Word Ordering with Phrase-Based Grammars

Adrià de Gispert
Cambridge University
Engineering Department
University of Cambridge
ad465@cam.ac.uk

Marcus Tomalin
Cambridge University
Engineering Department
University of Cambridge
mt126@cam.ac.uk

William Byrne
Cambridge University
Engineering Department
University of Cambridge
wjb31@cam.ac.uk

Abstract

We describe an approach to word ordering using modelling techniques from statistical machine translation. The system incorporates a phrase-based model of string generation that aims to take unordered bags of words and produce fluent, grammatical sentences. We describe the generation grammars and introduce parsing procedures that address the computational complexity of generation under permutation of phrases. Against the best previous results reported on this task, obtained using syntax driven models, we report huge quality improvements, with BLEU score gains of 20+ which we confirm with human fluency judgements. Our system incorporates dependency language models, large n-gram language models, and minimum Bayes risk decoding.

1 Introduction

Word ordering is a fundamental problem in NLP and has been shown to be NP-complete in discourse ordering (Althaus et al., 2004) and in SMT with arbitrary word reordering (Knight, 1999). Typical solutions involve constraints on the space of permutations, as in multi-document summarisation (Barzilay and Elhadad, 2011) and preordering in SMT (Tromble and Eisner, 2009; Genzel, 2010).

Some recent work attempts to address the fundamental word ordering task directly, using syntactic models and heuristic search. Wan et al. (2009) use a dependency grammar to address word ordering, while Zhang and Clark (2011; 2012) use CCG and large-scale n-gram language models. These techniques are applied to the unconstrained problem of generating a sentence from a multi-set of input words.

We describe GYRO (**G**et **Y**our **O**rder **R**ight), a phrase-based approach to word ordering. Given a bag of words, the system first scans a large, trusted text collection and extracts phrases consisting of words from the bag. Strings are then generated by concatenating these phrases in any order, subject to the constraint that every string is a valid reordering of the words in the bag, and the results are scored under an n-gram language model (LM). The motivation is that it is an easier modelling problem to make fluent sentences from phrases (snippets of fluent text) than from words in isolation.

GYRO builds on approaches developed for syntactic SMT (Chiang, 2007; de Gispert et al., 2010; Iglesias et al., 2011). The system generates strings in the form of weighted automata which can be rescored using higher-order n-gram LMs, dependency LMs (Shen et al., 2010), and Minimum Bayes Risk decoding, either using posterior probabilities obtained from GYRO or SMT systems.

We report extensive experiments using BLEU and conclude with human assessments. We show that despite its relatively simple formulation, GYRO gives BLEU scores over 20 points higher than the best previously reported results, generated by a syntax-based ordering system. Human fluency assessments confirm these substantial improvements.

EACL 2014, to appear

2 Phrase-based Word Ordering

We take as input a bag of N words $\Omega = \{w_1, \ldots, w_N\}$. The words are sorted, e.g. alphabetically, so that it is possible to refer to the i^{th} word in the bag, and repeated words are distinct tokens. We also take a set of phrases, $\mathcal{L}(\Omega)$ that are extracted from large text collections, and contain only words from Ω. We refer to phrases as u, i.e. $u \in \mathcal{L}(\Omega)$. The goal is to generate all permutations of Ω that can be formed by concatenation of phrases from $\mathcal{L}(\Omega)$.

2.1 Word Order Generation Grammar

Consider a subset $A \subset \Omega$. We can represent A by an N-bit binary string $I(A) = I_1(A) \ldots I_N(A)$, where $I_i(A) = 1$ if $w_i \in A$, and $I_i(A) = 0$ otherwise. A Context-Free Grammar (CFG) for generation can then be defined by the following rules:

Phrase-based Rules: $\forall A \subset \Omega$ and $\forall u \in \mathcal{L}(A)$

$$I(A) \rightarrow u$$

Concatenation Rules: $\forall A \subset \Omega, B \subset A, C \subset A$ such that $I(A) = I(B) + I(C)$ and $I(B) \bullet I(C) = 0$

$$I(A) \rightarrow I(B)\, I(C)$$

where $\bullet$ is the bit-wise logical AND

Root: $S \rightarrow I(\Omega)$

We use this grammar to 'parse' the list of the words in the bag Ω. The grammar has one nonterminal per possible binary string, so potentially 2^N distinct nonterminals might be needed to generate the language. Each nonterminal can produce either a phrase $u \in \mathcal{L}(A)$, or the concatenation of two binary strings that share no bits in common. A derivation is sequence of rules that starts from the bit string $I(\Omega)$. Rules are unweighted in this basic formulation.

