Flexible Non-Terminals for Dependency Tree-to-Tree Reordering

John Richardson[†], **Fabien Cromières**[‡], **Toshiaki Nakazawa**[‡] and **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

A major benefit of tree-to-tree over tree-to-string translation is that we can use target-side syntax to improve reordering. While this is relatively simple for binarized constituency parses, the reordering problem is considerably harder for dependency parses, in which words can have arbitrarily many children. Previous approaches have tackled this problem by restricting grammar rules, reducing the expressive power of the translation model.

In this paper we propose a general model for dependency tree-to-tree reordering based on flexible non-terminals that can compactly encode multiple insertion positions. We explore how insertion positions can be selected even in cases where rules do not entirely cover the children of input sentence words. The proposed method greatly improves the flexibility of translation rules at the cost of only a 30% increase in decoding time, and we demonstrate a 1.2–1.9 BLEU improvement over a strong tree-to-tree baseline.

1 Introduction

Translation is most commonly performed by splitting an input sentence into manageable parts, translating these segments, then arranging them in an appropriate order. The first two steps have roughly the same difficulty for close and distant language pairs, however the reordering step is considerably more challenging for language pairs with dissimilar syntax. We need to be able to make linguistic generalizations, such as learning to translate between SVO and SOV clauses and converting post-modifying prepositional and pre-modifying postpositional phrases (Quirk et al., 2005). Such generalizations often require syntactically motivated long-distance reordering.

The first approaches to reordering were based on linear distortion (Koehn et al., 2003), which models the probability of swapping pairs of phrases over some given distance. The linear distance is the only parameter, ignoring any contextual information, however this model has been shown to work well for string-to-string translation. Linear reordering was improved with lexical distortion (Tillmann, 2004), which characterizes reordering in terms of type (monotone, swap, or discontinuous) as opposed to distance. This approach however is prone to sparsity problems, in particular for distant language pairs.

In order to improve upon linear string-based approaches, syntax-based approaches have also been proposed. Tree-to-string translation has been the most popular syntax-based paradigm in recent years, which is reflected by a number of reordering approaches considering source-only syntax (Liu et al., 2006; Neubig, 2013). One particularly interesting approach is to project source dependency parses to the target side and then learn a probability model for reordering children using features such as source and target head words (Quirk et al., 2005).

While tree-to-tree translation (Graehl and

Proceedings of NAACL-HLT 2016, pages 11–19,
San Diego, California, June 12-17, 2016.

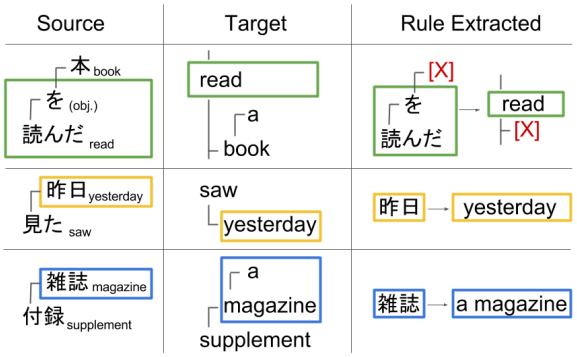

Figure 1: Examples of tree-to-tree translation rules extracted from an aligned and parsed bitext. Colored boxes represent aligned phrases and [X] is a non-terminal.

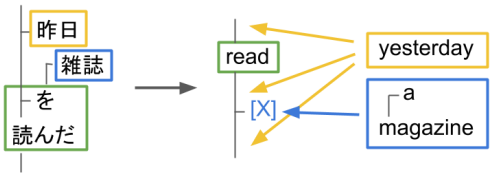

Figure 2: Combination of translation rules, demonstrating non-terminal substitution and multiple possible insertion positions for a non-matching input phrase ('昨日').

Knight, 2004; Cowan and Collins, 2006; Chiang, 2010) has been somewhat less popular than tree-to-string translation, we believe there are many benefits of considering target-side syntax. In particular, reordering can be defined naturally with non-terminals in the target-side grammar. This is relatively simple when the target structure of rules is restricted to 'well-formed' dependencies (Shen et al., 2008), however in this paper we consider more general rules with flexible non-terminal insertion positions.

