KyotoEBMT: An Example-Based Dependency-to-Dependency Translation Framework

John Richardson[†] **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@nlp.ist.i.kyoto-u.ac.jp, {fabien, nakazawa}@pa.jst.jp,
kuro@i.kyoto-u.ac.jp

Abstract

This paper introduces the KyotoEBMT Example-Based Machine Translation framework. Our system uses a *tree-to-tree* approach, employing syntactic dependency analysis for both source and target languages in an attempt to preserve non-local structure. The effectiveness of our system is maximized with online example matching and a flexible decoder. Evaluation demonstrates BLEU scores competitive with state-of-the-art SMT systems such as Moses. The current implementation is intended to be released as open-source in the near future.

1 Introduction

Corpus-based approaches have become a major focus of Machine Translation research. We present here a fully-fledged Example-Based Machine Translation (EBMT) platform making use of both source-language and target-language dependency structure. This paradigm has been explored comparatively less, as studies on Syntactic-based SMT/EBMT 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.

Our experiments 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 translation examples 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 incorporates a web-based translation interface for ease of use. The web interface (see Figure 1) also displays information useful for error analysis such as the list of translation examples used. Experiments are facilitated through the inclusion of a curses-based graphical interface for performing tuning and evaluation. The decoder supports multiple threads.

We are currently making preparations for the project to be released with an open-source license. The code will be available at http://nlp.ist.i.kyoto-u.ac.jp/kyotoebmt/.

2 System Overview

Figure 2 shows the basic structure of the proposed translation pipeline.

The training process begins with parsing and aligning parallel sentences from the train-

Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 79–84,
Baltimore, Maryland USA, June 23-24, 2014.

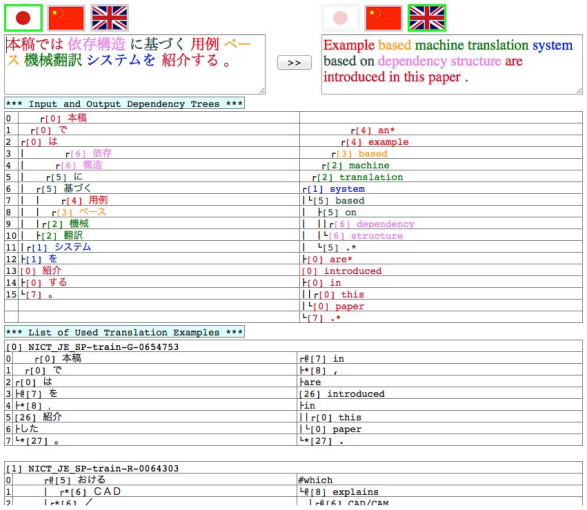

Figure 1: A screenshot of the web interface showing a Japanese-English translation. The interface provides the source and target side dependency tree, as well as the list of examples used with their alignments. The web interface facilitates easy and intuitive error analysis, and can be used as a tool for computer-aided translation.

ing corpus. Alignment uses a Bayesian subtree alignment model based on dependency trees. This contains a tree-based reordering model and can capture non-local reorderings, which sequential word-based models often cannot handle effectively. The alignments are then used to build an example database ('translation memory') 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 decoder that optimizes a log linear model score. The example retrieval and decoding steps are explained in more detail in sections 3 and 4 respectively. The choice of features and the tuning of the log linear model is described in section 5.

Figure 3 shows the process of combining examples matching the input tree to create an output sentence.

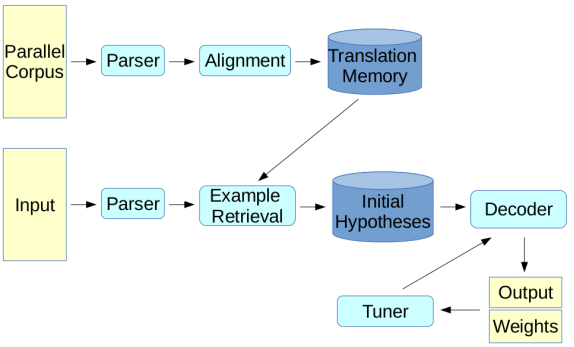

Figure 2: Translation pipeline. An example database is first trained from a parallel corpus. Translation is performed by the decoder, which combines initial hypotheses generated by the example retrieval module. Weights can be improved with batch tuning.

