Rule Markov Models for Fast Tree-to-String Translation

Ashish Vaswani
Information Sciences Institute
University of Southern California
avaswani@isi.edu

Haitao Mi
Institute of Computing Technology
Chinese Academy of Sciences
htmi@ict.ac.cn

Liang Huang and **David Chiang**
Information Sciences Institute
University of Southern California
{lhuang,chiang}@isi.edu

Abstract

Most statistical machine translation systems rely on *composed rules* (rules that can be formed out of smaller rules in the grammar). Though this practice improves translation by weakening independence assumptions in the translation model, it nevertheless results in huge, redundant grammars, making both training and decoding inefficient. Here, we take the opposite approach, where we only use *minimal rules* (those that cannot be formed out of other rules), and instead rely on a *rule Markov model* of the derivation history to capture dependencies between minimal rules. Large-scale experiments on a state-of-the-art tree-to-string translation system show that our approach leads to a slimmer model, a faster decoder, yet the same translation quality (measured using B) as composed rules.

1 Introduction

Statistical machine translation systems typically model the translation process as a sequence of translation steps, each of which uses a translation rule, for example, a phrase pair in phrase-based translation or a tree-to-string rule in tree-to-string translation. These rules are usually applied independently of each other, which violates the conventional wisdom that translation should be done in context. To alleviate this problem, most state-of-the-art systems rely on *composed rules*, which are larger rules that can be formed out of smaller rules (including larger phrase pairs that can be formerd out of smaller phrase pairs), as opposed to *minimal rules*, which are rules that cannot be formed out of other rules. Although this approach does improve translation quality dramatically by weakening the independence assumptions in the translation model, they suffer from two main problems. First, composition can cause a combinatorial explosion in the number of rules. To avoid this, ad-hoc limits are placed during composition, like upper bounds on the number of nodes in the composed rule, or the height of the rule. Under such limits, the grammar size is manageable, but still much larger than the minimal-rule grammar. Second, due to large grammars, the decoder has to consider many more hypothesis translations, which slows it down. Nevertheless, the advantages outweigh the disadvantages, and to our knowledge, all top-performing systems, both phrase-based and syntax-based, use composed rules. For example, Galley et al. (2004) initially built a syntax-based system using only minimal rules, and subsequently reported (Galley et al., 2006) that composing rules improves B by 3.6 points, while increasing grammar size 60-fold and decoding time 15-fold.

The alternative we propose is to replace composed rules with a *rule Markov model* that generates rules conditioned on their context. In this work, we restrict a rule's context to the vertical chain of ancestors of the rule. This ancestral context would play the same role as the context formerly provided by rule composition. The dependency treelet model developed by Quirk and Menezes (2006) takes such an approach within the framework of dependency translation. However, their study leaves unanswered whether a rule Markov model can take the place of composed rules. In this work, we investigate the use of rule Markov models in the context of tree-

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 856–864,
Portland, Oregon, June 19-24, 2011.

to-string translation (Liu et al., 2006; Huang et al., 2006). We make three new contributions.

First, we carry out a detailed comparison of rule Markov models with composed rules. Our experiments show that, using trigram rule Markov models, we achieve an improvement of 2.2 B over a baseline of minimal rules. When we compare against *vertically* composed rules, we find that our rule Markov model has the same accuracy, but our model is much smaller and decoding with our model is 30% faster. When we compare against *full* composed rules, we find that our rule Markov model still often reaches the same level of accuracy, again with savings in space and time.

Second, we investigate methods for pruning rule Markov models, finding that even very simple pruning criteria actually improve the accuracy of the model, while of course decreasing its size.

Third, we present a very fast decoder for tree-to-string grammars with rule Markov models. Huang and Mi (2010) have recently introduced an efficient incremental decoding algorithm for tree-to-string translation, which operates top-down and maintains a derivation history of translation rules encountered. This history is exactly the vertical chain of ancestors corresponding to the contexts in our rule Markov model, which makes it an ideal decoder for our model.

We start by describing our rule Markov model (Section 2) and then how to decode using the rule Markov model (Section 3).

