KyotoEBMT System Description for the 2nd Workshop on Asian Translation

John Richardson[†] **Raj Dabre**[†] **Chenhui Chu**[‡]
Fabien Cromières[‡] **Toshiaki Nakazawa**[‡] **Sadao Kurohashi**[†]
[†]Graduate School of Informatics, Kyoto University, Kyoto 606-8501
[‡]Japan Science and Technology Agency, Kawaguchi-shi, Saitama 332-0012
{john, dabre}@nlp.ist.i.kyoto-u.ac.jp, {fabien, nakazawa, chu}@pa.jst.jp,
kuro@i.kyoto-u.ac.jp

Abstract

This paper introduces the KyotoEBMT example-based machine translation framework. Since last year's workshop we have replaced input trees with forests, improved alignment, added new features, and introduced bilingual neural network reranking. The major benefits of our system include online example retrieval and flexible reordering. We also use syntactic dependency analysis for both source and target languages in the hope of learning how to translate non-local structure. The system implementation (this paper refers to version 1.0) is available as open-source.

1 Introduction

This paper describes the KyotoEBMT system used in the 2nd Workshop on Asian Translation (Nakazawa et. al, 2015).

Our system is a fully-fledged Example-Based Machine Translation (EBMT) platform making use of both source-language and target-language dependency structure. This approach has been explored comparatively less in studies on syntax-based SMT/EBMT, which tend to focus on constituent trees rather than dependency trees, and on tree-to-string rather than tree-to-tree approaches. Furthermore, we employ separate dependency parsers for each language rather than projecting the dependencies from one language to another, as in (Quirk et. al, 2005).

The dependency structure information is used end-to-end: for improving the quality of the alignment of the translation examples, for constraining the translation rule extraction and for guiding the decoding. We believe that dependency structure, which considers more than just local context, is important in order to generate fluent and accurate translations of complex sentences across distant language pairs.

The experiments described in this paper focus on technical domain translation for Japanese-Chinese and Japanese-English, however our implementation is applicable to any domain and language pair for which there exist parallel sentences and dependency parsers.

A further unique characteristic of our system is that, again contrary to the majority of similar systems, it does not rely on precomputation of translation rules. Instead it matches each input sentence to the full database of translation examples before extracting translation rules online. This has the merit of maximizing the information available when creating and combining translation rules, while retaining the ability to produce excellent translations for input sentences similar to an existing translation example.

The system is mostly developed in C++ and is available as open source. The code and documentation are available from http://nlp.ist.i.kyoto-u.ac.jp/kyotoebmt/. Experiments are facilitated through the inclusion of an end-to-end experiment management system (EMS) which has been greatly improved in this version. The framework is simple to use and supports model training with multiple threads or across a cluster.

2 System Overview

Figure 1 shows the basic structure of the KyotoEBMT translation pipeline.

The training process begins with parsing and aligning parallel sentences from the training corpus. The alignments are then used to build an example database ('translation mem-

Proceedings of the 2nd Workshop on Asian Translation (WAT2015), pages 54-60,
Kyoto, Japan, 16th October 2015.

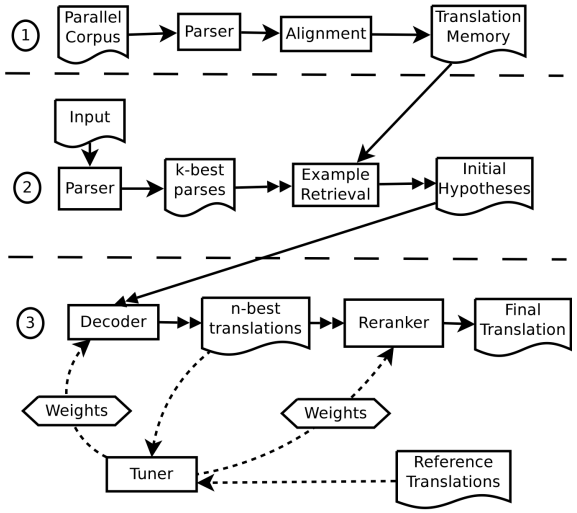

Figure 1: The translation pipeline can be roughly divided in 3 steps. Step 1 is the creation of the example database, trained from a parallel corpus. Step 2 is the parsing of an input sentence and the generation of sets of initial hypotheses. Step 3 consists in decoding and reranking. The tuning of the weights for decoding and reranking is done by a modified version of step 3.

