Patterns for Simplifying Complex Sentences in English-Japanese Machine Translation

Chinh To Kevin Duh Mamoru Komachi Shuhei Kondo Yuji Matsumoto
Nara Institute of Science and Technology

Abstract: This study explores in detail the question of complex sentences in English-to-Japanese Statistical Machine Translation (SMT). Complex sentences pose the most difficulty to existing SMT systems, due to large word order differences between SVO and SOV languages. In order to overcome this problem, we take a "divide and rewrite" approach: a complex sentence is divided into simple clauses based on syntactic patterns; then simple clauses are translated and the results are pieced together to form the final output.

The main challenge of such "divide and rewrite" preprocessing approaches is the construction of syntactic patterns. While previous works focus on either automatic or manual methods, we pursue a semi-automatic approach. First, we automatically extract and cluster patterns of dependent clauses based on source-side parses. Our novel definition of pattern templates enables us to reduce all sources of syntactic variations into a small set of 100 clusters. Then it becomes feasible to perform a manual construction of corresponding target-side patterns. In our experiments, we demonstrate that this cost-effective approach covers 82 percent of all complex sentences and improves BLEU by over 2 points over the baseline.
Keywords: sentence pattern, complex sentence, functional word, machine translation

1. Introduction

Complex sentences present significant challenges to existing Statistical Machine Translation (SMT) systems, especially for languages pairs with considerable re-ordering. A complex sentence is a sentence containing one or more subordinating (dependent) clauses. For a language pair such as English (SVO) and Japanese (SOV), the issue of long-distance word re-ordering, compounded with the possibility of intervening subordinating clauses, makes it hard to place the verb, for example, in the correct position.

Much effort has been put into the re-ordering problem, with both preprocessing and integrated approaches being effective. The former "rewrites" the original input E to an alternative sentence E' that is easier to handle (Xia and McCord, 2004), while the latter incorporates a reordering model directly into search (Chiang, 2005). Surprisingly, although SMT re-ordering is an active research area, only a few previous works focused on the related problem of complex sentence structure (Sudoh et. al., 2010; Ramanathan et. al., 2011).

In this paper, we adopt the preprocessing approach of (Sudoh et. al., 2010), where a complex sentence is divided into simple clauses and translated independently. This approach has proven very effective for language pairs such as English-Japanese. It requires a source-side (English) syntactic parser to identify clauses (via SBAR). Identified clauses are replaced by non-terminals as place-holders; after translation, the independent results are reconstructed using non-terminals.

A limitation of this previous method is that it treats all kinds of dependent clauses in the same way. The corresponding target-side rules are learned automatically from word alignments, as in conventional phrase-based SMT (Koehn et. al., 2003). It is not clear whether noisy word alignments are reliable for extracting generalizable target-side rules. We believe it is much more desirable to process dependent clauses differently based on context. In fact, we find it is possible to manually-craft these contextual target-side rules based on the set of pattern clusters and function word indicators on the source-side.

The main contribution of this paper is a semi-automatic and linguistically motivated methodology for developing effective "divide and rewrite" patterns for complex sentences. By semi-automatic, we mean that our source-side patterns are extracted and clustered automatically, while our target- side patterns are developed manually based on linguistic observation. This is in contrast to most preprocessing approaches, which extract patterns either fully automatically (Genzel, 2010; Visweswariah, 2011) or fully manually (Xu, 2009; Chao et. al., 2007).

A second contribution is a novel definition of pattern templates that proved effective for English- Japanese. Our source-side patterns are based on SBAR, it's parents, sisters, daughters and related function words. Though simple, these patterns are easy to extract and cluster, leading to cost- effective manual analysis. The list of patterns useful for translation will be available at http://blind.review.

Finally, we note that our emphasis is on ***clausal*** reordering in complex sentences. The numerous previous research on ***word*** ordering are orthogonal to our work here. Certainly high-level clausal rewrites and low-level word reordering can work in tandem.

A second contribution is a novel definition of pattern templates that proved effective for English-Japanese. Our source-side patterns are based on each subordinating clause, it's parent, sisters, daughters and related function words. Though simple, these patterns are easy to extract and cluster, leading to cost-effective manual analysis.

Finally, we note that our emphasis is on *clausal* reordering in complex sentences. The numerous previous researches on *word* ordering are orthogonal to our work here. Certainly high-level clausal rewrites and low-level word reordering can work in tandem.

2. Divide and rewrite approach

This study deals with translation of complex sentences from English to Japanese by using the typical patterns extracted to divide and rewrite a complex sentence into simple structures. The Oxford Dictionary (2000) defines that a complex sentence is a sentence containing one or more subordinating clauses.

In order to examine and extract typical patterns of complex sentences, we produce a parse tree for each sentence in the training data using Berkeley Parser. Subordinating clauses, which are introduced by a (possibly empty) subordinating conjunction, are marked as "SBAR" [1] in a Berkeley parse tree. Subordinating clauses will be referred to as "SBAR".

The data used for experiments is data from the magazine Hiragana Times [2]. Hiragana Times is a bilingual magazine written in Japanese and English to introduce Japan to non-Japanese, covering a wide ranges of topics including culture, society, history, politics... The data contains 172,098 sentences crawled in October 2011.

