Head Finalization: A Simple Reordering Rule for SOV Languages

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, Kevin Duh
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seikacho, Sorakugun, Kyoto, 619-0237, Japan
{isozaki,sudoh,tsukada,kevinduh}@cslab.kecl.ntt.co.jp

Abstract

English is a typical SVO (Subject-Verb-Object) language, while Japanese is a typical SOV language. Conventional Statistical Machine Translation (SMT) systems work well within each of these language families. However, SMT-based translation from an SVO language to an SOV language does not work well because their word orders are completely different. Recently, a few groups have proposed rule-based preprocessing methods to mitigate this problem (Xu et al., 2009; Hong et al., 2009). These methods rewrite SVO sentences to derive more SOV-like sentences by using a set of handcrafted rules. In this paper, we propose an alternative single reordering rule: **Head Finalization**. This is a syntax-based preprocessing approach that offers the advantage of simplicity. We do not have to be concerned about part-of-speech tags or rule weights because the powerful Enju parser allows us to implement the rule at a general level. Our experiments show that its result, **Head Final English (HFE)**, follows almost the same order as Japanese. We also show that this rule improves automatic evaluation scores.

1 Introduction

Statistical Machine Translation (SMT) is useful for building a machine translator between a pair of languages that follow similar word orders. However, SMT does not work well for distant language pairs such as English and Japanese, since English is an SVO language and Japanese is an SOV language.

Some existing methods try to solve this word-order problem in language-independent ways. They usually parse input sentences and learn a reordering decision at each node of the parse trees. For example, Yamada and Knight (2001), Quirk et al. (2005), Xia and McCord (2004), and Li et al. (2007) proposed such methods.

Other methods tackle this problem in language-dependent ways (Katz-Brown and Collins, 2008; Collins et al., 2005; Nguyen and Shimazu, 2006). Recently, Xu et al. (2009) and Hong et al. (2009) proposed rule-based preprocessing methods for SOV languages. These methods parse input sentences and reorder the words using a set of handcrafted rules to get SOV-like sentences.

If we could completely reorder the words in input sentences by preprocessing to match the word order of the target language, we would be able to greatly reduce the computational cost of SMT systems.

In this paper, we introduce a single reordering rule: **Head Finalization**. *We simply move syntactic heads to the end of the corresponding syntactic constituents (e.g., phrases and clauses).* We use only this reordering rule, and we do not have to consider part-of-speech tags or rule weights because the powerful Enju parser allows us to implement the rule at a general level.

Why do we think this works? The reason is simple: Japanese is a typical **head-final** language. That is, a syntactic head word comes after non-head (dependent) words. SOV is just one aspect of head-final languages. In order to implement this idea, we need a parser that outputs **syntactic heads**. **Enju** is such a parser from the University of Tokyo (http://www-tsujii.is.s.u-tokyo.ac.jp/enju). We discuss other parsers in section 5.

There is another kind of head: **semantic heads**. Hong et al. (2009) used Stanford parser (de Marneffe et al., 2006), which outputs semantic head-based dependencies; Xu et al. (2009) also used the same representation.

The use of syntactic heads and the number of **dependents** are essential for the simplicity of

Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR, pages 244–251,
Uppsala, Sweden, 15-16 July 2010.

Head Finalization (See Discussion). Our method simply checks whether a tree node is a syntactic head. We do not have to consider what we are moving and how to move it. On the other hand, Xu et al. had to introduce dozens of weighted rules, probably because they used the semantic head-based dependency representation without restriction on the number of dependents.

The major difference between our method and the above conventional methods, other than its simplicity, is that our method moves not only verbs and adjectives but also functional words such as prepositions.

2 Head Finalization

Figure 1 shows Enju's XML output for the simple sentence: "`John hit a ball`." The tag `<cons>` indicates a nonterminal node and `<tok>` indicates a terminal node or a word (token). Each node has a unique `id`. Head information is given by the node's `head` attribute. For instance, node `c0`'s head is node `c3`, and `c3` is a `VP`, or verb phrase. Thus, Enju treats not only words but also non-terminal nodes as heads.

