · POS(2),	
POS($size-1$), POS($size-1$) · POS($size-2$),	
POS($size-1$) · POS($size-2$) · POS($size-3$),	(*VBD*, *NNP*, *NN*)

the allowed POS tag is assigned. For unconstrained words, we assign all possible POS tags according to a tag dictionary compiled from the training data, following standard practice for POS-tagging (Ratnaparkhi 1996).

When an edge is expanded, it is combined with all edges in the chart in all possible ways to generate new edges. Two edges can be combined by concatenation of the surface strings in both orders and, in each case, constructing a dependency link between their heads in two ways (corresponding to the two options for the head of the new link). When there is a head constraint on the dependent word, a dependency link can be constructed only if it is consistent with the constraint. This algorithm implements abstract word ordering, partial-tree linearization, and full tree linearization—all generalized word ordering tasks—in a unified method.

Pseudocode for the decoder is shown as Algorithm 4. Many of the functions have the same definition as for Algorithm 1: INITAGENDA, INITCHART, TIMEOUT, POPBEST, GOALTEST, ADD. CANCOMBINE checks whether two edges do not contain an overlapping word (i.e., they do not contain a word more times than its count in the input); unlike the CCG case, all pairs of words are allowed to combine according to the dependency model. COMBINE creates a dependency link between two words, with the word order determined by the order in which the arguments are supplied to the function, and the head coming from either the first (*HeadLeft*) or second (*HeadRight*) argument (so there are four combinations considered and COMBINE is called four times in Algorithm 4).

4.4 The Learning Algorithm

As for the CCG system, an online large-margin learning algorithm based on the decoding process is used to train the model. At each step, the expanded edge e is compared with the gold standard. If it is a gold edge, decoding continues; otherwise e is taken as a negative example $e\_$ and the lowest-scored gold edge in the agenda is taken

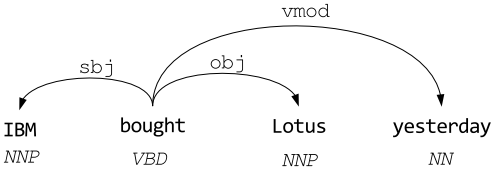

Figure 3
Feature template example.

Algorithm 4 The decoding algorithm for partial-tree linearization.

```
a ← INITAGENDA( )
c ← INITCHART( )
while not TIMEOUT( ) do
    new ← []
    e ← POPBEST(a)
    if GOALTEST(e) then
        return e
    end if
    for ẽ ∈ c do
        if CANCOMBINE(e, ẽ) then
            e′ ← COMBINE(e, ẽ, HeadLeft)
            APPEND(new, e′)
            e′ ← COMBINE(e, ẽ, HeadRight)
            APPEND(new, e′)
        end if
        if CANCOMBINE(ẽ, e) then
            e′ ← COMBINE(ẽ, e, HeadLeft)
            APPEND(new, e′)
            e′ ← COMBINE(ẽ, e, HeadRight)
            APPEND(new, e′)
        end if
    end for
    for e′ ∈ new do
        ADD(a, e′)
    end for
    ADD(c, e)
end while
```

as a positive example e_+, and a parameter update is executed (repeated here from Section 3.4):

$$\vec{\theta} \leftarrow \vec{\theta}_0 + \frac{\hat{f}(e_-) - \hat{f}(e_+) + 1}{\| \frac{\Phi(e_+)}{|e_+|} - \frac{\Phi(e_-)}{|e_-|} \|^2} \left(\frac{\Phi(e_+)}{|e_+|} - \frac{\Phi(e_-)}{|e_-|} \right)$$

The training process is essentially the same as in Algorithm 3, but with the CCG grammar and model replaced with the dependency-based grammar and model.

In our conference paper describing the earlier version of the dependency-based system (Zhang 2013), the decoding step is finished immediately after the parameter update; in this article we expand the negative example, as in Algorithm3, putting it onto the chart and thereby exploring a larger part of the search space (in particular that part containing negative examples). Our later experiments show that this method yields improved results, consistent with the CCG system.

Table 3
Training, development, and test data from the Penn Treebank.

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

5. Experiments

We use CCGBank (Hockenmaier and Steedman 2007) and the Penn Treebank (Marcus, Santorini, and Marcinkiewicz 1993) for CCG and dependency data, respectively. CCGbank is the CCG version of the Penn Treebank. Standard splits were used for both: Sections 02–21 for training, Section 00 for development, and Section 23 for the final test. Table 3 gives statistics for the Penn Treebank.

For the CCG experiments, original sentences from CCGBank are transformed into bags of words, with sequence information removed, and passed to our system as input data. The system outputs are compared to the original sentences for evaluation. Following Wan et al. (2009), we use the BLEU metric (Papineni et al. 2002) for string comparison. Although BLEU is not the perfect measure of fluency or grammaticality, being based on *n*-gram precision, it is currently widely used for automatic evaluation and allows us to compare directly with existing work (Wan et al. 2009). Note also that one criticism of BLEU for evaluating machine translation systems (i.e., that it can only register exact matches between the same words in the system and reference translation), does not apply here, because the system output always contains the same words as the original reference sentence. For the dependency-based experiments, gold-standard dependency trees were derived from bracketed sentences in the treebank using the Penn2Malt tool.[2]

For fair comparison with Wan et al. (2009), we keep base NPs as atomic units when preparing the input. Wan et al. used base NPs from the Penn Treebank annotation, and we follow this practice for the dependency-based experiments. For the CCG experiments we extract base NPs from CCGBbank by taking as base NPs those NPs that do not recursively contain other NPs. These base NPs mostly correspond to the base NPs from the Penn Treebank: In the training data, there are 242,813 Penn Treebank base NPs with an average size of 1.09, and 216,670 CCGBank base NPs with an average size of 1.19.

