Syntax-Based Word Ordering Incorporating a Large-Scale Language Model

Yue Zhang
University of Cambridge
Computer Laboratory
yz360@cam.ac.uk

Graeme Blackwood
University of Cambridge
Engineering Department
gwb24@eng.cam.ac.uk

Stephen Clark
University of Cambridge
Computer Laboratory
sc609@cam.ac.uk

Abstract

A fundamental problem in text generation is word ordering. Word ordering is a computationally difficult problem, which can be constrained to some extent for particular applications, for example by using synchronous grammars for statistical machine translation. There have been some recent attempts at the unconstrained problem of generating a sentence from a multi-set of input words (Wan et al., 2009; Zhang and Clark, 2011). By using CCG and learning guided search, Zhang and Clark reported the highest scores on this task. One limitation of their system is the absence of an N-gram language model, which has been used by text generation systems to improve fluency. We take the Zhang and Clark system as the baseline, and incorporate an N-gram model by applying online large-margin training. Our system significantly improved on the baseline by 3.7 BLEU points.

1 Introduction

One fundamental problem in text generation is word ordering, which can be abstractly formulated as finding a grammatical order for a multi-set of words. The word ordering problem can also include word choice, where only a subset of the input words are used to produce the output.

Word ordering is a 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 et al., 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 et al., 2010), parts of translation hypotheses identified as having high local confidence are held fixed, so that word ordering elsewhere is strictly local.

Some recent work attempts to address the fundamental word ordering task directly, using syntactic models and heuristic search. Wan et al. (2009) uses a dependency grammar to solve word ordering, and Zhang and Clark (2011) uses CCG (Steedman, 2000) for word ordering and word choice. The use of syntax models makes their search problems harder than word permutation using an N-gram language model only. Both methods apply heuristic search. Zhang and Clark developed a bottom-up best-first algorithm to build output syntax trees from input words, where search is guided by learning for both efficiency and accuracy. The framework is flexible in allowing a large range of constraints to be added for particular tasks.

We extend the work of Zhang and Clark (2011) (Z&C) in two ways. First, we apply online large-margin training to guide search. Compared to the perceptron algorithm on "constituent level features" by Z&C, our training algorithm is theoretically more elegant (see Section 3) and converges more smoothly empirically (see Section 5). Using online large-margin training not only improves the output quality, but also allows the incorporation of an N-gram language-model into

Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics, pages 736–746,
Avignon, France, April 23 - 27 2012.

the system. N-gram models have been used as a standard component in statistical machine translation, but have not been applied to the syntactic model of Z&C. Intuitively, an N-gram model can improve local fluency when added to a syntax model. Our experiments show that a four-gram model trained using the English GigaWord corpus gave improvements when added to the syntax-based baseline system.

The contributions of this paper are as follows. First, we improve on the performance of the Z&C system for the challenging task of the general word ordering problem. Second, we develop a novel method for incorporating a large-scale language model into a syntax-based generation system. Finally, we analyse large-margin training in the context of learning-guided best-first search, offering a novel solution to this computationally hard problem.

2 The statistical model and decoding algorithm

We take Z&C as our baseline system. Given a multi-set of input words, the baseline system builds a CCG derivation by choosing and ordering words from the input set. The scoring model is trained using CCGBank (Hockenmaier and Steedman, 2007), and best-first decoding is applied. We apply the same decoding framework in this paper, but apply an improved training process, and incorporate an N-gram language model into the syntax model. In this section, we describe and discuss the baseline statistical model and decoding framework, motivating our extensions.

2.1 Combinatory Categorial Grammar

CCG, and parsing with CCG, has been described elsewhere (Clark and Curran, 2007; Hockenmaier and Steedman, 2002); here we provide only a short description.