Enju outputs at most two child nodes for each node. One child is a head and the other is a dependent. `c3`'s head is `c4`, which is `VX`, or a fragment of a verb phrase. `c4`'s head is `t1` or `hit`, which is `VBD` or a past-tense verb. The upper picture of Figure 2 shows the parse tree graphically. Here, ⋆ indicates an edge that is linked from a 'head.'

Our **Head Finalization** rule simply swaps two children when the head child appears before the dependent child. In the upper picture of Fig. 2, `c3` has two children `c4` and `c5`. Here, `c3`'s head `c4` appears before `c5`, so `c4` and `c5` are swapped.

The lower picture shows the swapped result. Then we get `John a ball hit`, which has the same word order as its Japanese translation *jon wa bohru wo utta* except for the functional words `a`, *wa*, and *wo*.

We have to add Japanese particles *wa* (topic marker) or *ga* (nominative case marker) for `John` and *wo* (objective case marker) for `ball` to get an acceptable Japanese sentence.

It is well known that SMT is not good at generating appropriate particles from English, whitch does not have particles. Particle generation was tackled by a few research groups (Toutanova and Suzuki, 2007; Hong et al., 2009).

Here, we use Enju's output to generate seeds

```
⟨sentence id="s0" parse_status="success"⟩
  ⟨cons id="c0" cat="S" xcat="" head="c3"⟩
    ⟨cons id="c1" cat="NP" xcat="" head="c2"⟩
      ⟨cons id="c2" cat="NX" xcat="" head="t0"⟩
        ⟨tok id="t0" cat="N" pos="NNP"
             base="john"⟩John⟨/tok⟩
      ⟨/cons⟩
    ⟨/cons⟩
    ⟨cons id="c3" cat="VP" xcat="" head="c4"⟩
      ⟨cons id="c4" cat="VX" xcat="" head="t1"⟩
        ⟨tok id="t1" cat="V" pos="VBD" base="hit"
             arg1="c1" arg2="c5"⟩hit⟨/tok⟩
      ⟨/cons⟩
      ⟨cons id="c5" cat="NP" xcat="" head="c7"⟩
        ⟨cons id="c6" cat="DP" xcat="" head="t2"⟩
          ⟨tok id="t2" cat="D" pos="DT" base="a"
               arg1="c7"⟩a⟨/tok⟩
        ⟨/cons⟩
        ⟨cons id="c7" cat="NX" xcat="" head="t3"⟩
          ⟨tok id="t3" cat="N" pos="NN"
               base="ball"⟩ball⟨/tok⟩
        ⟨/cons⟩
      ⟨/cons⟩
    ⟨/cons⟩
  ⟨/cons⟩
.⟨/sentence⟩
```

Figure 1: Enju's XML output (some attributes are removed for readability).

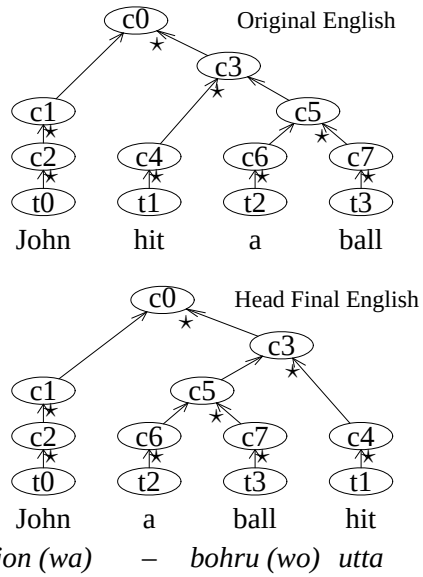

Figure 2: Head Finalization of a simple sentence (⋆ indicates a head).

