A General-Purpose Rule Extractor for SCFG-Based Machine Translation

Greg Hanneman and **Michelle Burroughs** and **Alon Lavie**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213 USA
{ghannema, mburroug, alavie}@cs.cmu.edu

Abstract

We present a rule extractor for SCFG-based MT that generalizes many of the contraints present in existing SCFG extraction algorithms. Our method's increased rule coverage comes from allowing multiple alignments, virtual nodes, and multiple tree decompositions in the extraction process. At decoding time, we improve automatic metric scores by significantly increasing the number of phrase pairs that match a given test set, while our experiments with hierarchical grammar filtering indicate that more intelligent filtering schemes will also provide a key to future gains.

1 Introduction

Syntax-based machine translation systems, regardless of the underlying formalism they use, depend on a method for acquiring bilingual rules in that formalism to build the system's translation model. In modern syntax-based MT, this formalism is often synchronous context-free grammar (SCFG), and the SCFG rules are obtained automatically from parallel data through a large variety of methods.

Some SCFG rule extraction techniques require only Viterbi word alignment links between the source and target sides of the input corpus (Chiang, 2005), while methods based on linguistic constituency structure require the source and/or target side of the input to be parsed. Among such techniques, most retain the dependency on Viterbi word alignments for each sentence (Galley et al., 2004; Zollmann and Venugopal, 2006; Lavie et al., 2008; Chiang, 2010) while others make use of a general, corpus-level statistical lexicon instead of individual alignment links (Zhechev and Way, 2008). Each method may also place constraints on the size, format, or structure of the rules it returns.

This paper describes a new, general-purpose rule extractor intended for cases in which two parse trees and Viterbi word alignment links are provided for each sentence, although compatibility with single-parse-tree extraction methods can be achieved by supplying a flat "dummy" parse for the missing tree. Our framework for rule extraction is thus most similar to the Stat-XFER system (Lavie et al., 2008; Ambati et al., 2009) and the tree-to-tree situation considered by Chiang (2010). However, we significantly broaden the scope of allowable rules compared to the Stat-XFER heuristics, and our approach differs from Chiang's system in its respect of the linguistic constituency constraints expressed in the input tree structure. In summary, we attempt to extract the greatest possible number of syntactically motivated rules while not allowing them to violate explicit constituent boundaries on either the source or target side. This is achieved by allowing creation of virtual nodes, by allowing multiple decompositions of the same tree pair, and by allowing extraction of SCFG rules beyond the minimial set required to regenerate the tree pair.

After describing our extraction method and comparing it to a number of existing SCFG extraction techniques, we present a series of experiments examining the number of rules that may be produced from an input corpus. We also describe experiments on Chinese-to-English translation that suggest that filtering a very large extracted grammar to a more

Proceedings of SSST-5, Fifth Workshop on Syntax, Semantics and Structure in Statistical Translation, pages 135–144,
ACL HLT 2011, Portland, Oregon, USA, June 2011.