CCG (Steedman, 2000) is a lexicalized grammar formalism, which associates each word in a sentence with a lexical category. There is a small number of basic lexical categories, such as noun (N), noun phrase (NP), and prepositional phrase (PP). Complex lexical categories are formed recursively from basic categories and slashes, which indicate the directions of arguments. The CCG grammar used by our system is read off the derivations in CCGbank, following Hockenmaier and Steedman (2002), meaning that the CCG combinatory rules are encoded as rule instances, together with a number of additional rules which deal with punctuation and type-changing. Given a sentence, its CCG derivation can be produced by first assigning a lexical category to each word, and then recursively applying CCG rules bottom-up.

2.2 The decoding algorithm

In the decoding algorithm, a hypothesis is an *edge*, which corresponds to a sub-tree in a CCG derivation. Edges are built bottom-up, starting from leaf edges, which are generated by assigning all possible lexical categories to each input word. 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 which 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 combinatory rule, and covers the concatenated surface strings of the two sub-edges in their order or combination. New edges can also be generated by applying unary rules to a single existing edge. Starting from the leaf edges, the bottom-up process is repeated until a goal edge is found, and its surface string is taken as the output.

This derivation-building process is reminiscent of a bottom-up CCG parser in the edge combination mechanism. However, it is fundamentally different from a bottom-up parser. Since, for the generation problem, the order of two edges in their combination is flexible, the search problem is much harder than that of a parser. With no input order specified, no efficient dynamic-programming algorithm is available, and less contextual information is available for disambiguation due to the lack of an input string.

In order to combat the large search space, best-first search is applied, where candidate hypotheses are ordered by their scores, and kept in an agenda, and a limited number of accepted hypotheses are recorded in a chart. Here the chart is essentially a set of beams, each of which contains the highest scored edges covering a particular number of words. Initially, all leaf edges are generated and scored, before they are put onto the agenda. During each step in the decoding process, the top edge from the agenda is expanded. If it is a goal edge, it is returned as the output, and the

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
    for e′ ∈ UNARY(e, grammar) do
        APPEND(new, e)
    end for
    for ẽ ∈ c do
        if CANCOMBINE(e, ẽ) then
            e′ ← BINARY(e, ẽ, grammar)
            APPEND(new, e′)
        end if
        if CANCOMBINE(ẽ, e) 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
```

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 onto the agenda, while the original edge is put onto the chart. The process repeats until a goal edge is found, or a timeout limit is reached. In the latter case, a default output is produced using existing edges in the chart.

Pseudocode for the decoder is shown as Algorithm 1. Again it is reminiscent of a best-first parser (Caraballo and Charniak, 1998) in the use of an agenda and a chart, but is fundamentally different due to the fact that there is no input order.

2.3 Statistical model and feature templates

The baseline system uses a linear model to score hypotheses. For an edge e, its score is defined as:

$$f(e) = \Phi(e) \cdot \theta,$$

where $\Phi(e)$ represents the feature vector of e and θ is the parameter vector of the model.

During decoding, feature vectors are computed incrementally. When an edge is constructed, its score is computed from the scores of its sub-edges and the incrementally added structure:

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

In the 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. The feature vector $\phi(e)$ represents the incremental structure when e is constructed over its sub-edges. It is called the "constituent-level feature vector" by Z&C. 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. The score $f(e)$ is therefore the sum of $f(e_s)$ (for all $e_s \in e$) plus $\phi(e) \cdot \theta$. The feature templates we use are the same as those in the baseline system.

An important aspect of the scoring model is that edges with different sizes are compared with each other during decoding. Edges with different sizes can have different numbers of features, which can make the training of a discriminative model more difficult. For example, a leaf edge with one word can be compared with an edge over the entire input. One way of reducing the effect of the size difference is to include the size of the edge as part of feature definitions, which can improve the comparability of edges of different sizes by reducing the number of features they have in common. Such features are applied by Z&C, and we make use of them here. Even with such features, the question of whether edges with different sizes are linearly separable is an empirical one.

3 Training

The efficiency of the decoding algorithm is dependent on the statistical model, since the best-

first search is *guided* to a solution by the model, and a good model will lead to a solution being found more quickly. In the ideal situation for the best-first decoding algorithm, the model is perfect and the score of any gold-standard edge is higher than the score of any non-gold-standard edge. As a result, the top edge on the agenda is always a gold-standard edge, and therefore all edges on the chart are gold-standard before the gold-standard goal edge is found. In this oracle procedure, the minimum number of edges is expanded, and the output is correct. The best-first decoder is perfect in not only accuracy, but also speed. In practice this ideal situation is rarely met, but it determines the goal of the training algorithm: to produce the perfect model and hence decoder.

If we take gold-standard edges as positive examples, and non-gold-standard edges as negative examples, the goal of the training problem can be viewed as finding a large separating margin between the scores of positive and negative examples. However, it is infeasible to generate the full space of negative examples, which is factorial in the size of input. Like Z&C, we apply online learning, and generate negative examples based on the decoding algorithm.

Our training algorithm is shown as Algorithm 2. 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 chart. Similar to the decoding algorithm, the agenda is intialized using all possible leaf edges. During each step, the top of the agenda e is popped. If it is a gold-standard edge, it is expanded in exactly the same way as the decoder, with the newly generated edges being put onto 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 udpate to the model parameter vector θ. Our parameter update algorithm is different from the baseline perceptron algorithm, as will be discussed later. After updating the parameters, the scores of 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. By not putting any non-gold-standard edges onto the chart, the training speed is much faster; on the other hand a wide range of negative examples is pruned. We leave for further work possible alternative methods to generate more negative examples during training.

Algorithm 2 The training algorithm.