For example, assume the following bag $\Omega = \{a, b, c, d, e\}$, which we sort alphabetically. Assume the phrases are $\mathcal{L}(\Omega) = \{$"$a\,b$", "$b\,a$", "$d\,e\,c$"$\}$. The generation grammar contains the following 6 rules:

R^1: 11000$\rightarrow ab$
R^2: 11000$\rightarrow ba$
R^3: 00111$\rightarrow dec$
R^4: 11111$\rightarrow$ 11000 00111
R^5: 11111$\rightarrow$ 00111 11000
R^6: $S\rightarrow$ 11111

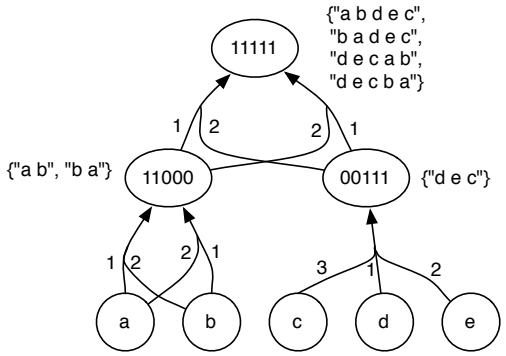

Figure 1: Hypergraph representing generation from $\{a, b, c, d, e\}$ with phrases $\{$"$a\,b$", "$b\,a$", "$d\,e\,c$"$\}$.

Figure 1 represents all the possible derivations in a hypergraph, which generate four alternative strings. For example, string "$d\,e\,c\,b\,a$" is obtained with derivation R^{6}R^{5}R^{3}R^2, whereas string "$a\,b\,d\,e\,c$" is obtained via R^{6}R^{4}R^{1}R^3.

2.2 Parsing a Bag of Words

We now describe a general algorithm for parsing a bag of words with phrase constraints. The search is organized along a two-dimensional grid $M[x, y]$ of 2^N–1 cells, where each cell is associated with a unique nonterminal in the grammar (a bit string I with at least one bit set to 1). Each row x in the grid has $\binom{N}{x}$ cells, representing all the possible ways of covering exactly x words from the bag. There are N rows in total.

For a bit string I, $X(I)$ is the length of I, i.e. the number of 1's in I. In this way $X(I(A))$ points to the row associated with set A. There is no natural ordering of cells within a row, so we introduce a second function $Y(I)$ which indicates which cell in row $X(I)$ is associated with I. Hence $M[X(I), Y(I)]$ is the cell associated with bit string I. In the inverse direction, we using the notation $I_{x,y}$ to indicate a bit string associated with the cell $M[x, y]$.

The basic parsing algorithm is given in Figure 2. We first initialize the grid by filling the cells linked to phrase-based rules (lines 1-4 of Figure 2). Then parsing proceeds as follows. For each row in increasing order (line 5), and for each of the non-empty cells in the row (line 6), try to combine its bit string with any other bit strings (lines 7-8). If combination is admitted, then form the resultant bit string and add the con-

PARSE-BAG-OF-WORDS

Input: bag of words Ω of size N
Input: list of phrases $\mathcal{L}(\Omega)$
Initialize - Add phrase-based rules:
1 $M[x, y] \leftarrow \emptyset$
2 **for each** subset $A \in \Omega$
3 **for each** phrase $u \in \mathcal{L}(A)$
4 **add** rule $I(A) \rightarrow u$ to cell $M[X(I(A)), Y(I(A))]$
Parse:
5 **for each** row $x = 1, \ldots, N$
6 **for each** $y = 1, \ldots, \binom{N}{x}$
7 **for each** valid $A \in \Omega$
8 **if** $I_{x,y} \bullet I(A) = 0$, **then**
9 $I' \leftarrow I_{x,y} + I(A)$
10 **add** rule $I' \rightarrow I_{x,y}\, I(A)$ to cell $M[X(I'), Y(I')]$
11 **if** $|M[N, 1]| > 0$, **success**.

Figure 2: Parsing algorithm for a bag of words.

catenation rule to the associated cell in the grid (lines 9-10). The combination will always yield a bit string that resides in a higher row of the grid, so search is exhaustive. If a rule is found in cell $M[N, 1]$, there is a parse (line 11); otherwise none exists. The complexity of the algorithm is $O(2^N \cdot K)$. If back-pointers are kept, traversing these from cell $M[N, 1]$ yields all the generated word sequences.

The number of cells will grow exponentially as the bag grows in size. In practice, the number of cells actually used in parsing can be smaller than $2^N - 1$. This depends strongly on the number of distinct phrase-based rules and the distinct subsets of Ω they cover. For example, if we consider 1-word subsets of Ω, then all cells are needed and GYRO attempts all word permutation. However, if only 10 distinct 5-word phrases and 20 distinct 4-word phrases are considered for a bag of N=9 words, then fewer than 431 cells will be used (20 + 10 for the initial cells at rows 4 and 5; plus all combinations of 4-word subsets into row 8, which is less than 400; plus 1 for the last cell at row 9).

2.3 Generation from Exact Parsing

We are interested in producing the space of word sequences generated by the grammar, and in scoring each of the sequences according to a word-based n-gram LM. Assuming that parsing the bag of words suceeded, this is a very similar scenario to that of syntax-based approaches to SMT: the output is a large collection of word sequences, which are built by putting together smaller units and which can be found by a process of *expansion*, i.e. by traversing the back-pointers from an initial cell in a grid structure. A significant difference is that in syntax-based approaches the parsing stage tends to be computationally easier than the parsing stage has only a quadratic dependency on the length of the input sentence.

