Travatar: A Forest-to-String Machine Translation Engine based on Tree Transducers

Graham Neubig
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi, Nara, Japan
neubig@is.naist.jp

Abstract

In this paper we describe Travatar, a forest-to-string machine translation (MT) engine based on tree transducers. It provides an open-source C++ implementation for the entire forest-to-string MT pipeline, including rule extraction, tuning, decoding, and evaluation. There are a number of options for model training, and tuning includes advanced options such as hypergraph MERT, and training of sparse features through online learning. The training pipeline is modeled after that of the popular Moses decoder, so users familiar with Moses should be able to get started quickly. We perform a validation experiment of the decoder on English-Japanese machine translation, and find that it is possible to achieve greater accuracy than translation using phrase-based and hierarchical-phrase-based translation. As auxiliary results, we also compare different syntactic parsers and alignment techniques that we tested in the process of developing the decoder.

Travatar is available under the LGPL at http://phontron.com/travatar

1 Introduction

One of the recent trends in statistical machine translation (SMT) is the popularity of models that use syntactic information to help solve problems of long-distance reordering between the source and target language text. These techniques can be broadly divided into pre-ordering techniques, which first parse and reorder the source sentence into the target order before translating (Xia and McCord, 2004; Isozaki et al., 2010b), and tree-based decoding techniques, which take a tree or forest as input and choose the reordering and translation jointly (Yamada and Knight, 2001; Liu et al., 2006; Mi et al., 2008). While pre-ordering is not able to consider both translation and reordering in a joint model, it is useful in that it is done before the actual translation process, so it can be performed with a conventional translation pipeline using a standard phrase-based decoder such as Moses (Koehn et al., 2007). For tree-to-string systems, on the other hand, it is necessary to have available or create a decoder that is equipped with this functionality, which becomes a bottleneck in the research and development process.

In this demo paper, we describe Travatar, an open-source tree-to-string or forest-to-string translation system that can be used as a tool for translation using source-side syntax, and as a platform for research into syntax-based translation methods. In particular, compared to other decoders which mainly implement syntax-based translation in the synchronous context-free grammar (SCFG) framework (Chiang, 2007), Travatar is built upon the tree transducer framework (Graehl and Knight, 2004), a richer formalism that can help capture important distinctions between parse trees, as we show in Section 2. Travatar includes a fully documented training and testing regimen that was modeled around that of Moses, making it possible for users familiar with Moses to get started with Travatar quickly. The framework of the software is also designed to be extensible, so the toolkit is applicable for other tree-to-string transduction tasks.

In the evaluation of the decoder on English-Japanese machine translation, we perform a comparison to Moses's phrase-based, hierarchical-phrase-based, and SCFG-based tree-to-string

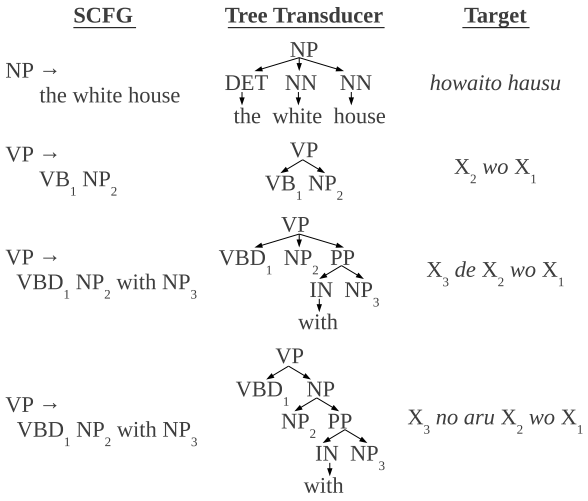

Figure 1: Tree-to-string translation rules for SCFGs and tree transducers.

translation. Based on the results, we find that tree-to-string, and particularly forest-to-string, translation using Travatar provides competitive or superior accuracy to all of these techniques. As auxiliary results, we also compare different syntactic parsers and alignment techniques that we tested in the process of developing the decoder.

2 Tree-to-String Translation

2.1 Overview

Tree-to-string translation uses syntactic information to improve translation by first parsing the source sentence, then using this source-side parse tree to decide the translation and reordering of the input. This method has several advantages, including efficiency of decoding, relatively easy handling of global reordering, and an intuitive representation of de-lexicalized rules that express general differences in order between the source and target languages. Within tree-to-string translation there are two major methodologies, synchronous context-free grammars (Chiang, 2007), and tree transducers (Graehl and Knight, 2004).

An example of tree-to-string translation rules supported by SCFGs and tree transducers is shown in Figure 1. In this example, the first rule is a simple multi-word noun phrase, the second example is an example of a delexicalized rule expressing translation from English SVO word order to Japanese SOV word order. The third and fourth examples are translations of a verb, noun phrase, and prepositional phrase, where the third rule has the preposition attatched to the verb, and the fourth has the preposition attached to the noun.