ory') containing 'examples' or 'treelets' that form the hypotheses to be combined during decoding.

Translation is performed by first parsing an input sentence then searching for treelets matching entries in the example database. The retrieved treelets are combined by a lattice-based decoder that optimizes a log linear model score. Finally, we use a reranker to select the optimal translation from the n-best list provided by the decoder using additional non-local features (see section 3.4).

Figure 2 shows the process of combining examples matching the input tree to create an output sentence.

2.1 Example retrieval and translation hypothesis construction

An important characteristic of our system is that we do not extract and store translation rules in advance: the alignment of translation examples is performed offline. However, for a given input sentence i, the steps for finding examples partially matching i and extracting their translation hypotheses is an online process. This approach could be considered to be more faithful to the original EBMT approach advocated by Nagao (1984). It has already been proposed for phrase-based (Callison-Burch et al., 2005), hierarchical (Lopez, 2007), and syntax-based (Cromières and Kurohashi, 2011) systems. It does not however, seem to be very commonly integrated in syntax-based MT.

This approach has several benefits. The first is that we are not required to impose a limit on the size of translation hypotheses. Systems extracting rules in advance typically restrict the size and number of extracted rules for fear of becoming unmanageable. In particular, if an input sentence is the same or very similar to one of our translation examples, we will be able to retrieve a perfect translation. A second advantage is that we can make use of the full context of the example to assign features and scores to each translation hypothesis.

The main drawback of our approach is that it can be computationally more expensive to retrieve arbitrarily large matchings in the example database online than it is to match precomputed rules. We use the techniques described in Cromières and Kurohashi (2011) to perform this step as efficiently as possible.

Once we have found an example translation (s, t) for which s partially matches i, we proceed to extract a translation hypothesis from it. A translation hypothesis is defined as a generic translation rule for a part p of the input sentence that is represented as a target-language treelet, with non-terminals representing the insertion positions for the translations of other parts of the sentence. A translation hypothesis is created from a translation example as follows:

1. We project the part of s that is matched into the target side t using the alignment of s and t. This is trivial if each word of s and t is aligned, but this is not typically the case. Therefore our translation hypotheses will often have some target words/nodes marked as *optionals*: this means that we will decide if they should be added to the final translation only at the moment of combination.