5.1 Convergence of Training

The plots in Figure 4 show the development test scores of three CCG models by the number of training iterations. The three curves represent the scaled model of this article, the online large-margin model from Zhang, Blackwood, and Clark (2012), and the perceptron model from Zhang and Clark (2011), respectively. For each curve, the BLEU score generally increases as the number of training iterations increases, until it reaches its maximum at a particular iteration. We use the number of training iterations

2 http://w3.msi.vxu.se/~nivre/research/Penn2Malt.html.

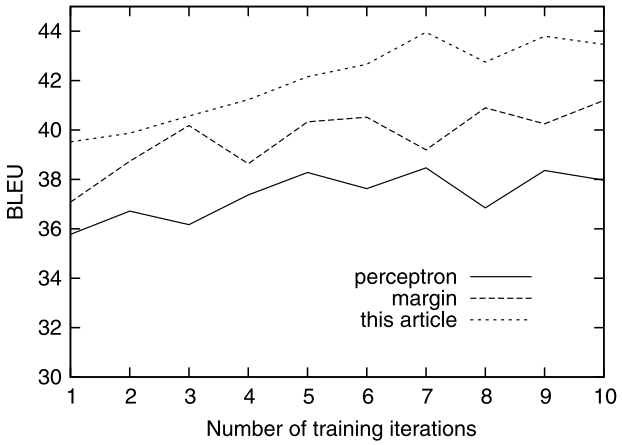

Figure 4
BLEU scores of the perceptron, large-margin, and scaled large-margin CCG models by the number of training iterations.

that gives the best development test scores for the training of our model when testing on the test data.

Another way to observe the convergence of training is to measure the training times for each iteration at different numbers of iterations. The per-iteration training times for the large-margin and the scaled CCG models are shown in Figure 5. For each model, the training time for each iteration decreases as the number of training iterations increases, reflecting the convergence of learning-guided search. When the model gets better, fewer non-gold hypotheses are expanded before gold hypotheses, and hence it takes less time for the decoder to find the gold goal edge. Figure 6 shows the corresponding curve for dependency-based word ordering, with similar observations.

Because of the expanding of negative examples, the systems of this article took more time to train than those of our previous conference papers. However, the convergence

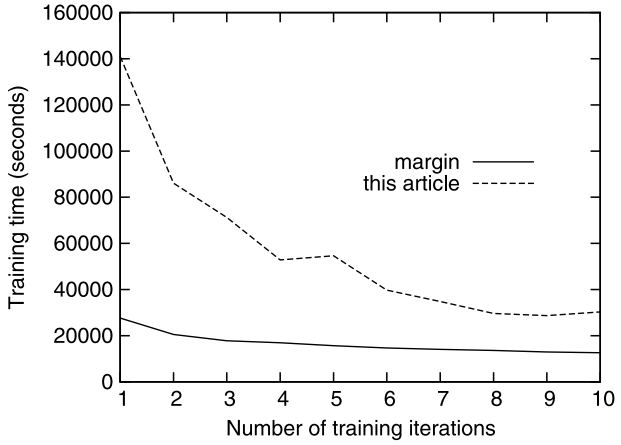

Figure 5
Training times of the large-margin model and the scaled CCG models by the number of training iterations.

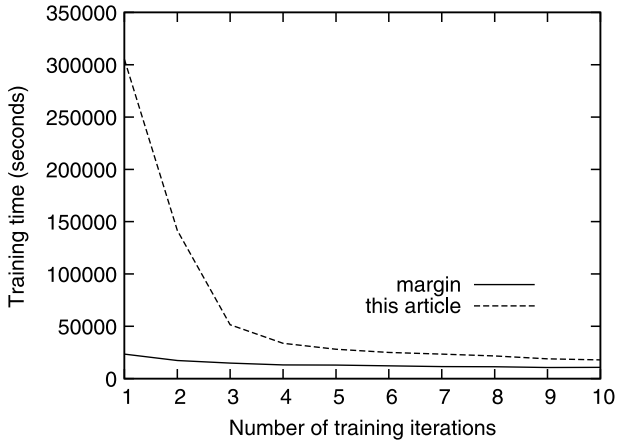

Figure 6
Training times of the large-margin and scaled dependency models by the number of training iterations.

rate is also faster when negative training examples are expanded, as demonstrated by the rate of speed improvement as the number of training iterations increases. The training times of the perceptron algorithm are close to those of the large-margin algorithm, and hence are omitted from Figures 5 and 6. The new model gives the best development test scores, as shown in Figure 4. The next section investigates the effects of two of the innovations of this article: use of negative examples during training and the scaling of the model by hypothesis size.

5.2 The Effect of the Scaled Model and Negative Examples

Table 4 shows a set of CCG development experiments to measure the effect of the scaled model and the expansion of negative examples during training. With the standard linear model (Zhang, Blackwood, and Clark 2012) and no expansions of negative examples, our system obtained a BLEU score of 39.04. The scaled model improved the BLEU score by 1.41 BLEU points to 40.45, and the expansion of negative examples gave a further improvement of 3.02 BLEU points.