For the SCFGs, it can be seen that on the source side of the rule, there are placeholders corresponding to syntactic phrases, and on the target side of the rule there corresponding placeholders that do not have a syntactic label. On the other hand in the example of the translation rules using tree transducers, it can be seen that similar rules can be expressed, but the source rules are richer than simple SCFG rules, also including the internal structure of the parse tree. This internal structure is important for achieving translation results faithful to the input parse. In particular, the third and fourth rules show an intuitive example in which this internal structure can be important for translation. Here the full tree structures demonstrate important differences in the attachment of the prepositional phrase to the verb or noun. While this is one of the most difficult and important problems in syntactic parsing, the source side in the SCFG is identical, losing the ability to distinguish between the very information that parsers are designed to disambiguate.

In traditional tree-to-string translation methods, the translator uses a single one-best parse tree output by a syntactic parser, but parse errors have the potential to degrade the quality of translation. An important advance in tree-to-string translation that helps ameliorate this difficulity is forest-to-string translation, which represents a large number of potential parses as a packed forest, allowing the translator to choose between these parses during the process of translation (Mi et al., 2008).

2.2 The State of Open Source Software

There are a number of open-source software packages that support tree-to-string translation in the SCFG framework. For example, Moses (Koehn et al., 2007) and NiuTrans (Xiao et al., 2012) support the annotation of source-side syntactic labels, and taking parse trees (or in the case of NiuTrans, forests) as input.

There are also a few other decoders that support other varieties of using source-side syntax to help improve translation or global reordering. For example, the cdec decoder (Dyer et al., 2010) supports the context-free-reordering/finite-state-translation framework described by Dyer and Resnik (2010). The Akamon decoder (Wu et al., 2012) supports translation using head-driven

phrase structure grammars as described by Wu et al. (2010).

However, to our knowledge, while there is a general-purpose tool for tree automata in general (May and Knight, 2006), there is no open-source toolkit implementing the SMT pipeline in the tree transducer framework, despite it being a target of active research (Graehl and Knight, 2004; Liu et al., 2006; Huang et al., 2006; Mi et al., 2008).

3 The Travatar Machine Translation Toolkit

In this section, we describe the overall framework of the Travatar decoder, following the order of the training pipeline.

3.1 Data Preprocessing

This consists of parsing the source side sentence and tokenizing the target side sentences. Travatar can decode input in the bracketed format of the Penn Treebank, or also in forest format. There is documentation and scripts for using Travatar with several parsers for English, Chinese, and Japanese included with the toolkit.

3.2 Training

Once the data has been pre-processed, a tree-to-string model can be trained with the training pipeline included in the toolkit. Like the training pipeline for Moses, there is a single script that performs alignment, rule extraction, scoring, and parameter initialization. Language model training can be performed using a separate toolkit, and instructions are provided in the documentation.

For word alignment, the Travatar training pipeline is integrated with GIZA++ (Och and Ney, 2003) by default, but can also use alignments from any other aligner.

Rule extraction is performed using the GHKM algorithm (Galley et al., 2006) and its extension to rule extraction from forests (Mi and Huang, 2008). There are also a number of options implemented, including rule composition, attachment of null-aligned target words at either the highest point in the tree, or at every possible position, and left and right binarization (Galley et al., 2006; Wang et al., 2007).

Rule scoring uses a standard set of forward and backward conditional probabilities, lexicalized translation probabilities, phrase frequency, and word and phrase counts. Rule scores are stored as sparse vectors by default, which allows for scoring using an arbitrarily large number of feature functions.

3.3 Decoding

Given a translation model Travatar is able to decode parsed input sentences to generate translations. The decoding itself is performed using the bottom-up forest-to-string decoding algorithm of Mi et al. (2008). Beam-search implemented using cube pruning (Chiang, 2007) is used to adjust the trade-off between search speed and translation accuracy.

The source side of the translation model is stored using a space-efficient trie data structure (Yata, 2012) implemented using the marisa-trie toolkit.[1] Rule lookup is performed using left-to-right depth-first search, which can be implemented as prefix lookup in the trie for efficient search.

The language model storage uses the implementation in KenLM (Heafield, 2011), and particularly the implementation that maintains left and right language model states for syntax-based MT (Heafield et al., 2011).

3.4 Tuning and Evaluation

For tuning the parameters of the model, Travatar natively supports minimum error rate training (MERT) (Och, 2003) and is extension to hypergraphs (Kumar et al., 2009). This tuning can be performed for evaluation measures including BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010a), with an easily extendable interface that makes it simple to support other measures.