2. We insert the non-terminals as child nodes of the projected subtree. This is

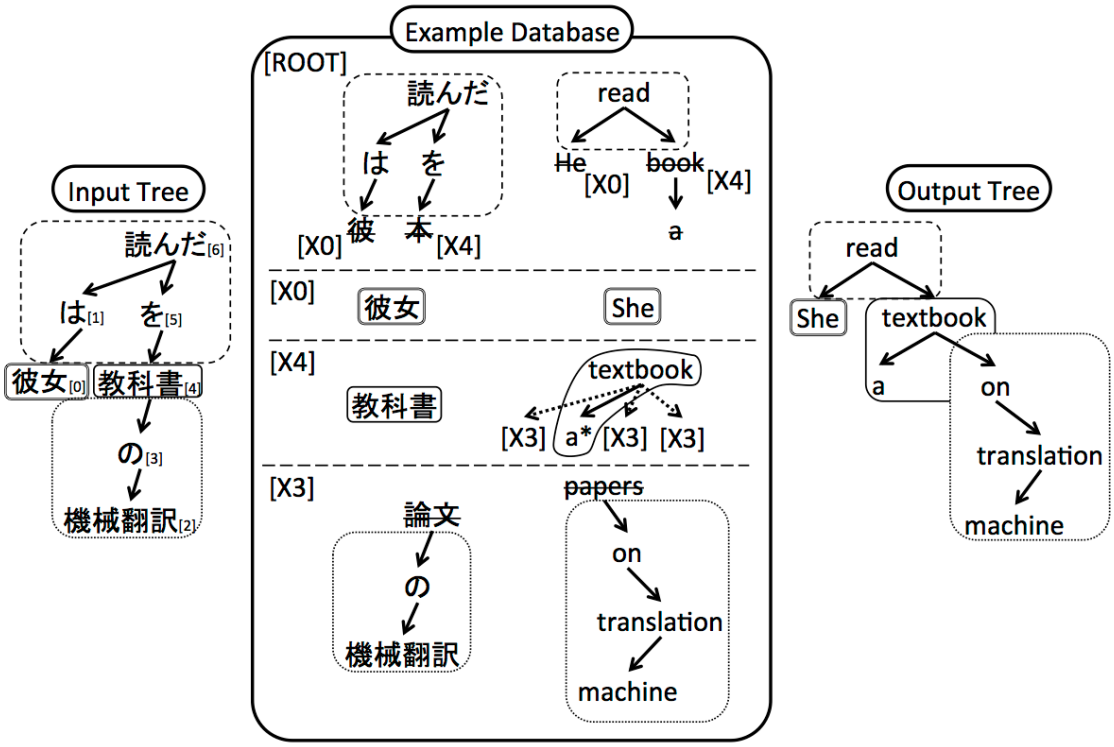

Figure 2: The process of translation. The source sentence is parsed and matching subtrees from the example database are retrieved. From the examples, we extract translation hypotheses than can contain optional target words and several position for each non-terminals. For example the translation hypothesis containing "textbook" has three possible position for the non-terminal X3 (as a left-child before "a", as a left-child after "a" or as a right-child). The translation hypotheses are then combined during decoding. Choice of optional words and final non-terminal positions is also done during decoding.

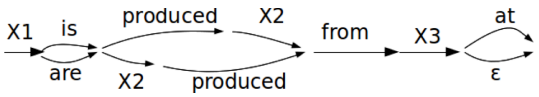

Figure 3: A translation hypothesis endoded as a lattice. This representation allows us to handle efficiently the ambiguities of our translation rules. Note that each path in this lattice corresponds to different choices of insertion position for X2, morphological forms of "be", and the optional insertion of "at".

simple if i, s and t have the same structure and are perfectly aligned, but again this is not typically the case. A consequence is that we will sometimes have *several possible insertion positions* for each non-terminal. The choice of insertion position is again made during combination.

2.2 Decoding

After having extracted translation hypotheses for as many parts of the input tree as possible, we need to decide how to select and combine them. Our approach here is similar to what has been proposed for Corpus-Based Machine Translation. We first choose a number of features and create a linear model scoring each possible combination of hypotheses (see Section 3.3). We then attempt to find the combination that maximizes this model score.

The combination of rules is constrained by the structure of the input dependency tree. If we only consider local features[1], then a simple bottom-up dynamic programming approach can efficiently find the optimal combination with linear $O(|\mathcal{H}|)$ complexity[2]. However, non-local features (such as language models)

[1]The score of a combination will be the sum of the local scores of each translation hypothesis.

[2]$\mathcal{H}$ = set of translation hypotheses

will force us to prune the search space. This pruning is done efficiently through a variation of cube-pruning (Chiang, 2007). We use KenLM[3] (Heafield, 2011) for computing the target language model score. Decoding is made more efficient by using some of the more advanced features of KenLM such as state-reduction ((Li and Khudanpur, 2008), (Heafield et al., 2011)) and rest-cost estimations(Heafield et al., 2012).