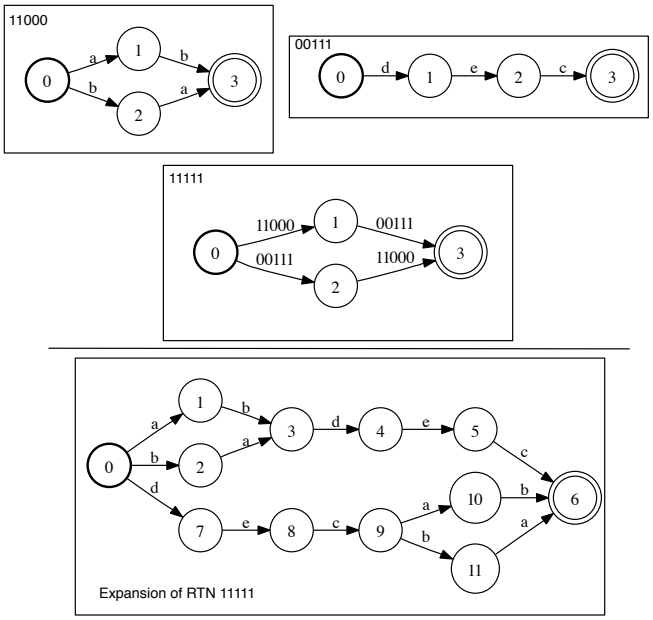

Figure 3: RTN representing generation from $\{a, b, c, d, e\}$ with phrases $\{$"$a\,b$", "$b\,a$", "$d\,e\,c$"$\}$ (top) and its expansion as an FSA (bottom).

We borrow techniques from SMT to represent and manipulate the space of generation hypotheses. Here we follow the approach of expanding this space onto a Finite-State Automata (FSA) described in (de Gispert et al., 2010; Iglesias et al., 2011). This means that in parsing, each cell $M[x, y]$ is associated with an FSA $F_{x,y}$, which encodes all the sequences generated by the grammar when covering the words marked by the bit string of that cell. When a rule is added to a cell, a new path from the initial to the final state of $F_{x,y}$ is created so that each FSA is the union of all paths arising from the rules added to the cell. Importantly, when an instance of the concatenation rule is added to a cell, the new path is built with only two arcs. These point to other FSAs at lower rows in the grid so that the result has the form of a Recursive Transition Network with a finite depth of recursion. Following the example from Section 2.1, the top three FSAs in Figure 3 represent the RTN for example from Figure 1.

The parsing algorithm is modified as follows:

```
 4   add rule I(A) → u
         as path to FSA F_{X(I(A)),Y(I(A))}
     ...
10   add rule I' → I_{x,y} I(A)
         as path to FSA F_{X(I'),Y(I')}
11   if NumStates(F_{N,1}) > 1, success.
```

At this point we specify two strategies:

Algorithm 1: Full expansion is described by the pseudocode in Figure 4, excluding lines 2-3. A recursive FSA replacement operation (Allauzen et al., 2007) can be used to expand the FSA in the top-most cell. In our running example, the result is the FSA at the bottom of Figure 3. We then apply a word-based LM to the resulting FSA via standard FSA composition. This outputs the complete (unpruned) language of interest, where each word sequence generated from the bag according to the phrasal constraints is scored by the LM.

Algorithm 2: Pruned expansion is described by the pseudocode in Figure 4, now including lines 2-3. We introduce pruning because full, unpruned expansion may not be feasible for large bags with many phrasal rules. Once parsing is done, we introduce the following bottom-up pruning strategy. For each row starting at row r, we union all FSAs of the row and expand the unioned FSA through the recursive replacement operation. This yields the space of all generation hypotheses of length r. We then apply the language model to this lattice and reduce it under likelihood-based pruning at weight β. We then update each cell in the row with a new FSA obtained as the intersection of its original FSA and the pruned FSA.[1] This intersection may yield an empty FSA for a particular cell (meaning that all its hypotheses were pruned out of the row), but it will always leave at least one surviving FSA per row, guaranteeing that if parsing succeeds, the top-most cell will expand into a non-empty FSA. As we process higher rows, the replacement operation will yield smaller FSAs because some back-pointers will point to empty FSAs. In this way memory usage can be controlled through parameters r and β. Of course, when pruning in this way, the final output lattice L will not contain the complete space of hypotheses that could be generated by the grammar.

[1] This step can be performed much more efficiently with a single forward pass of the resultant lattice. This is possible because the replace operation can yield a transducer where the input symbols encode a pointer to the original FSA, so in traversing the arcs of the pruned lattice, we know which arcs will belong to which cell FSAs. However, for ease of explanation we avoid this detail.