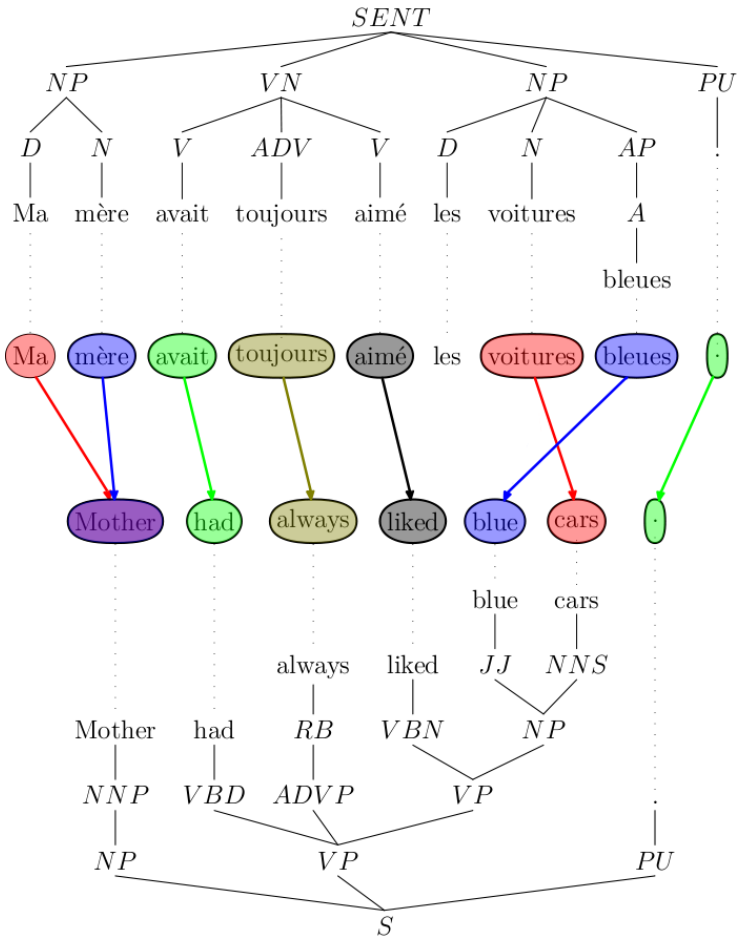

Figure 1: Sample input for our rule extraction algorithm. It consists of a source-side parse tree (French) and a target-side parse tree (English) connected by a Viterbi word alignment.

moderate-sized translation model is an important consideration for obtaining strong results. Finally, this paper concludes with some suggestions for future work.

2 Rule Extraction Algorithm

We begin with a parallel sentence consisting of a source-side parse tree S, a target-side parse tree T, and a Viterbi word alignment between the trees' leaves. A sample sentence of this type is shown in Figure 1. Our goal is to extract a number of SCFG rules that are licensed by this input.

2.1 Node Alignment

Our algorithm first computes a node alignment between the parallel trees. A node s in tree S is aligned to a node t in tree T if the following constraints are met. First, all words in the yield of s must either be aligned to words within the yield of t, or they must be unaligned. Second, the reverse must also hold: all words in the yield of t must be aligned to words within the yield of s or again be unaligned. This is analogous to the word-alignment consistency constraint of phrase-based SMT phrase extraction (Koehn et al., 2003). In Figure 1, for example, the NP dominating the French words *les voitures bleues* is aligned to the equivalent English NP node dominating *blue cars*.

As in phrase-based SMT, where a phrase in one language may be consistent with multiple possible phrases in the other language, we allow parse nodes in both trees to have multiple node alignments. This is in contrast to one-derivation rule extractors such as that of Lavie et al. (2008), in which each node

in S may only be aligned to a single node in T and vice versa. The French NP node *Ma mère*, for example, aligns to both the NNP and NP nodes in English producing *Mother*.

Besides aligning existing nodes in both parse trees to the extent possible, we also permit the introduction of "virtual" nodes into either tree. Virtual nodes are created when two or more contiguous children of an existing node are aligned consistently to a node or a similar set of two or more contiguous children of a node in the opposite parse tree. Virtual nodes may be aligned to "original" nodes in the opposite tree or to other virtual nodes.

In Figure 1, the existing English NP node *blue cars* can be aligned to a new virtual node in French that dominates the N node *voitures* and the AP node *bleues*. The virtual node is inserted as the parent of N and AP, and as the child of the NP node directly above. In conjunction with node alignments between existing nodes, this means that the English NP *blue cars* is now aligned twice: once to the original French NP node and once to the virtual node N+AP. We thus replicate the behavior of "growing into the gaps" from phrase-based SMT in the presence of unaligned words. As another example, a virtual node in French covering the V node *avait* and the ADV node *toujours* could be created to align consistently with a virtual node in English covering the VBD node *had* and the ADVP node *always*.

Since virtual nodes are always created out of children of the same node, they are always consistent with the existing syntactic structure of the tree. Within the constraints of the existing tree structure and word alignments, however, all possible virtual nodes are considered. This is in keeping with our philosophy of allowing multiple alignments without violating constituent boundaries. Near the top of the trees in Figure 1, for example, French virtual nodes NP+VN+NP (aligned to English NP+VP) and VN+NP+PU (aligned to VP+PU) both exist, even though they overlap. In our procedure, we do allow a limit to be placed the number of child nodes that can be combined into a virtual node. Setting this limit to two, for instance, will constrain node alignment to the space of possible synchronous binarizations consistent with the Viterbi word alignments.

