Improving fast_align by Reordering

Chenchen Ding, Masao Utiyama, Eiichiro Sumita
Multilingual Translation Laboratory
National Institute of Information and Communications Technology
3-5 Hikaridai, Seikacho, Sorakugun, Kyoto, 619-0289, Japan
{chenchen.ding, mutiyama, eiichiro.sumita}@nict.go.jp

Abstract

fast_align is a simple, fast, and efficient approach for word alignment based on the IBM model 2. fast_align performs well for language pairs with relatively similar word orders; however, it does not perform well for language pairs with drastically different word orders. We propose a *segmenting-reversing reordering* process to solve this problem by alternately applying fast_align and reordering source sentences during training. Experimental results with Japanese-English translation demonstrate that the proposed approach improves the performance of fast_align significantly without the loss of efficiency. Experiments using other languages are also reported.

1 Introduction

Aligning words in a parallel corpus is a basic task for almost all state-of-the-art statistical machine translation (SMT) systems. Word alignment is used to extract translation rules in various way, such as the phrase pairs used in a phrase-based (PB) SMT system (Koehn et al., 2003), the hierarchical rules used in a HIERO system (Chiang, 2007), and the sophisticated translation templates used in tree-based SMT systems (Liu et al., 2006).

Among different approaches, GIZA++[1] (Och and Ney, 2003), which is based on the IBM translation models, is the most widely used word alignment tool. Other well-known tools are the BerkeleyAligner[2], Nile[3] (Riesa et al., 2011), and pialign[4] (Neubig et al., 2011).

fast_align[5] (Dyer et al., 2013) is a recently proposed word alignment approach based on the reparameterization of the IBM model 2, which is usually referred to as a *zero-order* alignment model (Och and Ney, 2003). Taking advantage of the simplicity of the IBM model 2, fast_align introduces a "*tension*" parameter to model the overall accordance of word orders and an efficient parameter re-estimation algorithm is devised. It has been reported that the fast_align approach is more than 10 times faster than baseline GIZA++, with comparable results in end-to-end French-, Chinese-, and Arabic-to-English translation experiments.

However, the simplicity of the IBM model 2 also leads to a limitation. As demonstrated in this study, fast_align does not perform well when applied to language pairs with drastically different word orders, e.g., Japanese and English. The problem is because of the IBM model 2's intrinsic inability to handle complex distortions. In this study, we propose a simple and efficient reordering approach to improve the fast_align's performance in such situations, referred to as *segmenting-reversing* (seg_rev). Our motivation is to apply a rough but robust reordering to make the source and target sentences have more similar word orders, where fast_align can show its power. Specifically, seg_rev first segments a source-target sentence pair into a sequence of *minimal monotone* chunk pairs[6] based on the automatically generated word alignment. Within the chunk pairs, source word sequences are examined to determine whether they should be *completely reversed* or the original order should be retained. The objective of this step is to convert the source sentence to a roughly target-like word order. The seg_rev process is applied recursively but not deeply (only twice in our ex-

[1] http://www.statmt.org/moses/giza/GIZA++.html

[2] https://code.google.com/p/berkeleyaligner/

[3] http://jasonriesa.github.io/nile/

[4] http://www.phontron.com/pialign/

[5] https://github.com/clab/fast_align

[6] same as the "*tuple*" used in Mariño et al. (2006)

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1046–1051,
Lisbon, Portugal, 17-21 September 2015. ©2015 Association for Computational Linguistics.

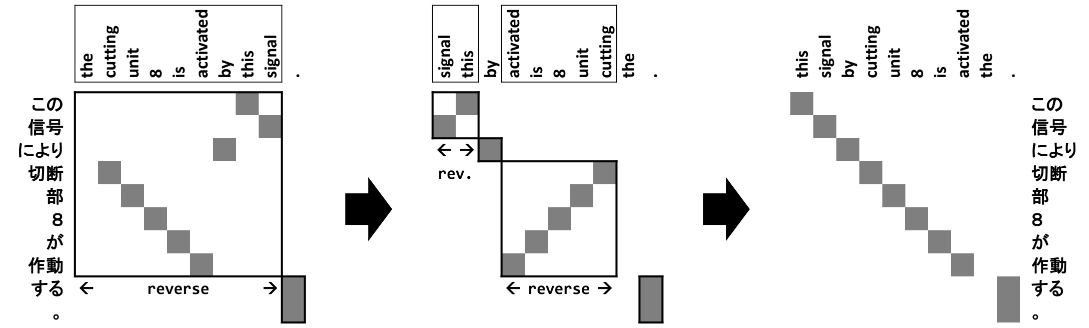

Figure 1: Example of seg_rev applied to a word-aligned English-Japanese sentence pair. Based on the word alignment, the source sentence is reordered in a target-like order after applying seg_rev **twice**.

periments) for each source sentence in the training data. Consequently, the seg_rev process is lightweight and shallow. Local word sequences, except those at chunk boundaries, are not scrambled, while global word orders are re-arranged if there are *large* chunks.

