A Method for Extracting Translation Patterns from Translation Examples

Hideo Watanabe
IBM Research, Tokyo Research Laboratory
1623-14, Shimotsuruma, Yamato-shi, Kanagawa-ken 242, JAPAN
watanabe@trl.vnet.ibm.com

Abstract

When the example-based approach is used for machine translation, it is important to collect a large volume of translation patterns, because most systems employ a pair of source and target parsed structures as a translation pattern, and such translation pairs are hard to collect. This paper describes a method to find out translation patterns which are valid for the translation pattern base by comparing wrong translations and corresponding correct translations.

1 Introduction

An example-based approach (EBA), an emerging machine translation technology proposed by Nagao [3], has been extensively used in recent studies [8, 5, 10, 7, 2, 1, 11]. A typical EBA translation system uses a large volume of translation examples or translation patterns, each of which is a pair of parsed structures in two languages. Collecting a large volume of such patterns is a difficult and time-consuming task. This paper describes a method of extracting translation patterns that are valid for the current translation pattern base from translation examples.

In Figure 1, (a) is a dependency structure of an input Japanese sentence, (r1), (r2), and (r3) are given translation patterns, and (b) is the English dependency structure produced by these patterns. Clearly, (b) is not a correct translation of (a), whereas (c) is. Therefore, what we want to do here is, by comparing the current output (b) and a correct output (c), to extract a translation pattern (r4) that can be added to the given translation pattern base.

Given an input string S_s, let S_t be an output string produced by a translation system TS, and let S_c be a correct translation of S_s. The purpose of the proposed method is to obtain translation patterns that produce the correct translation string S_c. We assume the following:

- TS is an example-based transfer system that can use translation patterns directly.
- Parsing is error-free, that is, the parsed structures of the source and target languages are correct.
- Translation pattern base includes some translation patterns.

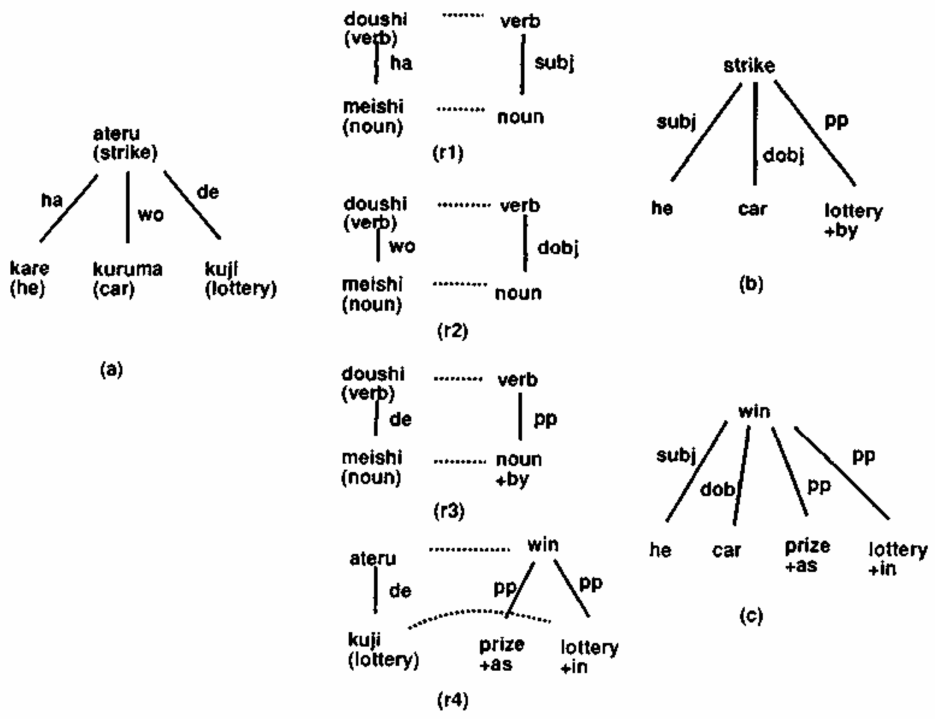

Figure 1: Example of extracting a new translation pattern

The second condition might sound rather severe. However it does not mean that parsing must be done entirely by a parsing program, but rather that human supervision is allowed. Thus, we can break down the problem of obtaining translation patterns into the following two sub-problems:

- making a mapping M_c from D_s to D_c, and
- extracting translation patterns by comparing $\langle D_s, M_t, D_t\rangle$ with $\langle D_s, M_c, D_c\rangle$,

where D_s is a dependency structure of an input string S_s, D_t is a dependency structure produced by TS, D_c is a dependency structure of S_c, and M_t is a mapping from D_s to D_t, which are linked by TS.