There is also a preliminary implementation of online learning methods such as the structured perceptron algorithm (Collins, 2002), and regularized structured SVMs trained using FOBOS (Duchi and Singer, 2009). There are plans to implement more algorithms such as MIRA or AROW (Chiang, 2012) in the near future.

The Travatar toolkit also provides an evaluation program that can calculate the scores of translation output according to various evaluation measures, and calculate the significance of differences between systems using bootstrap resampling (Koehn, 2004).

[1] `http://marisa-trie.googlecode.com`

4 Experiments

4.1 Experimental Setup

In our experiments, we validated the performance of the translation toolkit on English-Japanese translation of Wikipedia articles, as specified by the Kyoto Free Translation Task (KFTT) (Neubig, 2011). Training used the 405k sentences of training data of length under 60, tuning was performed on the development set, and testing was performed on the test set using the BLEU and RIBES measures. As baseline systems we use the Moses[2] implementation of phrase-based (MOSES-PBMT), hierarchical phrase-based (MOSES-HIER), and tree-to-string translation (MOSES-T2S). The phrase-based and hierarchical phrase-based models were trained with the default settings according to tutorials on each web site.

For all systems, we use a 5-gram Kneser-Ney smoothed language model. Alignment for each system was performed using either GIZA++[3] or Nile[4] with main results reported for the aligner that achieved the best accuracy on the dev set, and a further comparison shown in the auxiliary experiments in Section 4.3. Tuning was performed with minimum error rate training to maximize BLEU over 200-best lists. Tokenization was performed with the Stanford tokenizer for English, and the KyTea word segmenter (Neubig et al., 2011) for Japanese.

For all tree-to-string systems we use Egret[5] as an English parser, as we found it to achieve high accuracy, and it allows for the simple output of forests. Rule extraction was performed using one-best trees, which were right-binarized, and lower-cased post-parsing. For Travatar, composed rules of up to size 4 and a maximum of 2 non-terminals and 7 terminals for each rule were used. Null-aligned words were only attached to the top node, and no count normalization was performed, in contrast to Moses, which performs count normalization and exhaustive null word attachment. Decoding was performed over either one-best trees (TRAV-T2S), or over forests including all edges included in the parser 200-best list (TRAV-F2S), and a pop limit of 1000 hypotheses was used for cube pruning.

	BLEU	RIBES	Rules	Sent/s.
MOSES-PBMT	22.27	68.37	10.1M	5.69
MOSES-HIER	22.04	70.29	34.2M	1.36
MOSES-T2S	**23.81**	72.01	52.3M	1.71
TRAV-T2S	23.15	72.32	9.57M	3.29
TRAV-F2S	**23.97**	**73.27**	9.57M	1.11

Table 1: Translation results (BLEU, RIBES), rule table size, and speed in sentences per second for each system. Bold numbers indicate a statistically significant difference over all other systems (bootstrap resampling with $p > 0.05$) (Koehn, 2004).

4.2 System Comparison

The comparison between the systems is shown in Table 1. From these results we can see that the systems utilizing source-side syntax significantly outperform the PBMT and Hiero, validating the usefulness of source side syntax on the English-to-Japanese task. Comparing the two tree-to-string sytems, we can see that TRAV-T2S has slightly higher RIBES and slightly lower BLEU than MOSES-T2S. One reason for the slightly higher BLEU of MOSES-T2S is because Moses's rule extraction algorithm is more liberal in its attachment of null-aligned words, resulting in a much larger rule table (52.3M rules vs. 9.57M rules) and memory footprint. In this setting, TRAV-T2S is approximately two times faster than MOSES-T2S. When using forest based decoding in TRAV-F2S, we see significant gains in accuracy over TRAV-T2S, with BLEU slightly and RIBES greatly exceeding that of MOSES-T2S.

4.3 Effect of Alignment/Parsing

In addition, as auxiliary results, we present a comparison of Travatar's tree-to-string and forest-to-string systems using different alignment methods and syntactic parsers to examine the results on translation (Table 2).

For parsers, we compared Egret with the Stanford parser.[6] While we do not have labeled data to calculate parse accuracies with, Egret is a clone of the Berkeley parser, which has been reported to achieve higher accuracy than the Stanford parser on several domains (Kummerfeld et al., 2012). From the translation results, we can see that STAN-

[2] http://statmt.org/moses/

[3] http://code.google.com/p/giza-pp/

[4] http://code.google.com/p/nile/ As Nile is a supervised aligner, we trained it on the alignments provided with the KFTT.

[5] http://code.google.com/p/egret-parser/

[6] http://nlp.stanford.edu/software/lex-parser.shtml

	GIZA++		Nile	
	BLEU	RIBES	BLEU	RIBES
MOSES-PBMT	22.28	68.37	22.37	68.43
MOSES-HIER	22.05	70.29	21.77	69.31
STAN-T2S	21.47	70.94	22.44	72.02
EGRET-T2S	22.82	71.90	23.15	72.32
EGRET-F2S	23.35	71.77	**23.97**	**73.27**

Table 2: Translation results (BLEU, RIBES), for several translation models (PBMT, Hiero, T2S, F2S), aligners (GIZA++, Nile), and parsers (Stanford, Egret).

