A Pattern-based Machine Translation System Extended by Example-based Processing

Hideo Watanabe and Koichi Takeda
IBM Research, Tokyo Research Laboratory
1623-14 Shimotsuruma, Yamato, Kanagawa 242-8502, Japan
{watanabe,takeda}@trl.ibm.co.jp

Abstract

In this paper, we describe a machine translation system called PalmTree which uses the "pattern-based" approach as a fundamental framework. The pure pattern-based translation framework has several issues. One is the performance due to using many rules in the parsing stage, and the other is inefficiency of usage of translation patterns due to the exact-matching. To overcome these problems, we describe several methods; pruning techniques for the former, and introduction of example-based processing for the latter.

1 Introduction

While the World-Wide Web (WWW) has quickly turned the Internet into a treasury of information for every *netizen*, non-native English speakers now face a serious problem that textual data are more often than not written in a foreign language. This has led to an explosive popularity of machine translation (MT) tools in the world.

Under these circumstances, we developed a machine translation system called *PalmTree*[1] which uses the pattern-based translation [6, 7] formalism.

The key ideas of the pattern-based MT is to employ a massive collection of diverse transfer knowledge, and to select the best translation among the translation candidates (ambiguities). This is a natural extension of the example-based MT in the sense that we incorporate not only sentential correspondences (bilingual corpora) but every other level of linguistic (lexical, phrasal, and collocational) expressions into the transfer knowledge. It is also a rule-based counterpart to the word n-grams of the stochastic MT since our patterns intuitively captures the frequent collocations.

Although the pattern-based MT framework is promising, there are some drawbacks. One is the speed, since it uses many rules when parsing. The other is inefficiency of usage of translation patterns, since it uses the exact-match when matching translation patterns with the input. We will describe several methods for accelerating the system performance for the former, and describe the extension by using the example-based processing [4, 8] for the latter.

2 Pattern-based Translation

Here, we briefly describe how the pattern-based translation works. (See [6, 7] for details.) A translation pattern is a pair of source CFG-rule and its corresponding target CFG-rule. The followings are examples of translation patterns.

(p1) take:VERB:1 a look at NP:2 $\Rightarrow$ VP:1
VP:1 $\Leftarrow$ NP:2 wo(dobj) miru(see):VERB:1
(p2) NP:1 VP:2 $\Rightarrow$ S:2 S:2 $\Leftarrow$ NP:1 ha VP:2
(p3) PRON:1 $\Rightarrow$ NP:1 NP:1 $\Leftarrow$ PRON:1

The (p1) is a translation pattern of an English colloquial phrase "take a look at," and (p2) and (p3) are general syntactic translation patterns. In the above patterns, a left-half part (like "A B C $\Rightarrow$ D") of a pattern is a source CFG-rule, the right-half part (like "A $\Leftarrow$ B C D") is a target CFG-rule, and an index number represents correspondence of terms in the source and target sides and is also used to indicate a head term (which is a term having the same index as the left-hand side[2] of a CFG-rule). Further, some features can be attached as matching conditions for each term.

The pattern-based MT engine performs a CFG-parsing for an input sentence with using source sides of translation patterns. This is done by using chart-type CFG-parser. The target structure is constructed by the synchronous derivation which generates a target structure by combining target sides of translation patterns which are used to make a parse.

Figure 2 shows how an English sentence "She takes a look at him" is translated into Japanese.

[1]Using this system, IBM Japan releases a MT product called "Internet King of Translation" which can translate an English Web pages into Japanese.

[2]we call the destination of an arrow of a CFG rule description the left-hand side or LHS, on the other hand, we call the source side of an arrow the right-hand side or RHS.