Compared with the original cube-pruning algorithm, our decoder is designed to handle an arbitrary number of non-terminals. In addition, as we have seen in Section 2.1, the translation hypotheses we initially extract from examples are ambiguous in term of which target word is going to be used and which will be the final position of each non-terminal. In order to handle such ambiguities, we use a lattice-based internal representation that can encode them efficiently (see Figure 3). This lattice representation also allows the decoder to make choices between various morphological variations of a word (e.g. be/is/are). We use the decoding algorithm described in (Cromières and Kurohashi, 2014).

3 Improvements from WAT2014

3.1 Alignment

Based on the findings of Neubig and Duh (2014), we experimented with supervised alignment using Nile (Riesa et al., 2011) as part of our translation framework. We found that using supervised alignments made a considerable improvement to translation quality. Since Nile supports only constituency parses, we also perform constituency parsing for source and target languages for generating bidirectional word alignments.

For the initial alignments for Nile, we use the alignments generated from the model described in last year's system description (Richardson et al., 2014), which makes use of our dependency parses in order to capture non-local reorderings.

3.2 Forest Input

We found that the quality of the source-side dependency parsing had an important impact on translation quality. Unfortunately, parsing errors are unavoidable. Chinese parsing is maybe especially challenging and our Chinese parser still produces a significant number of parsing errors. In order to mitigate this problem, last year we used a k-best list of input parses. We found this was somewhat successful but inefficient, and therefore have moved from a k-best list representation of multiple parses to a more compact and efficient forest representation.

In the future, we will consider also using forests for all the translation examples (and not just the input sentence).

3.3 Features

During decoding we use a linear model to score each possible combination of hypotheses. This linear model is based on a linear combination of both local features (local to each translation hypothesis) and non-local features (such as a 5-gram language model score of the final translation). Despite our already relatively large set of dense features, we found there were a number of cases where these features were not enough to differentiate between good and bad translation hypotheses.

This year we have added ten new features, now reaching a total of 52, a selection of which are shown below:

- Forest parse scores
- Number of content/function words aligned to content/function words
- Number of times a subtree is inserted in a position (left or right of parent) that is not the most common in the training data
- Number of examples sharing the same information used to create an initial hypothesis
- Similarity between source and input word embeddings (Mikolov et al., 2013)

The optimal weights for each feature are as before estimated using the implementation of k-best batch MIRA (Cherry and Foster, 2012) included in Moses.

[3]http://kheafield.com/code/kenlm/

3.4 Reranking

A final reranking step allows us to use more advanced features for selecting the best translations. We reranked the n-best output of our system using several additional language models: a standard 7-gram language model with Modified Kneser-Ney smoothing, a Recurrent Neural Network Language Model (RNNLM) (Mikolov et. al, 2010) and several variations of a Bilingual Recurrent Neural Network Language Model.

The RNNLM model was trained with hidden layer size 200, and 5000 sentences from the training fold were used as validation data.

For the bilingual language model, we used the Neural Machine Translation Model of (Bahdanau et. al, 2014) which has an open source implementation in the GroundHog/Theano framework[4]. For each language pair we trained two models, one for each translation direction. In addition, for Japanese and Chinese, we considered two types of segmentation: the segmentation produced by our morphological analyzer, and a character-level segmentation. We had therefore up to four models per language pair. Rescoring our translations with these models gave up to four additional features. It is interesting to note that although trying to directly translate our input sentences using these neural MT models typically resulted in a comparatively low BLEU score, they turned out to be useful for reranking in our system. This is probably due to the fact that, since they represent a very different approach to translation, the models tend to learn different aspects of the translation and make different mistakes to our system. Using a character-based segmentation further ensured the neural models learned a different kind of information. The models took two to four days each to train on a GPU. The settings we used were mostly the defaults of the implementations[5].