2.2 Grammar Extraction

Given the final set of node alignments between the source tree and the target tree, SCFG rules are obtained via a grammar extraction step. Rule extraction proceeds in a depth-first manner, such that rules are extracted and cached for all descendents of a source node s before rules in which s is the left-hand side are considered. Extracting rules where source node s is the left-hand side consists of two phases: decomposition and combination.

The first phase is decomposition of node s into all distinct sets $D = \{d_1, d_2, \ldots, d_n\}$ of descendent nodes such that D spans the entire yield of node s, where $d_i \in D$ is node-aligned or is an unaligned terminal for all i, and d_i has no ancestor a where a is a descendent of s and a is node-aligned. Each D thus represents the right-hand side of a minimal SCFG rule rooted at s. Due to the introduction of overlapping virtual nodes, the decomposition step may involve finding multiple sets of decomposition points when there are multiple nodes with the same span at the same level of the tree.

The second phase involves composition of all rules derived from each element of D subject to certain constraints. Rules are constructed using s, the set of nodes $T_s = \{t \mid s \text{ is aligned to } t\}$, and each decomposed node set D. The set of left-hand sides is $\{s\} \times T_s$, but there may be many right-hand sides for a given t and D. Define $rhs(d)$ as the set of right-hand sides of rules that are derived from d, plus all alignments of d to its aligned set T_d. If d is a terminal, word alignments are used in the place of node alignments. To create a set of right-hand sides, we generate the set $R = rhs(d_1) \times \ldots \times rhs(d_n)$. For each $r \in R$, we execute a $combine$ operation such that $combine(r)$ creates a new right-hand side by combining the component right-hand sides and recalculating co-indexes between the source- and target-side nonterminals. Finally, we insert any unaligned terminals on either side.

We work through a small example of grammar extraction using Figure 2, which replicates a fragment of Figure 1 with virtual nodes included. The English node JJ is aligned to the French nodes A and AP, the English node NNS is aligned to the French node N and the virtual node D+N, and the English node NP is aligned to the French node NP and the

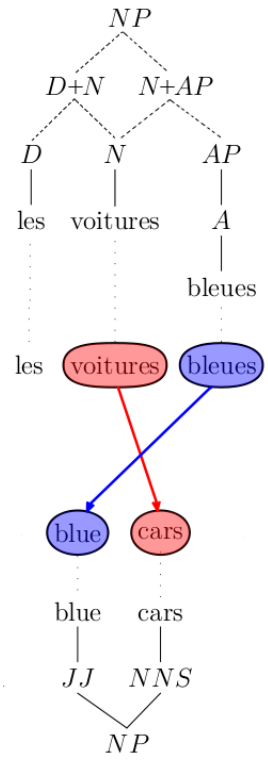

Figure 2: A fragment of Figure 1 with virtual nodes (symbolized by dashed lines) added on the French side. Nodes D, N, and AP are all original children of the French NP.

virtual node N+AP. To extract rules from the French node NP, we consider two potential decompositions: $D_1 = \{\text{D+N}, \text{AP}\}$ and $D_2 = \{\text{les}, \text{N+AP}\}$. Since the French NP is aligned only to the English NP, the set of left-hand sides is {NP::NP}, where we use the symbol "::" to separate the source and target sides of joint nonterminal label or a rule.

In the next step, we use cached rules and alignments to generate all potential right-hand-side pieces from these top-level nodes:

$$rhs(\text{D+N}) = \left\{ \begin{array}{l} [\text{D+N}^1] :: [\text{NNS}^1], \\ [\text{les voitures}] :: [\text{cars}] \end{array} \right\}$$

$$rhs(\text{AP}) = \left\{ \begin{array}{l} [\text{AP}^1] :: [\text{JJ}^1], \\ [\text{A}^1] :: [\text{JJ}^1], \\ [\text{bleues}] :: [\text{blue}] \end{array} \right\}$$