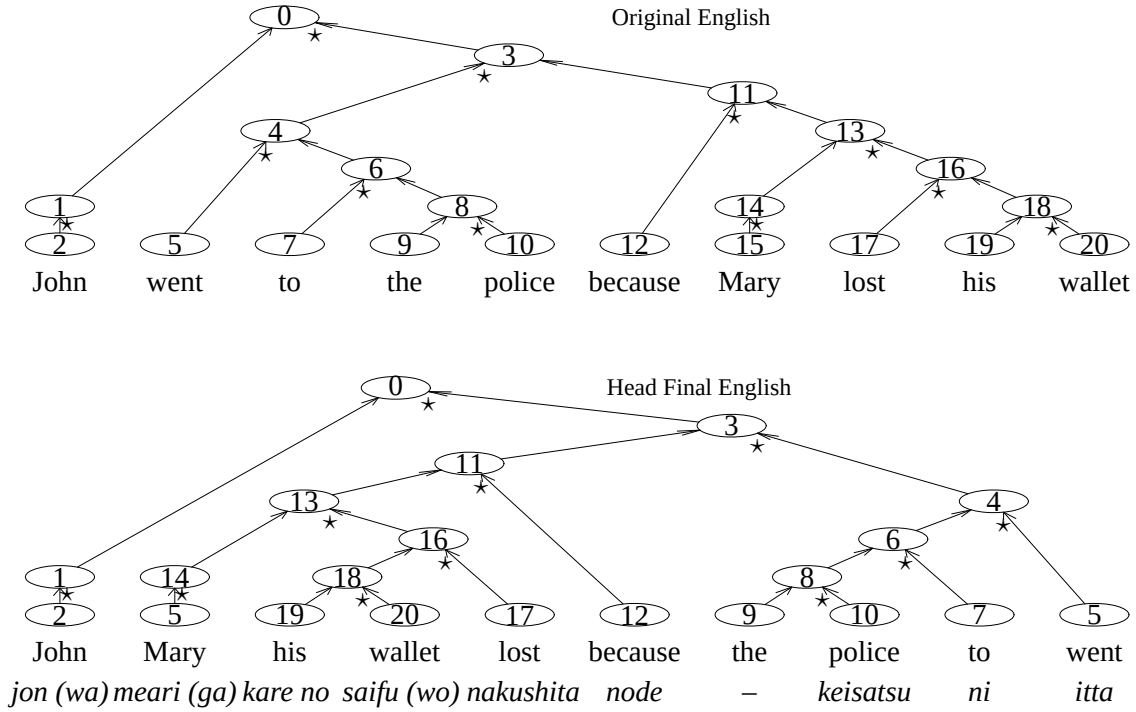

Figure 3: Head-Finalizing a complex sentence.

for particles. As Fig. 1 shows, the verb `hit` has `arg1="c1"` and `arg2="c5"`. This indicates that `c1 (John)` is the subject of `hit` and `c5 (a ball)` is the object of `hit`. We add seed words `va1` after `arg1` and `va2` after `arg2`. Then, we obtain `John va1 a ball va2 hit`. We do not have to add `arg2` for `be` because `be`'s `arg2` is not an object but a complement. We introduced the idea of particle seed words independently but found that it is very similar to Hong et al. (2009)'s method for Korean.

Figure 3 shows Enju's parse tree for a more complicated sentence "`John went to the police because Mary lost his wallet.`" For brevity, we hide the terminal nodes, and we removed the nonterminal nodes' prefix `C`.

Conventional Rule-Based Machine Translation (RBMT) systems swap X and Y of "X because Y" and move verbs to the end of each clause. Then we get "Mary his wallet lost because John the police to went." Its word-to-word translation is a fluent Japanese sentence: *meari (ga) kare no saifu (wo) nakushita node jon (wa) keisatsu ni itta*.

On the other hand, our Head Finalization with particle seed words yields a slightly different word order "John va1 Mary va1 his wallet va2 lost because the police to went." Its word-to-word translation is *jon wa meari ga kare no saifu wo nakushita node keisatsu ni itta*. This is also an acceptable Japanese sentence.