```
a ← INITAGENDA( )
c ← INITCHART( )
while not TIMEOUT( ) do
    new ← []
    e ← POPBEST(a)
    if GOLDSTANDARD(e) and GOALTEST(e)
then return e
    end if
    if not GOLDSTANDARD(e) then
        e_ ← e
        e+ ← MINGOLD(a)
        UPDATEPARAMETERS(e+, e_)
        RECOMPUTESCORES(a, c)
        continue
    end if
    for e' ∈ UNARY(e, grammar) do
        APPEND(new, e)
    end for
    for ẽ ∈ c do
        if CANCOMBINE(e, ẽ) then
            e' ← BINARY(e, ẽ, grammar)
            APPEND(new, e')
        end if
        if CANCOMBINE(ẽ, e) 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
```

Another 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-standard edges. This is the view described by Z&C. Given a positive example e_+ and a negative example e_-, they use the perceptron algorithm to penalize the score for $\phi(e_-)$ and reward the score of $\phi(e_+)$, but do not update parameters for the sub-edges of e_+ and e_-. An argument for not penalizing the sub-edge scores for e_- is that the sub-edges must be gold-standard edges (since the training process is constructed so that only gold-standard edges are expanded). From

the perspective of correctness, it is unnecessary to find a margin between the sub-edges of e_+ and those of e_-, since both are gold-standard edges.

However, since the score of an edge not only represents its correctness, but also affects its priority on the agenda, promoting the sub-edge of e_+ can lead to "easier" edges being constructed before "harder" ones (i.e. those that are less likely to be correct), and therefore improve the output accuracy. This perspective has been observed by other works of learning-guided-search (Shen et al., 2007; Shen and Joshi, 2008; Goldberg and Elhadad, 2010). Intuitively, the score difference between easy gold-standard and harder gold-standard edges should not be as great as the difference between gold-standard and non-gold-standard edges. The perceptron update cannot provide such control of separation, because the amount of update is fixed to 1.

As described earlier, we treat parameter update as finding a separation between correct and incorrect edges, in which the global feature vectors Φ, rather than ϕ, are considered. Given a positive example e_+ and a negative example e_-, we make a minimum update so that the score of e_+ is higher than that of e_- with some margin:

$$\theta \leftarrow \arg\min_{\theta'} \parallel \theta' - \theta_0 \parallel, \text{s.t.} \Phi(e_+)\theta' - \Phi(e_-)\theta' \geq 1$$

where θ_0 and θ denote the parameter vectors before and after the udpate, respectively. The update is similar to the update of online large-margin learning algorithms such as 1-best MIRA (Crammer et al., 2006), and has a closed-form solution:

$$\theta \leftarrow \theta_0 + \frac{f(e_-) - f(e_+) + 1}{\parallel \Phi(e_+) - \Phi(e_-) \parallel^2}\big(\Phi(e_+) - \Phi(e_-)\big)$$

In this update, the global feature vectors $\Phi(e_+)$ and $\Phi(e_-)$ are used. Unlike Z&C, the scores of sub-edges of e_+ and e_- are also udpated, so that the sub-edges of e_- are less prioritized than those of e_+. We show empirically that this training algorithm significantly outperforms the perceptron training of the baseline system in Section 5. An advantage of our new training algorithm is that it enables the accommodation of a separately trained N-gram model into the system.

4 Incorporating an N-gram language model

Since the seminal work of the IBM models (Brown et al., 1993), N-gram language models have been used as a standard component in statistical machine translation systems to control output fluency. For the syntax-based generation system, the incorporation of an N-gram language model can potentially improve the local fluency of output sequences. In addition, the N-gram language model can be trained separately using a large amount of data, while the syntax-based model requires manual annotation for training.

The standard method for the combination of a syntax model and an N-gram model is linear interpolation. We incorporate fourgram, trigram and bigram scores into our syntax model, so that the score of an edge e becomes:

$$\begin{aligned} F(e) &= f(e) + g(e) \\ &= f(e) + \alpha \cdot g_{four}(e) + \beta \cdot g_{tri}(e) + \gamma \cdot g_{bi}(e), \end{aligned}$$