$$rhs(\text{les}) = \emptyset$$

$$rhs(\text{N+AP}) = \left\{ \begin{array}{l} [\text{N+AP}^1] :: [\text{NP}^1], \\ [\text{N}^1\ \text{AP}^2] :: [\text{JJ}^2\ \text{NNS}^1], \\ [\text{N}^1\ \text{A}^2] :: [\text{JJ}^2\ \text{NNS}^1], \\ [\text{voitures AP}^1] :: [\text{JJ}^1\ \text{cars}], \\ [\text{voitures A}^1] :: [\text{JJ}^1\ \text{cars}], \\ [\text{N}^1\ \text{bleues}] :: [\text{blue NNS}^1], \\ [\text{voitures bleues}] :: [\text{blue cars}] \end{array} \right\}$$

Next we must combine these pieces. For example, from D_1 we derive the full right-hand sides

1. *combine*([les voitures]::[cars], [bleues]::[blue])
   = [les voitures bleues]::[blue cars]
2. *combine*([les voitures]::[cars], [A^1]::[JJ^1])
   = [les voitures A^1]::[JJ^1 cars]
3. *combine*([les voitures]::[cars], [AP^1]::[JJ^1])
   = [les voitures AP^1]::[JJ^1 cars]
4. *combine*([D+N^1]::[NNS^1], [bleues]::[blue])
   = [D+N^1 bleues]::[blue NNS^1]
5. *combine*([D+N^1]::[NNS^1], [A^1]::[JJ^1])
   = [D+N^1 A^2]::[JJ^2 NNS^1]
6. *combine*([D+N^1]::[NNS^1], [AP^1]::[JJ^1])
   = [D+N^1 AP^2]::[JJ^2 NNS^1]

Similarly, we derive seven full right-hand sides from D_2. Since $rhs(\text{les})$ is empty, rules derived have right-hand sides equivalent to $rhs(\text{N+AP})$ with the unaligned *les* added on the source side to complete the span of the French NP. For example, *combine*([N+AP^1]::[NP^1]) = [les N+AP^1]::[NP^1].

In the final step, the left-hand side is added to each full right-hand side. Thus,

$$\text{NP} :: \text{NP} \rightarrow [\text{les voitures A}^1] :: [\text{JJ}^1\ \text{cars}]$$

is one example rule extracted from this tree.

The number of rules can grow rapidly: if the parse tree has a branching factor of b and a depth of h, there are potentially $O(2^{b^h})$ rules extracted. To control this, we allow certain constraints on the rules extracted that can short-circuit right-hand-side formation. We allow separate restrictions on the number of items that may appear on the right-hand side of phrase pair rules (max_p) and hierarchical grammar rules (max_g). We also optionally allow the exclusion of parallel unary rules — that is, rules whose right-hand sides consist solely of a pair of aligned nonterminals.

System	Tree Constraints	Multiple Alignments	Virtual Nodes	Multiple Derivations
Hiero	No	—	—	Yes
Stat-XFER	Yes	No	Some	No
GHKM	Yes	No	No	Yes
SAMT	No	No	Yes	Yes
Chiang (2010)	No	No	Yes	Yes
This work	Yes	Yes	Yes	Yes

Table 1: Comparisons between the rule extractor described in this paper and other SCFG rule extraction methods.

3 Comparison to Other Methods

Table 1 compares the rule extractor described in Section 2 to other SCFG extraction methods described in the literature. We include comparisons of our work against the Hiero system (Chiang, 2005), the Stat-XFER system rule learner most recently described by Ambati et al. (2009), the composed version of GHKM rule extraction (Galley et al., 2006), the so-called Syntax-Augmented MT (SAMT) system (Zollmann and Venugopal, 2006), and a Hiero–SAMT extension with source- and target-side syntax described by Chiang (2010). Note that some of these methods make use of only target-side parse trees — or no parse trees at all, in the case of Hiero — but our primary interest in comparison is the constraints placed on the rule extraction process rather than the final output form of the rules themselves. We highlight four specific dimensions along these lines.