2 Rule Markov models

Our model which conditions the generation of a rule on the vertical chain of its ancestors, which allows it to capture interactions between rules.

Consider the example Chinese-English tree-to-string grammar in Figure 1 and the example derivation in Figure 2. Each row is a derivation step; the tree on the left is the derivation tree (in which each node is a rule and its children are the rules that substitute into it) and the tree pair on the right is the source and target derived tree. For any derivation node r, let $anc_1(r)$ be the parent of r (or ϵ if it has no parent), $anc_2(r)$ be the grandparent of node r (or ϵ if it has no grandparent), and so on. Let $anc_1^n(r)$ be the chain of ancestors $anc_1(r) \cdots anc_n(r)$.

The derivation tree is generated as follows. With probability $\mathrm{P}(r_1 \mid \epsilon)$, we generate the rule at the root node, r_1. We then generate rule r_2 with probability $\mathrm{P}(r_2 \mid r_1)$, and so on, always taking the leftmost open substitution site on the English derived tree, and generating a rule r_i conditioned on its chain of ancestors with probability $\mathrm{P}(r_i \mid anc_1^n(r_i))$. We carry on until no more children can be generated. Thus the probability of a derivation tree T is

$$\mathrm{P}(T) = \prod_{r \in T} \mathrm{P}(r \mid anc_1^n(r)) \quad (1)$$

For the minimal rule derivation tree in Figure 2, the probability is:

$$\begin{aligned}\mathrm{P}(T) = \mathrm{P}(r_1 \mid \epsilon) \cdot \mathrm{P}(r_2 \mid r_1) \cdot \mathrm{P}(r_3 \mid r_1) \\ \cdot \mathrm{P}(r_4 \mid r_1, r_3) \cdot \mathrm{P}(r_6 \mid r_1, r_3, r_4) \\ \cdot \mathrm{P}(r_7 \mid r_1, r_3, r_4) \cdot \mathrm{P}(r_5 \mid r_1, r_3) \quad (2)\end{aligned}$$

Training We run the algorithm of Galley et al. (2004) on word-aligned parallel text to obtain a single derivation of minimal rules for each sentence

pair. (Unaligned words are handled by attaching them to the highest node possible in the parse tree.) The rule Markov model

can then be trained on the path set of these derivation trees.

Smoothing We use interpolation with absolute discounting (Ney et al., 1994):

$$\begin{aligned}\mathrm{P}_{abs}(r \mid anc_1^n(r)) = \frac{\max\left\{c(r \mid anc_1^n(r)) - D_n, 0\right\}}{\sum_{r'} c(r' \mid anc_1^n(r'))} \\ + \ (1 - \lambda_n)\mathrm{P}_{abs}(r \mid anc_1^{n-1}(r)), \quad (3)\end{aligned}$$

where $c(r \mid anc_1^n(r))$ is the number of times we have seen rule r after the vertical context $anc_1^n(r)$, D_n is the discount for a context of length n, and $(1 - \lambda_n)$ is set to the value that makes the smoothed probability distribution sum to one.

We experiment with bigram and trigram rule Markov models. For each, we try different values of D_1 and D_2, the discount for bigrams and trigrams, respectively. Ney et al. (1994) suggest using the following value for the discount D_n:

$$D_n = \frac{n_1}{n_1 + n_2} \quad (4)$$

rule id	translation rule
r_1	IP(x_1:NP x_2:VP) → x_1 x_2
r_2	NP(Bùshí) → Bush
r_3	VP(x_1:PP x_2:VP) → x_2 x_1
r_4	PP(x_1:P x_2:NP) → x_1 x_2
r_5	VP(VV(jǔxíng) AS(le) NPB(huìtán)) → held talks
r_6	P(yǔ) → with
r'_6	P(yǔ) → and
r_7	NP(Shālóng) → Sharon

Figure 1: Example tree-to-string grammar.