Our approach for complex sentence simplification can be divided into two stages. In the preparation stage, we first extract source patterns (Section 3.1), cluster them into a manageable number (Section 3.2), and develop corresponding target patterns (Section 3.3). In the deployment stage, we simply look for source pattern matches in the input, translate the clauses, and recombine the results using target patterns.

2.1 Extracting patterns of complex sentences

For each sentence in the training data we first produce a parse tree to examine its patterns. We assume that the structure of a subordinating clause in a complex sentence together with its functional words and its relations to other components in the sentence can be modeled to a set of manageable patterns. It is expected that these patterns will be used as useful clues either to divide and rewrite sentences or to create rules for the translation of particular patterns having ambiguity.

For each SBAR node in a parse tree, a pattern is extracted as follows: for the SBAR's parent and sister nodes, the phrase names are extracted. This information would help to define where a SBAR belong to and which functions it may have. For SBAR's child node that contains function words, *all* of the phrase name, the POS tag and the surface word are extracted. The function words are extracted because they are important elements, which can express the grammatical relationship between clauses in a sentence. Figures 1 and 2 illustrate how the pattern "(NP (NP) (SBAR (WHNP (WP who)) (S (VP))))" is extracted.

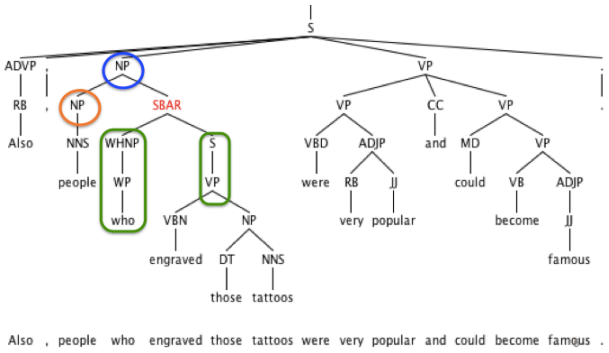

Figure 1: Example of a complex sentence and its SBAR

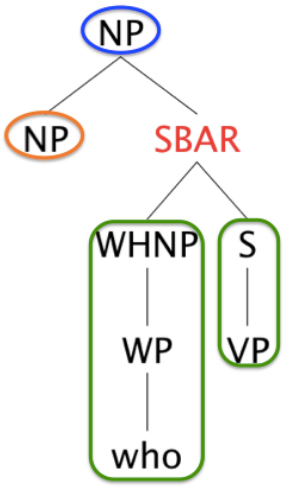

Figure 2: Example of an extracted pattern

In Figures 1 and 2, the pattern "(NP (NP) (SBAR (WHNP (WP who)) (S (VP))))" is extracted, the SBAR has a parent node as "NP" and a sister node as "(NP)", the functional word of SBAR is "who", and this SBAR has the structure as a verb phrase, "VP". With the related information, the pattern can be described as a subordinating clause composed of the functional word "who" and having a child node that consists only of a verb phrase, "VP". The clause modifies a noun phrase "(NP)" and belongs to a noun phrase "NP".

By producing a parse tree for each sentence, each pattern is extracted as described above. Table 1 shows that we generated 21k distinct patterns from our dataset of 70k complex sentences.

Number of sentences	171,098
Number of complex sentences	70,134
Number of SBARs	114,840
Number of distinct SBAR patterns	21,090

Table 1 - Statistics of patterns in complex sentences in training data.

2.2 Acquiring a compact set of typical patterns

The raw number of extracted source patterns is too large for

1 http://bulba.sdsu.edu/jeanette/thesis/PennTags.html
2 http://www.hiraganatimes.com/index-j.html

manual construction of corresponding target patterns. Thus an important question is, are there patterns that translate into similar structures? Is it possible to cluster them in an organized, easily-interpretable way?

Each pattern has its frequency number showing how many times it appears in the training data. There are patterns that appear much more than others. Though the number of all distinct patterns is 21,090, the top 10 distinct patterns, which have the highest frequencies, account for 25.8 percent of all the times when a SBAR is found (Table 2). Sequentially, the top 100 and 200 distinct patterns account for 53 and 59 percent for instance.

Rank	SBAR Pattern	Freq.
1	(NP (NP) (SBAR (WHNP (WP who)) (S (VP))))	6880
2	(NP (NP) (SBAR (S (NP) (VP))))	4928
3	(NP (NP) (SBAR (WHNP (WDT that)) (S (VP))))	4741
4	(NP (NP) (SBAR (S (VP))))	2998
5	(VP (VBP) (SBAR (S (NP) (VP))))	2743
6	(VP (VBD) (SBAR (IN that) (S (NP) (VP))))	1552
7	(VP (VB) (SBAR (IN that) (S (NP) (VP))))	1549
8	(VP (VBD) (SBAR (S (NP) (VP))))	1520
9	(VP (VBZ) (SBAR (IN that) (S (NP) (VP))))	1388
10	(VP (VBP) (SBAR (IN that) (S (NP) (VP))))	1343
	(sum)	29642 / 114840 (25.8%)

Table 2: The top 10 patterns of highest frequencies

Actually, it is interesting to note that the patterns exhibit Zipf-like distributions. We can see cumulative frequency of all distinct patterns in Figure 3. The graph shows the number of distinct patterns on X-axis versus frequency on the Y-axis. It is clear that a only a small number of top-ranking patterns can model a big portion of all the patterns found in training data.