These CCG development experiments show that the expansion of negative examples during training is an important factor in achieving good performance. When no negative examples are expanded, the higher score of the scaled linear model demonstrates the effectiveness of fair comparison between edges with different sizes. However,

Table 4
The effect of the scaled model and expansion of negative examples during training for the CCG system.

	standard linear model	scaled linear model
no expansion of negative examples	39.04	40.45
expansion of negative examples	no convergence	43.47

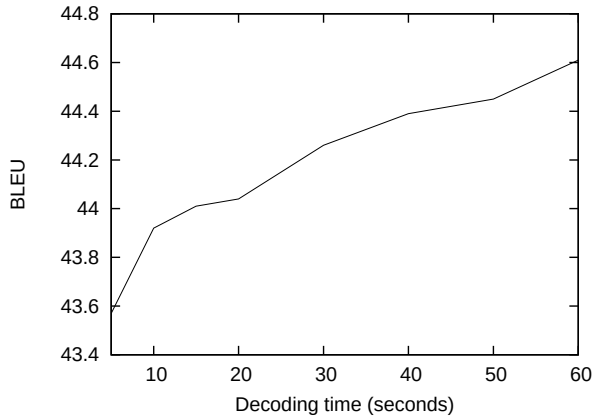

Figure 7
The effect of search time for the CCG system on the development test data.

it is a more important advantage of the scaled linear model that it allows the expansion of negative examples during training, which was not possible with the standard linear model. In the latter case, training failed to converge when negative examples were expanded, reflecting the limitations of the standard linear model in separating the training data. Similar results were found for dependency-based word ordering, where the best development BLEU score improved from 44.71 (Zhang 2013) to 46.44 with the expansion of negative training examples.

5.3 The Effect of Search Time

Figure 7 shows the BLEU scores for the CCG system on the development data when the timeout limit for decoding a single sentence is set to 5 sec, 10 sec, 15 sec, 20 sec, 30 sec, 40 sec, 50 sec, and 60 sec, respectively. The timeout was applied during decoding at test time. The scaled model with negative training examples was used for this set of experiments, and the same model was used for all timeout settings. The results demonstrate that better outputs can be recovered given more search time, which is expected for a time-constrained best-first search framework. Recall that output is created greedily by combining the largest available edges, when the system times out. Similar results were obtained with the dependency-based system of Zhang (2013), where the development BLEU scores improved from 42.89 to 43.42, 43.58, and 43.72 when the timeout limit increased from 5 sec to 10 sec, 30 sec, and 60 sec, respectively. The scaled dependency-based model without expansion of negative examples was used in this set of experiments.

5.4 Example Outputs

Example output for sentences in the development set is shown in Tables 5 and 6, grouped by sentence length. The CCG systems of our previous conference papers and this article are compared, all with the timeout value set to 5 sec. All three systems perform relatively better with smaller sentences. For longer sentences, the fluency of

Table 5
Example development output for the CCG-based systems and sentences with fewer than 20 words.

Zhang and Clark (2011)	Zhang, Blackwood, and Clark (2012)	this article	reference
. A Lorillard spokewoman said This is an old story,	A Lorillard spokewoman said, This is an old story.	A Lorillard spokewoman said, This is an old story.	A Lorillard spokewoman said, " This is an old story.
It today has no bearing on our work force.	It today has no bearing on our work force.	It has no bearing on our work force today.	It has no bearing on our work force today.
No price for the new shares has been set.	No price for the new shares has been set.	No price has been set for the new shares.	No price for the new shares has been set.
Previously he was vice president of Eastern Edison.	Previously he was vice president of Eastern Edison.	Previously the was vice president of Eastern Edison.	Previously he was vice president of Eastern Edison.
Big mainframe computers for had for years been around business.	had been Big mainframe computers for business for years around.	Big mainframe computers had for years been around for business.	Big mainframe computers for business had been around for years.
The field of reserves has 21 million barrels.	The field has 21 million barrels of reserves.	The field has 21 million barrels of reserves.	The field has reserves of 21 million barrels.
is some heavy-duty competition Behind all the hoopla.	all the hoopla Behind is some heavy-duty competition.	some heavy-duty competition is Behind all the hoopla.	Behind all the hoopla is some heavy-duty competition.
Pamela Sebastian contributed to New York in this article.	Pamela Sebastian in New York contributed to this article.	Pamela Sebastian in New York contributed to this article.	Pamela Sebastian in New York contributed to this article.
painted oils in by Lighthouse II was the playwright in 1901 ...	Lighthouse II was the playwright in 1901 ... in oils	oils in Lighthouse II was painted by the playwright in 1901 ...	Lighthouse II was painted in oils by the playwright in 1901 ...

Example output for the CCG-based systems and sentences with fewer than ten words.

Zhang and Clark (2011)	Zhang, Blackwood, and Clark (2012)	this article	reference
Mr. Vinken is Elsevier N.V., chairman of the Dutch publishing group.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.
asbestos-related diseases have Four, including three of the five surviving workers with recently diagnosed cancer.	have Four of asbestos-related diseases with the five surviving workers, including recently diagnosed cancer. three	Four with the five surviving workers, including three of asbestos-related diseases have recently diagnosed cancer.	Four of the five surviving workers have asbestos-related diseases, including three with recently diagnosed cancer.
, interest rates expect to slide amid further declines in Yields on money-market mutual funds that portfolio managers continued. signs	further declines expect signs that Yields continued to slide in interest rates on money-market mutual funds, amid portfolio managers.	Yields expect further declines in interest rates on money-market mutual funds, amid signs that portfolio managers continued to slide.	Yields on money-market mutual funds continued to slide, amid signs that portfolio managers expect further declines in interest rates.
The thrift holding company expects it said obtain regulatory approval by year-end and to complete the transaction.	The thrift holding company said it expects regulatory approval by to complete the transaction and obtain year-end.	The thrift holding company said it expects the transaction to complete by year-end and obtain regulatory approval.	The thrift holding company said it expects to obtain regulatory approval and complete the transaction by year-end.
Finmeccanica is an Italian state-owned holding company with interests in the mechanical engineering industry.	Finmeccanica is an Italian state-owned holding company with interests in the mechanical engineering industry.	Finmeccanica is an Italian state-owned holding company with interests in the mechanical engineering industry.	Finmeccanica is an Italian state-owned holding company with interests in the mechanical engineering industry.

Example output for the CCG-based systems and sentences between 11 and 20 words.