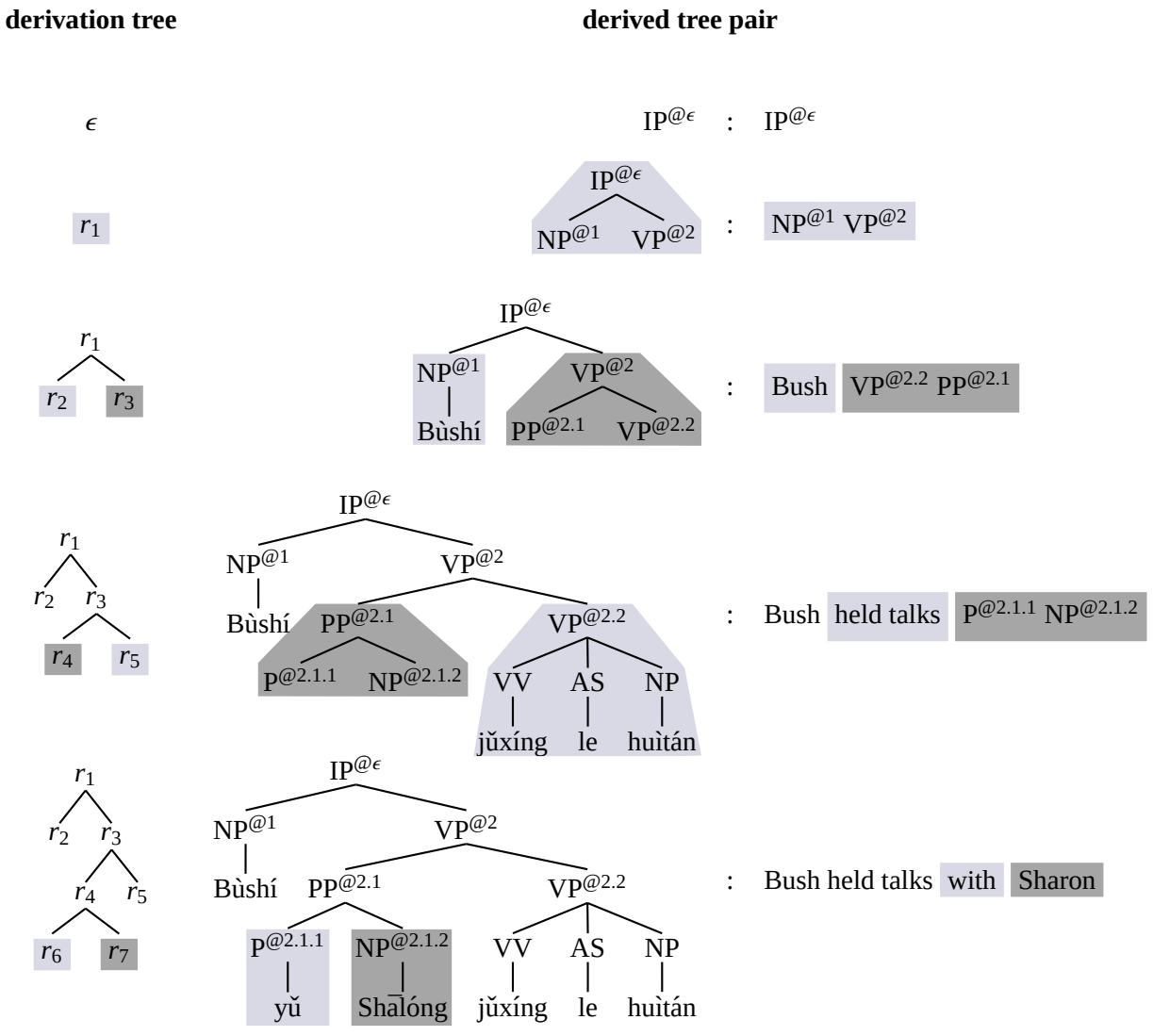

Figure 2: Example tree-to-string derivation. Each row shows a rewriting step; at each step, the leftmost nonterminal symbol is rewritten using one of the rules in Figure 1.

Here, n_1 and n_2 are the total number of n-grams with exactly one and two counts, respectively. For our corpus, $D_1 = 0.871$ and $D_2 = 0.902$. Additionally, we experiment with 0.4 and 0.5 for D_n.