FULL-PARSE-EXPANSION

```
     Input: bag of words Ω of size N
     Input: list phrases L(Ω)
     Input: word-based LM G
     Output: word lattice L of generated sequences
     Generate:
 1   PARSE-BAG-OF-WORDS(Ω)
 2   for each row x = r, ..., N − 1
 3     PRUNE-ROW(x)
 4   F ← FSA-REPLACE(F_{N,1})
 5   return L ← F ∘ G

 6   function PRUNE-ROW(x) :
 7         F ← ⋃_y F_{x,y}
 8         F ← FSA-REPLACE(F)
 9         F ← F ∘ G
10         F ← FSA-PRUNE(F, β)
11         for each cell y = 1, ..., (N choose x)
12           F_{x,y} ← F_{x,y} · F
13   return
```

Figure 4: Pseudocode for Algorithm 1 (excluding lines 2-3) and Algorithm 2 (including all lines).

2.4 Algorithm 3: Pruned Parsing and Generation

The two generation algorithms presented above rely on a completed initial parsing step. However, given that the complexity of the parsing stage is $O(2^N \cdot K)$, this may not be achievable in practice. Leaving aside time considerations, the memory required to store 2^N FSAs will grow exponentially in N, even if the FSAs contain only pointers to other FSAs. Therefore we also describe an algorithm to perform bottom-up pruning guided by the LM during parsing. The pseudocode is identical to that of Algorithm 1 except for the following changes: in parsing (Figure 2) we pass G as input and we call the row pruning function of Figure 4 after line 5 if $x \geq r$.

We note that there is a strong connection between GYRO and the IDL approach of Soricut and Marcu (2005; 2006). Our bag of words parser could be cast in the IDL-formalism, and the FSA 'Replace' operation would be expressed by an IDL 'Unfold' operation. However, whereas their work applies pruning in the creation of the IDL-expression prior to LM application, GYRO uses unweighted phrase constraints so the LM must be considered for pruning while parsing.

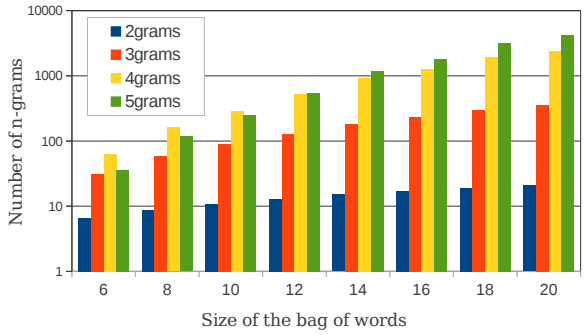

Figure 5: Average number of extracted phrases as a function of the bag of word size.

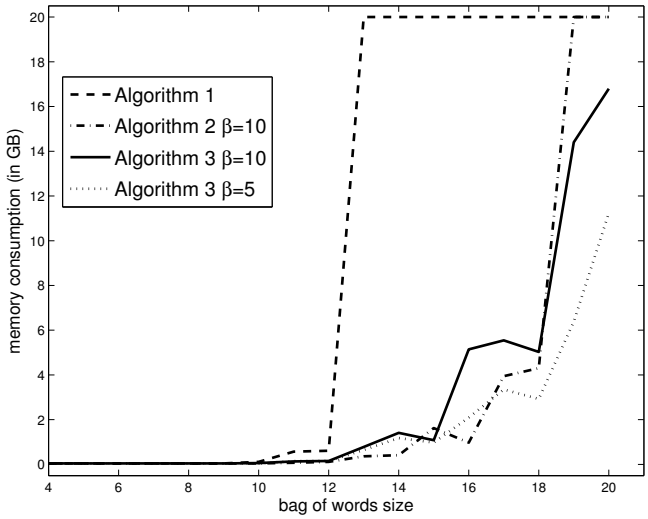

Figure 6: Worst-case memory required (GB) by each GYRO algorithm relative to the size of the bags.

3 Experimental Results

We now report various experiments evaluating the performance of the generation approach described above. The system is evaluated using the MT08-nw, and MT09-nw testsets. These correspond to the first English reference of the newswire portion of the Arabic-to-English NIST MT evaluation sets[2]. They contain 813 and 586 sentences respectively (53,325 tokens in total; average sentence length = 38.1 tokens after tokenization). In order to reduce the computational complexity, all sentences with more than 20 tokens were divided into sub-sentences, with 20 tokens being the upper limit. Between 70-80% of the sentences in the testsets were divided in this way. For each of these sentences we create a bag.

The GYRO system uses a n-gram LM estimated over 1.3 billion words of English text, including the AFP and Xinhua portions of the GigaWord corpus version 4 (1.1 billion words) and the English side of various Arabic-English parallel corpora typically used in MT evaluations (0.2 billion words).

Phrases of up to length 5 are extracted for each bag from a text collection containing 10.6 billion words of English news text. We use efficient Hadoop-based look-up techniques to carry out this extraction step and to retrieve rules for generation (Pino et al., 2012). The average number of phrases extracted as a function of the size of the bag is shown in Figure 5. These are the phrase-based rules of our generation grammar.