Table 6
Example development output for the CCG-based systems and sentences with more than 20 words.

Zhang and Clark (2011)	Zhang, Blackwood, and Clark (2012)	this article	reference
Dr. Talcott from Boston University and a team led researchers and of the National Cancer Institute of the medical schools Harvard University	the medical schools from Dr. Talcott and Harvard University of the National Cancer Institute and a team of led Boston University	researchers led Dr. Talcott of Harvard University and the National Cancer Institute and a team of the medical schools from Boston University.	Dr. Talcott led a team of researchers from the National Cancer Institute and the medical schools of Harvard University and Boston University.
because they indicate portfolio managers declining interest rates permit to retain a longer period for Longer maturities are. to relatively higher rates thought	because declining interest rates thought Longer maturities permit relatively higher rates for to retain to indicate portfolio managers are a longer period. they	declining interest rates permit portfolio managers to retain relatively higher rates for Longer maturities because they are thought to indicate a longer period.	Longer maturities are thought to indicate declining interest rates because they permit portfolio managers to retain relatively higher rates for a longer period.
Royal Trustco Ltd. said Pacific First Financial Corp. approved $ 27 or a share, for its acquisition of shareholders by Toronto. $ 212 million	by Toronto shareholders said Pacific First Financial Corp. approved its acquisition of Royal Trustco Ltd. for $ 212 million, or $ 27 a share.	Toronto said Pacific First Financial Corp. approved its acquisition of Royal Trustco Ltd. for shareholders by $ 212 million, or $ 27 a share.	Pacific First Financial Corp. said shareholders approved its acquisition by Royal Trustco Ltd. of Toronto for $ 27 a share, or $ 212 million.

Example output for the CCG-based systems and sentences between 21 and 40 words.

Zhang and Clark (2011)	Zhang, Blackwood, and Clark (2012)	this article	reference
European history, an educational research organization, the test and only French history questions, Education, a European history class, John Cannell, an Albuquerque, N.M., psychiatrist and founder their kids standardized testing	European history, only French history questions, an educational research organization, John Cannell, Education, France and a European history class when they decided, standardized testing has studied an Albuquerque, N.M., psychiatrist of their kids and founder. says the test	when an Albuquerque, N.M., psychiatrist, say an educational research organization decided to give students and a European history class, which has studied Friends for their kids and founder of only French history questions, European history the test standardized testing John Cannell	It's as if France decided to give only French history questions to students in a European history class, and when everybody aces the test, they say their kids are good in European history, says John Cannell, an Albuquerque, N.M., psychiatrist and founder of an educational research organization, Friends for Education, which has studied standardized testing.
which than at about # 15 billion over United Kingdom institutional funds (Still, Richard Barfield, said, Much less {index-arbitrage activity} manages $ 23.72 billion.) Standard Life Assurance Co. chief investment manager the U.S.	Standard Life Assurance Co. in United Kingdom institutional funds, chief investment manager here is done at about # 15 billion than (, Still, Richard Barfield said Much less {index-arbitrage activity} manages $ 23.72 billion in the U.S..) over which	done in Much less {index-arbitrage activity} in the U.S., Richard Barfield (United Kingdom institutional funds, which manages $ 23.72 billion), chief investment manager at Standard Life Assurance Co. about # 15 billion	Still, Much less {index-arbitrage activity} is done over here than in the U.S. said Richard Barfield, chief investment manager at Standard Life Assurance Co., which manages about # 15 billion ($ 23.72 billion) in United Kingdom institutional funds.

Example output for the CCG-based systems and sentences with more than 40 words.

the output is significantly reduced. One source of errors is confusion between different noun phrases, and where they should be positioned, which becomes more severe with increased sentence length and adds to the difficulty in reading the outputs. The system of this article gave observably improved outputs compared with the two other systems.

Table 7
Development BLEU scores for partial-tree linearization, with different proportions of input POS and dependency information randomly selected from full gold-standard trees.

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
42.89	43.41	44.71	50.46	51.40	52.18	73.34	74.68	76.28

5.5 Partial-Tree Linearization

In the previous section, the same input settings were used for both training and testing, and the assumption was made that the input to the system would be a bag of words, with no constraints on the output structure. This somewhat artificial assumption allows a standardized evaluation but, as discussed previously, text generation applications are unlikely to satisfy this assumption and, in practice, the realization problem is likely to be *easier* compared with our previous set-up. In this section, we simulate practical situations in dependency-based pipelines by measuring the performance of our system using randomly chosen input POS tags and dependency relations. For maximum flexibility, so that the same system can be applied to different input scenarios, our system is trained without input POS tags or dependencies. However, if POS tags and dependencies are made available during testing, they will be used to provide hard constraints on the output (i.e., the output sentence with POS tags and dependencies must contain those in the input). From the perspective of search, input POS tags and dependencies greatly constrain the search space and lead to an easier search problem, with correspondingly improved outputs.

Table 7 shows a set of development results with varying amounts of POS and dependency information in the input. For each test, we randomly sampled a percentage of words for which the gold-standard POS tags or dependencies are given in the input. As can be seen from the table, increased amounts of POS and dependency information in the input lead to higher BLEU scores, and dependencies were more effective than POS tags in determining the word order in the output. When all POS tags and dependencies are given, our constraint-enabled system gave a BLEU score of 76.28.[3]

Table 8 shows the output of our system for the first nine development test sentences with different input settings. These examples illustrate the positive effect of input dependencies in specifying the outputs. Consider the second sentence as an example. When only input words are given, the output of the system is largely grammatical but nonsensical. With increasing amounts of dependency relations, the output begins to look more fluent, sometimes with the system reproducing the original sentence when all dependencies are given.