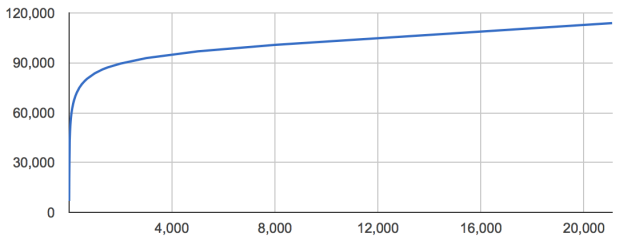

FIGURE 3: Cumulative frequency of all distinct patterns (x axis: Number of distinct patterns; y axis: Cumulative frequency)

So, it is expected that a small number of patterns can be used to summarize most of complex sentences. How such a manageable set of typical patterns would be extracted?

Examining the extracted patterns, we found that there are patterns that share similar structure and features. These patterns can be grouped together as one cluster; this will help to summarize the extracted pattern in an interpretable and organized way.

We define the following criteria for grouping together source patterns: Their SBAR nodes must have (1) a child node containing the same function words, (2) the same parent node label, which would tell where they belong to and what kind of function they may have, and (3) a matching non-terminal structure.

Rank	SBAR Pattern	Freq.
1	(NP (NP) (SBAR (WHNP (WP who)) (S (VP))))	6880
21	(NP (NP) (,) (SBAR (WHNP (WP who)) (S (VP))))	646
151	(NP (NP) (,) (SBAR (WHNP (WP who)) (S (ADVP) (VP))))	51
557	(NP (NP) (ADJP) (SBAR (WHNP (WP who)) (S (VP))))	13

TABLE 3: Examples of distinct patterns grouped together (underlined elements must be shared)

Table 3 shows an example of distinct patterns that are grouped together into one grouped pattern. Each distinct pattern has its frequency ranked as seen in the left column of the table. In this case, in each pattern, one SBAR child node contain the same functional word "who", the basic structure of each SBAR contains a verb phrase component as "(VP)", and all the SBAR in each pattern belongs to a noun phrase "NP", modifying a noun phrase "(NP)".

We give a note about functional words here. We have a list of all function words, which are related to SBAR and often considered as subordinating conjunctions (e.g. "after", "although", "as", "because", "before", "if", "though", "unless", "what", "when", "who",…). We can consider and analyze all patterns with their function words. As for function words like "of", we do not consider this kind of function words because it is not related (or connected) directly to a SBAR.

The result of grouping similar pattern together is that we get an organized compact set of typical patterns in complex sentences. The set includes 100 typical pattern clusters, which cover 82% of all the patterns extracted in complex sentences in the training data. (See Appendix 1 for the whole list of the typical patterns.). For the reason we decided the number of pattern clusters to 100, as shown in Figure 3, we plotted all the patterns and cumulative frequency and the patterns have a Zipf-like distribution. It means that a number of the top high frequency patterns can cover a big number of all sentences. For example, looking at the coverage of the top ranking patterns, we can see that the top 100 patterns have the coverage 82%, while the top 200 increase this number to 85%, and the top 300 can cover 87% of all sentences. So, 100 is a good and convenient start to summarize patterns, we can edit further, for instance, 300 top patterns for higher coverage.

We describe the algorithm to classify similar patterns together into one group as follows.

A SBAR normally have the basic structure as follows.

i. (SBAR (functional words X) (S (NP) (VP))) or

ii. (SBAR (functional words X) (S (VP)))

Subsequently, each pattern of SBAR normally has the basic structure as follows.

i. (PARENT (SISTER)* (SBAR (functional words X) (S (NP) (VP)))) or

ii. (PARENT (SISTER)* (SBAR (functional words X) (S (VP))))

Each of the basic structures above can have various modifying parts such as adverbs, prepositions, parenthetical phrases... For examples (SBAR (functional words X) (S (NP) (VP))) would have additional modifying parts as (SBAR (functional words X) (S (ADVP) (PP) (NP) (VP))), while (PARENT (SISTER)* (SBAR (function words X) (S (VP)))) would have additional modifying parts as (PARENT (ADVP) (PP) (SISTER)* (SBAR (function words X) (S (VP))))...

Pseudo-code to group similar patterns together:

```
for pattern_k = '(PAR (SIS)* (SBAR (functionwords X)(S (VP))))'  do
add(.*)→(PAR(.*)(SIS)*(.*)(SBAR(.*)(function_wordsX)(.*)(S(.*  )(VP)
(.* )))
pattern_string=(PAR(.*)(SIS)*(.*)(SBAR(.*)(function_wordsX)(.*)(S(.*
 )(VP)(.* ))
        regular_exp_pattern_k = Regexp.new(pattern_string)
        for i in 0 ... set_of_patterns.size do
            if regexp_No.k =~ pattern[i] then
                group pattern[i] into the same group of pattern_k
                delete pattern[i] out of set_of_patterns
```

When grouping patterns, we use regular expression and we order the top 100 patterns in the way that conflicting match problems are solved. For example, concerning the following patterns:

i. (PARENT (SISTER)* (SBAR (function words X) (S (NP) (VP)))) or

ii. (PARENT (SISTER)* (SBAR (function words X) (S (VP))))

Both of them can match with patterns like "(PARENT (SISTER)* (SBAR (function words X) (S (ADVP) (NP) (VP)))) ".