Our primary experimental results for Japanese-English translation show that applying seg_rev significantly improves fast_align's performance to a level comparable to GIZA++. The training time becomes 2–4 times that of a baseline fast_align, which is still at least 2 – 4 times faster than the training time required by baseline GIZA++. Results for German-, French-, and Chinese-English translations are also reported.

2 Segmenting-Reversing Reordering

The seg_rev is inspired by the "*REV preorder*" (Katz-Brown and Collins, 2008), which is a simple pre-reordering approach originally designed for the Japanese-to-English translation task. More efficient pre-reordering approaches usually require trained parsers and sophisticated machine learning frameworks (de Gispert et al., 2015; Hoshino et al., 2015). We adopt the REV method in Katz-Brown and Collins (2008) considering it is the simplest and lightest pre-reordering approach (to our knowledge), which may bring a minimal effect on the efficiency of fast_align.

An example seg_rev process, where the word alignment is generated by fast_align, is illustrated in Fig. 1. The example we selected has relatively correct word alignment and seg_rev performs well. In general cases, the alignment has significant noise and the reordering is rougher .

Algorithm 1 describes the repeated (δ times) application of the seg_rev process, and Algorithm 2 describes a single application. Specifically, Algorithm 1 applies Algorithm 2 δ times. For each application of Algorithm 2, source sentence S and source indices in alignment A are reordered, and the overall permutation R_I is updated and recorded. In Fig. 1, the original English sentence had 10 words (including the period), being indexed as $[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$. After the first application of seg_rev, R_I was $[\boxed{8,7,6,5,4,3,2,1,0}, 9]$, and after the second application, R_I was $[\boxed{7,8}, 6, \boxed{1,2,3,4,5}, 0, 9]$ (reversed parts are boxed).

In Algorithm 2, the main **for** loop (line 3) scans the source sentence from the beginning to the end to obtain monotone segmentation. The **foreach** (line 5) and **if** (line 11) are general phrase pair extraction process. The **if** (line 13) guarantees that the chunk is monotone on the target side. The rev function (line 16), which is described in Algorithm 3, determines whether the sub-sequence from s_{start} to s_{end} should be reversed by examining the related alignment A_{sub}. For example, in the first application shown in Fig.1, two sub-sequences $[0:8]$ and $[9:9]$ are processed by rev and $[0:8]$ is reversed. Four sub-sequences $[0:1]$, $[2:2]$, $[3:7]$, and $[9:9]$ are processed in the second application and $[0:1]$ and $[3:7]$ are reversed.[7] Finally, source sentence S and source indices in alignment A are reordered (lines 19 – 20).[8]

Algorithm 3 performs the reversal. We count the concordant and discordant pairs[9] and reverse

[7] The sub-sequences are based on the input, not the original sentence, e.g., sub-sequence $[0:1]$ contains the 8th and 7th word of the original source sentence in the 2nd application.

[8] Unaligned words between chunks on the source side are problematic. They are not touched by line 18. Although they can be attached to preceding or succeeding chunks, we do not use further heuristics to handle them. An example is the drifting "*the*" in the English sentence in Fig. 1, which our approach cannot handle properly.

[9] As used in Kendall's τ or Goodman and Kruskal's γ.

Algorithm 1: `seg_rev`$^{\delta}$

```
input   : same as Algorithm 2 and a depth $\delta$;
output  : same as Algorithm 2 except $R_A$;
$R_I \leftarrow [0,\cdots,I]$; $R_S \leftarrow S$; $R_A \leftarrow A$;
for $i \leftarrow 1$ to $\delta$ do
    $R_S, R_A, R_{I'} \leftarrow$ seg_rev $(R_S, T, R_A)$;
    permute $R_I$ with $R_{I'}$;
return $R_S, R_I$;
```

