NiuTrans: An Open Source Toolkit for Phrase-based and Syntax-based Machine Translation

Tong Xiao[†‡], Jingbo Zhu[†‡], Hao Zhang[†] and Qiang Li[†]

[†]Natural Language Processing Lab, Northeastern University
[‡]Key Laboratory of Medical Image Computing, Ministry of Education
{xiaotong,zhujingbo}@mail.neu.edu.cn
{zhanghao1216,liqiangneu}@gmail.com

Abstract

We present a new open source toolkit for phrase-based and syntax-based machine translation. The toolkit supports several state-of-the-art models developed in statistical machine translation, including the phrase-based model, the hierachical phrase-based model, and various syntax-based models. The key innovation provided by the toolkit is that the decoder can work with various grammars and offers different choices of decoding algrithms, such as phrase-based decoding, decoding as parsing/tree-parsing and forest-based decoding. Moreover, several useful utilities were distributed with the toolkit, including a discriminative reordering model, a simple and fast language model, and an implementation of minimum error rate training for weight tuning.

1 Introduction

We present NiuTrans, a new open source machine translation toolkit, which was developed for constructing high quality machine translation systems. The NiuTrans toolkit supports most statistical machine translation (SMT) paradigms developed over the past decade, and allows for training and decoding with several state-of-the-art models, including: the phrase-based model (Koehn et al., 2003), the hierarchical phrase-based model (Chiang, 2007), and various syntax-based models (Galley et al., 2004; Liu et al., 2006). In particular, a unified framework was adopted to decode with different models and ease the implementation of decoding algorithms. Moreover, some useful utilities were distributed with the toolkit, such as: a discriminative reordering model, a simple and fast language model, and an implementation of minimum error rate training that allows for various evaluation metrics for tuning the system. In addition, the toolkit provides easy-to-use APIs for the development of new features. The toolkit has been used to build translation systems that have placed well at recent MT evaluations, such as the NTCIR-9 Chinese-to-English PatentMT task (Goto et al., 2011).

We implemented the toolkit in C++ language, with special consideration of extensibility and efficiency. C++ enables us to develop efficient translation engines which have high running speed for both training and decoding stages. This property is especially important when the programs are used for large scale translation. While the development of C++ program is slower than that of the similar programs written in other popular languages such as Java, the modern compliers generally result in C++ programs being consistently faster than the Java-based counterparts.

The toolkit is available under the GNU general public license [1]. The website of NiuTrans is http://www.nlplab.com/NiuPlan/NiuTrans.html.

2 Motivation

As in current approaches to statistical machine translation, NiuTrans is based on a log-linear

[1] http://www.gnu.org/licenses/gpl-2.0.html

Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics, pages 19–24,
Jeju, Republic of Korea, 8-14 July 2012.

model where a number of features are defined to model the translation process. Actually NiuTrans is not the first system of this kind. To date, several open-source SMT systems (based on either phrase-based models or syntax-based models) have been developed, such as Moses (Koehn et al., 2007), Joshua (Li et al., 2009), SAMT (Zollmann and Venugopal, 2006), Phrasal (Cer et al., 2010), cdec (Dyer et al., 2010), Jane (Vilar et al., 2010) and SilkRoad [2], and offer good references for the development of the NiuTrans toolkit. While our toolkit includes all necessary components as provided within the above systems, we have additional goals for this project, as follows:

- *It fully supports most state-of-the-art SMT models*. Among these are: the phrase-based model, the hierarchical phrase-based model, and the syntax-based models that explicitly use syntactic information on either (both) source and (or) target language side(s).
- *It offers a wide choice of decoding algorithms*. For example, the toolkit has several useful decoding options, including: standard phrase-based decoding, decoding as parsing, decoding as tree-parsing, and forest-based decoding.
- *It is easy-to-use and fast*. A new system can be built using only a few commands. To control the system, users only need to modify a configuration file. In addition to the special attention to usability, the running speed of the system is also improved in several ways. For example, we used several pruning and multithreading techniques to speed-up the system.