5.6 Final Results

Table 9 shows the test results of various systems. For the system of this article, we take the optimal setting from the development tests, using the scaled linear model and

3 When all POS tags and dependencies are also provided *during training*, the BLEU score is reduced to 74.79, showing the value in the system, which can adapt to varying amounts of POS and dependency information in the input at test time.

Table 8
Partial-tree linearization outputs for the first nine development test sentences with various input information.

no POS, no dep	50% unlabeled dep, no POS	gold unlabeled dep, 50% POS	reference
the board as a nonexecutive director will join Pierre Vinken, 61 years old, Nov. 29.	Pierre Vinken, 61 years old, will join the board Nov. 29 as a nonexecutive director.	Pierre Vinken, 61 years old, will join the board Nov. 29 as a nonexecutive director.	Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.
Mr. Vinken, the Dutch publishing group of Elsevier N.V. is chairman.	chairman of Elsevier N.V., the Dutch publishing group is Mr. Vinken.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.	Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.
and Consolidated Gold Fields PLC, former chairman of Rudolph Agnew, 55 years old was named a nonexecutive director of this British industrial conglomerate.	, Rudolph Agnew, was 55 years old and former chairman of Consolidated Gold Fields PLC named a nonexecutive director of this British industrial conglomerate.	Rudolph Agnew, 55 years old and former chairman of Consolidated Gold Fields PLC, was named a nonexecutive director of this British industrial conglomerate.	Rudolph Agnew, 55 years old and former chairman of Consolidated Gold Fields PLC, was named a nonexecutive director of this British industrial conglomerate.
it reported A form of asbestos, a high percentage of Kent cigarette filters used to make once exposed to researchers among workers a group of cancer deaths has caused more than 30 years ago.	, reported has A form asbestos once used to make Kent cigarette filters caused a high percentage of cancer deaths among a group of workers exposed to researchers more than 30 years ago.	, researchers reported A form of asbestos once used to make Kent cigarette filters has caused a high percentage of cancer deaths among a group of workers exposed to it more than 30 years ago.	A form of asbestos once used to make Kent cigarette filters has caused a high percentage of cancer deaths among a group of workers exposed to it more than 30 years ago, researchers reported.
The asbestos fiber with the lungs once it is causing even brief exposures to show researchers that enters up decades later.	it enters The asbestos fiber causing it later show up with the lungs that is unusually resilient to researchers even brief exposures.	once it enters the lungs crocidolite, is unusually resilient, with symptoms that show up decades later causing even brief exposures to it, researchers said.	The asbestos fiber, crocidolite, is unusually resilient once it enters the lungs, with even brief exposures to it causing symptoms that show up decades later, researchers said.
crocidolite stopped its Micronite cigarette filters in Kent cigarettes that makes Lorillard Inc. in 1956, the unit of New York-based Loews Corp., using.	Lorillard Inc. stopped using the unit of New York-based Loews Corp., that makes Kent cigarettes, in 1956.	Lorillard Inc. the unit of New York-based Loews Corp., that makes Kent cigarettes, stopped using crocidolite in its Micronite cigarette filters in 1956.	Lorillard Inc., the unit of New York-based Loews Corp. that makes Kent cigarettes, stopped using crocidolite in its Micronite cigarette filters in 1956.
new attention Although a forum more than a year ago, the problem were likely to appear to bring preliminary findings of Medicine in today's New England Journal, the latest results.	of Medicine the problem bring new attention to preliminary findings in a forum, more than a year the latest results were reported to today's New England Journal.	Although preliminary findings were reported more than a year ago, the latest results appear in today's New England Journal of Medicine, a forum likely to bring new attention to the problem.	Although preliminary findings were reported more than a year ago, the latest results appear in today's New England Journal of Medicine, a forum likely to bring new attention to the problem.
A Lorillard spokewoman said, "This is an old story.	A Lorillard spokewoman said, "This is an old story.	A Lorillard spokewoman said, "This is an old story.	A Lorillard spokewoman said, "This is an old story.
We're anyone before talking about asbestos of having any questionable properties heard years ago.	We're talking about anyone having heard before any questionable properties of asbestos years ago.	We're talking about years ago before anyone heard of having asbestos any questionable properties.	We're talking about years ago before anyone heard of asbestos having any questionable properties.

Table 9
Final test results on the standard word ordering task.

System	BLEU
Wan et al. (2009) (dependency)	33.7
Zhang and Clark (2011) (CCG)	40.1
Zhang, Blackwood, and Clark (2012) (CCG)	42.5
Zhang, Blackwood, and Clark (2012) +LM (CCG)	43.8
Zhang (2013) (dependency)	46.8
This article (CCG)	46.5
This article (dependency)	**48.7**

expansion of negative examples during training. For direct comparison with previous work, the timeout threshold was set to 5 sec. Our new system of this article significantly outperforms all previous systems and achieves the best published BLEU score on this task. It is worth noting that our systems without a language model outperform the system of our 2012 paper using a large-scale language model.

Interestingly, the dependency-based systems performed better than the CCG systems of this article. One of the main reasons is that the CCG systems generated shorter outputs by not finding full spanning derivations for a larger proportion of inputs. Because of the rigidity in combinatory rules, not all hypotheses in the chart can be combined with the hypothesis being expanded, leading to an increased likelihood of full spanning derivations being unreachable. Overall, the CCG system recovered 93.98% of the input words in the test set, and the dependency system recovered 97.71%.

5.7 Shared Task Evaluation

The previous sections report evaluations on the task of word ordering, an abstract yet fundamental problem in text generation. One question that is not addressed by these experiments is how the abstract task can be utilized to benefit full text generation, for which more considerations need to be taken into account in addition to word ordering. We investigate this question using the 2011 Generation Challenge shared task data, which provide a common-ground for the evaluation of text generation systems (Belz et al. 2011).