Reranking was conducted by first calculating the various language model scores for each translation in the n-best list. These features were added to those used in the first round of tuning, then one final iteration of tuning was run. The tuning algorithm and settings were the same as for standard tuning. This retuning step was added in order to find an optimal combination of the additional features with related features such as sentence length and the score given by the 5-gram language model used inside the decoder.

4 Experiments

We conducted translation experiments on the four language pairs in the scientific papers subtask: Japanese-English (JA–EN), English-Japanese (EN–JA), Japanese-Chinese (JA–ZH) and Chinese-Japanese (ZH–JA).

The proposed system used the following dependency parsers and show below their approximate parsing accuracies (micro-average), which were evaluated by hand on a random subset of sentences from the test data. The parsers were trained on domains different to those used in the experiments.

- English: NLParser[6] (92%) (Charniak and Johnson, 2005)
- Japanese: KNP (96%) (Kawahara and Kurohashi, 2006)
- Chinese: SKP (88%) (Shen et al., 2012)

For generating input for Nile we used the following constituency parsers:

- English: Berkeley Parser (Petrov et al., 2006)
- Japanese: Cyklark (Oda et al., 2015)
- Chinese: Berkeley Parser (Petrov et al., 2006)

Forests were created by packing the 200-best dependency parses for Japanese and English, and 50-best parses for Chinese.

4.1 Results

Table 1 shows the results of our proposed system (WAT15) and a comparison with the system from last year (WAT14) (Richardson

[4]https://github.com/lisa-groundhog/GroundHog

[5]More precisely, the hidden layer size was 1000. Training done with a minibatch size of 64 and the adadelta algorithm (rho = 0.95, eps = 1e-6). Vocabulary size was reduced to 20,000 for the word-segmented model. Backpropagation through time number of steps increased to up to 100 for the character-based models.

[6]Converted to dependency parses with in-house tool.

Language Pair	System	BLEU	RIBES	HUMAN
JA–EN	Baseline	18.45	64.51	n/a
	WAT15	21.31	70.65	16.50
	WAT15+Rerank	**22.89**	**72.46**	**32.50**
	WAT14	20.60	70.12	(21.50)
	WAT14+Rerank	21.07	69.90	(25.00)
EN–JA	Baseline	27.48	68.37	n/a
	WAT15	30.69	76.78	40.50
	WAT15+Rerank	**33.06**	**78.95**	**51.00**
	WAT14	29.76	75.21	(33.75)
	WAT14+Rerank	31.09	75.96	(38.00)
JA–ZH	Baseline	27.96	78.90	n/a
	WAT15	29.99	80.71	**16.00**
	WAT15+Rerank	**31.40**	**82.70**	12.50
	WAT14	27.21	79.13	(-0.75)
	WAT14+Rerank	27.67	78.83	(-8.75)
ZH–JA	Baseline	34.65	77.25	n/a
	WAT15	36.30	81.97	16.75
	WAT15+Rerank	**38.53**	**84.07**	**18.50**
	WAT14	33.57	80.10	(6.00)
	WAT14+Rerank	34.75	80.26	(7.50)

Table 1: Official evaluation results for BLEU/RIBES/HUMAN. *(NB: Human evaluation scores of WAT2014 and WAT2015 are not comparable.)*

et al., 2014) and official baseline (phrase-based SMT, for details see Nakazawa et al. (2015)). We give results for evaluation on the test set after tuning (WAT15, WAT14) and tuning plus reranking (WAT15+Rerank, WAT14+Rerank). Tuning was conducted over 10 iterations on the development set using an n-best list of length 500, and we used the 1000-best for reranking.

WAT15+Rerank was the strongest system in our comparison, outperforming the official baseline, non-reranked system (WAT15) and last year's systems in all metrics for all languages, with the minor exception of JA–ZH human evaluation for reranked vs. non-reranked.

5 Conclusion

In this paper we have described the latest version of the KyotoEBMT example-based translation system. Since last year we have improved alignment, introduced forest input, added new features and used bilingual neural network features in reranking.