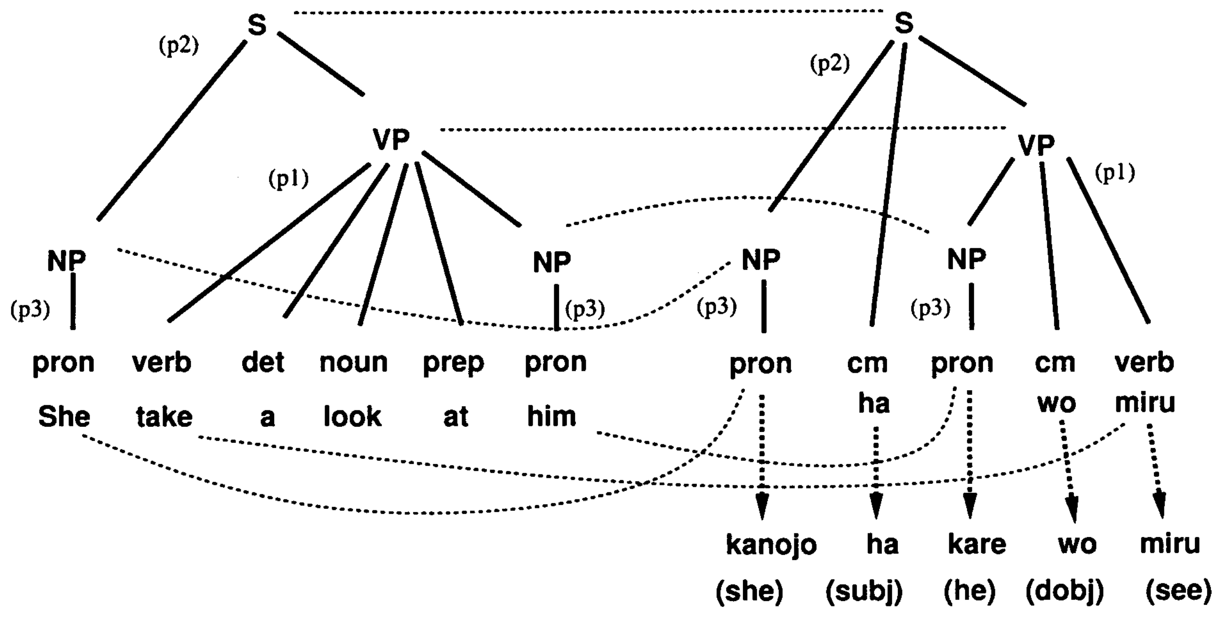

Figure 1: Translation Example by Pattern-based MT

In this figure, a dotted line represents the correspondence of terms in the source side and the target side. The source part of (p3) matches "She" and "him," the source part of (p1) matches a segment consisting "take a look at" and a NP("him") made from (p3), and finally the source part of (p2) matches a whole sentence. A target structure is constructed by combining target sides of (p1), (p2), and (p3). Several terms without lexical forms are instantiated with translation words, and finally a translated Japanese sentence "kanojo(she) ha(subj) kare(he) wo(dobj) miru(see)" will be generated.

3 Pruning Techniques

As mentioned earlier, our basic principle is to use many lexical translation patterns for producing natural translation. Therefore, we use more CFG rules than usual systems. This causes the slow-down of the parsing process. We introduced the following pruning techniques for improving the performance.

3.1 Lexical Rule Preference Principle

We call a CFG rule which has lexical terms in the right-hand side (RHS) a *lexical rule*, otherwise a *normal rule*. The *lexical rule preference principle (or LRPP)* invalidates arcs made from normal rules in a span in which there are arcs made from both normal rules and lexical rules.

Further, lexical rules are assigned cost so that lexical rules which has more lexical terms are preferred.

For instance, for the span [take, map] of the following input sentence,

He takes a look at a map.

if the following rules are matched,

(r1) take:verb a look at NP
(r2) take:verb a NP at NP
(r3) take:verb NP at NP
(r4) VERB NP PREP NP

then, (r4) is invalidated, and (r1),(r2), and (r3) are preferred in this order.

3.2 Left-Bound Fixed Exclusive Rule

We generally use an exclusive rule which invalidates competitive arcs made from general rules for a very special expression. This is, however, limited in terms of the matching ability since it is usually implemented as both ends of rules are lexical items. There are many expression such that left-end part is fixed but right-end is open, but these expressions cannot be expressed as exclusive rules. Therefore, we introduce here a *left-bound fixed exclusive (or LBFE) rule* which can deal with right-end open expressions.