Algorithm 2: `seg_rev`

```
input   : source sentence $S = s_0, \cdots, s_I$;
          target sentence $T = t_0 \cdots, t_J$;
          word alignment $A = \{(i,j)|$
            $s_i, t_j$ are aligned, $i \in [0,I], j \in [0,J]\}$;
output  : reordered source sentence $R_S$;
          source index reordered alignment $R_A$;
          reordered indices $R_I = \pi(0,\cdots,I)$,
            which is a permutation from 0 to $I$;
$R_I \leftarrow [0,\cdots,I]$;
$s_{start} \leftarrow 0$; $t_{pre\_end} \leftarrow -1$;
for $s_{end} \leftarrow s_{start}$ to $I$ do
    $t_{start} \leftarrow J+1$; $t_{end} \leftarrow -1$;
    foreach $(i,j) \in A$ do
        if $i \in [s_{start}, s_{end}]$ then
            if $j < t_{start}$ then
                $t_{start} \leftarrow j$;
            if $j > t_{end}$ then
                $t_{end} \leftarrow j$;
    if $\exists (i,j) \in A$ :
          $j \in [t_{start}, t_{end}] \wedge i \notin [s_{start}, s_{end}]$ then
        continue;
    if $\exists (i,j) \in A : j \in (t_{pre\_end}, t_{start})$ then
        continue;
    $A_{sub} \leftarrow \{(i,j)|(i,j) \in A,$
          $i \in [s_{start}, s_{end}] \wedge j \in [t_{start}, t_{end}]\}$;
    rev $(R_I[s_{start} : s_{end}], A_{sub})$;
    $t_{pre\_end} \leftarrow t_{end}$;
    $s_{start} \leftarrow min(\{i|i > s_{end} \wedge \exists (i,j) \in A\})$;
$R_S \leftarrow$ permute $S$ according to $R_I$;
$R_A \leftarrow$ permute $i$ in $A$ according to $R_I$;
return $R_S, R_A, R_I$;
```

Algorithm 3: `rev`

```
input   : index sequence $I_{sub}$; word alignment $A_{sub}$;
$con \leftarrow 0$; $dis \leftarrow 0$;
foreach unordered tuple $((i_0,j_0),(i_1,j_1))$ :
      $(i_0,j_0) \in A_{sub} \wedge (i_1,j_1) \in A_{sub}$ do
    $x \leftarrow (i_1 - i_0) \times (j_1 - j_0)$;
    if $x > 0$ then
        $con \leftarrow con + 1$;
    else if $x < 0$ then
        $dis \leftarrow dis + 1$;
    else
        continue;
if $dis > con$ then
    $I_{sub} \leftarrow$ reverse $I_{sub}$;
```

Algorithm 4: `fast_align` with `seg_rev`

```
   input   : parallel corpus $\mathcal{C}$ with $N$ sentence pairs
               $\mathcal{C} = \{(S^1,T^1)\cdots,(S^N,T^N)\}$;
             maximum iteration $M$; depth $\delta$ for seg_rev;
   output  : word alignment $\mathcal{A} = \{A^1,\cdots,A^N\}$;
 1 $\mathcal{A} \leftarrow \varnothing$;
 2 for $iter \leftarrow 1$ to $M$ do
 3     $\mathcal{I} \leftarrow \varnothing$; $\mathcal{C}' \leftarrow \varnothing$;
 4     if $\mathcal{A} \neq \varnothing$ then
 5         for $n \leftarrow 1$ to $N$ do
 6             $R_S, R_I \leftarrow$ seg_rev$^{\delta}(S^n, T^n, A^n)$;
 7             append $(R_S, T^n)$ to $\mathcal{C}'$;
 8             append $R_I$ to $\mathcal{I}$;
 9     else
10         $\mathcal{C}' \leftarrow \mathcal{C}$;
11     $\mathcal{A} \leftarrow$ sym $\circ$ fast_align $(\mathcal{C}')$;
12     if $\mathcal{I} \neq \varnothing$ then
13         for $n \leftarrow 1$ to $N$ do
14             recover $A^n$ by $\mathcal{I}[n]$;
15 return $\mathcal{A}$;
```

the sub-sequence if and only if there are more discordant pairs than the concordant pairs. In Fig.1, the sub-sequence $[0:8]$ in the first application has $C_8^2 = 28$ pairs of aligned word pair (i.e., 28 gray block pairs for eight gray blocks); however, only 11 pairs are concordant ($C_5^2 = 10$ pairs in $[1:5]$ and one pair in $[7:8]$), Consequently, the sub-sequence $[0:8]$ is reversed because there are more discordant pairs ($17 = 28 - 11$). The two reversed sub-sequences in the second application are obvious.