We order so that the pattern " (SBAR (function words X) (S (NP) (VP)))" would be checked and processed before " (SBAR (function words X) (S(VP))) ". Also, once a pattern is found matched with any in the clusters it will be deleted, so it would not matched with any other pattern cluster.

All the patterns in one cluster will have the same reordering because they share the same main structure, the different parts they have are mainly modifying parts like prepositions, and adverbs.

2.3 Using patterns to simplify and rewrite

2.3.1 Dividing and rewriting sentences based on patterns

There are several considerations when developing target-side patterns. First, one needs to decide which part of the pattern should be separated for translation. For example with patterns such as '(NP (NP) (SBAR (WHNP (WP who)) (S (VP))))', the whole node (include "who") should be translated to Japanese as a unit. On the other hand, for patterns such as '(VP (VB*) (SBAR (IN that) (S (SBAR) (NP) (VP)))', only the SBAR part '(S (SBAR) (NP) (VP))' needs to be translated (without "that"). (VB* stands for VP clusters, i.e., VB, VBP, VPZ, VPD…)

In order to do that, we decide whether a separated part can be treated as an independent structure in relation to other components; or the separated part is treated as an integral part in relation to other components in a sentence. (Here an integral part means a part that must be considered together with other related parts in certain context. Whereas an independent structure is a unit that can convey an independent meaning, it has independent relations with other components of the same level in a sentence.)

So, for each separated part, related information including its parents and children are kept and used. Table 4.1 shows an example of 2 matched patterns in a sentence, information about each pattern and its related elements are kept to divide, rewrite and later reconstruct the sentence.

How patterns are used to rewrite a sentence is described as follows.

First, the sentence is checked if it matches any of the patterns. The result is the sentence has 2 matches.

Sentence: If the Sea Group wins it means there will be a good haul during the year.
Match1: Pattern (VP (VB) (SBAR (S (NP) (VP)))) Part to be divided (SBAR (S (NP) (VP))): there will be a good haul during the year Parents: Match2; Children: no
Match2: Pattern (S (SBAR (IN If) (S (NP) (VP))) (NP) (VP)) Part to be divided (S (SBAR (IN If) (S (NP) (VP))) (NP) (VP)): If the Sea Group wins it means there will be a good haul during the year. Parents: Sentence; Children: Match1

TABLE 4.1: A sentence with two matched patterns, each match keeps information about itself and the relation to other matches and the sentence.

Next, for each match the pattern is examined. Which part of the pattern will be divided and rewritten is decided based on the matched pattern. Regarding the matched pattern "(VP (VB) (SBAR (S (NP) (VP))))", the part "(SBAR (S (NP) (VP))))"is divided and the sentence is rewritten using non-terminal symbols X0.

Then the pattern "(S (SBAR (IN if | If) (S (NP) (VP))) (NP) (VP))" is examined and the pattern is divided into two simple parts, which correspond to "(SBAR (IN if | If) (S (NP) (VP)))" and "(NP) (VP)" parts.

The sentence and related parts are rewritten in Table 4.2 as follows.

Rewrite	when process Match. 1	when process Match. 2
Sentence	If the Sea Group wins it means X0.	S1 { position_marker : S1, S1_1, S1_2}
Match1	X0: there will be a good haul during the year	X0: there will be a good haul during the year
Match2	S1: If the Sea Group wins it means X0.	S1_1: If the Sea Group wins; S1_2: it means X0.

TABLE 4.2: The patterns of Match1, 2 are processed in sequence. When a pattern is processed, its Parent (if exist) and the sentence will be updated.

Finally, the divided and rewritten parts are translated. The translation results are reconstructed using non-terminal symbols and information of the position marker. Table 4.3 illustrates the divided and rewritten parts are translated, and reconstructed. When all the patterns of Match1, 2 are processed in sequence and the sentence and related parts are updated we get the final parts for translation and reconstruction.

Sentence: If the Sea Group wins it means there will be a good haul during the year. (position marker : S1, S1_1, S1_2)	
Divided & Rewritten: X0: there will be a good haul during the year S1_1: If the Sea Group wins S1_2: it means X0.	Translation results: 年度中に豊漁があるだろ 海グループが勝てば それはX0を意味します。
Reconstruct: 海グループが勝てばそれは年度中に豊漁があるだろうを意味します。	

TABLE 4.3: The divided and rewritten parts are translated, non-terminal symbols (e.g. X0) are replaced by translation results, and position marker is used to reconstruct the parts.

We present the pseudo-code to divide, rewrite and reconstruct a sentence having matched patterns as follows.

```
#get relations among matched parts (order all matches in sequence 0,1,...,n)
    find the relations (Parents, Children ) among matches :
    if match_i include match_j then
        match_i.Child = match_j, match_j.Parents = match i
    if match_i. Parent = nil then
        set match_i.Parents = Sentence
#divide, translate (process each match in sequence 0,1,2,..,n) :
    for match_i do:
        if pattern is decided as an 'integral part':
            seperate the related part: X_i = separated_part;
            update sentence, X_i's Parents;
            translate X_i → X_i_ja (Japanese translation);
        if pattern is considered as an 'independent structure':
            separate the related part: S_i=separated_part;
            save the relative position of S_i in position_holder;
            divide S_i (if necessary) to simpler parts as S_i*
            (e.g. S_i.1, S_i.2,… );
            save the position of S_i* to position holder;
            update sentence, S_i's Parents with S_i, S_i*;
            translate S_i* → S_i*.ja
        translate rewritten_sentence → sentence.ja
#reconstruct:
    replace non-terminal symbols X_i in sentence.ja with X_i_ja
    use position_holder {S_i, S_i*}to reconstruct S_i*.ja to S_i.ja;
    put S_i.ja to sentence
```