Given a span [x y] for which an LBFE rule matched, in a span [i j] such that i<x and x<j<y, and in all sub-spans inside [x y],

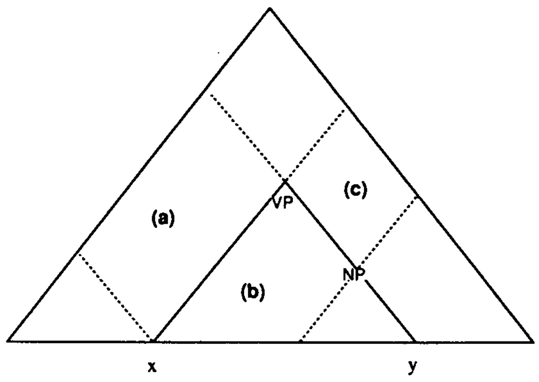

Figure 2: The Effect of an LBFE Rule

- Rules other than exclusive rules are not applied, and
- Arcs made from non-exclusive rules are invalidated.

Fig.2 shows that an LBFE rule "VP ⇐ VERB NP"[3] matches an input. In spans of (a),(b), and (c), arcs made from non-exclusive rules are invalidated, and the application of non-exclusive rules are inhibited.

Examples of LBFE rules are as follows:

NP ← DET own NP
NOUN ← as many as NP
NP ← most of NP

3.3 Preprocessing

Preprocessing includes local bracketing of proper nouns, monetary expressions, quoted expressions, Internet addresses, and so on. Conversion of numeric expressions and units, and decomposition of unknown hyphenated words are also included in the preprocessing. A bracketed span works like an exclusive rule, that is, we can ignore arcs crossing a bracketed span. Thus, accurate preprocessing not only improved the translation accuracy, but it visibly improved the translation speed for longer sentences.

3.4 Experiments

To evaluate the above pruning techniques, we have tested the speed and the translation quality for three documents. Table 1 shows the speed to translate documents with and without the above pruning techniques.[4] The fourth row shows the number of sentences tested with pruning which become worse than sentences without pruning and sentences with pruning which become better than without pruning.

This shows the speed with pruning is about 2 times faster than one without pruning at the same time the translation quality with pruning is kept in the almost same level as one without pruning..

[3]This is not an LBFE rule in practice.

[4]Please note that the time shown in this table was recorded about two years ago and the latest version is much faster.

4 Extension by Example-based Processing

One drawback of our pattern-based formalism is to have to use many rules in the parsing process. One of reasons to use such many rules is that the matching of rules and the input is performed by the exact-matching. It is a straightforward idea to extend this exact-matching to fuzzy-matching so that we can reduce the number of translation patterns by merging some patterns identical in terms of the fuzzy-matching. We made the following extensions to the pattern-based MT to achieve this example-based processing.

4.1 Example-based Parsing

If a term in a RHS of source part of a pattern has a lexical-form and a corresponding term in the target part, then it is called a *fuzzy-match term*, otherwise an *exact-match term*. A pattern writer can intentionally designate if a term is a fuzzy-match term or an exact-match term by using a double-quoted string (for fuzzy-match) or a single-quoted string (for exact-match).

For instance, in the following example, a word *make* is usually a fuzzy-match term since it has a corresponding term in the target side (*ketsudan-suru*), but it is a single-quoted string, so it is an exact-match term. Words *a* and *decision* are exact-match terms since they has no corresponding terms in the target side.

'make':VERB:1 a decision ⇒ VP:1
VP:1 ⇐ ketsudan-suru:1

Thus, the example-based parsing extends the term matching mechanism of a normal parsing as follows: A term T_B matches another matched-term T_A (Lex_A,Pos_B)[5] if one of the following conditions holds.

(1) When a term T_B has both Lex_B and Pos_B,

(1-1) Lex_B is the same as Lex_A, and Pos_B is the same as Pos_A.

[5]A matched-term inherits a lexical-form of a term it matches.

	Sample 1	Sample 2	Sample 3
Num of Sentences	13	41	50
Time with Pruning (sec.)	16	23	44
Time without Pruning (sec.)	20	48	67
Num of Changed Sentences (Worse/Better)	1/2	5/4	4/6

Table 1: Result of peformance experiment of pruning techniques

(1-2) T_B is a fuzzy-match term, the semantic distance of Lex_B and Lex_A is smaller than a criterion, and Pos_B is the same as Pos_A.