Pruning In addition to full n-gram Markov models, we experiment with three approaches to build smaller models to investigate if pruning helps. Our results will show that smaller models indeed give a higher B score than the full bigram and trigram models. The approaches we use are:

- RM-A: We keep only those contexts in which more than P unique rules were observed. By optimizing on the development set, we set $P = 12$.
- RM-B: We keep only those contexts that were observed more than P times. Note that this is a superset of RM-A. Again, by optimizing on the development set, we set $P = 12$.
- RM-C: We try a more principled approach for learning variable-length Markov models inspired by that of Bejerano and Yona (1999), who learn a Prediction Suffix Tree (PST). They grow the PST in an iterative manner by starting from the root node (no context), and then add contexts to the tree. A context is added if the KL divergence between its predictive distribution and that of its parent is above a certain threshold and the probability of observing the context is above another threshold.

3 Tree-to-string decoding with rule Markov models

In this paper, we use our rule Markov model framework in the context of tree-to-string translation. Tree-to-string translation systems (Liu et al., 2006; Huang et al., 2006) have gained popularity in recent years due to their speed and simplicity. The input to the translation system is a source parse tree and the output is the target string. Huang and Mi (2010) have recently introduced an efficient incremental decoding algorithm for tree-to-string translation. The decoder operates top-down and maintains a derivation history of translation rules encountered. The history is exactly the vertical chain of ancestors corresponding to the contexts in our rule Markov model. This

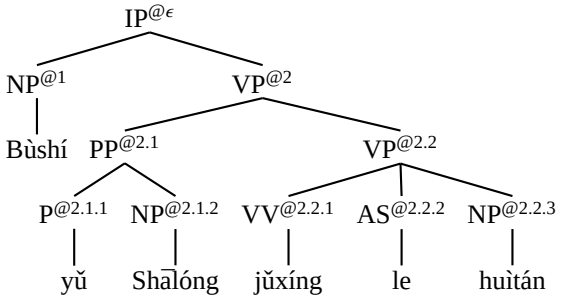

Figure 3: Example input parse tree with tree addresses.

makes incremental decoding a natural fit with our generative story. In this section, we describe how to integrate our rule Markov model into this incremental decoding algorithm. Note that it is also possible to integrate our rule Markov model with other decoding algorithms, for example, the more common non-incremental top-down/bottom-up approach (Huang et al., 2006), but it would involve a non-trivial change to the decoding algorithms to keep track of the vertical derivation history, which would result in significant overhead.

Algorithm Given the input parse tree in Figure 3, Figure 4 illustrates the search process of the incremental decoder with the grammar of Figure 1. We write $X^{@\eta}$ for a tree node with label X at tree address η (Shieber et al., 1995). The root node has address ϵ, and the ith child of node η has address $\eta.i$. At each step, the decoder maintains a stack of active rules, which are rules that have not been completed yet, and the rightmost $(n-1)$ English words translated thus far (the hypothesis), where n is the order of the word language model (in Figure 4, $n = 2$). The stack together with the translated English words comprise a state of the decoder. The last column in the figure shows the rule Markov model probabilities with the conditioning context. In this example, we use a trigram rule Markov model.

After initialization, the process starts at step 1, where we *predict* rule r_1 (the shaded rule) with probability $\mathrm{P}(r_1 \mid \epsilon)$ and push its English side onto the stack, with variables replaced by the corresponding tree nodes: x_1 becomes $\mathrm{NP}^{@1}$ and x_2 becomes $\mathrm{VP}^{@2}$. This gives us the following stack:

$$s = [\mathbf{.}\ \mathrm{NP}^{@1}\ \mathrm{VP}^{@2}]$$

The dot (**.**) indicates the next symbol to process in

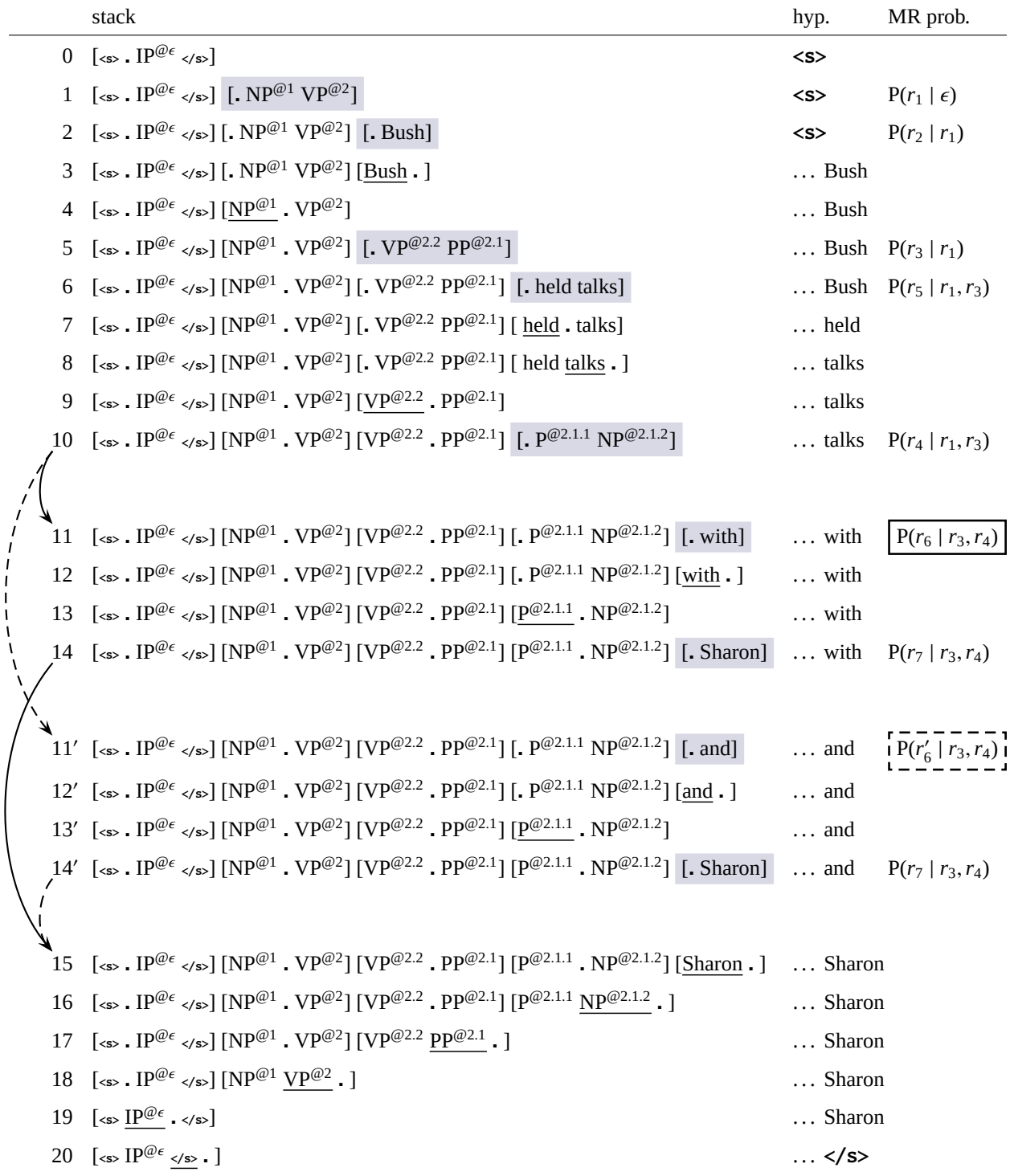

Figure 4: Simulation of incremental decoding with rule Markov model. The solid arrows indicate one path and the dashed arrows indicate an alternate path.