3 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 ex-

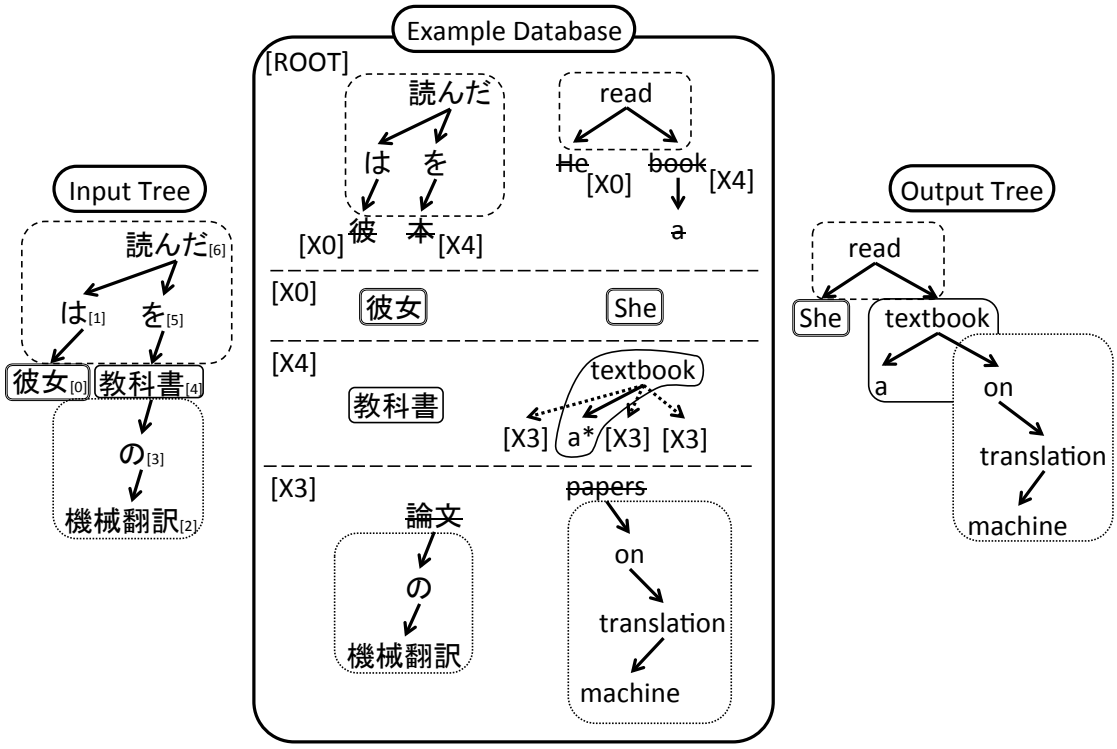

Figure 3: The process of translation. The source sentence is parsed and matching subtrees from the example database are retrieved. From the examples, we extract translation hypotheses than can contain optional target words and several position for each non-terminals. For example the translation hypothesis 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.

ample 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 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.

4 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

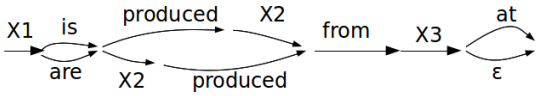

Figure 4: A translation hypothesis endoded as a lattice. This representation allows us to handle efficiently the ambiguities of our translation rules. Note that each path in this lattice corresponds to different choices of insertion position for X2, morphological forms of "be", and the optional insertion of "at".

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 5). 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) 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 3, 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 4). This lattice representation also allows the decoder to make choices between various morphological variations of a word (e.g. be/is/are).

5 Features and Tuning

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). The decoder considers in total a combination of 34 features, a selection of which are given below.

- Example penalty and example size
- Translation probability
- Language model score
- Optional words added/removed

The optimal weights for each feature are estimated using the Pairwise Ranking Optimization (PRO) algorithm (Hopkins and May, 2011) and parameter optimization with MegaM[4]. We use the implementation of PRO that is provided with the Moses SMT system and the default settings of MegaM.

6 Experiments

In order to evaluate our system, we conducted translation experiments on four language pairs: Japanese-English (JA–EN), English-Japanese (EN–JA), Japanese-Chinese (JA–ZH) and Chinese-Japanese (ZH–JA).