In our preparation for this workshop we have focused mainly on improving Japanese–Chinese and Chinese–Japanese translation, particularly in terms of dealing with poor quality Chinese dependency parses. As future work we plan to perform more extensive error analysis on the other language pairs. We also found that despite using forest input there are still many issues caused by incorrect parsing and will consider in the future how best to overcome this.

References

Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *arXiv preprint arXiv:1409.0473*. .

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 2005. Scaling phrase-based statistical machine translation to larger corpora and longer phrases. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 255–262. Association for Computational Linguistics, 2005.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics, ACL 2005*.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *HLT-NAACL*, 2012.

David Chiang. 2007. Hierarchical phrase-based translation. In *Computational Linguistics*.

Fabien Cromières and Sadao Kurohashi. 2011. Efficient retrieval of tree translation examples for syntax-based machine translation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*.

Fabien Cromières and Sadao Kurohashi. 2014. Translation Rules with Right-Hand Side Lattices. In *Proceedings of EMNLP 2014*

Isao Goto, Ka Po Chow, Bin Lu, Eiichiro Sumita and Benjamin Tsou. 2013. Overview of the Patent Machine Translation Task at the NTCIR-10 Workshop. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies (NTCIR-10)*.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, 2011.

Kenneth Heafield, Hieu Hoang, Philipp Koehn, Tetsuo Kiso, and Marcello Federico. 2011. Left language model state for syntactic machine translation. In *Proceedings of the International Workshop on Spoken Language Translation*, 2011.

Kenneth Heafield, Philipp Koehn, and Alon Lavie. 2012. Language model rest costs and space-efficient storage. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 2012.

Daisuke Kawahara and Sadao Kurohashi. 2006. A Fully-Lexicalized Probabilistic Model for Japanese Syntactic and Case Structure Analysis. In *Proceedings of the Human Language Technology Conference of the NAACL*.

Zhifei Li and Sanjeev Khudanpur. 2008. A scalable decoder for parsing-based machine translation with equivalent language model state maintenance. In *Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation*. Association for Computational Linguistics, 2008.

Adam Lopez. 2007. Hierarchical phrase-based translation with suffix arrays. In *EMNLP-CoNLL*, 2007.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proceedings of the 11th Annual Conference of the International Speech Communication Association*, 2010.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at ICLR, 2013*.

Toshiaki Nakazawa, Hideya Mino, Isao Goto, Graham Neubig, Sadao Kurohashi and Eiichiro Sumita. 2015. Overview of the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*.

Makoto Nagao. 1984. A framework of a mechanical translation between Japanese and English by analogy principle. In *A. Elithorn and R. Banerji. Artificial and Human Intelligence*.

Toshiaki Nakazawa and Sadao Kurohashi. 2012. Alignment by bilingual generation and monolingual derivation. In *Proceedings of COLING 2012*.

Graham Neubig and Kevin Duh. 2014. On the Elements of an Accurate Tree-to-String Machine Translation System. In *The 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Yusuke Oda, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Ckylark: A More Robust PCFG-LA Parser. In *Proceedings of NAACL 2015: Demo Track*.

Slav Petrov, Leon Barrett, Romain Thibaux and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of COLING-ACL 2006*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005.

John Richardson, Fabien Cromières, Toshiaki Nakazawa, Sadao Kurohashi. 2014. KyotoEBMT System Description for the 1st Workshop on Asian Translation. In *Proceedings of the 1st Workshop on Asian Translation*.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-Rich Language-Independent Syntax-Based Alignment for Statistical Machine Translation. In *Proceedings of EMNLP 2011*.

Mo Shen, Daisuke Kawahara and Sadao Kurohashi. 2012. A Reranking Approach for Dependency Parsing with Variable-sized Subtree Features. In *Proceedings of 26th Pacific Asia Conference on Language Information and Computing*.