2.3.2 Creating rules for patterns with multiple translations

The second consideration is that there may be patterns that have multiple translations in the target language. By examining different sample sentences containing the same pattern, we found that a number of patterns have multiple translations in the target language. One example is the case of patterns "(S (SBAR (IN Since | since) (S (NP) (VP))) (NP) (VP))", "(VP (VB*) (NP) (SBAR (IN since) (S (NP) (VP))))", the meaning of the function word 'since' can be either time-related or reason-related, and therefore translate to different Japanese patterns. The meaning of the functional word "since" in this pattern is defined with the meaning as "in the intervening period between (the time mentioned) and the time under consideration, typically the present" or the meaning as "for that reason that; because " (Oxford Dictionary, 2000). Table 5 shows a list of patterns, which have multiple translations.

Sbar patterns having multiple translations
(S (SBAR (IN As | as) (S (NP) (VP))) (NP) (VP))
(S (SBAR (IN Since | since) (S (NP) (VP))) (NP) (VP))
(S (SBAR (IN While | while) (S (NP) (VP))) (NP) (VP))
(VP (VB*) (NP) (SBAR (IN while) (S (NP) (VP))))
(VP (VB*) (SBAR (IN since) (S (NP) (VP))))
(VP (VB*) (SBAR (IN if) (S (NP) (VP))))
(VP (VB*) (SBAR (IN that) (S (NP) (VP))))
(VP (VB*) (SBAR (WHNP (WDT what)) (S (NP) (VP))))
(VP (VB*) (SBAR (WHADVP (WRB when)) (S (NP) (VP))))
(VP (VB*) (SBAR (WHADVP (WRB where)) (S (NP) (VP))))
(ADJP (JJ) (SBAR (IN that) (S (NP) (VP))))
(ADVP (ADVP) (SBAR (IN that) (S (NP) (VP))))
(NP (NN) (SBAR (IN that) (S (NP) (VP))))

Table 5: Examples of patterns having multiple translations

It is necessary to identify a correct translation when a pattern has multiple translations. The information and features of nodes in the pattern can be used as clues to choose a correct translation.

For instance, the pattern "(VP (VB*) (NP) (SBAR (IN since) (S (NP) (VP))))", the functional word "since" have two different translations in the target language. We observed that the tense in the node "(VB*)" and the node "(VP)" are important factors to affect the meaning of "since". Indeed, out of 95 sample sentences, which have the tenses in the node "(VB*)" and the node "(VP)" as present perfect tense and past tense, 90 sentences have the meaning related to time. The present perfect tense and the past tense combination in the pattern actually expresses the time–related meaning of "since". On the other hand, when the two nodes have the same tense as "simple present tense- simple present tense" or "simple past tense – simple past tense" then all 45 sample sentences have "since" translated as "for the reason" in the target language. For each pattern, it is possible to examine and extract the features of nodes. Tense in "VP" nodes, type and features of verbs and other nodes can help to identify a correct translation.

3. Experiment

To train a translation model, we use Moses 3 as a decoder and GIZA++ 1.0.7 4 as an alignment tool. We use the bilingual general-domain magazine corpus which consists of around 170,000 sentence pairs from Hiragana Times, and performed the standard preprocessing steps (e.g. sentence length filtering) in the Moses training pipeline. We split the parallel data into two parts: 1000 sentence pairs for testing and development (500 pairs for testing, 500 for development), and the rest 171,098 sentence pairs for training. We also used Google Translate as a second MT engine.

BLEU score is used to evaluate the translation results of test sentences with and without preprocessing. In testing, those sentences with patterns matched are divided and rewritten for translation. The complex sentences that have matched patterns will be preprocessed (e.g. divided and rewritten into simple parts). Simple sentences and complex sentences which do not have matched patterns would not be preprocessed at all. Around 80% of all complex sentences have matched patterns with the pattern set we extracted. Actually, there are 232 complex sentences found in the test data of 500, and 185 complex sentences (covering 79.74 percent) have patterns matched with those of the typical pattern set.

3.1 Experiment results

Table 6 shows the score for the translation results of test data when preprocessing is performed and when no preprocessing is performed. In the table, the columns without* indicate scores of the whole test data set, the *columns indicate BLEU score for only complex sentences. With the proposed method, we get an improvement of 2.23 point in BLEU score, if considering only complex sentences, the increase is over 3 points with Moses.

	Moses	Moses*	Google	Google*
w/o preprocessing	15.26	12.84	24.36	15.61
w/ preprocessing	**17.49**	**15.97**	**24.73**	**16.43**

TABLE 6: BLEU scores using Moses and Google Translate.

With Google Translate, significant improvements are shown clearly at long and/or complicated sentences. In sentences, where there are more than one dependent clauses, SMT systems fail to recognize the borders of these clauses, therefore create wrong word order if no preprocessing is performed.