2 Dependency Tree-To-Tree Translation

Dependency tree-to-tree translation begins with the extraction of translation rules from a bilingual corpus that has been parsed and word aligned. Figure 1 shows an example of three rules that can be extracted from aligned and parsed sentence pairs. In this paper we consider rules similar to previous work on tree-to-tree dependency MT (Richardson et al., 2014).

The simplest type of rule, containing only terminal symbols, can be extracted trivially from aligned subtrees (see rules 2 and 3 in Figure 1). Non-terminals can be added to rules (see rule 1 in Figure 1) by omitting aligned subtrees and replacing on each side with non-terminal symbols. We can naturally express phrase reordering as the source/target-side non-terminals are aligned.

Decoding is performed by combining these rules to form a complete translation, as shown in Figure 2. We are able to translate part of the sentence with non-ambiguous reordering ('read a magazine'), as we can insert '雑誌 → a magazine' into the rule '[X] を 読んだ → read [X]'.

We cannot however decide clearly where to insert the rule '昨日 → yesterday' as there is no matching non-terminal in the rule containing its parent in the input sentence ('読んだ'). We use the term *floating* to describe words such as 'yesterday' in this example, i.e. for an input subtree matched to the source side of a rule, children of the input root that are not contained in the source side of the rule as terminals and cannot be inserted using fixed-position non-terminals in the rule.

Previous work deals with this problem by either using simple glue rules (Chiang, 2005) or limiting rules in a way to avoid isolated floating children (Shen et al., 2008). For example, it is possible to disallow the first rule in Figure 1 when translating a sentence such as that in Figure 2 with uncovered children (in this case the word 'yesterday'). This method greatly reduces the expressiveness and flexibility of translation rules.

In our generalized model, we allow any number of terminals and non-terminals and permit arbitrarily many floating children in each rule. To our knowledge this is the first study to take this more comprehensive approach.

Note that in the case of constituency-based tree-to-tree translation it is possible to binarize the input tree and therefore gluing floating children becomes simpler, as we only have to choose between pre-insertion and post-insertion. In the dependency case it is in general much more dif-

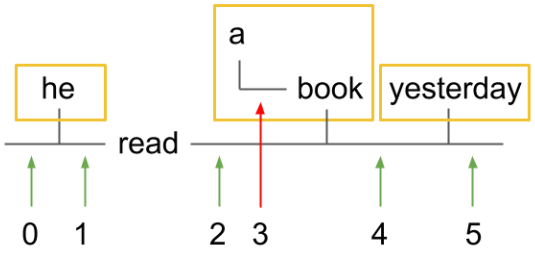

Figure 3: Possible insertion positions for flexible non-terminals with target-side head 'read'. Allowed positions are shown in green and disallowed positions are shown in red. We do not allow insertion position 3 because it could allow a non-projective dependency structure.

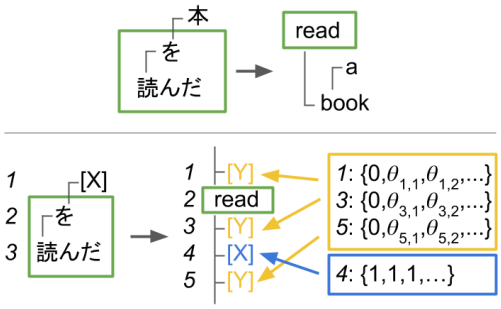

Figure 4: Example of translation rule with flexible non-terminals generated from the first parallel sentence in Figure 1. [X] has a fixed position (4) but [Y] can have multiple positions (1, 3, 5). Each position has an associated set of features shown in curly brackets, where $\theta_{i,j}$ is the jth feature for insertion position i. The first feature (0 or 1) shows whether the insertion position is unambiguous.

ficult because we must order an arbitrarily large group of children sharing a common head.

3 Flexible Non-Terminals

In this paper we propose flexible non-terminals in order to create generalized tree-to-tree translation rules that can overcome the problems described in the previous section. Rather than fixed insertion positions for child nodes, we instead consider multiple possible insertion positions and give features to each position. These are stored in a compact representation allowing for efficient decoding.

We define *flexible non-terminals* as non-terminals with multiple possible insertion positions and associated features. During decoding we select the most promising insertion position for each non-terminal.