Algorithm 4 describes the training framework, where `fast_align` and `seg_rev` are applied alternately. To generate word alignment, `fast_align` is run bi-directionally and symmetrization heuristics are applied to reduce noise (line 11). In each iteration, the source sentences for `seg_rev` are the original sentences, and `fast_align` uses the reordered sentences with the exception of the first iteration. The word alignment generated is thus based on the reordered source sentences; consequently, the recorded permutation (line 14) is used to recover word alignment before the next iteration. The permutation is a one-to-one mapping; therefore, recovering is realized by the inverse mapping of the permutation, which transfers the source-side word alignment indices to match the original source sentences.

The time complexity of Algorithm 3 is $O(l^2)$, where l is the size of A_{sub} that is related to the chunk size. If the average chunk size is a constant C depending on languages pairs or data sets, then the time complexity of Algorithm 2 is $O(C \cdot I^2)$ assuming J and the size of A are both linear against I. The average chunk size will be reduced when `seg_rev` is applied successively; therefore, the time required for subsequent `seg_rev` processes will decrease. In practice, compared with the training time required by `fast_align`, seg_rev processing time is negligible. Note that seg_rev processes are accelerated easily by parallel processing.

3 Experiments and Discussion

We applied the proposed approach to Japanese-English translation, a language pair with dramatically different word orders. In addition, we applied the approach to German-English translation, a language pair with relatively different word orders among European languages.

For Japanese-English translation, we used **NTCIR-7 PAT-MT** data (Fujii et al., 2008). For German-English translation, we used the **Europarl v7** corpus[10] (Koehn, 2005) for training, the **WMT 08**[11] **/ WMT 09**[12] test sets for development / testing, respectively. Default settings for the PB SMT in MOSES[13] (Koehn et al., 2007) were used, except for Japanese-English translations where the *distortion-limit* was set to 12 to reach a recently reported baseline (Isozaki et al., 2012). MERT (Och, 2003) was used to tune development set parameter weights and BLEU (Papineni et al., 2002) was used on test sets to evaluate the translation performance. *Bootstrap sampling* (Koehn, 2004) was employed to test statistical significance using bleu_kit[14].

We compared GIZA++ and fast_align with default settings. GIZA++ was used as a module of MOSES. The bi-directional outputs of fast_align were symmetrized by atools in cdec[15] (Dyer et al., 2010), and further training steps were conducted using MOSES. *grow-diag-final-and* symmetrization was used consistently in the experiments. For the the proposed approach, we set $\delta = 2$ and $M = 4$ in Algorithm 4. Note that δ can be set to a larger value and seg_rev could be applied repeatedly until no additional reordering is possible. As mentioned, the word alignment is noisy and our intention is a robust and rough process; therefore, we restricted seg_rev to two applications and did not consider the difference in sentence lengths or different languages during training. Within each iteration, fast_align was run with default settings, except *initial diagonal-tension* (λ^{ini}) was set to 0.1 in the first iteration, to avoid overly strong monotone preference at the beginning of training.

	ja-en	**en-ja**	**de-en**	**en-de**
GIZA++	$\mathbf{28.8}$	$\mathbf{30.8}$	$\mathbf{18.2}$	$\mathbf{12.9}$
FA$_{\lambda^{ini}=4.0}$	$28.1^{\ddagger}$	$29.5^{\ddagger}$	$18.0^{\dagger}$	$12.7^{\dagger}$
FA$_{\lambda^{ini}=0.1}$	$28.0^{\ddagger}$	$29.8^{\ddagger}$	$17.5^{\ddagger}$	$12.5^{\ddagger}$
iteration 2	$28.3^{\dagger}$	$\mathbf{30.9}$	$17.9^{\ddagger}$	$\mathbf{12.8}$
iteration 3	$28.4^{\dagger}$	$30.1^{\ddagger}$	$\mathbf{18.1}$	$12.7^{\dagger}$
iteration 4	$\mathbf{28.8}$	$\mathbf{30.7}$	$\mathbf{18.1}$	$12.7^{\dagger}$

Table 1: Test set BLEU scores for Japanese-English and German-English translations. (‡, statistical significance at $p < 0.01$; †, at $p < 0.05$; boldface, no significance; all compared with GIZA++)

Experimental results for Japanese-English and German-English translations in both directions are listed in Table 1. The first two rows show the baseline performance. fast_align (using a default $\lambda^{ini} = 4.0$) performance was statistically significantly lower than GIZA++, particularly for Japanese-English translation. The following four rows show the results of the proposed approach. For the first iteration, λ^{ini} was set to 0.1, and the performance did not change significantly. The translations **from** English improved (equal to GIZA++) at the second iteration. However, translations **to** English improved more slowly. We attribute the difference in improvement rates between translation to and from English to the relatively fixed word order of English, whereby the reordering process is easier and more consistent. Note that once translations **from** English improved in the second iteration, performance decreased in the following iterations. The results in Table 1 were obtained using *predictable-seed* for tuning, which generated determinate results. Another attempt using random seeds to tune returned test set BLEU scores of $\mathbf{30.5}$, $\mathbf{30.4}$ on **en-ja** and $\mathbf{12.8}$, $\mathbf{12.8}$ on **en-de**, for iterations 3 and 4, respectively. These four scores had no statistical significance against GIZA++. The instability is largely due to the alignment of function words, which affects translation performance (Riesa et al., 2011). The alignment does not change significantly after the second iteration; however, it is unstable around function words,[16] because seg_rev does not process