For Japanese-English, we evaluated on the NTCIR-10 PatentMT task data (patents) (Goto et al., 2013) and compared our system with the official baseline scores. For Japanese-Chinese, we used parallel scientific paper excerpts from the ASPEC[5] corpus and compared against the same baseline system as for Japanese-English. The corpora contain 3M parallel sentences for Japanese-English and 670K for Japanese-Chinese.

The two baseline systems are based on the open-source GIZA++/Moses pipeline. The baseline labeled "Moses" uses the classic phrase-based engine, while "Moses-Hiero" uses the Hierarchical Phrase-Based decoder. These

[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

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

[4]http://www.umiacs.umd.edu/~hal/megam/

[5]http://orchid.kuee.kyoto-u.ac.jp/ASPEC/

System	JA–EN	EN–JA	JA–ZH	ZH–JA
Moses	28.86	**33.61**	32.90	**42.79**
Moses-Hiero	28.56	32.98	—	—
Proposed	**29.00**	32.15	**32.99**	37.64

Table 1: Scores

System	BLEU	Translation
Moses	31.09	Further, the expansion stroke, the sectional area of the inner tube 12, and the oil is supplied to the lower oil chamber S2 from the oil reservoir chamber R × stroke.
Moses-Hiero	21.49	Also, the expansion stroke, the cross-sectional area of the inner tube 12 × stroke of oil supplied from the oil reservoir chamber R lower oil chamber S2.
Proposed	44.99	Further in this expansion stroke, the oil at an amount obtained by multiplying cross sectional area of the inner tube 12 from the oil reservoir chamber R is resupplied to the lower oil chamber S2.
Reference	100.00	In this expansion stroke, oil in an amount obtained by multiplying the cross sectional area of the inner tube 12 by the stroke is resupplied from the upper oil reservoir chamber R to the lower oil chamber S2.

Table 2: Example of JA–EN translation with better translation quality than baselines.

correspond to the highest performing official baselines for the NTCIR-10 PatentMT task.

As it appeared Moses was giving similar and slightly higher BLEU scores than Moses-Hiero for Japanese-English, we restricted evaluation to the standard settings for Moses for our Japanese-Chinese experiments.

The following dependency parsers were used. The scores in parentheses are the 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)

6.1 Results

The results shown are for evaluation on the test set after tuning. Tuning was conducted over 50 iterations on the development set using an n-best list of length 500.

Table 2 shows an example sentence showing significant improvement over the baseline. In particular, non-local structure has been preserved by the proposed system, such as the modification of 'oil' by the 'in an amount... by the stroke' phrase. Another example is the incorrect location of '× stroke' in the Moses output. The proposed system produces a much more fluent output than the hierarchical-based baseline Moses-Hiero.

The proposed system also outperforms the baseline for JA–ZH, however falls short for ZH–JA. We believe this is due to the low quality of parsing for Chinese input.

The decoder requires on average 0.94 seconds per sentence when loading from precompiled hypothesis files. As a comparison, Moses (default settings) takes 1.78 seconds per sentence, loading from a binarized and filtered phrase table.

7 Conclusion

This paper introduces an example-based translation system exploiting both source and target dependency analysis and online example retrieving, allowing the availability of full translation examples at translation time.

We believe that the use of dependency parsing is important for accurate translation across distant language pairs, especially in settings such as ours with many long sentences. We have designed a complete translation frame-

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

work around this idea, using dependency-parsed trees at each step from alignment to example retrieval to example combination.

The current performance (BLEU) of our system is similar to (or even slightly better than) state-of-the-art open-source SMT systems. As we have been able to obtain steady performance improvements during development, we are hopeful that this trend will continue and we will shortly obtain even better results. Future plans include enriching the feature set, adding a tree-based language model and considering forest input for multiple parses to provide robustness against parsing errors. When the code base is sufficiently stable, we intend to release the entire system as open-source, in the hope of providing a more syntactically-focused alternative to existing open-source SMT engines.

Acknowledgements

This work was partially supported by the Japan Science and Technology Agency. The first author is supported by a Japanese Government (MEXT) research scholarship. We would like to thank the anonymous reviewers.

References

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.*

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.*

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).*

Mark Hopkins and Jonathan May. 2011. Tuning as Ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing.*

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.*

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.*

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.*

Chris Callison-Burch, Colin Bannard, and Josh Schroeder. 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.

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

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.

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.

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.