3 Toolkit

The toolkit serves as an end-to-end platform for training and evaluating statistical machine translation models. To build new translation systems, all you need is a collection of word-aligned sentences [3], and a set of additional sentences with one or more reference translations for weight tuning and test. Once the data is prepared, the MT system can be created using a sequence of commands. Given a number of sentence-pairs and the word alignments between them, the toolkit first extracts a phrase table and two reordering models for the phrase-based system, or a Synchronous Context-free/Tree-substitution Grammar (SCFG/STSG) for the hierarchical phrase-based and syntax-based systems. Then, an *n*-gram language model is built on the target-language corpus. Finally, the resulting models are incorporated into the decoder which can automatically tune feature weights on the development set using minimum error rate training (Och, 2003) and translate new sentences with the optimized weights.

In the following, we will give a brief review of the above components and the main features provided by the toolkit.

3.1 Phrase Extraction and Reordering Model

We use a standard way to implement the phrase extraction module for the phrase-based model. That is, we extract all phrase-pairs that are consistent with word alignments. Five features are associated with each phrase-pair. They are two phrase translation probabilities, two lexical weights, and a feature of phrase penalty. We follow the method proposed in (Koehn et al., 2003) to estimate the values of these features.

Unlike previous systems that adopt only one reordering model, our toolkit supports two different reordering models which are trained independently but jointly used during decoding.

- The first of these is a *discriminative reordering model*. This model is based on the standard framework of maximum entropy. Thus the reordering problem is modeled as a classification problem, and the reordering probability can be efficiently computed using a (log-)linear combination of features. In our implementation, we use all boundary words as features which are similar to those used in (Xiong et al., 2006).
- The second model is the *MSD reordering model*[4] which has been successfully used in the Moses system. Unlike Moses, our toolkit supports both the word-based and phrase-based methods for estimating the

[2] http://www.nlp.org.cn/project/project.php?proj_id=14

[3] To obtain word-to-word alignments, several easy-to-use toolkits are available, such as GIZA++ and Berkeley Aligner.

[4] Term MSD refers to the three orientations (reordering types), including Monotone (M), Swap (S), and Discontinuous (D).

probabilities of the three orientations (Galley and Manning, 2008).

3.2 Translation Rule Extraction

For the hierarchical phrase-based model, we follow the general framework of SCFG where a grammar rule has three parts – a source-side, a target-side and alignments between source and target non-terminals. To learn SCFG rules from word-aligned sentences, we choose the algorithm proposed in (Chiang, 2007) and estimate the associated feature values as in the phrase-based system.

For the syntax-based models, all non-terminals in translation rules are annotated with syntactic labels. We use the GHKM algorithm to extract (minimal) translation rules from bilingual sentences with parse trees on source-language side and/or target-language side[5]. Also, two or more minimal rules can be composed together to obtain larger rules and involve more contextual information. For unaligned words, we attach them to all nearby rules, instead of using the most likely attachment as in (Galley et al., 2006).

3.3 *N*-gram Language Modeling

The toolkit includes a simple but effective *n*-gram language model (LM). The LM builder is basically a "sorted" trie structure (Pauls and Klein, 2011), where a map is developed to implement an array of key/value pairs, guaranteeing that the keys can be accessed in sorted order. To reduce the size of resulting language model, low-frequency *n*-grams are filtered out by some thresholds. Moreover, an *n*-gram cache is implemented to speed up *n*-gram probability requests for decoding.

3.4 Weight Tuning

We implement the weight tuning component according to the minimum error rate training (MERT) method (Och, 2003). As MERT suffers from local optimums, we added a small program into the MERT system to let it jump out from the coverage area. When MERT converges to a (local) optimum, our program automatically conducts the MERT run again from a random starting point near the newly-obtained optimal point. This procedure is repeated for several times until no better weights (i.e., weights with a higher BLEU score) are found. In this way, our program can introduce some randomness into weight training. Hence users do not need to repeat MERT for obtaining stable and optimized weights using different starting points.