[10] http://www.statmt.org/europarl/

[11] http://www.statmt.org/wmt08/

[12] http://www.statmt.org/wmt09/

[13] http://www.statmt.org/moses/

[14] http://www.nlp.mibel.cs.tsukuba.ac.jp/bleu_kit/

[15] http://www.cdec-decoder.org/

[16] Specifically, *to*, articles, and prepositions in English-Japanese; *of*, *have*, and relative pronouns in English-German.

	fr-en	**zh-en**
GIZA++	23.3	31.4
$\texttt{FA}_{\lambda^{ini}=4.0}$	23.1	31.7
iteration 2	23.1	31.7

Table 2: Test set BLEU scores for French-to-English and Chinese-to-English translations. For **fr-en**, the data sets were the same as for **de-en**. For **zh-en**, NIST 2006 OpenMT data were used for training and test; test data from 2002 to 2005 OpenMT were used for tuning.

unaligned function words between chunks. Our approach is too rough to handle function words precisely. We plan to address this in future.

We also tested our approach on French- and Chinese-to-English translations. The results are listed in Table 2. GIZA++ and fast_align showed no statistically significant difference in performance, which is consistent with Dyer et al. (2013). The proposed approach did not affect performance for French- and Chinese-to-English translations. These results are expected as these language pairs have similar word orders.

With regard to processing time, a naïve, single-thread implementation of seg_rev in C++ took approximately 60*s* / 40*s* in the first / second application on the entire Japanese-English corpus[17]. The recover process took less than 30*s* in each iteration. In contrast, fast_align, although very fast, took approximately one hour for one round of training (using five iterations for its log-linear model) on the same corpus. Therefore, the additional time required in our approach is quite small and can be ignored compared with the training time of fast_align.[18]

4 Conclusion and Future Work

We have proposed a simple and efficient approach to improve the performance of fast_align on language pairs with drastically different word orders. With the proposed approach, fast_align obtained results comparable with GIZA++, and its efficiency is retained. We are investigating further properties of seg_rev and plan to extend it to achieve greater stability and efficiency.[19]

Acknowledgments

We thank Dr. Atsushi Fujita for his helpful discussions on this work.

References

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

Adrià de Gispert, Gonzalo Iglesias, and Bill Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proc. of NAACL-HLT*, pages 1012–1017.

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

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proc. of NAACL-HLT*, pages 644–648.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, Terumasa Ehara, Hiroshi Echizen-ya, and Sayori Shimohata. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Proc. of NTCIR*, pages 389–400.

Sho Hoshino, Yusuke Miyao, Katsuhito Sudoh, Katsuhiko Hayashi, and Masaaki Nagata. 2015. Discriminative preordering meets Kendall's Tau maximization. In *Proc. of ACL (Short Papers)*, pages 139–144.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2012. HPSG-based preprocessing for English-to-Japanese translation. *ACM Transactions on Asian Language Information Processing*, 11(3):8.

Jason Katz-Brown and Michael Collins. 2008. Syntactic reordering in preprocessing for Japanese → English translation: MIT system description for NTCIR-7 patent translation task. In *Proc. of NTCIR*, pages 409–414.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 48–54.

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

[17] 1.8 *M* sentence pairs with an average length of 35 words.

[18] GIZA++ actually took around 18 hours to align the Japanese-English corpus (parallel processes for two directions, including mkcls, five iterations for the model 1 and the HMM model, three iteration for the model 3 and 4).

[19] Ongoing experiments using seg_rev with GIZA++ returned negative results. BLEU decreases by approx. 1 point.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. of EMNLP*, pages 388–395.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT summit*, pages 79–86.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. of ACL*, pages 609–616.

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Graham Neubig, Taro Watanabe, Eiichiro Sumita, Shinsuke Mori, and Tatsuya Kawahara. 2011. An unsupervised model for joint phrase alignment and extraction. In *Proc. of ACL-HLT*, pages 632–641.

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

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

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

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proc. of EMNLP*, pages 497–507.