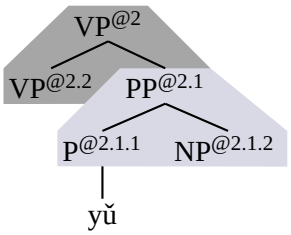

Figure 5: Vertical context r_3 r_4 which allows the model to correctly translate *yǔ* as *with*.

the English word order. We expand node $\text{NP}^{@1}$ first with English word order. We then predict lexical rule r_2 with probability $\text{P}(r_2 \mid r_1)$ and push rule r_2 onto the stack:

$$[\text{. NP}^{@1}\ \text{VP}^{@2}\ ]\ [\text{. Bush}]$$

In step 3, we perform a *scan* operation, in which we append the English word just after the dot to the current hypothesis and move the dot after the word. Since the dot is at the end of the top rule in the stack, we perform a *complete* operation in step 4 where we pop the finished rule at the top of the stack. In the *scan* and *complete* steps, we don't need to compute rule probabilities.

An interesting branch occurs after step 10 with two competing lexical rules, r_6 and r'_6. The Chinese word *yǔ* can be translated as either a preposition *with* (leading to step 11) or a conjunction *and* (leading to step 11′). The word *n*-gram model does not have enough information to make the correct choice, *with*. As a result, good translations might be pruned because of the beam. However, our rule Markov model has the correct preference because of the conditioning ancestral sequence (r_3, r_4), shown in Figure 5. Since $\text{VP}^{@2.2}$ has a preference for *yǔ* translating to *with*, our corpus statistics will give a higher probability to $\text{P}(r_6 \mid r_3, r_4)$ than $\text{P}(r'_6 \mid r_3, r_4)$. This helps the decoder to score the correct translation higher.

Complexity analysis With the incremental decoding algorithm, adding rule Markov models does not change the time complexity, which is $O(nc|V|^{g-1})$, where n is the sentence length, c is the maximum number of incoming hyperedges for each node in the translation forest, V is the target-language vocabulary, and g is the order of the n-gram language model (Huang and Mi, 2010). However, if one were to use rule Markov models with a conventional CKY-style bottom-up decoder (Liu et al., 2006), the complexity would increase to $O(nC^{m-1}|V|^{4(g-1)})$, where C is the maximum number of outgoing hyperedges for each node in the translation forest, and m is the order of the rule Markov model.

4 Experiments and results

4.1 Setup

The training corpus consists of 1.5M sentence pairs with 38M/32M words of Chinese/English, respectively. Our development set is the newswire portion of the 2006 NIST MT Evaluation test set (616 sentences), and our test set is the newswire portion of the 2008 NIST MT Evaluation test set (691 sentences).

We word-aligned the training data using GIZA++ followed by link deletion (Fossum et al., 2008), and then parsed the Chinese sentences using the Berkeley parser (Petrov and Klein, 2007). To extract tree-to-string translation rules, we applied the algorithm of Galley et al. (2004). We trained our rule Markov model on derivations of minimal rules as described above. Our trigram word language model was trained on the target side of the training corpus using the SRILM toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing. The base feature set for all systems is similar to the set used in Mi et al. (2008). The features are combined into a standard log-linear model, which we trained using minimum error-rate training (Och, 2003) to maximize the B score on the development set.

At decoding time, we again parse the input sentences using the Berkeley parser, and convert them into translation forests using rule pattern-matching (Mi et al., 2008). We evaluate translation quality using case-insensitive IBM B -4, calculated by the script `mteval-v13a.pl`.

4.2 Results

Table 1 presents the main results of our paper. We used grammars of minimal rules and composed rules of maximum height 3 as our baselines. For decoding, we used a beam size of 50. Using the best bigram rule Markov models and the minimal rule grammar gives us an improvement of 1.5 B over the minimal rule baseline. Using the best trigram rule Markov model brings our gain up to 2.3 B .

grammar	rule Markov model	max rule height	parameters ($\times 10^6$) full	parameters ($\times 10^6$) dev+test	B test	time (sec/sent)
minimal	None	3	4.9	0.3	24.2	1.2
	RM-B bigram	3	4.9+4.7	0.3+0.5	25.7	1.8
	RM-A trigram	3	4.9+7.6	0.3+0.6	26.5	2.0
vertical composed	None	7	176.8	1.3	26.5	2.9
composed	None	3	17.5	1.6	26.4	2.2
	None	7	448.7	3.3	27.5	6.8
	RM-A trigram	7	448.7+7.6	3.3+1.0	28.0	9.2

Table 1: Main results. Our trigram rule Markov model strongly outperforms minimal rules, and performs at the same level as composed and vertically composed rules, but is smaller and faster. The number of parameters is shown for both the full model and the model filtered for the concatenation of the development and test sets (dev+test).