3.1 Rule Augmentation

As is standard practice in phrase-based SMT, before translation we filter translation rules to those relevant to the input sentence. At this time, for each accepted rule we check the input sentence for floating children, and flexible non-terminals are added for each floating child.

We allow all insertion positions between the children (along with their descendants) of the target-side head for each floating child, including insertion before the first child and after the last child. We do not allow insertion positions between deeper descendants of the head to avoid non-projective dependencies. See Figure 3 for an example of allowed/disallowed positions.

Features are then set for each insertion position and these are used to determine the best insertion position during decoding (see Section 3.2). Figure 4 shows an example of the proposed rule augmentation.

3.2 Features

In previous work reordering is mostly decided by the combination of a standard distortion model and language model to score possible insertion positions. We instead consider the following four features and combine them during decoding to find the most appropriate insertion positions for floating children. All features are real numbers between 0 and 1.

3.2.1 Insertion Position Features

We first define a set of features to estimate the likelihood of each insertion position for some given non-terminal. The features for inserting the translation f of a source phrase into the target-side e of a rule at insertion position i are defined as follows, for surface forms (S) and POS tags (P):

- Reordering probability:
  $P_S(i \mid f, e)$, $P_P(i \mid f, e)$

- Marginalized over target-side: $P_S(i \mid f)$, $P_P(i \mid f)$
- Marginalized over source-side: $P_S(i \mid e)$, $P_P(i \mid e)$

The probabilities $P(i \mid X)$ are calculated by counting insertions of X in each position i across the whole training corpus (aligned and parsed bitext). The exact formula is given below, for position i (X is one of $\{f\}$, $\{e\}$ or $\{f, e\}$):

$$P(i \mid X) = \frac{count(i, X)}{\sum_j count(j, X)} \quad (1)$$

Instead of applying smoothing, in order to reduce sparsity issues we use both the full probability $P(i \mid f, e)$ and also probabilities marginalized over the source/target phrases. We also consider both probabilities trained on surface forms (S) and POS tags (P).

While traditional models use linear distance for i, this is impractical for long-distance reordering. Instead we restrict insertion types i to one of the following 6 types: first-pre-child, mid-pre-child, final-pre-child, first-post-child, mid-post-child, and final-post-child. These correspond to the first (first), last (final) or central (mid) children on the left (pre) or right (post) side of the parent word. We found this was more effective than using either linear distance or a binary (pre/post) position type.

3.2.2 Relative Position Feature

We also consider a relative position, or 'swapping' feature, inspired by the swap operation of classic lexical distortion (Tillmann, 2004).

Let T be the children of the root word of the target-side of a rule. We also include in T a pseudo-token M splitting the left and right children of the target-side root to differentiate between pre-insertion and post-insertion.

We first learn a model describing the probability of the translation of input phrase I appearing to the left ($P_L(I, t)$) or right ($P_R(I, t)$) of word t in the target-side of a translation rule. The probabilities are calculated by counting occurrences of I being translated to the left/right sides of t over the aligned and parsed training bitext.

The relative position feature is calculated by considering the relative position of the translation of I with all the target-side root children T. For each insertion position i, let $T_{i,L}$ be the $t \in T$ to the left of position i and $T_{i,R}$ the $t \in T$ to the right of position i. Then we have:

$$P(i \mid I, T) = \prod_{t \in T_{i,R}} P_L(I, t) \prod_{t \in T_{i,L}} P_R(I, t) \quad (2)$$

3.2.3 Left/Right Attachment Preference

We also set an attachment direction preference feature for each rule, specifying whether we prefer to insert the rule as a left child or right child of the root of a parent rule.

The attachment preference is determined by the position of the target-side of the rule in the target-side of the parallel sentence from which it was extracted. For example, in Figure 1 the rule '昨日 → yesterday' was extracted from a parallel sentence in which 'yesterday' was a right-side child of its head ('saw'), so we set the attachment preference to 'right'. In cases when we cannot determine the attachment preference (for example 'read' in the first rule in Figure 1), because it is the sentence root), we arbitrarily choose 'right'.