where f is the syntax model score, and g is the N-gram model score. g consists of three components, g_{four}, g_{tri} and g_{bi}, representing the log-probabilities of fourgrams, trigrams and bigrams from the language model, respectively. α, β and γ are the corresponding weights.

During decoding, $F(e)$ is computed incrementally. Again, denoting the sub-edges of e as e_s,

$$\begin{aligned} F(e) &= f(e) + g(e) \\ &= \big(\sum_{e_s \in e} F(e_s)\big) + \phi(e)\theta + g_\delta(e) \end{aligned}$$

Here $g_\delta(e) = \alpha \cdot g_{\delta four}(e) + \beta \cdot g_{\delta tri}(e) + \gamma \cdot g_{\delta bi}(e)$ is the sum of log-probabilities of the new N-grams resulting from the construction of e. For leaf edges and unary-branching edges, no new N-grams result from their construction (i.e. $g_\delta = 0$). For a binary-branching edge, new N-grams result from the surface-string concatenation of its sub-edges. The sum of log-probabilities of the new fourgrams, trigrams and bigrams contribute to g_δ with weights α, β and γ, respectively.

For training, there are at least three methods to tune α, β, γ and θ. One simple method is to train the syntax model θ independently, and select α, β, and γ empirically from a range of candidate values according to development tests. We call this method test-time interpolation. An alternative is to select α, β and γ first, initializing the vector θ as all zeroes, and then run the training algorithm for θ taking into account the N-gram language model. In this process, g is considered when finding a separation between positive and

negative examples; the training algorithm finds a value of θ that best suits the precomputed α, β and γ values, together with the N-gram language model. We call this method g-precomputed interpolation. Yet another method is to initialize α, β, γ and θ as all zeroes, and run the training algorithm taking into account the N-gram language model. We call this method g-free interpolation.

The incorporation of an N-gram language model into the syntax-based generation system is weakly analogous to N-gram model insertion for syntax-based statistical machine translation systems, both of which apply a score from the N-gram model component in a derivation-building process. As discussed earlier, polynomial-time decoding is typically feasible for syntax-based machine translation systems without an N-gram language model, due to constraints from the grammar. In these cases, incorporation of N-gram language models can significantly increase the complexity of a dynamic-programming decoder (Bar-Hillel et al., 1961). Efficient search has been achieved using chart pruning (Chiang, 2007) and iterative numerical approaches to constrained optimization (Rush and Collins, 2011). In contrast, the incorporation of an N-gram language model into our decoder is more straightforward, and does not add to its asymptotic complexity, due to the heuristic nature of the decoder.

5 Experiments

We use sections 2–21 of CCGBank to train our syntax model, section 00 for development and section 23 for the final test. Derivations from CCGBank are transformed into inputs by turning their surface strings into multi-sets of words. Following Z&C, we treat base noun phrases (i.e. NPs that do not recursively contain other NPs) as atomic units for the input. Output sequences are compared with the original sentences to evaluate their quality. We follow previous work and use the BLEU metric (Papineni et al., 2002) to compare outputs with references.

Z&C use two methods to construct leaf edges. The first is to assign lexical categories according to a dictionary. There are 26.8 lexical categories for each word on average using this method, corresponding to 26.8 leaf edges. The other method is to use a pre-processing step — a CCG supertagger (Clark and Curran, 2007) — to prune candidate lexical categories according to the gold-standard sequence, assuming that for some problems the ambiguities can be reduced (e.g. when the input is already partly correctly ordered). Z&C use different probability cutoff levels (the β parameter in the supertagger) to control the pruning. Here we focus mainly on the dictionary method, which leaves lexical category disambiguation entirely to the generation system. For comparison, we also perform experiments with lexical category pruning. We chose $\beta = 0.0001$, which leaves 5.4 leaf edges per word on average.