3.5 Decoding

Chart-parsing is employed to decode sentences in development and test sets. Given a source sentence, the decoder generates 1-best or *k*-best translations in a bottom-up fashion using a CKY-style parsing algorithm. The basic data structure used in the decoder is a *chart*, where an array of *cells* is organized in topological order. Each cell maintains a list of hypotheses (or *items*). The decoding process starts with the minimal cells, and proceeds by repeatedly applying translation rules or composing items in adjunct cells to obtain new items. Once a new item is created, the associated scores are computed (with an integrated *n*-gram language model). Then, the item is added into the list of the corresponding cell. This procedure stops when we reach the final state (i.e., the cell associates with the entire source span).

The decoder can work with all (hierarchical) phrase-based and syntax-based models. In particular, our toolkit provides the following decoding modes.

- *Phrase-based decoding*. To fit the phrase-based model into the CKY paring framework, we restrict the phrase-based decoding with the ITG constraint (Wu, 1996). In this way, each pair of items in adjunct cells can be composed in either monotone order or inverted order. Hence the decoding can be trivially implemented by a three-loop structure as in standard CKY parsing. This algorithm is actually the same as that used in parsing with bracketing transduction grammars.
- *Decoding as parsing (or string-based decoding)*. This mode is designed for decoding with SCFGs/STSGs which are used in the hierarchical phrase-based and syntax-based systems. In the general framework of synchronous grammars and tree transducers, decoding can be regarded as a parsing problem. Therefore, the above chart-based decoder is directly applicable to

[5] For tree-to-tree models, we use a natural extension of the GHKM algorithm which defines admissible nodes on tree-pairs and obtains tree-to-tree rules on all pairs of source and target tree-fragments.

the hierarchical phrase-based and syntax-based models. For efficient integration of *n*-gram language model into decoding, rules containing more than two variables are binarized into binary rules. In addition to the rules learned from bilingual data, glue rules are employed to glue the translations of a sequence of chunks.

- *Decoding as tree-parsing (or tree-based decoding)*. If the parse tree of source sentence is provided, decoding (for tree-to-string and tree-to-tree models) can also be cast as a tree-parsing problem (Eisner, 2003). In tree-parsing, translation rules are first mapped onto the nodes of input parse tree. This results in a translation tree/forest (or a hypergraph) where each edge represents a rule application. Then decoding can proceed on the hypergraph as usual. That is, we visit in bottom-up order each node in the parse tree, and calculate the model score for each edge rooting at the node. The final output is the 1-best/*k*-best translations maintained by the root node of the parse tree. Since tree-parsing restricts its search space to the derivations that exactly match with the input parse tree, it in general has a much higher decoding speed than a normal parsing procedure. But it in turn results in lower translation quality due to more search errors.
- *Forest-based decoding*. Forest-based decoding (Mi et al., 2008) is a natural extension of tree-based decoding. In principle, forest is a data structure that can encode exponential number of trees efficiently. This structure has been proved to be helpful in reducing the effects caused by parser errors. Since our internal representation is already in a hypergraph structure, it is easy to extend the decoder to handle the input forest, with little modification of the code.

4 Other Features

In addition to the basic components described above, several additional features are introduced to ease the use of the toolkit.

4.1 Multithreading

The decoder supports multithreading to make full advantage of the modern computers where more than one CPUs (or cores) are provided. In general, the decoding speed can be improved when multiple threads are involved. However, modern MT decoders do not run faster when too many threads are used (Cer et al., 2010).

4.2 Pruning

To make decoding computational feasible, *beam pruning* is used to aggressively prune the search space. In our implementation, we maintain a beam for each cell. Once all the items of the cell are proved, only the top-*k* best items according to model score are kept and the rest are discarded. Also, we re-implemented the *cube pruning* method described in (Chiang, 2007) to further speed-up the system.