3.2.4 Unambiguous Insertion Preference

In cases where we have a single unambiguous insertion position for a non-terminal (e.g. [X] in Figure 4), we set an additional binary feature to the value 1 (otherwise 0) to specify that this position is unambiguous. We found that a large positive weight is almost always given to this feature, which is to be expected as we would prefer to use fixed non-terminals if possible. We set all features related to insertion position choice to the maximum value (1).

3.3 Decoding

The flexible non-terminals that we are proposing can lead to some interesting challenges when it comes to decoding. A naive approach is to expand each translation rule containing flexible non-terminals into a set of 'simple' rules with fixed non-terminals, and then apply classic decoding with cube-pruning.

However, this can be quite inefficient in practice. Due to the combinatorial aspect, a single rule can expand into a very large number of simple rules. It is common for our translation rules to have more than four flexible non-terminals, each with more than four possible insertion positions. Such rules will already generate hundreds of simple rules. In the most extreme cases, we may encounter rules having more than ten flexible non-terminals, leading to the generation of many millions of simple rules. This explosion of rules can lead to impractical decoding time and memory usage.

It is therefore important to make use of the compact encoding of many simple rules provided by the concept of flexible non-terminals in the decoding process itself. We use the decoding approach of right-hand lattices (Cromières and Kurohashi, 2014), an efficient way of encoding many simple rules. The idea is to encode the translation rules into a lattice form, then use this lattice to decode efficiently without the need to expand the flexible non-terminals explicitly.

Figure 5 shows how the concept of flexible non-terminals can be efficiently encoded into lattice form. The top half shows a target-side tree translation rule with flexible non-terminals X1, X2, X3 and X4 allowed to be inserted at any position that is a child of the word 'a', with the constraint that X1 comes before X2 and that X2 comes before X3. X5 is another flexible non-terminal that will be a child of the word 'f'. The lower half shows a lattice compactly encoding all the possible combinations of non-terminal positions. Each path from the top-left to the bottom right in this lattice represents a choice for the insertion positions of the non-terminals. For example, the path marked with a dotted line represents the flattened sequence 'b c X1 X2 a X3 X4 d e f X5 g'. The lattice form has only 48 edges, while an explicit enumeration of all combinations of insertion positions for the flexible non-terminals would force the decoder to consider ${}^8C_4 \times 3 \times 12 = 2520$ edges.

The insertion position features described above are added to the edges of the lattice. They are combined alongside the standard set of features, such as word penalty and language model score, using a standard log-linear model. The weights for the reordering features are tuned together with the standard features.

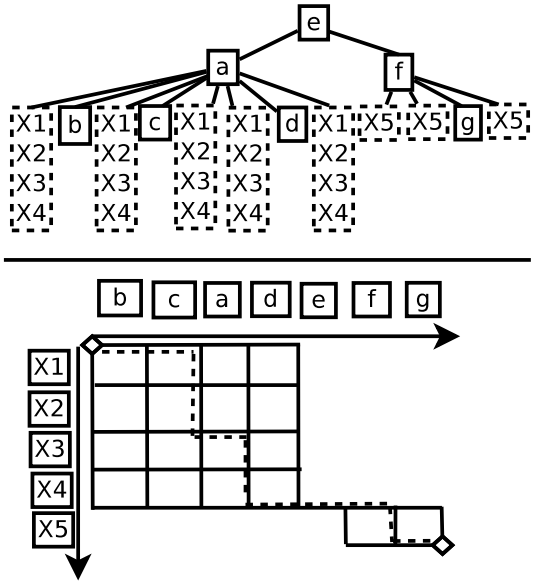

Figure 5: Example showing how a rule containing many flexible non-terminals is encoded into lattice form for decoding.

	JA–EN	EN–JA	JA–ZH	ZH–JA
Train	3M	3M	676K	676K
Dev	1790	1790	2123	2123
Test	1812	1812	2171	2171

Table 1: Translation experiment data (number of sentences).

4 Experiments

4.1 Data and Settings

We performed translation experiments on four distant language pairs, Japanese–English (JA–EN), English–Japanese (EN–JA), Japanese–Chinese (JA–ZH) and Chinese–Japanese (ZH–JA), from the Asian Scientific Paper Excerpt Corpus (ASPEC)[1]. The data was split into training, development and test folds as shown in Table 1.

Our experiments were conducted using a state-of-the-art dependency tree-to-tree framework KyotoEBMT (Richardson et al., 2014).