CCGBank	Sentences	Tokens
training	39,604	929,552
development	1,913	45,422
GigaWord v4	**Sentences**	**Tokens**
AFP	30,363,052	684,910,697
XIN	15,982,098	340,666,976

Table 1: Number of sentences and tokens by language model source.

We used the SRILM Toolkit (Stolcke, 2002) to build a true-case 4-gram language model estimated over the CCGBank training and development data and a large additional collection of fluent sentences in the Agence France-Presse (AFP) and Xinhua News Agency (XIN) subsets of the English GigaWord Fourth Edition (Parker et al., 2009), a total of over 1 billion tokens. The GigaWord data was first pre-processed to replicate the CCGBank tokenization. The total number of sentences and tokens in each LM component is shown in Table 1. The language model vocabulary consists of the 46,574 words that occur in the concatenation of the CCGBank training, development, and test sets. The LM probabilities are estimated using modified Kneser-Ney smoothing (Kneser and Ney, 1995) with interpolation of lower n-gram orders.

5.1 Development experiments

A set of development test results without lexical category pruning (i.e. using the full dictionary) is shown in Table 2. We train the baseline system and our systems under various settings for 10 iterations, and measure the output BLEU scores after each iteration. The timeout value for each sentence is set to 5 seconds. The highest score (max BLEU) and averaged score (avg. BLEU) of each system over the 10 training iterations are shown in the table.

Method	max BLEU	avg. BLEU
baseline	38.47	37.36
margin	41.20	39.70
margin +LM (g-precomputed)	41.50	40.84
margin +LM ($\alpha = 0,\ \beta = 0,\ \gamma = 0$)	40.83	—
margin +LM ($\alpha = 0.08,\ \beta = 0.016,\ \gamma = 0.004$)	38.99	—
margin +LM ($\alpha = 0.4,\ \beta = 0.08,\ \gamma = 0.02$)	36.17	—
margin +LM ($\alpha = 0.8,\ \beta = 0.16,\ \gamma = 0.04$)	34.74	—

Table 2: Development experiments without lexical category pruning.

The first three rows represent the baseline system, our largin-margin training system (margin), and our system with the N-gram model incorporated using g-precomputed interpolation. For interpolation we manually chose $\alpha = 0.8$, $\beta = 0.16$ and $\gamma = 0.04$, respectively. These values could be optimized by development experiments with alternative configurations, which may lead to further improvements. Our system with large-margin training gives higher BLEU scores than the baseline system consistently over all iterations. The N-gram model led to further improvements.

The last four rows in the table show results of our system with the N-gram model added using test-time interpolation. The syntax model is trained with the optimal number of iterations, and different α, β, and γ values are used to integrate the language model. Compared with the system using no N-gram model (margin), test-time interpolation did not improve the accuracies.

The row with $\alpha, \beta, \gamma = 0$ represents our system with the N-gram model loaded, and the scores g_{four}, g_{tri} and g_{bi} computed for each N-gram during decoding, but the scores of edges are computed without using N-gram probabilities. The scoring model is the same as the syntax model (margin), but the results are lower than the row "margin", because computing N-gram probabilities made the system slower, exploring less hypotheses under the same timeout setting.[1]

The comparison between g-precomputed interpolation and test-time interpolation shows that the system gives better scores when the syntax model takes into consideration the N-gram model during training. One question that arises is whether g-free interpolation will outperform g-precomputed interpolation. g-free interpolation offers the freedom of α, β and γ during training, and can potentially reach a better combination of the parameter values. However, the training algorithm failed to converge with g-free interpolation. One possible explanation is that real-valued features from the language model made our large-margin training harder. Another possible reason is that our training process with heavy pruning does not accommodate this complex model.