This difference comes from the syntactic role of '`because`.' In our method, Enju states that `because` is a dependent of `went`, whereas RBMT systems treat `because` as a clause conjunction.

When we use Xu et al.'s preprocessing method, '`because`' moves to the beginning of the sentence. We do not know a good monotonic translation of the result.

Preliminary experiments show that HFE looks good as a first approximiation of Japanese word order. However, we can make it better by introducing some heuristic rules. (We did not see the test set to develop these heuristic rules.)

From a preliminary experiment, we found that **coordination** expressions such as `A and B` and `A or B` are reordered as `B and A` and `B or A`. Although `A` and `B` have syntactically equal positions, the order of these elements sometimes matters. Therefore, we decided to stop swapping them at coordination nodes, which are indicated `cat` and `xcat` attributes of the Enju output. We call this the **coordination exception rule**. In addition, we avoid Enju's splitting of numerical expressions such as "`12,345`" and "`(1)`" because this splitting leads to inappropriate word orders.

3 Experiments

In order to show how closely our Head Finalization makes English follow Japanese word order, we measured Kendall's τ, a rank correlation coefficient. We also measured BLEU (Papineni et al., 2002) and other automatic evaluation scores to show that Head Finalization can actually improve the translation quality.

We used NTCIR7 PAT-MT's Patent corpus (Fujii et al., 2008). Its training corpus has 1.8 million sentence pairs. We used MeCab (`http://mecab.sourceforge.net/`) to segment Japanese sentences.

3.1 Rough evaluation of reordering

First, we examined rank correlation between Head Final English sentences produced by the Head Finalization rule and Japanese reference sentences. Since we do not have handcrafted word alignment data for an English-to-Japanese bilingual corpus, we used GIZA++ (Och and Ney, 2003) to get automatic word alignment.

Based on this automatic word alignment, we measured Kendall's τ for the word order between HFE sentences and Japanese sentences. Kendall's τ is a kind of rank correlation measure defined as follows. Suppose a list of integers such as L = [2, 1, 3, 4]. The number of all integer pairs in this list is ${}_4C_2 = 4 \times 3/(2 \times 1) = 6$. The number of increasing pairs is five: $(2,3)$, $(2,4)$, $(1,3)$, $(1,4)$, and $(3,4)$. Kendall's τ is defined by

$$\tau = \frac{\text{\#increasing pairs}}{\text{\#all pairs}} \times 2 - 1.$$

In this case, we get $\tau = 5/6 \times 2 - 1 = 0.667$.

For each sentence in the training data, we calculate τ based on a GIZA++ alignment file, `en-ja.A3.final`. (We also tried `ja-en.A3.final`, but we got similar results.) It looks something like this:

```
John hit a ball .
NULL ({3}) jon ({1}) wa ({}) bohru ({4})
  wo ({}) utta ({2}) . ({5})
```

Numbers in `({ })` indicate corresponding English words. The article 'a' has no corresponding word in Japanese, and such words are listed in `NULL ({ })`. From this alignment information, we get an integer list [1, 4, 2, 5]. Then, we get $\tau = 5/{}_4C_2 \times 2 - 1 = 0.667$.

For HFE in Figure 2, we will get the following alignment.

```
John va1 a ball va2 hit .
NULL ({3}) jon ({1}) wa ({2}) bohru ({4})
wo ({5}) utta ({6}) . ({7})
```

Then, we get [1, 2, 4, 5, 6, 7] and $\tau = 1.0$. We use $\overline{\tau}$ or the average of τ over all training sentences to observe the tendency.

Sometimes, one Japanese word corresponds to an English phrase:

```
John went to Costa Rica .
NULL ({}) jon ({1}) wa ({}) kosutarika ({4 5})
ni ({3}) itta ({2}) . ({6})
```

We get [1, 4, 5, 3, 2, 6] from this alignment.