The data are based on the CoNLL 2008 shared task data (Surdeanu et al. 2008), which consist of selected sections of the Penn WSJ Treebank, converted to syntactic dependencies via the LTH tool (Johansson and Nugues 2007). Sections 2–21 are used for training, Section 24 for development, and Section 23 for testing. A small number of sentences from the original WSJ sections are not included in this set. The input format of the shared task is an unordered syntactic dependency tree, with nodes being lemmas, and dependency relations on the arcs. Named entities and hyphenated words are broken into individual nodes, and special dependency links are used to mark them. Information on coarse-grained POS, number, tense, and participle features is given to each node where applicable. The output is a fully ordered and inflected sentence.

We developed a full-text generation system according to this task specification, with the core component being the dependency-based word ordering system of Section 4. In addition to minor engineering details that were required to adapt the system to this new

task, one additional task that the generation system needs to carry out is morphological generation—finding the appropriate inflected form for each input lemma. Our approach is to perform joint word ordering and inflection using the learning-guided search framework, letting one statistical model decide the best order as well as the inflections of ambiguous lemmas. For a lemma, we generate one or more candidate inflections by using a lexicon and a set of inflection rules. Candidate inflections for an input lemma are generated according to the lemma itself and its input attributes, such as the number and tense. Some lemmas are unambiguous, which are inflected before being passed to the word ordering system. For the other lemmas, more than one candidate's inflections are passed as input words to the word ordering system. To ensure that each lemma occurs only once in the output, a unique ID is given to all the inflections of the same lemma, making them mutually exclusive.

Four types of lemmas need morphological generation, including nouns, verbs, adjectives, and miscellaneous cases. The last category includes *a* (*a* or *an*) and *not* (*not* or *n't*), for which the best inflection can be decided only when *n*-gram information is available. For these lemmas, we pass all possible inflections to the search module. For nouns and adjectives, the inflection is relatively straightforward, since the number (e.g., singular, plural) of a lemma is given as an attribute of the input node, and comparative and superlative adjectives have specific parts of speech. For those cases where the necessary information is not available from the input, all possible inflections are handed over to the search module for further disambiguation. The most ambiguous lemma types are verbs, which can be further divided into *be* and other verbs. The uniqueness of *be* is that the inflections for the first and second person can be different. All verb inflections are disambiguated according to the tense and participle attributes of the input node. In addition, for verbs in the present tense, the subject needs to be determined in order to differentiate between third-person singular verbs and others. This can be straightforward when the subject is a noun or pronoun, but can be ambiguous when the subject is a *wh*-pronoun, in which case the real subject might not be directly identifiable from the dependency tree. We leave all possible inflections of *be* and other verbs to the word ordering system whenever the ambiguity is not directly solvable from the subject dependency link. Overall, the pre-processing step generates 1.15 inflections for each lemma on average.

For word ordering, the search procedure of Algorithm 4 is applied directly, and the feature templates of Table 2 are used with additional labeled dependency features described subsequently. The main reason that the dependency-based word ordering algorithm can perform joint morphological disambiguation is that it uses rich syntactic and *n*-gram features to score candidate hypotheses, which can also differentiate between correct and incorrect inflections under particular contexts. For example, *an honest person* and *a honest person* can be differentiated by *n*-gram features, while *Tom and Sally is* and *Tom and Sally are* can be differentiated by higher-order dependency features.

In addition to lemma-formed inputs, one other difference between the shared task and the word ordering problem solved by Algorithm 4 is that the former uses labeled dependencies whereas Algorithm 4 constructs unlabeled dependency trees. We address this issue by assigning dependency labels in the construction of dependency links, and applying an extra set of features. The new features are defined by making a duplicate of all the features from Table 2 that contain *dir* information, and associating each feature in the new copy with a dependency label.

The training of the word ordering system requires fully ordered dependency trees, while references in the shared task data are raw sentences. We perform a pre-processing

Table 10
Results and comparison with the top-performing systems on the shared task data

System	BLEU
STUMABA	89.1
DCU	85.8
this article	89.6

step to obtain gold-standard training data by matching the input lemmas to the reference sentence in order to obtain their gold-standard order. More specifically, given a training instance, we generate all candidate inflections for each lemma, resulting in an exponential set of possible mappings between the input tree and the reference sentence. We then prune these mappings bottom–up, assuming that the dependency tree is projective, and therefore that each word dominates a continuous span in the reference. After such pruning, only one correct mapping is found for the majority of the cases. For the cases where more than one mapping is found, we randomly choose one as the gold-standard. There are also instances for which no correct ordering can be found, and these are mostly due to non-projectivity in the shared task data, with a few cases being due to conflicts between our morphological generation system and the shared task data, or inconsistency in the data itself. Out of the 39K training instances, 2.8K conflicting instances are discarded, resulting in 36.2K gold-standard ordered dependency trees.

Table 10 shows the results of our system and the top two participating systems of the shared task. Our system outperforms the STUMABA system by 0.5 BLEU points, and the DCU system by 3.8 BLEU points. More evaluation of the system was published in Song et al. (2014).

6. Related Work

There is a recent line of research on text-to-text generation, which studies the linearization of dependency structures (Barzilay and McKeown 2005; Filippova and Strube 2007, 2009; He et al. 2009; Bohnet et al. 2010; Guo, Hogan, and van Genabith 2011). On the other hand, Wan et al. (2009) study the ordering of a bag of words without any dependency information given. We generalize the word ordering problem, and formulate it as a task of ordering a multi-set of words, regardless of input syntactic constraints.