In addition, we develop another method that prunes the search space using punctuations. The idea is to divide the input sentence into a sequence of segments according to punctuations. Then, each segment is translated individually. The MT outputs are finally generated by composing the translations of those segments.

4.3 APIs for Feature Engineering

To ease the implementation and test of new features, the toolkit offers APIs for experimenting with the features developed by users. For example, users can develop new features that are associated with each phrase-pair. The system can automatically recognize them and incorporate them into decoding. Also, more complex features can be activated during decoding. When an item is created during decoding, new features can be introduced into an internal object which returns feature values for computing the model score.

5 Experiments

5.1 Experimental Setup

We evaluated our systems on NIST Chinese-English MT tasks. Our training corpus consists of 1.9M bilingual sentences. We used GIZA++ and the "grow-diag-final-and" heuristics to generate word alignment for the bilingual data. The parse trees on both the Chinese and English sides were

Entry			BLEU4[%]	
			Dev	Test
Moses: phrase			36.51	34.93
Moses: hierarchical phrase			36.65	34.79
NiuTrans	phrase		36.99	35.29
	hierarchical phrase		37.41	35.35
	t2s	parsing	36.48	34.71
		tree-parsing	35.54	33.99
		forest-based	36.14	34.25
	t2t	parsing	35.99	34.01
		tree-parsing	35.04	33.21
		forest-based	35.56	33.45
	s2t	parsing	37.63	35.65

Table 1: BLEU scores of various systems. t2s, t2t, and s2t represent the tree-to-string, tree-to-tree, and string-to-tree systems, respectively.

generated using the Berkeley Parser, which were then binarized in a head-out fashion [6]. A 5-gram language model was trained on the Xinhua portion of the Gigaword corpus in addition to the English part of the LDC bilingual training data. We used the NIST 2003 MT evaluation set as our development set (919 sentences) and the NIST 2005 MT evaluation set as our test set (1,082 sentences). The translation quality was evaluated with the case-insensitive IBM-version BLEU4.

For the phrase-based system, phrases are of at most 7 words on either source or target-side. For the hierarchical phrase-based system, all SCFG rules have at most two variables. For the syntax-based systems, minimal rules were extracted from the binarized trees on both (either) language-side(s). Larger rules were then generated by composing two or three minimal rules. By default, all these systems used a beam of size 30 for decoding.

5.2 Evaluation of Translations

Table 1 shows the BLEU scores of different MT systems built using our toolkit. For comparison, the result of the Moses system is also reported. We see, first of all, that our phrase-based and hierarchical phrase-based systems achieve competitive performance, even outperforms the Moses system over 0.3 BLEU points in some cases. Also, the syntax-based systems obtain very promising results. For example, the string-to-tree system significantly outperforms the phrase-based and hierarchical phrase-based counterparts. In addition, Table 1 gives a test of different decoding methods (for syntax-based systems). We see that the parsing-based method achieves the best BLEU score. On the other hand, as expected, it runs slowest due to its large search space. For example, it is 5-8 times slower than the tree-parsing-based method in our experiments. The forest-based decoding further improves the BLEU scores on top of tree-parsing. In most cases, it obtains a +0.6 BLEU improvement but is 2-3 times slower than the tree-parsing-based method.

[6] The parse trees follow the nested bracketing format, as defined in the Penn Treebank. Also, the NiuTrans package includes a tool for tree binarization.

Entry	BLEU4[%]		Speed (sent/sec)
	Dev	Test	
Moses: phrase	36.69	34.99	0.11
+ cube pruning	36.51	34.93	0.47
NiuTrans: phrase	37.14	35.47	0.14
+ cube pruning	36.98	35.39	0.60
+ cube & punct pruning	36.99	35.29	3.71
+ all pruning & 8 threads	36.99	35.29	21.89
+ all pruning & 16 threads	36.99	35.29	22.36

Table 2: Effects of pruning and multithreading techniques.