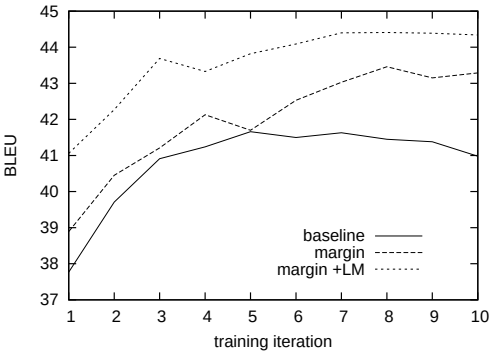

Figure 1: Development experiments with lexical category pruning ($\beta = 0.0001$).

Figure 1 shows a set of development experiments with lexical category pruning (with the supertagger parameter $\beta = 0.0001$). The scores of the three different systems are calculated by varying the number of training iterations. The large-margin training system (margin) gave consistently better scores than the baseline system, and adding a language model (margin +LM) improves the scores further.

Table 3 shows some manually chosen examples for which our system gave significant improvements over the baseline. For most other sentences the improvements are not as obvious. For each

[1]More decoding time could be given to the slower N-gram system, but we use 5 seconds as the timeout setting for all the experiments, giving the methods with the N-gram language model a slight disadvantage, as shown by the two rows "margin" and "margin +LM ($\alpha, \beta, \gamma = 0$).

baseline	margin	margin +LM
as a nonexecutive director Pierre Vinken , 61 years old , will join the board . 29 Nov.	61 years old , the board will join as a nonexecutive director Nov. 29 , Pierre Vinken .	as a nonexecutive director Pierre Vinken , 61 years old , will join the board Nov. 29 .
Lorillard nor smokers were aware of the Kent cigarettes of any research on the workers who studied the researchers	of any research who studied Neither the workers were aware of smokers on the Kent cigarettes nor the researchers	Neither Lorillard nor any research on the workers who studied the Kent cigarettes were aware of smokers of the researchers .
you But 35 years ago have to recognize that these events took place .	recognize But you took place that these events have to 35 years ago .	But you have to recognize that these events took place 35 years ago .
investors to pour cash into money funds continue in Despite yields recent declines	Despite investors , yields continue to pour into money funds recent declines in cash .	Despite investors , recent declines in yields continue to pour cash into money funds .
yielding The top money funds are currently well over 9 % .	The top money funds currently are yielding well over 9 % .	The top money funds are yielding well over 9 % currently .
where A buffet breakfast , held in the museum was food and drinks to . everyday visitors banned	everyday visitors are banned to where A buffet breakfast was held , food and drinks in the museum .	A buffet breakfast , everyday visitors are banned to where food and drinks was held in the museum .
A Commonwealth Edison spokesman said an administrative nightmare would be tracking down the past 3 12 years that the two million customers have . whose changed	tracking A Commonwealth Edison spokesman said that the two million customers whose addresses have changed down during the past 3 12 years would be an administrative nightmare .	an administrative nightmare whose addresses would be tracking down A Commonwealth Edison spokesman said that the two million customers have changed during the past 3 12 years .
The $ 2.5 billion Byron 1 plant , Ill. , was completed . near Rockford in 1985	The $ 2.5 billion Byron 1 plant was near completed in Rockford , Ill. , 1985 .	The $ 2.5 billion Byron 1 plant near Rockford , Ill. , was completed in 1985 .
will (During its centennial year , The Wall Street Journal report events of the past century that stand as milestones of American business history .)	as The Wall Street Journal (During its centennial year , milestones stand of American business history that will report events of the past century .)	During its centennial year events will report , The Wall Street Journal that stand as milestones of American business history (of the past century) .

Table 3: Some chosen examples with significant improvements (supertagger parameter $\beta = 0.0001$).

method, the examples are chosen from the development output with lexical category pruning, after the optimal number of training iterations, with the timeout set to 5s. We also tried manually selecting examples without lexical category pruning, but the improvements were not as obvious, partly because the overall fluency was lower for all the three systems.