[1] http://lotus.kuee.kyoto-u.ac.jp/ASPEC/

	JA–EN		EN–JA		JA–ZH		ZH–JA	
	BLEU	RIBES	BLEU	RIBES	BLEU	RIBES	BLEU	RIBES
Moses	18.09	63.97	27.48	68.37	27.96	79.03	34.65	77.25
Baseline	19.97	65.10	28.41	74.78	28.13	78.00	33.51	77.86
Flexible	21.23†	69.94†	**30.11**†	**77.11**†	29.42†	80.44†	35.37†	81.33†
+Pref	**21.66**‡	70.73‡	29.90†	76.85†	29.48†	80.43†	35.57‡	81.79‡
+Pref+Ins	21.47‡	**70.85**‡	30.03†	77.01†	29.64†	**80.65**†	35.71‡	**82.05**‡
+Pref+Ins+Rel	21.34†	70.69‡	29.99†	76.93†	**29.78**‡	80.51†	**35.81**‡	81.95‡

Table 2: Automatic evaluation of translation quality (BLEU and RIBES). Results marked with † are significantly higher than the baseline system and those marked with ‡ are significantly higher than the proposed system with no insertion position features ('Flexible'). Significance was calculated with bootstrapping for $p < 0.05$.

Experiments were performed with the default settings by adding the proposed non-terminal reordering features to the rules extracted with the baseline system. We used lattice-based decoding (Cromières and Kurohashi, 2014) to support multiple non-terminal insertion positions and default tuning using, k-best MIRA (Cherry and Foster, 2012). Dependency parsing was performed with: KNP (Kawahara and Kurohashi, 2006) (Japanese), SKP (Shen et al., 2012) (Chinese), NLParser (Charniak and Johnson, 2005) (English, converted to dependencies with hand-written rules). Alignment was performed with Nile (Riesa et al., 2011) and we used a 5-gram language model with modified Kneser-Ney smoothing built with KenLM (Heafield, 2011).

4.2 Evaluation

As our baseline ('Baseline'), we used the default tree-to-tree settings and features of KyotoEBMT, allowing only fixed-position non-terminals. We dealt with floating children not covered by any other rules by adding glue rules similar to those in hierarchical SMT (Chiang, 2005), joining floating children to the rightmost slots in the target-side parent. For reference, we also show results using Moses (Koehn et al., 2007) with default settings and distortion limit set to 20 ('Moses').

The proposed system ('Flexible') adds flexible non-terminals with multiple insertion positions, however we do not yet add the insertion choice features. This means that the insertion positions are in practice chosen by the language model. Note that we do not get a substantial hit in performance by adding the flexible non-terminals because of their compact lattice representation. The systems '+Pref', '+Pref+Ins' and '+Pref+Ins+Rel' show the results of adding insertion choice position features (left/right preference, insertion position choice, relative position choice).

We give translation scores measured in BLEU (Papineni et al., 2002) and RIBES (Isozaki et al., 2010), which is designed to reflect quality of translation word order more effectively than BLEU. The translation evaluation is shown in Table 2.

5 Discussion and Error Analysis

The experimental results showed a significantly positive improvement in terms of both BLEU and RIBES over the baseline tree-to-tree system. The baseline system uses fixed non-terminals and is competitive with the most popular string-to-string system (Moses).

The extensions of the proposed model (adding a variety of features) also all showed significant improvement over the baseline, and approximately half of the extended settings performed significantly better than the core proposed model. It is unclear however which of the extended settings is the most effective for all language pairs. There are a number of factors such as parse quality, corpus size and out-of-vocabulary occurrence that could affect the potential value of these features. Furthermore, Japanese is strongly left-branching (head-final), so the left/right preference distinction is likely to be less useful than for English and Chinese,

which contain both left-branching and right-branching structures.

Compared to the baseline, the flexible non-terminals gave around a 1.2–1.9 BLEU improvement at the cost of only a 30% increase in decoding time (approximately 2.04 vs. 2.66 seconds per sentence). This is made possible by the compact non-terminal representation combined with lattice decoding.