Tree Constraints. As we mentioned in this paper's introduction, we do not allow any part of our extracted rules to violate constituent boundaries in the input parse trees. This is in contrast to Hiero-derived techniques, which focus on expanding grammar coverage by extracting rules for all spans in the input sentence pair that are consistently word-aligned, regardless of their correspondence to linguistic constituents. Practitioners of both phrase-based and syntax-based SMT have reported severe grammar coverage issues when rules are required to exactly match parse constituents (Koehn et al., 2003; Chiang, 2010). In our work, we attempt to improve the coverage of the grammar by allowing multiple node alignments, virtual nodes, and multiple tree decompositions rather than ignoring structure constraints.

Multiple Alignments. In contrast to all other extraction methods in Table 1, ours allows a node in one parse tree to be aligned with multiple nodes in the other tree, as long as the word-alignment and structure constraints are satisfied. However, we do not allow a node to have multiple simultaneous alignments — a single node alignment must be chosen for extracting an individual rule. In practice, this prevents extraction of "triangle" rules where the same node appears on both the left- and right-hand side of the same rule.[1]

Virtual Nodes. In keeping with our philosophy of representing multiple alignments, our use of multiple and overlapping virtual nodes is less restrictive than the single-alignment constraint of Stat-XFER. Another key difference is that Stat-XFER requires all virtual nodes to be aligned to original nodes in the other language, while we permit virtual–virtual node alignments. In respecting existing tree structure constraints, our virtual node placement is more restrictive than SAMT or Chiang, where extracted nodes may cross existing constituent boundaries.

Multiple Derivations. Galley et al. (2006) argued that breaking a single tree pair into multiple decompositions is important for correct probability modeling. We agree, and we base our rule extractor's acquisition of multiple derivations per tree pair on techniques from both GHKM and Hiero. More specifically, we borrow from Hiero the idea of creating hierarchical rules by subtracting and abstracting all possible subsets of smaller phrases (aligned nodes in our case) from larger phrases. Like GHKM,

[1] Figure 2 includes a potential triangle rule, D+N :: NNS → [les N^1] :: [NNS^1], where the English NNS node appears on both sides of the rule. It is simultaneously aligned to the French D+N and N nodes.

we do this exhaustively within some limit, although in our case we use a rank limit on a rule's right-hand side rather than a limit on the depth of the subnode subtractions. Our constraint achieves the goal of controlling the size of the rule set while remaining flexibile in terms of depth depending on the shape of the parse trees.

4 Experiments

We conducted experiments with our rule extractor on the FBIS corpus, made up of approximately 302,000 Chinese–English sentence pairs. We parsed the corpus with the Chinese and English grammars of the Berkeley parser (Petrov and Klein, 2007) and word-aligned it with GIZA++ (Och and Ney, 2003). The parsed and word-aligned FBIS corpus served as the input to our rule extractor, which we ran with a number of different settings.

First, we acquired a baseline rule extraction ("xfer-orig") from our corpus using an implementation of the basic Stat-XFER rule learner (Lavie et al., 2008), which decomposes each input tree pair into a single set of minimal SCFG rules[2] using only original nodes in the parse trees. Next, we tested the effect of allowing multiple decompositions by running our own rule learner, but restricting its rules to also only make use of original nodes ("compatible"). Finally, we investigated the total number of extractable rules by allowing the creation of virtual nodes from up to four adjacent sibling nodes and placing two different limits on the length of the right-hand side ("full-short" and "full-long"). These configurations are summarized in Table 2.

Rule Set	max_p	max_g	**Virtual**	**Unary**
xfer-orig	10	∞	No	Yes
compatible	10	5	No	Yes
full-short	5	5	Yes	No
full-long	7	7	Yes	No

Table 2: Rule sets considered by a Stat-XFER baseline ("xfer-orig") and our own rule extractor.

[2]In practice, some Stat-XFER aligned nodes produce two rules instead of one: a minimal hierarchical SCFG rule is always produced, and a phrase pair rule will also be produced for node yields within the max_p cutoff.

4.1 Rules Extracted

As expected, we find that allowing multiple decompositions of each tree pair has a significant effect on the number of extracted rules. Table 3 breaks the extracted rules for each configuration down into phrase pairs (all terminals on the right-hand side) and hierarchical rules (containing at least one nonterminal on the right-hand side). We also count the number of extracted rule instances (tokens) against the number of unique rules (types). The results show that multiple decomposition leads to a four-fold increase in the number of extracted grammar rules, even when the length of the Stat-XFER baseline rules is unbounded. The number of extracted phrase pairs shows a smaller increase, but this is expected: the number of possible phrase pairs is proportional to the square of the sentence length, while the number of possible hierarchical rules is exponential, so there is more room for coverage improvement in the hierarchical grammar.