Our bottom–up, chart-based generation algorithm is inspired by the line of work on chart-based realization (Kay 1996; Carroll et al. 1999; White 2004, 2006; Carroll and Oepen 2005). Kay (1996) first proposed the concept of chart realization, drawing analogies between realization and parsing of free order languages. He discussed efficiency issues and provided solutions to specific problems. For the task of realization, efficiency improvement has been further investigated (Carroll et al. 1999; Carroll and Oepen 2005). The inputs to these systems are logical forms, which form natural constraints on the interaction between edges. In our case, one constraint that has been leveraged in the dependency system is a projectivity assumption—we assume that the dependents of a word must all have been attached before the word is attached to its head word, and that spans do not cross during combination. In addition, we assume that the right dependents of a word must have been attached before a left dependent of the word is attached. This constraint avoids spurious ambiguities. The projectivity assumption

is an important basis for the feasibility of the dependency system; it is similar to the chunking constraints of White (2006) for CCG-based realization.

White (2004) describes a system that performs CCG realization using best-first search. The search process of our algorithm is similar to that work, but the input is different: logical forms in the case of White (2004) and bags of words in our case. Further along this line, Espinosa, White, and Mehay (2008) also describe a CCG-based realization system, applying "hypertagging"—a form of supertagging—to logical forms in order to make use of CCG lexical categories in the realization process. White and Rajkumar (2009) further use perceptron reranking on n-best realization output to improve the quality.

The use of perceptron learning to improve search has been proposed in guided learning for easy-first search (Shen, Satta, and Joshi 2007) and LaSO (Daumé and Marcu 2005). LaSO is a general framework for various search strategies. Our learning algorithm is similar to LaSO with best-first inference, but the parameter updates are different. In particular, LaSO updates parameters when all correct hypotheses are lost, but our algorithm makes an update as soon as the top item from the agenda is incorrect. Our algorithm updates the parameters using a stronger precondition, because of the large search space. Given an incorrect hypothesis, LaSO finds the corresponding gold hypothesis for a perceptron update by constructing its correct sibling. In contrast, our algorithm takes the lowest scored gold hypothesis currently in the agenda to avoid updating parameters for hypotheses that may have not been constructed.

Our parameter update strategy is closer to the guided learning mechanism for the easy-first algorithm of Shen, Satta, and Joshi (2007), which maintains a queue of hypotheses during search, and performs learning to ensure that the highest-scored hypothesis in the queue is correct. However, in easy-first search, hypotheses from the queue are ranked by the score of their next action, rather than the hypothesis score. Moreover, Shen, Satta, and Joshi use aggressive learning and regenerate the queue after each update, but we perform non-aggressive learning, which is faster and is more feasible for our large and complex search space. Similar methods to Shen, Satta, and Joshi (2007) have also been used in Shen and Joshi (2008) and Goldberg and Elhadad (2010).

Another framework that closely integrates learning and search is SEARN (Daumé, Langford, and Marcu 2009), which addresses structured prediction problems that can be transformed into a series of simple classification tasks. The transformation is akin to greedy search in the sense that the complex structure is constructed by sequential classification decisions. The key problem that SEARN addresses is how to learn the tth decision based on the previous $t-1$ decisions, so that the overall loss in the resulting structure is minimized. Similar to our framework, SEARN allows arbitrary features. However, SEARN is more oriented to greedy search, optimizing local decisions. In contrast, our framework is oriented to best-first search, optimizing global structures.

Learning and search also interact with each other in a global discriminative learning and beam-search framework for incremental structured prediction (Zhang and Clark 2011). In this framework, an output is constructed incrementally by a sequence of transitions, while a beam is used to record the highest scored structures at each step. Online training is performed based on the search process, with the objective function being the margin between correct and incorrect structures. The method involves an early-update strategy, which stops search and updates parameters immediately when the gold structure falls out of the beam during training. It was first proposed by Collins and Roark (2004) for incremental parsing, and later gained popularity in the investigations of many NLP tasks, including POS-tagging (Zhang and Clark 2010), transition-based dependency parsing (Zhang and Clark 2008; Huang and Sagae 2010), and machine

translation (Liu 2013). Huang, Fayong, and Guo (2012) propose a theoretical analysis to the early-update training strategy, pointing out that it is a type of training method that fixes score violations in inexact search. When the score of a gold-standard structure is lower than that of a non-gold structure, a violation exists. Our parameter udpate strategy in this article can also be treated as a mechanism for violation fixing.

7. Conclusion

We investigated the general task of syntax-based word ordering, which is a fundamental problem for text generation, and a computationally very expensive search task. We provide a principled solution to this problem using learning-guided search, a framework that is applicable to other NLP problems with complex search spaces. We compared different methods for parameter updates, and showed that a scaled linear model gave the best results by allowing better comparisons between phrases of different sizes, increasing the separability of hypotheses and enabling the expansion of negative examples during training.

We formulate abstract word ordering as a spectrum of tasks with varying input specificity, from "pure" word ordering without any syntactic information to fully-informed word ordering with a complete unordered dependency tree given. Experiments show that our proposed method can effectively use available input constraints in generating output sentences.

Evaluation on the NLG 2011 shared task data shows that our system can be successfully applied to a more realistic application scenario, in particular one where some dependency constraints are provided in the input and word inflection is required as well as word ordering. Additional tasks that may be required in a practical text generation scenario include word selection, including the determination of content words and generation of function words. The joint modeling solution that we have proposed for word ordering and inflection could also be adopted for word selection, although the search space is greatly increased when the words themselves need deciding, particularly content words.

Acknowledgments

This work was carried out partly while Yue Zhang was a postdoctoral research associate at the University of Cambridge Computer Laboratory, where he and Stephen Clark were supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762, and partly after Yue Zhang joined Singapore University of Technology and Design, where he was supported by the MOE grant T2-MOE-2013-01. We thank Bill Byrne, Marcus Tomalin, Adrià de Gispert, and Graeme Blackwood for numerous discussions; Anja Belz and Mike White for kindly providing the NLG 2011 shared task data; Kai Song for helping with tables and figures in the draft; Yijia Liu for helping with the bibliography; and the anonymous reviewers for the many constructive comments that have greatly improved this article since the first draft.