[2]http://www.itl.nist.gov/iad/mig/tests/mt

3.1 Computational Analysis

We analyze here the computational requirements of the three alternative GYRO algorithms presented in Sections 2.3 and 2.4. We carry out this analysis on a subset of 200 random subsentences from MT08-nw and MT09-nw chosen to have the same sentence length distribution as the whole data set. For a fixed generation grammar comprised of 3-gram, 4-gram and 5-gram rules only, we run each algorithm with a memory limitation of 20GB. If the process reaches this limit, then it is killed. Figure 6 reports the worst-case memory memory required by each algorithm as a function of the size of the bag.

As shown, Full Expansion (Algorithm 1) is only feasible for bags that contain at most 12 words. By contrast, Pruned Expansion (Algorithm 2) with $\beta = 10$ is feasible for bags of up to 18 words. For bigger bags, the requirements of unpruned parsing make generation intractable under the memory limit. Finally, Pruned Parsing and Generation (Algorithm 3) is feasible at all bag sizes (up to 20 words), and its memory requirements can be controlled via the beam-width pruning parameter β. Harsher pruning (i.e. lower β) will incur more coverage problems, so it is desirable to use the highest feasible value of β.

We emphasise that Algorithm 3, with suitable pruning strategies, can scale up to larger problems quite readily and generate output from much larger input

LM	System	MT08-nw	MT09-nw
4g	CCG	48.0	48.8
3g	GYRO	59.0	58.4
	GYRO +3g	63.0	64.1
4g	GYRO +4g	65.5	65.9
	100-best oracle	76.1	76.1
	lattice oracle	80.4	80.2

Table 1: CCG and GYRO BLEU scores.

sets than reported here. We focus here on generation quality for moderate sized problems.

3.2 Generation Performance

We now compare the GYRO system with the Combinatory Categorial Grammar (CCG)-based system described in (Zhang et al., 2012). By means of extracted CCG rules, the CCG system searches for an optimal parse guided by large-margin training. Each partial hypothesis (or 'edge') is scored using the syntax model and a 4-gram LM trained similarly on one billion words of English Gigaword data. Both systems are evaluated using BLEU (Papineni et al., 2002; Espinosa et al., 2010).

For GYRO, we use the pruned parsing algorithm of Section 2.4 with $r = 6$ and $\beta = 10$ and a memory usage limit of 20G. The phrase-based rules of the grammar contain only 3-grams, 4-grams and 5-grams.[3] Under these conditions, GYRO finds an output for 91.4% of the bags. For the remainder, we obtain an output either by pruning less or by adding bigram rules (in 7.2% of the bags), or simply by adding all words as unigram rules (1.4% of the bags).

Table 1 gives the results obtained by CCG and GYRO under a 3-gram or a 4-gram LM. Because GYRO outputs word lattices as opposed to a 1-best hypothesis, we can reapply the same LM to the concatenated lattices of any sentences longer than 20 to take into account context in subsentence boundaries. This is the result in the third row in the Table, labeled 'GYRO +3g'. We can see that GYRO benefits significantly from this rescoring, beating the CCG system across both sets. This is possibly explained by the CCG system's dependence upon in-domain data that have been explicitly marked-up using the CCG formalism. The final row reports the positive impact of increasing the LM order to 4.

Impact of generation grammar. To measure the benefits of using high-order n-grams as constraints for generation, we also ran GYRO with unigram rules only. This effectively does permutation under the LM with the pruning mechanisms described. The BLEU scores are 54.0 and 54.5 for MT08-nw and MT09 respectively. This indicates that a strong GYRO grammar is very much needed for this type of parsing and generation.

Quality of generated lattices. We assess the quality of the lattices output by GYRO under the 4-gram LM by computing the oracle BLEU score of either the 100-best lists or the whole lattices[4] in the last two rows of Table 1. In order to compute the latter, we use the linear approximation to BLEU that allows an efficient FST-based implementation of an Oracle search (Sokolov et al., 2012). We draw two conclusions from these results: (a) that there is a significant potential for improvement from rescoring, in that even for small 100-best lists the improvement found by the Oracle can exceed 10 BLEU points; and (b) that the output lattices are not perfect in that the Oracle score is not 100.

3.2.1 Rescoring GYRO output

We now report on rescoring procedures intended to improve the first-pass lattices generated by GYRO.

Higher-order language models. The first row in Table 2 reports the result obtained when applying a 5-gram LM to the GYRO lattices generated under a 4-gram. The 5-gram is estimated over the complete 10.6 billion word collection using the uniform backoff strategy of (Brants et al., 2007). We find improvements of 3.0 and 1.9 BLEU with respect to the 4-gram baseline.