With virtual nodes included, there is again a large jump in both the number of extracted rule tokens and types, even at relatively short length limits. When both max_p and max_g are set to 7, our rule extractor produces 1.5 times as many unique phrase pairs and 20.5 times as many unique hierarchical rules as the baseline Stat-XFER system, and nearly twice the number of hierarchical rules as when using length limits of 5. Ambati et al. (2009) showed the usefulness of extending rule extraction from exact original–original node alignments to cases in which original–virtual and virtual–original alignments were also permitted. Our experiments confirm this, as only 60% (full-short) and 54% (full-long) of our extracted rule types are made up of only original–original node alignments. Further, we find a contribution from the new virtual–virtual case: approximately 8% of the rules extracted in the "full-long" configuration from Table 3 are headed by a virtual–virtual alignment, and a similar number have a virtual–virtual alignment on their right-hand sides.

All four of the extracted rule sets show Zipfian distributions over rule frequency counts. In the xfer-orig, full-short, and full-long configurations, between 82% and 86% of the extracted phrase pair rules, and between 88% and 92% of the extracted hierarchical rules, were observed only once. These

Rule Set	Extracted Instances		Unique Rules	
	Phrase	Hierarchical	Phrase	Hierarchical
xfer-orig	6,646,791	1,876,384	1,929,641	767,573
compatible	8,709,589	6,657,590	2,016,227	3,590,184
full-short	10,190,487	14,190,066	2,877,650	8,313,690
full-long	10,288,731	22,479,863	2,970,403	15,750,695

Table 3: The number of extracted rule instances (tokens) and unique rules (types) produced by the Stat-XFER system ("xfer-orig") and three configurations of our rule extractor.

percentages are remarkably consistent despite substantial changes in grammar size, meaning that our more exhaustive method of rule extraction does not produce a disproportionate number of singletons.[3] On the other hand, it does weaken the average count of an extracted hierarchical rule type. From Table 3, we can compute that the average phrase pair count remains at 3.5 when we move from xfer-orig to the two full configurations; however, the average hierarchical rule count drops from 2.4 to 1.7 (full-short) and finally 1.4 (full-long). This likely again reflects the exponential increase in the number of extractable hierarchical rules compared to the quadratic increase in the phrase pairs.

4.2 Translation Results

The grammars obtained from our rule extractor can be filtered and formatted for use with a variety of SCFG-based decoders and rule formats. We carried out end-to-end translation experiments with the various extracted rule sets from the FBIS corpus using the open-source decoder Joshua (Li et al., 2009). Given a source-language string, Joshua translates by producing a synchronous parse of it according to a scored SCFG and a target-side language model. A significant engineering challenge in building a real MT system of this type is selecting a more moderate-sized subset of all extracted rules to retain in the final translation model. This is an especially important consideration when dealing with expanded rule sets derived from virtual nodes and multiple decompositions in each input tree.

In our experiments, we pass all grammars through two preprocessing steps before any translation model scoring. First, we noticed that English cardinal numbers and punctuation marks in many languages tend to receive incorrect nonterminal labels during parsing, despite being closed-class items with clearly defined tags. Therefore, before rule extraction, we globally correct the nodel labels of all-numeral terminals in English and certain punctuation marks in both English and Chinese. Second, we attempt to reduce derivational ambiguity in cases where the same SCFG right-hand side appears in the grammar after extraction with a large number of possible left-hand-side labels. To this end, we sort the possible left-hand sides by frequency for each unique right-hand side, and we remove the least frequent 10 percent of the label distribution.