Table 4 shows a set of examples chosen randomly from the development test outputs of our system with the N-gram model. The optimal number of training iterations is used, and a timeout of 1 minute is used in addition to the 5s timeout for comparison. With more time to decode each input, the system gave a BLEU score of 44.61, higher than 41.50 with the 5s timout.

While some of the outputs we examined are reasonably fluent, most are to some extent fragmentary.[2] In general, the system outputs are still far below human fluency. Some samples are syntactically grammatical, but are semantically anomalous. For example, person names are often confused with company names, verbs often take unrelated subjects and objects. The problem is much more severe for long sentences, which have more ambiguities. For specific tasks, extra information (such as the source text for machine translation) can be available to reduce ambiguities.

6 Final results

The final results of our system without lexical category pruning are shown in Table 5. Row "W09 CLE" and "W09 AB" show the results of the maximum spanning tree and assignment-based algorithms of Wan et al. (2009); rows "margin" and "margin +LM" show the results of our large-margin training system and our system with the N-gram model. All these results are directly comparable since we do not use any lexical category pruning for this set of results. For each of our systems, we fix the number of training iterations according to development test scores. Consistent with the development experiments, our sys-

[2]Part of the reason for some fragmentary outputs is the default output mechanism: partial derivations from the chart are greedily put together when timeout occurs before a goal hypothesis is found.

timeout = 5s	timeout = 1m
drooled the cars and drivers , like Fortune 500 executives . over the race	After schoolboys drooled over the cars and drivers , the race like Fortune 500 executives .
One big reason : thin margins .	One big reason : thin margins .
You or accountants look around ... and at an eye blinks . professional ballplayers	blinks nobody You or accountants look around ... and at an eye . professional ballplayers
most disturbing And of it , are educators , not students , for the wrongdoing is who .	And blamed for the wrongdoing , educators , not students who are disturbing , much of it is most .
defeat coaching aids the purpose of which is , He and other critics say can to . standardized tests learning progress	gauge coaching aids learning progress can and other critics say the purpose of which is to defeat , standardized tests .
The federal government of government debt because Congress has lifted the ceiling on U.S. savings bonds suspended sales	The federal government suspended sales of government debt because Congress has n't lifted the ceiling on U.S. savings bonds .

Table 4: Some examples chosen at random from development test outputs without lexical category pruning.

System	BLEU
W09 CLE	26.8
W09 AB	33.7
Z&C11	40.1
margin	42.5
margin +LM	43.8

Table 5: Test results without lexical category pruning.

System	BLEU
Z&C11	43.2
margin	44.7
margin +LM	46.1

Table 6: Test results with lexical category pruning (supertagger parameter $\beta = 0.0001$).

tem outperforms the baseline methods. The accuracies are significantly higher when the N-gram model is incorporated.

Table 6 compares our system with Z&C using lexical category pruning ($\beta = 0.0001$) and a 5s timeout for fair comparison. The results are similar to Table 5: our large-margin training systems outperforms the baseline by 1.5 BLEU points, and adding the N-gram model gave a further 1.4 point improvement. The scores could be significantly increased by using a larger timeout, as shown in our earlier development experiments.