Dependency language models. We now investigate the benefits of applying a dependency LM (Shen et al., 2010) in a rescoring mode. We run the MALT dependency parser[5] on the generation hypotheses and rescore them according to $log(p_{LM}) + \lambda_d log(p_{depLM})$, i.e. a weighted combination of the word-based LM and the dependency LM scores. Since it is not possible to run the parser on the entire lattice, we carry out this experiment using the 100-best lists generated from the previous experiment

[3] Any word in the bag that does not occur in the large collection of English material is added as a 1-gram rule.

[4] Obtained by pruning at $\beta = 10$ in generation.

[5] Available at www.maltparser.org

('+5g'). The dependency LM is a 3-gram estimated on the entire GigaWord version 5 collection (∼5 billion words). Results are shown in rows 2 and 3 in Table 2, where in each row the performance over the set used to tune the parameter λ_d is marked with $\star$. In either case, we observe modest but consistent gains across both sets. We find this very promising considering that the parser has been applied to noisy input sentences.

Minimum Bayes Risk Decoding. We also use Lattice-based Minimum Bayes Risk (LMBR) decoding (Tromble et al., 2008; Blackwood et al., 2010a). Here, the posteriors over n-grams are computed over the output lattices generated by the GYRO system. The result is shown in row labeled '+5g +LMBR', where again we find modest but consistent gains across the two sets with respect to the 5-gram rescored lattices.

LMBR with MT posteriors. We investigate LMBR decoding when applying to the generation lattice a linear combination of the n-gram posterior probabilities extracted from (a) the same generation lattice, and (b) from lattices produced by an Arabic-to-English hierarchical-phrase based MT system developed for the NIST 2012 OpenMT Evaluation. As noted, LMBR relies on a posterior distribution over n-grams as part of its computation or risk. Here, we use LMBR with a posterior of the form $\alpha\, p_{\text{GYRO}} + (1{-}\alpha)\, p_{\text{MT}}$. This is effectively performing a system combination between the GYRO generation system and the MT system (de Gispert et al., 2009; DeNero et al., 2010) but restricting the hypothesis space to be that of the GYRO lattice (Blackwood et al., 2010b). Results are reported in the last two rows of Table 2. Relative to 5-gram LM rescoring alone, we see gains in BLEU of 2.3 and 4.4 in MT08-nw and MT09-nw, suggesting that posterior distributions over n-grams provided by SMT systems can give good guidance in generation. These results also suggest that if we knew what words to use, we could generate very good quality translation output.

3.3 Analysis and examples

Figure 7 gives GYRO generation examples. These are often fairly fluent, and it is striking how the output can be improved with guidance from the SMT system. The examples also show the harshness of BLEU, e.g.

4g GYRO rescoring:	MT08-nw	MT09-nw
+5g	68.5	67.8
+5g +depLM $\lambda_d = 0.4$	68.7 *	68.1
+5g +depLM $\lambda_d = 0.33$	68.7	68.2 *
+5g +LMBR	68.6	68.3
+5g +LMBR-mt $\alpha = 0.25$	70.8 *	72.2
+5g +LMBR-mt $\alpha = 0.25$	70.8	72.2 *

Table 2: Results in BLEU when rescoring the lattices generated by GYRO using various strategies. Tuning conditions are marked by *.

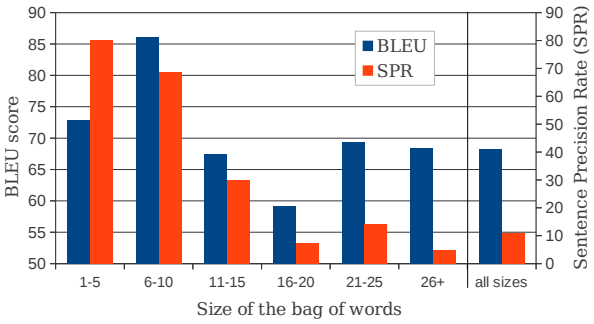

Figure 8: GYRO BLEU score and Sentence Precision Rate as a function of the bag of words size. Computed on the concatenation of MT08-nw and MT09-nw.

'german and turkish officials' is penalised with respect to ' turkish and german officials.' Metrics based on richer meaning representations, such as HyTER, could be valuable here (Dreyer and Marcu, 2012).

Figure 8 shows BLEU and Sentence Precision Rate (SPR), the percentage of exactly reconstructed sentences. As expected, performance is sensitive to length. For bags of up to 10, GYRO reconstructs the reference perfectly in over 65% of the cases. This is a harsh performance metric, and performance falls to less than 10% for bags of size 16-20. For bags of 6-10 words, we find BLEU scores of greater than 85. Performance is not as good for shorter segments, since these are often headlines and bylines that can be ambiguous in their ordering. The BLEU scores for bags of size 21 and higher are an artefact of our sentence splitting procedure. However, even for bag sizes of 16-to-20 GYRO has BLEU scores above 55.