T2S significantly underperforms EGRET-T2S, confirming that the effectiveness of the parser plays a large effect on the translation accuracy.

Next, we compared the unsupervised aligner GIZA++, with the supervised aligner Nile, which uses syntactic information to improve alignment accuracy (Riesa and Marcu, 2010). We held out 10% of the hand aligned data provided with the KFTT, and found that GIZA++ achieves 58.32% alignment F-measure, while Nile achieves 64.22% F-measure. With respect to translation accuracy, we found that for translation that does not use syntactic information, improvements in alignment do not necessarily increase translation accuracy, as has been noted by Ganchev et al. (2008). However, for all tree-to-string systems, the improved alignments result in significant improvements in accuracy, showing that alignments are, in fact, important in our syntax-driven translation setup.

5 Conclusion and Future Directions

In this paper, we introduced Travatar, an open-source toolkit for forest-to-string translation using tree transducers. We hope this decoder will be useful to the research community as a test-bed for forest-to-string systems. The software is already sufficiently mature to be used as is, as evidenced by the competitive, if not superior, results in our English-Japanese evaluation.

We have a number of plans for future development. First, we plan to support advanced rule extraction techniques, such as fuller support for count regularization and forest-based rule extraction (Mi and Huang, 2008), and using the EM algorithm to choose attachments for null-aligned words (Galley et al., 2006) or the direction of rule binarization (Wang et al., 2007). We also plan to incorporate advances in decoding to improve search speed (Huang and Mi, 2010). In addition, there is a preliminary implementation of the ability to introduce target-side syntactic information, either through hard constraints as in tree-to-tree translation systems (Graehl and Knight, 2004), or through soft constraints, as in syntax-augmented machine translation (Zollmann and Venugopal, 2006). Finally, we will provide better support of parallelization through the entire pipeline to increase the efficiency of training and decoding.

Acknowledgements: We thank Kevin Duh and an anonymous reviewer for helpful comments.

References

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

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *Journal of Machine Learning Research*, pages 1159–1187.

Michael Collins. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. HLT-NAACL*.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12.

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 *Proc. ACL*, pages 961–968.

Kuzman Ganchev, João V. Graça, and Ben Taskar. 2008. Better alignments = better translations? In *Proc. ACL*.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT*, pages 105–112.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proc. IWSLT*.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proc. WMT*, pages 187–197.

Liang Huang and Haitao Mi. 2010. Efficient incremental decoding for tree-to-string translation. In *Proc. 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.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *Proc. EMNLP*, pages 944–952.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head finalization: A simple reordering rule for SOV languages. In *Proc. WMT and MetricsMATR*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

Shankar Kumar, Wolfgang Macherey, Chris Dyer, and Franz Och. 2009. Efficient minimum error rate training and minimum Bayes-risk decoding for translation hypergraphs and lattices. In *Proc. ACL*, pages 163–171.

Jonathan K Kummerfeld, David Hall, James R Curran, and Dan Klein. 2012. Parser showdown at the wall street corral: an empirical investigation of error types in parser output. In *Proc. EMNLP*, pages 1048–1059.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. ACL*.

Jonathan May and Kevin Knight. 2006. Tiburon: A weighted tree automata toolkit. In *Implementation and Application of Automata*, pages 102–113. Springer.

Haitao Mi and Liang Huang. 2008. Forest-based translation rule extraction. In *Proc. EMNLP*, pages 206–214.

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

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proc. ACL*, pages 529–533, Portland, USA, June.

Graham Neubig. 2011. The Kyoto free translation task. `http://www.phontron.com/kftt`.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*.

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

Jason Riesa and Daniel Marcu. 2010. Hierarchical search for word alignment. In *Proc. ACL*, pages 157–166.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP*, pages 746–754.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2010. Fine-grained tree-to-string translation rule extraction. In *Proc. ACL*, pages 325–334.

Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Akamon: An open source toolkit for tree/forest-based statistical machine translation. In *Proceedings of the ACL 2012 System Demonstrations*, pages 127–132.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. COLING*.

Tong Xiao, Jingbo Zhu, Hao Zhang, and Qiang Li. 2012. Niutrans: An open source toolkit for phrase-based and syntax-based machine translation. In *Proceedings of the ACL 2012 System Demonstrations*, pages 19–24.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*.

Susumu Yata. 2012. Dictionary compression using nested prefix/Patricia tries (in Japanese). In *Proc. 17th NLP*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. WMT*.