These gains are statistically significant with $p < 0.01$, using bootstrap resampling with 1000 samples (Koehn, 2004). We find that by just using bigram context, we are able to get at least 1 B point higher than the minimal rule grammar. It is interesting to see that using just bigram rule interactions can give us a reasonable boost. We get our highest gains from using trigram context where our best performing rule Markov model gives us 2.3 B points over minimal rules. This suggests that using longer contexts helps the decoder to find better translations.

We also compared rule Markov models against composed rules. Since our models are currently limited to conditioning on vertical context, the closest comparison is against *vertically* composed rules. We find that our approach performs equally well using much less time and space.

Comparing against *full* composed rules, we find that our system matches the score of the baseline composed rule grammar of maximum height 3, while using many fewer parameters. (It should be noted that a parameter in the rule Markov model is just a floating-point number, whereas a parameter in the composed-rule system is an entire rule; therefore the difference in memory usage would be even greater.) Decoding with our model is 0.2 seconds faster per sentence than with composed rules.

These experiments clearly show that rule Markov models with minimal rules increase translation quality significantly and with lower memory requirements than composed rules. One might wonder if the best performance can be obtained by combining composed rules with a rule Markov model. This is straightforward to implement: the rule Markov model is still defined over derivations of minimal rules, but in the decoder's prediction step, the rule Markov model's value on a composed rule is calculated by decomposing it into minimal rules and computing the product of their probabilities. We find that using our best trigram rule Markov model with composed rules gives us a 0.5 B gain on top of the composed rule grammar, statistically significant with $p < 0.05$, achieving our highest score of 28.0.[1]

rule Markov model	D_1	B dev	time (sec/sent)
RM-A	0.871	29.2	1.8
RM-B	0.4	29.9	1.8
RM-C	0.871	29.8	1.8
RM-Full	0.4	29.7	1.9

Table 2: For rule bigrams, RM-B with $D_1 = 0.4$ gives the best results on the development set.

rule Markov model	D_1	D_2	B dev	time (sec/sent)
RM-A	0.5	0.5	30.3	2.0
RM-B	0.5	0.5	29.9	2.0
RM-C	0.5	0.5	30.1	2.0
RM-Full	0.4	0.5	30.1	2.2

Table 3: For rule bigrams, RM-A with $D_1, D_2 = 0.5$ gives the best results on the development set.

4.3 Analysis

Tables 2 and 3 show how the various types of rule Markov models compare, for bigrams and trigrams,

[1] For this experiment, a beam size of 100 was used.

parameters ($\times 10^6$) dev/test	B dev/test without RMM	with RMM	time (sec/sent) without/with RMM
2.6	31.0/27.0	31.1/27.4	4.5/7.0
2.9	31.5/27.7	31.4/27.3	5.6/8.1
3.3	31.4/27.5	31.4/28.0	6.8/9.2

Table 6: Adding rule Markov models to composed-rule grammars improves their translation performance.

D_2	D_1 0.4	0.5	0.871
0.4	30.0	30.0	
0.5	29.3	30.3	
0.902			30.0

Table 4: RM-A is robust to different settings of D_n on the development set.

parameters ($\times 10^6$) dev+test	B dev	test	time (sec/sent)
1.2	30.2	26.1	2.8
1.3	30.1	26.5	2.9
1.3	30.1	26.2	3.2

Table 5: Comparison of vertically composed rules using various settings (maximum rule height 7).

respectively. It is interesting that the full bigram and trigram rule Markov models do not give our highest B scores; pruning the models not only saves space but improves their performance. We think that this is probably due to overfitting.

Table 4 shows that the RM-A trigram model does fairly well under all the settings of D_n we tried. Table 5 shows the performance of *vertically* composed rules at various settings. Here we have chosen the setting that gives the best performance on the test set for inclusion in Table 1.