Our translation model scoring is based on the feature set of Hanneman et al. (2010). This includes the standard bidirectional conditional maximum-likelihood scores at both the word and phrase level on the right-hand side of rules. We also include maximum-likelihood scores for the left-hand-side label given all or part of the right-hand side. Using statistics local to each rule, we set binary indicator features for rules whose frequencies are ≤ 3, plus five additional indicator features according to the format of the rule's right-hand side, such as whether it is fully abstract. Since the system in this paper is not constructed using any non-syntactic rules, we do not include the Hanneman et al. (2010) "not labelable" maximum-likelihood features or the indicator features related to non-syntactic labels.

Beyond the above preprocessing and scoring common to all grammars, we experiment with three different solutions to the more difficult problem of selecting a final translation grammar. In any case, we separate phrase pair rules from hierarchical rules

[3] The compatible configuration is somewhat of an outlier. It has proportionally fewer singleton phrase pairs (80%) than the other variants, likely because it allows multiple alignments and multiple decompositions without allowing virtual nodes.

Rule Set	Filter	BLEU	TER	MET
xfer-orig	10k	24.39	68.01	54.35
xfer-orig	5k+100k	25.95	66.27	54.77
compatible	10k	24.28	65.30	53.58
full-short	10k	25.16	66.25	54.33
full-short	100k	25.51	65.56	54.15
full-short	5k+100k	26.08	64.32	54.58
full-long	10k	25.74	65.52	54.55
full-long	100k	25.53	66.24	53.68
full-long	5k+100k	25.83	64.55	54.35

Table 4: Automatic metric results using different rule sets, as well as different grammar filtering methods.

and include in the grammar all phrase pair rules matching a given tuning or testing set. Any improvement in phrase pair coverage during the extraction stage is thus directly passed along to decoding. For hierarchical rules, we experiment with retaining the 10,000 or 100,000 most frequently extracted unique rules. We also separate fully abstract hierarchical rules from partially lexicalized hierarchical rules, and in a further selection technique we retain the 5,000 most frequent abstract and 100,000 most frequent partially lexicalized rules.

Given these final rule sets, we tune our MT systems on the NIST MT 2006 data set using the minimum error-rate training package Z-MERT (Zaidan, 2009), and we test on NIST MT 2003. Both sets have four reference translations. Table 4 presents case-insensitive evaluation results on the test set according to the automatic metrics BLEU (Papineni et al., 2002), TER (Snover et al., 2006), and METEOR (Lavie and Denkowski, 2009).[4] The trend in the results is that including a larger grammar is generally better for performance, but filtering techniques also play a substantial role in determining how well a given grammar will perform at run time.

We first compare the results in Table 4 for different rule sets all filtered the same way at decoding time. With only 10,000 hierarchical rules in use ("10k"), the improvements in scores indicate that an important contribution is being made by the additional phrase pair coverage provided by each successive rule set. The original Stat-XFER rule extraction provides 244,988 phrase pairs that match the MT 2003 test set. This is already increased to 520,995 in the compatible system using multiple decompositions. With virtual nodes enabled, the full system produces 766,379 matching phrase pairs up to length 5 or 776,707 up to length 7. These systems both score significantly higher than the Stat-XFER baseline according to BLEU and TER, and the METEOR scores are likely statistically equivalent.

Across all configurations, we find that changing the grammar filtering technique — possibly combined with retuned decoder feature weights — also has a large influence on automatic metric scores. Larger hierarchical grammars tend to score better, in some cases to the point of erasing the score differences between rule sets. From this we conclude that making effective use of the extracted grammar, no matter its size, with intelligent filtering techniques is at least as important as the number and type of rules extracted overall. Though the filtering results in Table 4 are still somewhat inconclusive, the relative success of the "5k+100k" setting shows that filtering fully abstract and partially lexicalized rules separately is a reasonable starting approach. While fully abstract rules do tend to be more frequently observed in grammar extraction, and thus more reliably scored in the translation model, they also have the ability to overapply at decoding time because their use is not restricted to any particular lexical context.

5 Conclusions and Future Work

We demonstrated in Section 4.1 that the general SCFG extraction algorithm described in this paper is capable of producing very large linguistically motivated rule sets. These rule sets can improve automatic metric scores at decoding time. At the same time, we see the results in Section 4.2 as a springboard to more advanced and more intelligent methods of grammar filtering. Our major research question for future work is to determine how to make the best runtime use of the grammars we can extract.