(2) When a term T_B has only Lex_B,

(2-1) Lex_B is the same as Lex_A.

(2-2) Lex_B is a fuzzy-match term, the semantic distance of Lex_B and Lex_A is smaller than a criterion.

(3) When T_B has only Pos_B, then Pos_B is the same as Pos_A.

4.2 Prioritization of Rules

Many ambiguous results are given in the parsing, and the preference of these results are usually determined by the cost value calculated as the sum of costs of used rules. This example-based processing adds fuzzy-matching cost to this base cost. The fuzzy-matching cost is determined to keep the following order.

$$(1\text{-}1) < (1\text{-}2),(2\text{-}1) < (2\text{-}2) < (3)$$

The costs of (1-2) and (2-1) are determined by the fuzzy-match criterion value, since we cannot determine which one of (1-2) and (2-1) is preferable in general.

4.3 Modification of Target Side of Rules

Lexical-forms written in the target side may be different from translation words of matched input word, since the fuzzy-matching is used. Therefore, we must modify the target side before constructing a target structure.

Suppose that a RHS term t_t in the target side of a pattern has a lexical-form w_t, t_t has a corresponding term t_s in the source side, and t_s matches an input word w_i. If w_t is not a translation word of w_i, then w_t is replaced with translation words of w_i.

4.4 Translation Example

Figure 3 shows a translation example by using example-based processing described above.

In this example, the following translation patterns are used.

(p2) NP:1 VP:2 $\Rightarrow$ S:2 S:2 $\Leftarrow$ NP:1 ha VP:2
(p3) PRON:1 $\Rightarrow$ NP:1 NP:1 $\Leftarrow$ PRON:1
(p4) take:VERB:1 a bus:2 $\Rightarrow$ VP:1
VP:1 $\Leftarrow$ basu:2 ni noru:VERB:1

The pattern (p4) matches a phrase "take a taxi," since "taxi" and "bus" are semantically similar. By combining target parts of these translation patterns, a translation "PRON ha basu ni noru" is generated. In this translation, since "basu(bus)" is not a correct translation of a corresponding source word "taxi," it is changed to a correct translation word "takusi(taxi)." Further, PRON is instantiated by "watashi" which is a translation of "I." Then a correct translation "watashi ha takusi ni noru" is generated.

5 Discussion

Unlike most of existing MT approaches that consist of three major components[1, 2] - analysis, transfer, and generation - the pattern-based MT is based on a *synchronous model*[5, 3] of translation. That is, the analysis of a source sentence is directly connected to the generation of a target sentence through the translation knowledge (i.e., patterns). This simple architecture makes it much easier to customize a system for improving translation quality than the conventional MT, since the management of the ambiguities in 3-component architecture has to tackle the exponential combination of overall ambiguities. In this simple model, we can concentrate on a single module (a parser with synchronous derivation), and manage most of translation knowledge in a uniform way as translation patterns.

Although it is easier to add translation patterns in our system than previous systems, it is difficult for non-experts to specify detailed matching conditions (or features). Therefore, we made a *pattern compiler* which interprets a simple pattern which a non-expert writes and converts it into the full-scale patterns including necessary matching conditions,