Table 6 shows the performance of *fully* composed rules and fully composed rules with a rule Markov Model at various settings.[2] In the second line (2.9 million rules), the drop in B score resulting from adding the rule Markov model is not statistically significant.

5 Related Work

Besides the Quirk and Menezes (2006) work discussed in Section 1, there are two other previous efforts both using a rule bigram model in machine translation, that is, the probability of the current rule only depends on the immediate previous rule in the vertical context, whereas our rule Markov model can condition on longer and sparser derivation histories. Among them, Ding and Palmer (2005) also use a dependency treelet model similar to Quirk and Menezes (2006), and Liu and Gildea (2008) use a tree-to-string model more like ours. Neither compared to the scenario with composed rules.

Outside of machine translation, the idea of weakening independence assumptions by modeling the derivation history is also found in parsing (Johnson, 1998), where rule probabilities are conditioned on parent and grand-parent nonterminals. However, besides the difference between parsing and translation, there are still two major differences. First, our work conditions rule probabilities on parent and grandparent *rules*, not just nonterminals. Second, we compare against a composed-rule system, which is analogous to the Data Oriented Parsing (DOP) approach in parsing (Bod, 2003). To our knowledge, there has been no direct comparison between a history-based PCFG approach and DOP approach in the parsing literature.

6 Conclusion

In this paper, we have investigated whether we can eliminate composed rules without any loss in translation quality. We have developed a rule Markov model that captures vertical bigrams and trigrams of minimal rules, and tested it in the framework of tree-to-string translation. We draw three main conclusions from our experiments. First, our rule Markov models dramatically improve a grammar of minimal rules, giving an improvement of 2.3 B . Second, when we compare against *vertically* composed rules we are able to get about the same B score, but our model is much smaller and decoding with our

[2] For these experiments, a beam size of 100 was used.

model is faster. Finally, when we compare against full composed rules, we find that we can reach the same level of performance under some conditions, but in order to do so consistently, we believe we need to extend our model to condition on horizontal context in addition to vertical context. We hope that by modeling context in both axes, we will be able to completely replace composed-rule grammars with smaller minimal-rule grammars.

Acknowledgments

We would like to thank Fernando Pereira, Yoav Goldberg, Michael Pust, Steve DeNeefe, Daniel Marcu and Kevin Knight for their comments. Mi's contribution was made while he was visiting USC/ISI. This work was supported in part by DARPA under contracts HR0011-06-C-0022 (subcontract to BBN Technologies), HR0011-09-1-0028, and DOI-NBC N10AP20031, by a Google Faculty Research Award to Huang, and by the National Natural Science Foundation of China under contracts 60736014 and 90920004.

References

Gill Bejerano and Golan Yona. 1999. Modeling protein families using probabilistic suffix trees. In *Proc. RECOMB*, pages 15–24. ACM Press.

Rens Bod. 2003. An efficient implementation of a new DOP model. In *Proceedings of EACL*, pages 19–26.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of ACL*, pages 541–548.

Victoria Fossum, Kevin Knight, and Steve Abney. 2008. Using syntax to improve word alignment precision for syntax-based machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*.

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

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 COLING-ACL*, pages 961–968.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proceedings of EMNLP*, pages 273–283.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. Statistical syntax-directed translation with extended domain of locality. In *Proceedings of AMTA*, pages 66–73.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:613–632.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of EMNLP*, pages 388–395.

Ding Liu and Daniel Gildea. 2008. Improved tree-to-string transducer for machine translation. In *Proceedings of the Workshop on Statistical Machine Translation*, pages 62–69.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of COLING-ACL*, pages 609–616.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of ACL: HLT*, pages 192–199.

H. Ney, U. Essen, and R. Kneser. 1994. On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Franz Joseph Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*, pages 160–167.

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

Chris Quirk and Arul Menezes. 2006. Do we need phrases? Challenging the conventional wisdom in statistical machine translation. In *Proceedings of NAACL HLT*, pages 9–16.

Stuart Shieber, Yves Schabes, and Fernando Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Andreas Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proceedings of ICSLP*, volume 30, pages 901–904.