When the same word (or derivative words) appears twice or more in a single English sentence, two or more non-consecutive words in the English sentence are aligned to a single Japanese word:

```
rate of change of speed
NULL ({}) sokudo ({5}) henka ({3})
no ({2 4}) wariai ({1})
```

We excluded the ambiguously aligned words `(2 4)` from the calculation of τ. We use only [5, 3, 1] and get $\tau = -1.0$. The exclusion of these words will be criticized by statisticians, but even this rough calculation of τ sheds light on the weak points of Head Finalization.

Because of this exclusion, the best value $\tau = 1.0$ does not mean that we obtained the perfect word ordering, but low τ values imply failures. In section 4, we use τ to analyze failures.

By examining low τ sentences, we found that patent documents have a lot of expressions such as "`motor 2`." These are reordered (`2 motor`) and slightly degrade τ. We did not notice this problem until we handled the patent corpus because these expressions are rare in other documents such as news articles. Here, we added a rule to keep these expressions.

We did not use any dictionary in our experiment, but if we add dictionary entries to the training data, it raises $\overline{\tau}$ because most entries are short. One-word entries do not affect $\overline{\tau}$ because we cannot calculate τ. Most multi-word entries are short noun phrases that are not reordered (τ = 1.0). Therefore, we should exclude dictionary entries from the calculation of $\overline{\tau}$.

3.2 Quality of translation

It must be noted that the rank correlation does not directly measure the quality of translation. Therefore, we also measured BLEU and other automatic evaluation scores of the translated sentences. We used Moses (Koehn, 2010) for Minimum Error Rate Training and decoding.

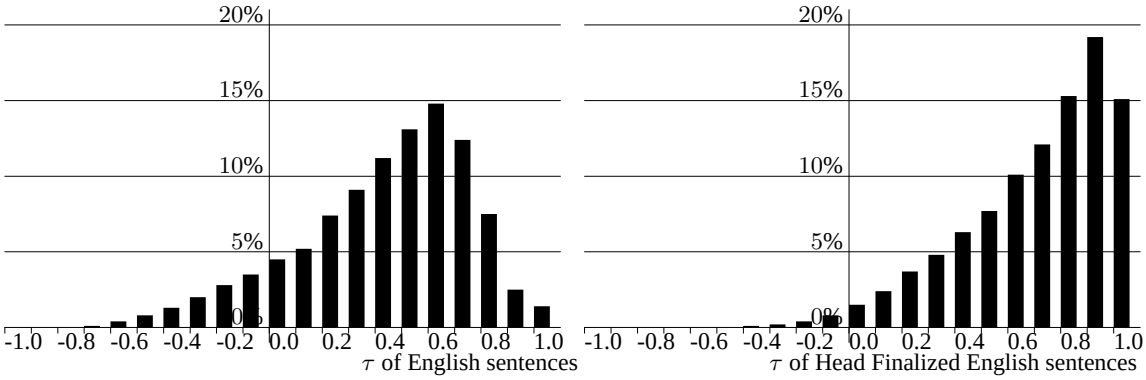

Figure 4: Distribution of τ

We used the development set (915 sentences) in the NTCIR7 PAT-MT PSD data as well as the formal run test set (1,381 sentences).

In the NTCIR7 PAT-MT workshop held in 2008, its participants used different methods such as hierarchical phrase-based SMT, RBMT, and EBMT (Example-Based Machine Translation). However, the organizers' Moses-based baseline system obtained the best BLEU score.

4 Results

First, we show τ values to evaluate word order, and then we show BLEU and other automatic evaluation scores.

4.1 Rank correlation

The original English sentences have $\overline{\tau} = 0.451$. Head Finalization improved it to 0.722. Figure 4 shows the distribution of τ for all training sentences. HFE reduces the percentage of low τ sentences: **49.6% of the 1.8 million HFE sentences have $\tau \geq 0.8$ and 15.1% have $\tau = 1.0$.**

We also implemented Xu et al.'s method with the Stanford parser 1.6.2. Its $\overline{\tau}$ was 0.624. The rate of the sentences with $\tau \geq 0.8$ was 30.6% and the rate of $\tau = 1.0$ was 4.3%.