As we saw in Section 2, multiple decompositions of a single parse tree allow the same constituent to be built in a variety of ways. This is generally good for coverage, but its downside at run time is that the decoder must manage a larger number of competing

[4]For METEOR scoring we use version 1.0 of the metric, tuned to HTER with the exact, stemming, and synonymy modules enabled.

derivations that, in the end, produce the same output string. Grammar filtering that explicitly attempts to limit the derivational ambiguity of the retained rules may prevent the translation model probabilities of correct outputs from getting fragmented into redundant derivations. So far we have only approximated this by using fully abstract rules as a proxy for the most derivationally ambiguous rules.

Filtering based on the content of virtual nodes may also be a reasonable strategy for selecting useful grammar rules and discarding those whose contributions are less necessary. For example, we find in our current output many applications of rules involving virtual nodes that consist of an open-class category and a mark of punctuation, such as VBD+COMMA and NN+PU. While there is nothing technically wrong with these rules, they may not be as helpful in translation as rules for nouns and adjectives such as JJ+NNP+NN or NNP+NNP in flat noun phrase structures such as *former U.S. president Bill Clinton*.

A final concern in making use of our large extracted grammars is the effect virtual nodes have on the size of the nonterminal set. The Stat-XFER baseline grammar from our "xfer-orig" configuration uses a nonterminal set of 1,577 unique labels. In our rule extractor so far, we have adopted the convention of naming virtual nodes with a concatenation of their component sibling labels, separated by "+"s. With the large number of virtual node labels that may be created, this gives our "full-short" and "full-long" extracted grammars nonterminal sets of around 73,000 unique labels. An undesirable consequence of such a large label set is that a particular SCFG right-hand side may acquire a large variety of left-hand-side labels, further contributing to the derivational ambiguity problems discussed above. In future work, the problem could be addressed by reconsidering our naming scheme for virtual nodes, by allowing fuzzy matching of labels at translation time (Chiang, 2010), or by other techniques aimed at reducing the size of the overall nonterminal set.

Acknowledgments

This research was supported in part by U.S. National Science Foundation grants IIS-0713402 and IIS-0915327 and the DARPA GALE program. We thank Vamshi Ambati and Jon Clark for helpful discussions regarding implementation details of the grammar extraction algorithm. Thanks to Chris Dyer for providing the word-aligned and preprocessed FBIS corpus. Finally, we thank Yahoo! for the use of the M45 research computing cluster, where we ran many steps of our experimental pipeline.

References

Vamshi Ambati, Alon Lavie, and Jaime Carbonell. 2009. Extraction of syntactic translation models from parallel data using syntax from source and target languages. In *Proceedings of the 12th Machine Translation Summit*, pages 190–197, Ottawa, Canada, August.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 263–270, Ann Arbor, MI, June.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *HLT-NAACL 2004: Main Proceedings*, pages 273–280, Boston, MA, May.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL*, pages 961–968, Sydney, Australia, July.

Greg Hanneman, Jonathan Clark, and Alon Lavie. 2010. Improved features and grammar selection for syntax-based MT. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 82–87, Uppsala, Sweden, July.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL 2003*, pages 48–54, Edmonton, Alberta, May–June.

Alon Lavie and Michael J. Denkowski. 2009. The METEOR metric for automatic evaluation of machine translation. *Machine Translation*, 23(2–3):105–115.

Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proceedings of the Second ACL Workshop on Syntax and Structure in Statistical Translation*, pages 87–95, Columbus, OH, June.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Juri Ganitkevitch, Sanjeev Khudanpur, Lane Schwartz, Wren N.G. Thornton, Jonathan Weese, and Omar F. Zaidan. 2009. Joshua: An open source toolkit for parsing-based machine translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 135–139, Athens, Greece, March.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, NY, April.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the Seventh Conference of the Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA, August.

Omar F. Zaidan. 2009. Z-MERT: A fully configurable open source tool for minimum error rate training of machine translation systems. *The Prague Bulletin of Mathematical Linguistics*, 91:79–88.

Ventsislav Zhechev and Andy Way. 2008. Automatic generation of parallel treebanks. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 1105–1112, Manchester, England, August.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 138–141, New York, NY, June.