3.4 Human Assessments

Finally, the CCG and 4g-GYRO+5g systems were compared using crowd-sourced fluency judgements gathered on CrowdFlower. Judges were asked 'Please

	Hypothesis	SBLEU
REF	a third republican senator joins the list of critics of bush 's policy in iraq .	
(a)	critics of bush 's iraq policy in a third of republican senator joins the list .	47.2
(b)	critics of bush 's policy in iraq joins the list of a third republican senator .	69.8
(c)	critics of bush 's iraq policy in a list of republican senator joins the third .	39.1
(d)	the list of critics of bush 's policy in iraq a third republican senator joins .	82.9
REF	it added that these messages were sent to president bashar al-asad through turkish and german officials .	
(a-c)	it added that president bashar al-asad through these messages were sent to german and turkish officials .	61.5
(d)	it added that these messages were sent to president bashar al-asad through german and turkish officials .	80.8
REF	a prominent republican senator has joined the ranks of critics of george bush 's policy in iraq , calling for a new strategy just days before a new confrontation in congress	
(a)	a prominent republican senator george has joined the ranks of critics of bush 's policy in iraq , just days before a new strategy in congress calling for a new confrontation	66.7
(b)	a prominent republican senator has joined the ranks of critics of george bush 's policy in iraq , just days before congress calling for a new strategy in a new confrontation	77.8
(c)	a prominent republican senator has joined the ranks of critics of george bush 's policy in iraq , just days before a new strategy in congress calling for a new confrontation	82.3
(d)	a prominent republican senator has joined the ranks of critics of george bush 's policy in iraq , calling for a new strategy just days before a new confrontation in congress	100

Figure 7: 4g GYRO (Table 2) output examples, with sentence level BLEU: (a) GYRO+4g; (b) GYRO+5g; (c) GYRO+5g+LMBR; (d) GYRO+5g+LMBR-mt. (a-c) indicates systems with identical hypotheses.

read the reference sentence and compare the fluency of items 1 & 2.' The test was a selection of 75 fluent sentences of 20 words or less taken from the MT dev sets. Each comparison was made by at least 3 judges. With an average selection confidence of 0.754, GYRO was preferred in 45 cases, CCG was preferred in 14 cases, and systems were tied 16 times. This is consistent with the significant difference in BLEU between these systems.

4 Related Work and Conclusion

Our work is related to surface realisation within natural language generation (NLG). NLG typically assumes a relatively rich input representation intended to provide syntactic, semantic, and other relationships to guide generation. Example input representations are Abstract Meaning Representations (Langkilde and Knight, 1998), attribute-value pairs (Ratnaparkhi, 2000), lexical predicate-argument structures (Bangalore and Rambow, 2000), Interleave-Disjunction-Lock (IDL) expressions (Nederhof and Satta, 2004; Soricut and Marcu, 2005; Soricut and Marcu, 2006), CCGbank derived grammars (White et al., 2007), meaning representation languages (Wong and Mooney, 2007) and unordered syntactic dependency trees (Guo et al., 2011; Bohnet et al., 2011; Belz et al., 2011; Belz et al., 2012)[6].

[6]Surface Realisation Task, Generation Challenges 2011, www.nltg.brighton.ac.uk/research/genchal11

These input representations are suitable for applications such as dialog systems, where the system maintains the information needed to generate the input representation for NLG (Lemon, 2011), or summarisation, where representations can be automatically extracted from coherent, well-formed text (Barzilay and Elhadad, 2011; Althaus et al., 2004). However, there are other applications, such as automatic speech recognition and SMT that could possibly benefit from NLG, but which do not generate reliable linguistic annotation in their output. For these problems it would be useful to have systems, as described in this paper, which do not require rich input representations. We plan to investigate these applications in future work.

There is much opportunity for future development. To improve coverage, the grammars of Section 2.1 could perform generation with overlapping, rather than concatenated, n-grams; and features could be included to define tuneable log-linear rule probabilities (Och and Ney, 2002; Chiang, 2007). The GYRO grammar could be extended using techniques from string-to-tree SMT, in particular by modifying the grammar so that output derivations respect dependencies (Shen et al., 2010); this will make it easier to integrate dependency LMs into GYRO. Finally, it would be interesting to couple the GYRO architecture with automata-based models of poetry and rhythmic text (Greene et al., 2010).

Acknowledgement

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7-ICT-2009-4) under grant agreement number 247762, the FAUST project faust-fp7.eu/faust/, and the EPSRC (UK) Programme Grant EP/I031022/1 (Natural Speech Technology) www.natural-speech-technology.org.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23, Prague, Czech Republic.

Ernst Althaus, Nikiforos Karamanis, and Alexander Koller. 2004. Computing locally coherent discourses. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 399. Association for Computational Linguistics.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics - Volume 1*, COLING '00, pages 42–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Noemie Elhadad. 2011. Inferring strategies for sentence ordering in multidocument news summarization. *arXiv preprint arXiv:1106.1820*.

Anja Belz, Mike 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 Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 217–226, Nancy, France.