5.1 Non-Terminal Matching Analysis

We found that roughly half of all our translation rules were augmented with flexible non-terminals, with one flexible non-terminal added per rule on average. This led to roughly half of non-terminals having flexible insertion positions. The decoder chose to use ambiguous insertion positions between 30%–60% of the time (depending on language pair), allowing for many more new translation hypotheses than the baseline system. For detailed results, see Table 3.

5.2 Translation Examples

The following translation is an example of an improvement achieved by using the proposed flexible non-terminals. There were multiple word order errors in the baseline translation that impeded understanding, and these have all been corrected.

- **Input:** 磁場入口と出口の温度差により生ずる磁性流体の圧力差と流速を測定した。
- **Reference:** The pressure difference and the flow velocity of the magnetized fluid caused by the temperature difference between the inlet and outlet of the magnetic field were measured.
- **Baseline:** We have measured the pressure difference and flow rate of a magnetic fluid generated by an entrance of a magnet and an exit temperature, and the difference between.
- **Proposed:** The pressure difference and the flow rate of a magnetic fluid generated by the temperature difference between the magnetic field inlet and exit were measured.

There are also cases where the proposed model decreases translation quality. In the example below, the proposed system output was selected by the decoder since it had a higher language model score than the baseline output, despite having incorrect word order. The incorrect translation was made available by the increased flexibility of the proposed model, and selected because the LM feature had a higher impact than the insertion position features.

- **Input:** このソフトウエアのR5バージョンの特徴, 利用マニュアルと設計文書をまとめた。
- **Reference:** The characteristics of R5 version of this software, instruction manual, and design document were summarized.
- **Baseline:** The R5 version of this software features, the manual for the utilization and design documents are summarized.
- **Proposed:** This software design documents of R5 version features, the manual for the utilization and summarized.

6 Conclusion and Future Work

In this paper we have proposed flexible non-terminals for dependency tree-to-tree translation. We plan to continue working on feature design for insertion position choice, and in the future would like to consider using neural networks for learning these features. We believe that it is important to continue to explore approaches that exploit more general target-side syntax, faithful to the tree-to-tree translation paradigm.

Flexible non-terminals allow multiple insertion positions to be expressed compactly and selected with features based on both source and target syntax. We have shown that a significant improvement in BLEU and RIBES scores can be gained by using the proposed model to increase the generality of dependency tree-to-tree translation rules.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback.

	JA–EN	EN–JA	JA–ZH	ZH–JA
% rules with flexible NTs	53.2	70.4	55.7	61.2
Average flexible NTs per rule	0.973	1.11	0.977	1.05
% all NTs that are flexible	48.0	48.6	54.5	56.1
% selected NTs that are flexible	32.2	35.1	40.5	58.4

Table 3: Results of non-terminal (NT) matching analysis.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180. Association for Computational Linguistics.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montréal, Canada, June. Association for Computational Linguistics.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270.

David Chiang. 2010. Learning to translate with source and target syntax. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1443–1452, Uppsala, Sweden, July. Association for Computational Linguistics.

Brooke Cowan and Michael Collins. 2006. A discriminative model for tree-to-tree translation. In *EMNLP*, pages 232–241.

Fabien Cromières and Sadao Kurohashi. 2014. Translation rules with right-hand side lattices. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 577–588, Doha, Qatar, October. Association for Computational Linguistics.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *HLT-NAACL 2004: Main Proceedings*, pages 105–112, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

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

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic evaluation of translation quality for distant language pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952, Cambridge, MA, October. Association for Computational Linguistics.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 176–183. Association for Computational Linguistics.

P. Koehn, F.J. Och, and D. Marcu. 2003. Statistical Phrase-Based Translation. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54, Morristown, NJ. Association for Computational Linguistics, Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July. Association for Computational Linguistics.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *ACL (Conference System Demonstra-*

tions), pages 91–96. The Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279.

John Richardson, Fabien Cromières, Toshiaki Nakazawa, and Sadao Kurohashi. 2014. KyotoEBMT: An example-based dependency-to-dependency translation framework. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Baltimore, Maryland, June. Association for Computational Linguistics.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 497–507. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph M Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Association for Computational Linguistics*.

Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. 2012. A reranking approach for dependency parsing with variable-sized subtree features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 308–317, Bali, Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Christoph Tillmann. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 101–104, Stroudsburg, PA, USA. Association for Computational Linguistics.