5.3 System Speed-up

We also study the effectiveness of pruning and multithreading techniques. Table 2 shows that all the pruning methods implemented in the toolkit is helpful in speeding up the (phrase-based) system, while does not result in significant decrease in BLEU score. On top of a straightforward baseline (only beam pruning is used), cube pruning and pruning with punctuations give a speed improvement of 25 times together[7]. Moreover, the decoding process can be further accelerated by using multithreading technique. However, more than 8 threads do not help in our experiments.

6 Conclusion and Future Work

We have presented a new open-source toolkit for phrase-based and syntax-based machine translation. It is implemented in C++ and runs fast. Moreover, it supports several state-of-the-art models ranging from phrase-based models to syntax-based models,

[7] The translation speed is tested on Intel Core Due 2 E8500 processors running at 3.16 GHz.

and provides a wide choice of decoding methods. The experimental results on NIST MT tasks show that the MT systems built with our toolkit achieve state-of-the-art translation performance.

The next version of NiuTrans will support ARPA-format LMs, MIRA for weight tuning and a beam-stack decoder which removes the ITG constraint for phrase decoding. In addition, a Hadoop-based MapReduce-parallelized version is underway and will be released in near future.

Acknowledgments

This research was supported in part by the National Science Foundation of China (61073140), the Specialized Research Fund for the Doctoral Program of Higher Education (20100042110031) and the Fundamental Research Funds for the Central Universities in China.

References

Daniel Cer, Michel Galley, Daniel Jurafsky and Christopher D. Manning. 2010. Phrasal: A Toolkit for Statistical Machine Translation with Facilities for Extraction and Incorporation of Arbitrary Model Features. In *Proc. of* HLT/NAACL 2010 demonstration Session, pages 9-12.

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

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, Philip Resnik. 2010. cdec: A Decoder, Alignment, and Learning Framework for Finite-State and Context-Free Translation Models. In *Proc. of* ACL 2010 System Demonstrations, pages 7-12.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proc. of* ACL 2003, pages 205-208.

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

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang and Ignacio Thayer. 2006. Scalable inferences and training of context-rich syntax translation models. In *Proc. of* COLING/ACL 2006, pages 961-968.

Michel Galley and Christopher D. Manning. 2008. A Simple and Effective Hierarchical Phrase Reordering Model. In *Proc. of* EMNLP2008, pages 848-856.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita and Benjamin K. Tsou. 2011. Overview of the Patent Machine Translation Task at the NTCIR-9 Workshop. In *Proc. of* NTCIR-9 Workshop Meeting, pages 559-578.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of* HLT/NAACL 2003, pages 127-133.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of* ACL 2007, pages 177–180.

Zhifei Li, Chris Callison-Burch, Chris Dyer, Sanjeev Khudanpur, Lane Schwartz, Wren Thornton, Jonathan Weese, and Omar Zaidan. 2009. Joshua: An Open Source Toolkit for Parsing-Based Machine Translation. In *Proc. of* the Workshop on Statistical Machine Translation, pages 135–139.

Yang Liu, Qun Liu and Shouxun Lin. 2006. Tree-to-String Alignment Template for Statistical Machine Translation. In *Proc. of* ACL 2006, pages 609-616.

Haitao Mi, Liang Huang and Qun Liu. 2008. Forest-Based Translation. In *Proc. of* ACL 2008, pages 192-199.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. of* ACL 2003, pages 160-167.

Adam Pauls and Dan Klein. 2011. Faster and Smaller N-Gram Language Models. In *Proc. of* ACL 2011, pages 258–267.

David Vilar, Daniel Stein, Matthias Huck and Hermann Ney. 2010. Jane: Open Source Hierarchical Translation, Extended with Reordering and Lexicon Models. In *Proc. of* the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR, pages 262-270.

Dekai Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of* ACL1996, pages 152–158.

Deyi Xiong, Qun Liu and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proc. of* ACL 2006, pages 521-528.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax Augmented Machine Translation via Chart Parsing. In *Proc. of* HLT/NAACL 2006, pages 138-141.