In addition, we implemented the word-level reordering preprocessing rules of Xu et al. [14] on top of our system. This led to very promising results, with a further BLEU increase of 2 points. When the two methods are applied together, the clause-level reordering would help to simplify sentences, while the word-level reordering rules would help to overcome the restriction of relative distance reordering in Moses decoder. So, the combination of different reordering approaches, at clause-level or at word-level would be important to achieve a significant improvement.

3.2 Analysis

We observed that test sentences of which the translations have been improved mainly fall to the kind of long complex sentences, complex sentences which have more than one subordinating clauses, complex sentences which have a specific expression, or complex sentences which have more complicated structures.

Table 7 shows examples of improved translations. For each cell the first English sentence is the one to be translated, the following two Japanese sentences are, in sequence, translations by MT system when no preprocessing is performed and when our proposed preprocessing is performed, the last one in the cell is the reference translation.

1. When I went to the hospital near the supermarket, a woman who was very beautiful looked at me and said hello. 私はスーパーマーケットの近くに病院、非常に美しかった私を見て挨拶した女性に行ったとき。(without preprocessing) 私はスーパーマーケット近くの病院に行ったとき、とても美しかった女性は私を見て挨拶した。(with preprocessing) 私はスーパーの近くにある病院に行った時、とても美しい女性が私を見て挨拶した。(Reference translation)
2. She said that he gave the boys and the girls who lived next door delicious red apples. 彼女は、彼が少年と隣の美味しい赤いリンゴに住んでいた女の子を与えたと述べた。 彼女は彼が男の子と隣に住んでいた女の子美味しい赤いリンゴを与えたと述べた。 彼女は彼が隣に住んでいた男の子と女の子たちに美味しくて赤いりんごをあげたと言った。
3. I can say for sure that you made a big mistake at the job interview .

3 http://www.statmt.org/moses
4 http://code.google.com/p/giza-pp/

私はあなたの就職の面接で大きなミスを犯していることを確認のために言うことができます。 私は確かにあなたは就職の面接で大きなミスを犯した言うことができます。 少なくとも、あなたは就職面接で、大きな失敗をしたと言えます。

TABLE 7: Examples of translation results

Sentence No.1 and sentence No.2 show translations are improved when complex sentences may have more than one subordinating clauses, MT systems failed to produce a correct word order for the target side because it could not recognize the correct borders of different clauses in a sentence. In both cases, MT systems get correct translations of phrases in the sentences, yet order them wrong.

The adverb phrase “for sure ” in sentence No.3 is translated incorrectly when no preprocessing is performed, yet when preprocessing is done, the “that ...” clause is separated and the sentence’s structure become short and simple, so the MT systems can handle it.

On the other hand, complex sentences that have simple structures get good quality translations even when no preprocessing is done. Table 8 shows examples of sentences which preprocessing does not help to improve translations.

Do you think that it is a right action? あなたは、これが正しい行動だと思いますか？(without preprocessing) あなたは、これは正しい動作と思いますか？ (with preprocessing) それは正しい行動だとあなた は 思いますか？(Reference Translation)
Punctuality is one of the disciplines you should observe in Japanese society. 時間厳守は、日本の社会の中で遵守すべき分野の一つです。 時間厳守は、あなたは日本社会で守るべき規律の一つです。 時間厳守は、日本社会の中での常識の一つといえます。

TABLE 8: Translations with no improvement when preprocessing

4. Conclusion and perspectives

We addressed the problem of complex sentences in MT by proposing a “divide and rewrite” approach. Our semi-automatic method automatically discovers a small set of typical source SBAR patterns, enabling us to craft linguistically-motivated target-side rules. We demonstrate that with merely 100 patterns, our approach covers 82 percent of complex sentences and improves BLEU by over 2 points over the baseline.

We would like to do further evaluation on other (in particular, out-of-domain) datasets to test and justify the efficiency of the set of typical patterns. Besides, the extracted typical patterns and related rules are expected to work well on not only English-Japanese translation but also English to other languages.

We plan to perform further experimentation on combination with word-based reordering patterns because this approach process and divide sentences at clause-level. It would be desirable to find out a combining model for a further substantial improvement.

We would like to develop alternative machine-learning based methods for clustering source patterns. This is because after patterns have been sorted into clusters, we further observed that there are clusters that share common features and behaviors. Therefore, we would like to learn the rules automatically to increase the number of patterns and raise the range of complex sentences, which would match with the patterns.