References

Auli, Michael and Adam Lopez. 2011. Training a log-linear parser with loss functions via softmax-margin. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 333–343, Edinburgh.

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

Belz, Anja, 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, ENLG '11, pages 217–226, Stroudsburg, PA.
Blackwood, Graeme, 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 (Coling 2010)*, pages 71–79, Beijing.
Bohnet, Bernd, Leo Wanner, Simon Mill, 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 (Coling 2010)*, pages 98–106, Beijing.
Bos, Johan, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of COLING-04*, pages 1240–1246, Geneva.
Caraballo, Sharon A. and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24:275–298.
Carroll, John, Ann Copestake, Dan Flickinger, and Victor Poznanski. 1999. An efficient chart generator for (semi-) lexicalist grammars. In *Proceedings of the 7th European Workshop on Natural Language Generation (EWNLG99)*, pages 86–95, Toulouse.
Carroll, John and Stephan Oepen. 2005. High efficiency realization for a wide-coverage unification grammar. In *Proceedings of the Second International Joint Conference on Natural Language Processing*, IJCNLP'05, pages 165–176, Berlin.
Chang, Pi-Chuan and Kristina Toutanova. 2007. A discriminative syntactic word order model for machine translation. In *Proceedings of ACL*, pages 9–16, Prague.
Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.
Clark, Stephen and James R. Curran. 2007a. Perceptron training for a wide-coverage lexicalized-grammar parser. In *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing*, pages 9–16, Prague.
Clark, Stephen and James R. Curran. 2007b. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.
Collins, Michael and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of ACL*, pages 111–118, Barcelona.
Crammer, Koby, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.
Daumé, Hal, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.
Daumé, Hal and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176, Bonn.
Espinosa, Dominic, Michael White, and Dennis Mehay. 2008. Hypertagging: Supertagging for surface realization with CCG. In *Proceedings of ACL-08: HLT*, pages 183–191, Columbus, OH.
Filippova, Katja and Michael Strube. 2007. Generating constituent order in German clauses. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 320–327, Prague.
Filippova, Katja and Michael Strube. 2009. Tree linearization in English: Improving language model based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 225–228, Boulder, CO.
Fowler, Timothy A. D. and Gerald Penn. 2010. Accurate context-free parsing with combinatory categorial grammar. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 335–344, Uppsala.
Goldberg, Yoav and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Los Angeles, CA.
Guo, Yuqing, Deirdre Hogan, and Josef van Genabith. 2011. DCU at generation challenges 2011 surface realisation track. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 227–229, Nancy.
He, Wei, 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, pages 809–816, Suntec.

Hockenmaier, Julia. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, School of Informatics, University of Edinburgh.

Hockenmaier, Julia and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

Huang, Liang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal.

Huang, Liang and Kenji Sagae. 2010. Dynamic programming for linear-time incremental parsing. In *Proceedings of ACL*, pages 1077–1086, Uppsala.

Johansson, Richard and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *16th Nordic Conference of Computational Linguistics*, pages 105–112, Tartu.

Kay, Martin. 1996. Chart generation. In *Proceedings of ACL*, pages 200–204, Santa Cruz, CA.

Koehn, Phillip. 2010. *Statistical Machine Translation*. Cambridge University Press.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague.

Koehn, Philipp, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Edmonton, Canada.

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

Lee, J. and S. Seneff. 2006. Automatic grammar correction for second-language learners. In *Proceedings of Interspeech*, pages 1978–1981, Pittsburgh, PA.

Liu, Yang. 2013. A shift-reduce parsing algorithm for phrase-based string-to-dependency translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1–10, Sofia.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Papineni, Kishore, 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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Ratnaparkhi, Adwait. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of EMNLP*, pages 133–142, Somerset, NJ.

Reiter, Ehud and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Shen, Libin and Aravind Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu. HI.

Shen, Libin, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL*, pages 760–767, Prague.

Song, Linfeng, Yue Zhang, Kai Song, and Qun Liu. 2014. Joint morphological generation and syntactic linearization. In *Proceedings of the Twenty-Eighth AAAI Conference*, Quebec.

Steedman, Mark. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Surdeanu, Mihai, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CONLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester.

Wan, Stephen, 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 ACL (EACL 2009)*, pages 852–860, Athens.

Weir, David. 1988. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. Ph.D. thesis, University of Pennsylvania.
White, Michael. 2004. Reining in CCG chart realization. In *Proceedings of INLG-04*, pages 182–191, Brockenhurst.
White, Michael. 2006. Efficient Realization of Coordinate Structures in Combinatory Categorial Grammar. *Research on Language & Computation*, 4(1):39–75.
White, Michael and Rajakrishnan Rajkumar. 2009. Perceptron reranking for CCG realization. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 410–419, Singapore.
Wu, Dekai. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.
Xu, Peng, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 191–198, Philadelphia, PA.
Zettlemoyer, Luke S. and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, pages 658–666, Edinburgh.
Zhang, Yue. 2013. Partial-tree linearization: Generalized word ordering for text synthesis. In *Proceedings of IJCAI*, pages 2232–2238, Beijing.
Zhang, Yue, 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, Avignon.
Zhang, Yue and Stephen Clark. 2008. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL/HLT*, pages 888–896, Columbus, OH.
Zhang, Yue and Stephen Clark. 2010. A fast decoder for joint word segmentation and POS-tagging using a single discriminative model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 843–852, Cambridge, MA.
Zhang, Yue and Stephen Clark. 2011. Syntactic processing using the generalized perceptron and beam search. *Computational Linguistics*, 37(1):105–151.
Zhang, Yue and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, OR.