Anja Belz, Bernd Bohnet, Simon Mille, Leo Wanner, and Michael White. 2012. The surface realisation task: Recent developments and future plans. In *Proceedings of the 7th International Natural Language Generation Conference*, pages 136–140, Utica, IL, USA.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010a. Efficient path counting transducers for minimum Bayes-risk decoding of statistical machine translation lattices. In *Proceedings of ACL: Short Papers*, pages 27–32, Uppsala, Sweden.

Graeme Blackwood, Adrià de Gispert, and William Byrne. 2010b. Fluency constraints for minimum Bayes-risk decoding of statistical machine translation lattices. In *Proceedings of COLING*, pages 71–79, Beijing, China.

Bernd Bohnet, Simon Mille, Benoît Favre, and Leo Wanner. 2011. <StuMaBa>: From deep representation to surface. In *Proceedings of the Generation Challenges Session at the 13th European Workshop on Natural Language Generation*, pages 232–235, Nancy, France.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of EMNLP-CoNLL*, pages 858–867, Prague, Czech Republic.

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

Adrià de Gispert, Sami Virpioja, Mikko Kurimo, and William Byrne. 2009. Minimum Bayes risk combination of translation hypotheses from alternative morphological decompositions. In *Proceedings of HLT-NAACL: Short Papers*, pages 73–76, Boulder, CO, USA.

Adrià de Gispert, Gonzalo Iglesias, Graeme Blackwood, Eduardo R. Banga, and William Byrne. 2010. Hierarchical phrase-based translation with weighted finite-state transducers and shallow-n grammars. *Computational Linguistics*, 36(3):505–533.

John DeNero, Shankar Kumar, Ciprian Chelba, and Franz Och. 2010. Model combination for machine translation. In *Proceedings of HTL-NAACL*, pages 975–983, Los Angeles, CA, USA.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of NAACL-HLT*, pages 162–171, Montréal, Canada.

Dominic Espinosa, Rajakrishnan Rajkumar, Michael White, and Shoshana Berleant. 2010. Further meta-evaluation of broad-coverage surface realization. In *Proceedings of EMNLP*, pages 564–574, Cambridge, MA, USA.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of COLING*, pages 376–384, Beijing, China.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of EMNLP*, pages 524–533, Cambridge, MA, USA.

Y. Guo, H. Wang, and J. Van Genabith. 2011. Dependency-based n-gram models for general purpose sentence realisation. *Natural Language Engineering*, 17(04):455–483.

Gonzalo Iglesias, Cyril Allauzen, William Byrne, Adrià de Gispert, and Michael Riley. 2011. Hierarchical

phrase-based translation representations. In *Proceedings of EMNLP*, pages 1373–1383, Edinburgh, Scotland, UK.

Kevin Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of ACL/COLING*, pages 704–710, Montreal, Quebec, Canada.

Oliver Lemon. 2011. Learning what to say and how to say it: Joint optimisation of spoken dialogue management and natural language generation. *Computer Speech & Language*, 25(2):210–221.

Mark-Jan Nederhof and Giorgio Satta. 2004. IDL-expressions: A formalism for representing and parsing finite languages in natural language processing. *Journal of Artificial Intelligence Research*, 21:287–317.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of ACL*, pages 295–302, Philadelphia, PA, USA.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, Philadelphia, PA, USA.

Juan Pino, Aurelien Waite, and William Byrne. 2012. Simple and efficient model filtering in statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, 98:5–24.

Adwait Ratnaparkhi. 2000. Trainable methods for surface natural language generation. In *Proceedings of NAACL*, pages 194–201, Seattle, WA, USA.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-dependency statistical machine translation. *Computational Linguistics*, 36(4):649–671.

Artem Sokolov, Guillaume Wisniewski, and Francois Yvon. 2012. Computing lattice bleu oracle scores for machine translation. In *Proceedings of EACL*, pages 120–129, Avignon, France.

Radu Soricut and Daniel Marcu. 2005. Towards developing generation algorithms for text-to-text applications. In *Proceedings of ACL*, pages 66–74, Ann Arbor, MI, USA.

Radu Soricut and Daniel Marcu. 2006. Stochastic Language Generation Using WIDL-Expressions and its Application in Machine Translation and Summarization. In *Proceedings of ACL*, pages 1105–1112, Sydney, Australia.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of EMNLP*, pages 1007–1016, Singapore.

Roy Tromble, Shankar Kumar, Franz Och, and Wolfgang Macherey. 2008. Lattice Minimum Bayes-Risk decoding for statistical machine translation. In *Proceedings of EMNLP*, pages 620–629, Honolulu, Hawaii, USA.

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 EACL*, pages 852–860, Athens, Greece.

Michael White, Rajakrishnan Rajkumar, and Scott Martin. 2007. Towards broad coverage surface realization with ccg. In *Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+ MT)*.

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. *Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT-07)*, pages 172–179.

Yue Zhang and Stephen Clark. 2011. Syntax-based Grammaticality Improvement using CCG and Guided Search. In *Proceedings of EMNLP*, pages 1147–1157, Edinburgh, Scotland, U.K.

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