We examined low τ sentences of our method and found the following reasons for low τ values.

- The sentence pair is not an exact one-to-one translation. A Japanese reference sentence for "`I bought the cake.`" can be something like "`The cake I bought.`" or "`The person who bought the cake is me.`"
- Mistakes in Enju's tagging or parsing. We encountered certain POS tag mistakes:
  - VBZ/NNS mistake: '`advances`' of "... `device advances along` ..." is VBZ, but NNS is assigned.
  - VBN/VBD mistake: '`encoded`' of "... `the error correction encoded data is supplied` ..." is VBN, but VBD is assigned.

  These tagging mistakes lead to global parsing mistakes. In addition, just like other parsers, Enju tends to make mistakes when a sentence has a comma or '`and`.'
- Mistakes/Ambiguity of GIZA++ automatic word alignment. Ambiguity happens when a single sentence has two or more occurrences of a word or derivatives of a word (e.g., difference/different/differential). As we described above, ambiguously aligned words are removed from calculation of τ, and small reordering mistakes in other words are emphasized.

main cause	count
tagging/parsing mistakes	12
VBN/VBD mistake	(4)
VBZ/NNS mistake	(2)
comma or `and`	(2)
inexact translation	7
wrong alignment	1

Table 1: Main causes of 20 worst sentences

We analyzed the 20 worst sentences with $\tau < -0.5$ when we used only 400,000 sentences for GIZA++. Their causes are summarized in Table 1. In general, low τ sentences have two or more causes, but here we show only the most influential cause for each sentence. This table shows that mistakes in tagging and parsing are major causes of low τ values. When we used all of 1.8 million

Method	BLEU	WER	TER
proposed (0)	30.79	0.663	0.554
proposed (3)	30.97	0.665	0.554
proposed (6)	**31.21**	**0.660**	**0.549**
proposed (9)	31.11	0.661	**0.549**
proposed (12)	30.98	0.662	0.551
proposed (15)	31.00	0.662	0.552
no va (6)	30.99	0.669	0.559
Organizer	*30.58*	*0.755*	*0.592*

Table 2: Automatic Evaluation of Translation Quality (Numbers in parentheses indicate distortion limits).

sentence pairs, only 11 sentences had $\tau < -0.5$ among the 1.8 million sentences.

4.2 Automatic Evaluation of Translation Quality

In general, it is believed that translation between English and Japanese requires a large *distortion limit* (dl), which restricts how far a phrase can move. SMT reasearchers working on E-J or J-E translation often use **dl=−1 (unlimited) as a default value, and this takes a long translation time.**

For PATMT J-E translation, Katz-Brown and Collins (2008) showed that dl=unlimited is the best and it requires a very long translation time. For PATMT E-J translation, Kumai et al. (2008) claimed that they achieved the best result "*when the distortion limit was 20 instead of* -1."

Table 2 compares the single-reference BLEU score of the proposed method and that of the Moses-based system by the NTCIR-7 PATMT organizers. This organizers' system was better than all participants (Fujii et al., 2008) in terms of BLEU. Here, we used Bleu Kit (`http://www.mibel.cs.tsukuba.ac.jp/norimatsu/bleu_kit/`) following the PATMT's overview paper (Fujii et al., 2008). The table shows that dl=6 gives the best result, and even dl=0 (no reordering in Moses) gives better scores than the organizers' Moses.

Table 2 also shows Word Error Rates (WER) and Translation Error Rates (TER) (Snover et al., 2006). Since they are error rates, smaller is better. Although the improvement of BLEU is not very impressive, the score of WER is greatly reduced. This difference comes from the fact that BLEU measures only local word order, while WER measures global word order. Another line 'no va' stands for our method without vas or particle seeds. Without particle seeds, all scores slightly drop.