Appendix

LIST OF 100 TYPICAL CLUSTERS OF PATTERNS

1. (S (SBAR (IN [A|a]fter) (S (NP) (VP))) (NP) (VP))
2. (S (SBAR (IN [A|a]lthough) (S (NP) (VP))) (NP) (VP))
3. (S (SBAR (IN [A|a]s) (S (NP) (VP))) (NP) (VP))
4. (S (SBAR (IN [B|b]ecause) (S (NP) (VP))) (NP) (VP))
5. (S (SBAR (IN [B|b]efore) (S (NP) (VP))) (NP) (VP))
6. (S (SBAR (RB [E|e]ven) (IN if) (S (NP) (VP))) (NP) (VP))
7. (S (SBAR (RB [E|e]ven) (IN though) (S (NP) (VP))) (NP) (VP))
8. (S (SBAR (IN [I|i]f) (S (NP) (VP))) (NP) (VP))
9. (S (SBAR (IN [I|i]f) (S (NP) (VP))) (VP))
10. (S (SBAR (IN [I|i]n) (NN order) (S (VP))) (NP) (VP))
11. (S (SBAR (IN [O|o]nce) (S (NP) (VP))) (NP) (VP))
12. (S (SBAR (IN [S|s]ince) (S (NP) (VP))) (NP) (VP))
13. (S (SBAR (IN [T|t]hough) (S (NP) (VP))) (NP) (VP))
14. (S (SBAR (IN [U|u]nless) (S (NP) (VP))) (NP) (VP))
15. (S (SBAR (WHNP (WP [W|w]hat)) (S (VP))) (VP))
16. (S (SBAR (WHNP (WP [W|w]hat)) (S (VP))) (NP) (VP))
17. (S (SBAR (WHNP (WP [W|w]hat)) (S (NP) (VP))) (VP))
18. (S (SBAR (WHADVP (WRB [W|w]hen)) (S (VP))) (NP) (VP))
19. (S (SBAR (WHADVP (WRB [W|w]hen)) (S (NP) (VP))) (NP) (VP))
20. (S (SBAR (WHADVP (WRB [W|w]henever)) (S (NP) (VP))) (NP) (VP))
21. (S (SBAR (IN [W|w]hile) (S (VP))) (NP) (VP))
22. (S (SBAR (IN [W|w]hile) (S (NP) (VP))) (NP) (VP))

23. (VP (VB*) (NP) (SBAR (IN as) (S (NP) (VP))))
24. (VP (VB*) (SBAR (IN as) (IN if) (S (NP) (VP))))
25. (VP (VB*) (NP) (SBAR (IN although) (S (NP) (VP))))
26. (VP (VB*) (SBAR (IN after) (S (NP) (VP))))
27. (VP (VB*) (SBAR (IN because) (S (NP) (VP))))
28. (VP (VB*) (NP) (SBAR (IN before) (S (NP) (VP))))
29. (VP (VB*) (ADJP) (SBAR (IN for) (S (NP) (VP))))
30. (VP (VB*) (NP) (SBAR (IN in) (NN order) (S (VP))))
31. (VP (VB*) (SBAR (IN if) (S (NP) (VP))))
32. (VP (VB*) (ADJP) (SBAR (IN if) (S (NP) (VP))))
33. (VP (VB*) (SBAR (IN like) (S (NP) (VP))))
34. (VP (VB*) (ADJP) (SBAR (IN that) (S (NP) (VP))))
35. (VP (VB*) (SBAR (IN since) (S (NP) (VP))))
36. (VP (VB*) (SBAR (IN so) (IN that) (S (NP) (VP))))
37. (VP (VB*) (NP) (SBAR (IN though) (S (NP) (VP))))
38. (VP (VB*) (NP) (SBAR (IN unless) (S (NP) (VP))))
39. (VP (VB*) (NP) (SBAR (IN until) (S (NP) (VP))))
40. (VP (VB*) (NP) (SBAR (IN while) (S (NP) (VP))))
41. (VP (VB*) (NP) (SBAR (IN while) (S (VP))))
42. (VP (VB*) (SBAR (WHADVP (WRB when)) (S (NP) (VP))))
43. (VP (VB*) (SBAR (IN that) (S (NP) (VP))))
44. (VP (VB*) (SBAR (WHADVP (WRB where)) (S (NP) (VP))))
45. (VP (VB*) (SBAR (WHADVP (WRB how)) (S (VP))))
46. (VP (VB*) (SBAR (WHADVP (WRB how)) (S (NP) (VP))))
47. (VP (VB*) (SBAR (WHNP (WP what)) (S (NP) (VP))))
48. (VP (VB*) (SBAR (WHNP (WP what)) (S (VP))))
49. (VP (VB*) (SBAR (WHNP (WDT whatever)) (S (NP) (VP))))
50. (VP (VB*) (SBAR (IN whether) (S (NP) (VP))))
51. (VP (VB*) (SBAR (WHADVP (WRB why)) (S (NP) (VP))))
52. (ADJP (ADJP) (SBAR (IN as) (S (NP) (VP))))
53. (ADVP (ADVP) (SBAR (IN as) (S (NP) (VP))))
54. (ADJP (ADJP) (SBAR (IN than) (S (NP) (VP))))
55. (ADJP (JJ) (SBAR (S (NP) (VP))))
56. (ADJP (JJ) (SBAR (IN that) (S (NP) (VP))))
57. (ADJP (ADJP) (SBAR (IN that) (S (NP) (VP))))
58. (ADVP (ADVP) (SBAR (IN that) (S (NP) (VP))))
59. (ADVP (ADVP) (SBAR (IN than) (S (NP) (VP))))
60. (ADVP (RB) (SBAR (IN that) (S (NP) (VP))))
61. (NP (NP) (SBAR (S (NP) (VP))))
62. (NP (DT) (NN) (SBAR (S (NP) (VP))))
63. (NP (DT) (NN) (SBAR (IN that) (S (NP) (VP))))