7 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; Filippova and Strube, 2009; Bohnet et al., 2010; Guo et al., 2011). Unlike our system, and Wan et al. (2009), input dependencies provide additional information to these systems. Although the search space can be constrained by the assumption of projectivity, permutation of modifiers of the same head word makes exact inference for tree linearization intractable. The above systems typically apply approximate inference, such as beam-search. While syntax-based features are commonly used by these systems for linearization, Filippova and Strube (2009) apply a trigram model to control local fluency within constituents. A dependency-based N-gram model has also been shown effective for the linearization task (Guo et al., 2011).

The best-first inference and timeout mechanism of our system is similar to that of White (2004), a surface realizer from logical forms using CCG.

8 Conclusion

We studied the problem of word-ordering using a syntactic model and allowing permutation. We took the model of Zhang and Clark (2011) as the baseline, and extended it with online large-margin training and an N-gram language model. These extentions led to improvements in the BLEU evaluation. Analyzing the generated sentences suggests that, while highly fluent outputs can be produced for short sentences ($\leq$ 10 words), the system fluency in general is still way below human standard. Future work remains to apply the system as a component for specific text generation tasks, for example machine translation.

Acknowledgements

Yue Zhang and Stephen Clark are supported by the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement no. 247762.

References

Yehoshua Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

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

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010. Fluency constraints for minimum Bayes-risk decoding of statistical machine translation lattices. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 71–79, Beijing, China, August. Coling 2010 Organizing Committee.

Bernd Bohnet, 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, China, August. Coling 2010 Organizing Committee.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Sharon A. Caraballo and Eugene Charniak. 1998. New figures of merit for best-first probabilistic chart parsing. *Comput. Linguist.*, 24:275–298, June.

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

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Katja Filippova 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, Czech Republic, June. Association for Computational Linguistics.

Katja Filippova 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, Colorado, June. Association for Computational Linguistics.

Yoav Goldberg 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, California, June. Association for Computational Linguistics.

Yuqing Guo, 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, France, September. Association for Computational Linguistics.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

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

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *International Conference on Acoustics, Speech, and Signal Processing, 1995. ICASSP-95*, volume 1, pages 181–184.

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

Philipp Koehn, 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, Czech Republic, June. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2009. English Gigaword Fourth Edition, Linguistic Data Consortium.

Alexander M. Rush and Michael Collins. 2011. Exact decoding of syntactic translation models through la-

grangian relaxation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 72–82, Portland, Oregon, USA, June. Association for Computational Linguistics.

Libin Shen 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, Hawaii, October. Association for Computational Linguistics.

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

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

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the International Conference on Spoken Language Processing*, pages 901–904.

Stephen Wan, 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, Greece, March. Association for Computational Linguistics.

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

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

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