Method	ROUGE-L	IMPACT	PER
proposed (6)	**0.480**	**0.369**	0.390
no va (6)	0.475	0.368	0.398
Organizer	*0.403*	*0.339*	**0.384**

Table 3: Improvement in word order

Echizen-ya et al. (2009) showed that IMPACT and ROUGE-L are highly correlated to human evaluation in evaluating J-E patent translation. Therefore, we also used these evaluation methods here for E-J translation. Table 3 shows that the proposed method is also much better than the organizers' Moses in terms of these measures. Without particle seeds, these scores also drop slightly.

On the other hand, Position-independent Word Error Rate (PER), which completely disregards word order, does not change very much. These facts indicate that our method improves word order, which is the most important problem in E-J translation.

The organizers' Moses uses dl=unlimited, and it has been reported that its MERT training took two weeks. On the other hand, our MERT training with dl=6 took only eight hours on a PC: Xeon X5570 2.93 GHz. Our method takes extra time to parse sentences by Enju, but it is easy to run the parser in parallel.

5 Discussion

Our method used an HPSG parser, which gives rich information, but it is not easy to build such a parser. It is much easier to build word dependency parsers and Penn Treebank-style parsers. In order use these parsers, we have to add some heuristic rules.

5.1 Word Dependency Parsers

At first, we thought that we could substitute a word dependency parser for Enju by simply rephrasing a **head** with a **modified word**. Xu et al. (2009) used a semantic head-based dependency parser for a similar purpose. Even when we use a syntactic head-based dependency parser instead, we encountered their 'excessive movement' problem.

A straightforward application of their rules changes

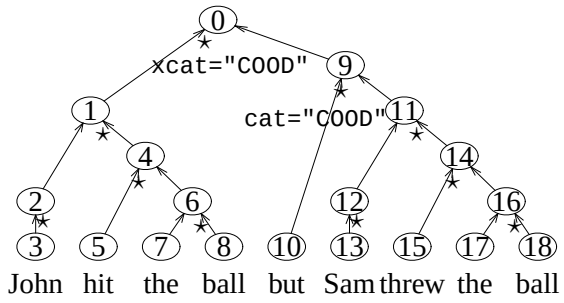

Figure 5: Head Finilization does not mix up clauses

`(0) John hit the ball but Sam threw the ball.`

to

`(1) John the ball but Sam the ball threw hit.`

Here, the two clauses are mixed up. To prevent this, they disallow any movement across punctuation and conjunctions. Then they get a better result:

`(2) John the ball hit but Sam the ball threw.`

When we used Enju, these clauses were not mixed up. Enju-based Head Finalization gave the same word order as (2):

`(3) John va1 ball va2 hit but Sam va1 ball va2 throw.`

Figure 5 shows Enju's parse tree. When Head Finalization swaps the children of a mother node, the children do not move beyond the range of the mother node. Therefore, Head Finalization based on Enju does not mix up the first clause `John hit the ball` covered by Node 1 with the second clause `Sam threw the ball` covered by Node 11. Moreover, our coordination exception rule keeps the order of these clauses. Thus, nonterminal nodes in Enju's output are useful to protect clauses.

When we use a word-dependency parser, we assume that the modified words are heads. Furthermore, the Head Finalization rule is rephrased as "move modified words after modifiers." Therefore, `hit` is moved after `threw` just like (2), and the two clauses become mixed up. Consequently, we need a heuristic rule like Xu's.

5.2 Penn Treebank-style parsers

We also tried Charniak-Johnson's parser (Charniak and Johnson, 2005). PyInputTree (`http://www.cs.brown.edu/~dmcc/software/PyInputTree/`) gives heads. Enju outputs at most two children for a mother node, but Penn Treebank-style parsers do not have such a limitation on the number of children. This fact causes a problem.

When we use Enju, '`This toy is popular in Japan`' is reordered as '`This toy va1 Japan in popular is`.' Its monotonic translation is fluent: *kono omocha wa nihon de ninki ga aru.*

On the other hand, Charniak-Johnson's parser outputs the following S-expression for this sentence (we added asterisks (`*`) to indicate heads).

```
(S (NP (DT This) (NN* toy))
  (VP* (AUX* is)
     (ADJP (JJ* popular))
     (PP (IN* in) (NP (NNP* Japan)))))
```

Simply moving heads to the end introduces '`Japan in`' between '`is`' and '`popular`': *this toy va1 popular Japan in is*. It is difficult to translate this monotonically because of this interruption.