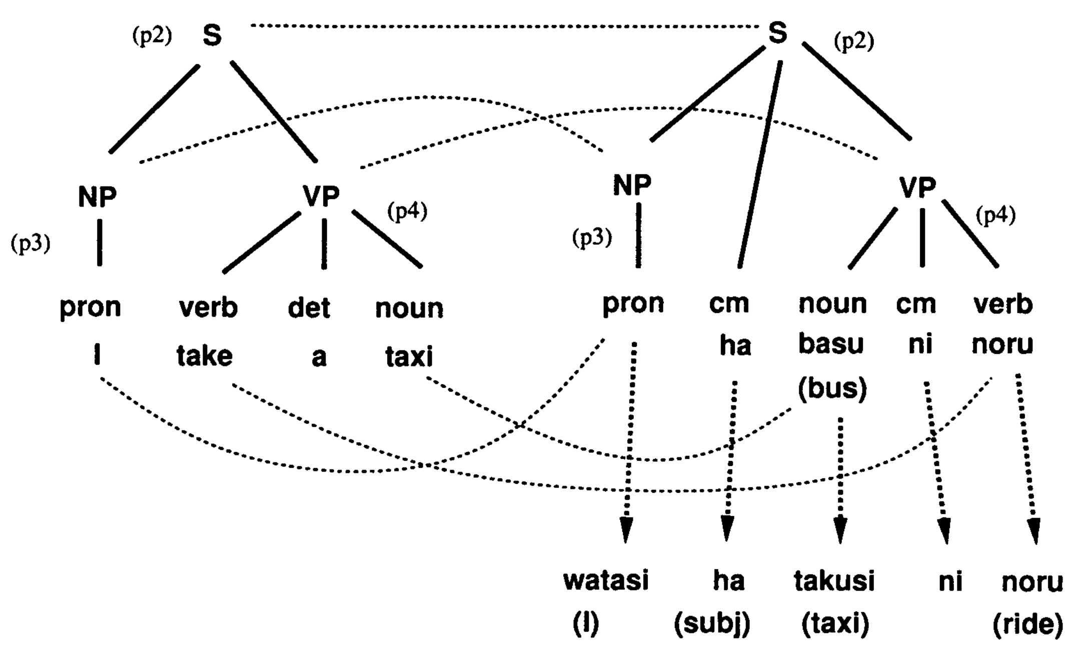

Figure 3: Translation Example by Example-based Processing

etc. For instance, the following E-to-J simple pattern (a) is converted into a full-scale pattern (b) by the pattern compiler.[6]

(a) [VP] hit a big shot = subarasii shotto wo utu
(b) hit:verb:1 a big shot ⇒ VP:1
    VP:1 ⇐ subarsii shotto wo utu:verb:1

Shown in the above example, it is very easy for non-experts to write these simple patterns. Thus, this pattern compiler enable non-experts to customize a system. In conventional MT systems, an expert is usually needed for each component (analysis, transfer, and generation).

These advantages can reduce the cost of development and customization of a MT system, and can largely contribute to rapidly improve the translation quality in a short time.

Further, we have shown the way to integrate example-based processing and pattern-based MT. In addition to reduce the total number of translation patterns, this combination enables us to make a more robust and human-like MT system thanks to the easy addition of translation pattern.

6 Conclusion

In this paper, we have described a pattern-based MT system called PalmTree. This system can break the current ceiling of MT technologies, and at the same time satisfy three essential requirements of the current market: efficiency, scalability, and ease-of-use.

We have described several pruning techniques for gaining better performance. Further we described the integration of example-based processing and pattern-based MT, which enables us to make more robust and human-like translation system.

[6]Practically, some conditional features are attached into verb terms.

References

[1] Nagao, M., Tsujii, J., and Nakamura, J., "The Japanese Government Project of Machine Translation," Computational Linguistics, 11(2-3):91–110, 1985.

[2] Nirenberg, S. editor: *Machine Translation - Theoretical and Methodological Issues*, Cambridge University Press, Cambridge, 1987.

[3] Rambow, O., and Satta, S., "Synchronous Models of Language," Proc. of the 34th of ACL, pp. 116–123, June 1996.

[4] Sato, S., and Nagao, M. "Toward Memory-based Translation," Proc. of 13th COLING, August 1990.

[5] Shieber, S. M., and Schabes Y., "Synchronous Tree-Adjoining Grammars," Proc. of the 13th COLING, pp. 253–258, August 1990.

[6] Takeda, K., "Pattern-Based Context-Free Grammars for Machine Translation," Proc. of 34th ACL, pp. 144–151, June 1996.

[7] Takeda, K., "Pattern-Based Machine Translation," Proc. of 16th COLING, Vol. 2, pp. 1155–1158, August 1996.

[8] Watanabe, H. "A Similarity-Driven Transfer System," Proc. of the 14th COLING, Vol. 2, pp. 770-776, 1992.