64. (NP (NP) (SBAR (IN that) (S (NP) (VP))))
65. (NP (DT) (JJ) (NN) (SBAR (IN that) (S (NP) (VP))))
66. (NP (NN) (SBAR (IN that) (S (NP) (VP))))
67. (NP (NP) (SBAR (WHNP (IN that)) (S (NP) (VP))))
68. (NP (NP) (SBAR (WHNP (WDT that)) (S (ADVP) (VP))))
69. (NP (NP) (SBAR (WHADVP (WRB when)) (S (NP) (VP))))
70. (NP (NP) (SBAR (WHADVP (WRB where)) (S (NP) (VP))))
71. (NP (NP) (SBAR (WHNP (WDT which)) (S (NP) (VP))))
72. (NP (NP) (SBAR (WHNP (WDT which)) (S (VP))))
73. (NP (NP) (SBAR (WHPP (IN in) (WHNP (WDT which))) (S (NP) (VP))))
74. (NP (NP) (SBAR (WHNP (WP who)) (S (NP) (VP))))
75. (NP (NP) (SBAR (WHNP (WP whom)) (S (NP) (VP))))
76. (NP (NP) (SBAR (WHNP (WP who)) (S (VP))))
77. (NP (NP) (SBAR (WHNP (WDT that)) (S (VP))))
78. (NP (NP) (SBAR (WHADVP (WRB why)) (S (NP) (VP))))
79. (NP (NP) (SBAR (IN for) (S (NP) (VP))))
80. (NP (NNS) (SBAR (IN that) (S (NP) (VP))))
81. (PP (IN) (SBAR (WHADVP (WRB how)) (S (VP))))
82. (PP (IN) (SBAR (WHADVP (WRB how)) (S (NP) (VP))))
83. (PP (IN) (SBAR (WHNP (WP what)) (S (NP) (VP))))
84. (PP (IN) (SBAR (WHNP (WP whatever)) (S (NP) (VP))))
85. (PP (IN) (SBAR (WHNP (WP what)) (S (VP))))
86. (PP (IN) (SBAR (WHADVP (WRB when)) (S (NP) (VP))))
87. (PP (IN) (SBAR (IN whether) (S (NP) (VP))))
88. (PP (IN) (SBAR (WHADVP (WRB where)) (S (NP) (VP))))
89. (PP (TO) (SBAR (WHNP (WP what)) (S (NP) (VP))))
90. (S (SBAR (IN If) (S (NP) (VP))) (S) (CC) (S))
91. (S (SBAR (WHADVP (WRB When)) (S (NP) (VP))) (S) (CC) (S))
92. (S (NP) (VP (VB*) (SBAR (S (NP) (VP)))))
93. (S (NP) (VP (VB*) (SBAR (IN that) (S (S) (CC) (S)))))
94. (S (NP) (VP (VB*) (SBAR (S (S) (CC) (S)))))
95. (S (NP) (VP (VB*) (SBAR (IN that) (S (SBAR) (NP) (VP)))))
96. (SBAR (SBAR (WHNP (WP who)) (S (VP))) (CC) (SBAR (WHNP (WP who)) (S (VP))))
97. (SBAR (SBAR (IN that) (S (NP) (VP))) (CC) (SBAR (IN that) (S (NP) (VP))))
98. (SBAR (SBAR (S (NP) (VP))) (CC) (SBAR (IN that) (S (NP) (VP))))
99. (SBAR (SBAR (WHADVP (WRB when)) (S (NP) (VP))) (CC) (SBAR (WHADVP (WRB when)) (S (NP) (VP))))
100. (SBAR (SBAR (IN whether) (S (NP) (VP))) (CC) (RB))

Reference

1) Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietria, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics, 19(2):263-311, 1993.

2) David Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the Association for Computational Linguistics '05*, pp. 263-270, 2005.

3) Dmitriy Genzel. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 376-384, 2010.

4) Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Procceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* , pp. 263-270, 2003.

5) Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callision-Burch, Miles Osborne, and David Talbot. Edinburg system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of International Workshop on Spoken Language Translation*, 2005.

6) Philipp Koehn. Statistical Machine Translation. Cambridge University Press, 2010. (pp. 127-148)

7) Adam Lopez. Statistical Machine Translation. Association for Computing Machinery Computing Surveys, Vol.40, No. 3, Article 8, August, 2008.

8) Ananthakrishnan Ramanathan, Pushpak Bhattacharyya, Kartik Visweswariah, Kushal Ladha, and Ankur Gandhe. Clause-based reordering constraints to improve statistical machine translation. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP),* pp.1351-1355, 2011.

9) Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Tsutomu Hirao and Masaaki Nagata. In *Proceedings of the Joint 5th Workshop on Statistical Machine Translation and MetricsMATR*, pp. 418-427, 2010.

10) Karthik Visweswariah, Rajakrishnan Rajkumar and Ankur Gandhe. A word reordering model for improved machine translation. In *Proceedings of the 2011 Conference Empirical Methods in Natural Language Processing*, pp. 486-496, 2010.

11) Chao Wang, Micheal Collins, and Philipp Koehn. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Conference Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 737-745, 2007.

12) Fei Xia and Micheal McCord. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics*, pp. 508-514, 2004.

13) Peng Xu, Jaeh Kang, Micheal Ringgaard and Franz Och. Using a dependency parser to improve SMT for subject-object-verb languages. In *Proceedings of the 2009 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2009.

14) Kenji Yamada and Kevin Knight. A syntax-based statistical machine translation model. In *Proceedings of the Association for Computational Linguistics '05*, pp. 523-530, 2001.

15) Hao Zhang, Liang Huang, Daniel Gildea, and Kevin Knight. Synchronous binarization for machine translation. In Proceedings of the Human Language Technology Conference of the *North American Chapter of the Association for Computational Linguistics*, Main Conference, pp. 256-263, 2006.