Reversing the children order (Xu et al., 2009) reconnects `is` and `popular`. We get '`This toy (va1) Japan in popular is`' from the following reversed S-expression.

```
(S (NP (DT This) (NN* toy))
  (VP* (PP (IN* in) (NP (NNP* Japan)))
     (ADJP (JJ* popular))
     (AUX* is)))
```

5.3 Limitation of Head Finalization

Head Finalization gives a good first approximation of Japanese word order in spite of its simplicity. However, it is not perfect. In fact, a small distortion limit improved the performance.

Sometimes, the Japanese language does not have an appropriate word for monotonic translation. For instance, '`I have no time`' becomes '`I va1 no time va2 have`.' Its monotonic translation is '`watashi wa nai jikan wo motteiru`,' but this sentence is not acceptable. An acceptable literal translation is '`watashi wa jikan ga nai`.' Here, '`no`' corresponds to '`nai`' at the end of the sentence.

6 Conclusion

To solve the word-order problem between SVO languages and SOV langugages, we introduced a new reordering rule called **Head Finalization**. This rule is simple, and we do not have to consider POS tags or rule weights. We also showed that this reordering improved automatic evaluation scores of English-to-Japanese translation. Improvement of the BLEU score is not very impressive, but other evaluation scores (WER, TER, LOUGE-L, and IMPACT) are greatly improved.

However, Head Finalization requires a sophisticated HPSG tagger such as Enju. We showed that severe failures are caused by Enju's POS tagging mistakes. We discussed the problems of other parsers and how to solve them.

Our future work is to build our own parser that makes fewer errors and to apply Head Finalization to other SOV languages such as Korean.

Acknowledgements

We would like to thank Dr. Yusuke Miyao for his useful advice on the usage of Enju. We also thank anonymous reviewers for their valuable suggestions.

References

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 173–180.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of the Language Resources and Evaluation Conference (LREC)*, pages 449–454.

Hiroshi Echizen-ya, Terumasa Ehara, Sayori Shimohata, Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, Takehito Utsuro, and Noriko Kando. 2009. Meta-evaluation of automatic evaluation methods for machine translation using patent translation data in NTCIR-7. In *Proceedings of the 3rd Workshop on Patent Translation*, pages 9–16.

Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. 2008. Overview of the patent translation task at the NTCIR-7 workshop. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, pages 389–400.

Gumwon Hong, Seung-Wook Lee, and Hae-Chang Rim. 2009. Bridging morpho-syntactic gap between source and target sentences for English-Korean statistical machine translation. In *Proc. of ACL-IJCNLP*, pages 233–236.

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 *Working Notes of the NTCIR Workshop Meeting (NTCIR)*.

Philipp Koehn, 2010. *MOSES, Statistical Machine Translation System, User Manual and Code Guide*.

Hiroyuki Kumai, Hirohiko Segawa, and Yasutsugu Morimoto. 2008. NTCIR-7 patent translation experiments at Hitachi. In *Working Notes of the NTCIR Workshop Meeting (NTCIR)*, pages 441–444.

Chi-Ho Li, Dongdong Zhang, Mu Li, Ming Zhou, Minghui Li, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 720–727.

Thai Phuong Nguyen and Akira Shimazu. 2006. Improving phrase-based statistical machine translation with morphosyntactic transformation. *Machine Translation*, 20(3):147–166.

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

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

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

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*.

Kristina Toutanova and Hisami Suzuki. 2007. Generating case markers in machine translation. In *Proc. of NAACL-HLT*, pages 49–56.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. of the International Conference on Computational Linguistics (COLING)*, pages 508–514.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve SMT for Subject-Object-Verb languages. In *Proc. of NAACL-HLT*, pages 245–253.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 523–530.