The data structures used in this paper is described in the next section. A method for finding correspondences mapping is given in Section 3, and a method for finding translation patterns in Section 4. Subsequently, Section 5 gives an example of the method's use, and Section 6 discusses related work. Some concluding remarks bring the paper to an end.

2 Data Structure

This section describes data structures used in this paper.

A dependency structure is expressed as a rooted labeled directed acyclic graph (in short rldag) indicating a graph which has only one root node, has nodes and arcs labeled, has arcs directed, and

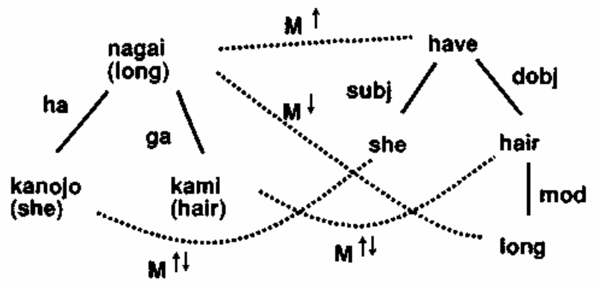

Figure 2: Example of a translation pattern

has no cyclic arc path. Each node consists of some features, and each arc is labeled its grammatical relation.

A translation pattern r is expressed as the following 3 tuples:

$$r = \langle G_s, M, G_t \rangle$$

where, G_s is a dependency structure of source language, and G_t is a dependency structure of target language. Further, M is a set of mappings as follows:

$$M = \langle M^L,\ M\uparrow,\ M\downarrow \rangle$$

where M^L is a *lexical mapping* from G_s nodes to G_t nodes, which indicates a G_t node which is a translation word of a G_s node, and $M\uparrow$ and $M\downarrow$ are structural mappings from G_s nodes to G_t nodes, which indicate a connection point (node) in G_t for a G_s node, on which some G_ts are merged; $M\uparrow$ is called a *upward mapping* and $M\downarrow$ is called a *downward mapping*. Figure 2 shows an example of a translation pattern which can translate a Japanese sentence "kanojo ha kami ga nagai" to an English sentence "she has long hair." In this figure, dotted lines represent correspondences between words. The Japanese word "nagai" ("long") has two corresponding nodes "have" and "long," as an upward mapping and a downward mapping, respectively. Suppose that "nagai" has the modifiers "totemo" ("very") and "itsumo" ("always"); then the English word "very" used to translate "totemo" should be related to "long," but the word "always" used to translate "itsumo" should be related to "have." This is one of the reasons why upward and downward mappings are needed.

We assume that an example-based transfer system runs as follows:

1. Gather some translation patterns, all G_ss of which cover the input dependency structure completely.
2. Combine all G_ts of the selected translation patterns so that G_t nodes whose corresponding G_s node is the same.

To find a corresponding G_s node for a G_t node, the above-mentioned structural mappings are used.

The data structure and behavior mentioned above is used in *SimTran*, an example-based transfer system we have developed. For more details, please refer to [11].

```
procedure mapping($D_s$,$D_t$) begin
    $M^L \leftarrow$ one-to-one-mapping($D_s$,$D_t$)                                              (1)
    $M\uparrow \leftarrow M^L$                                                                      (2)
    $M\downarrow \leftarrow M^L$                                                                    (3)
    for any node pair $s_1$ and $s_2$ in $D_s$ such that $s_1$ is an ancestor node of $s_2$ begin    (4)
        $t_1 \leftarrow M^L(s_1)$                                                                   (5)
        $t_2 \leftarrow M^L(s_2)$                                                                   (6)
        if $t_2$ is an ancestor node of $t_1$ then                                                  (7)
            remove $(s_1, t_1)$ from $M\uparrow$                                                    (8)
            $t_3 \leftarrow t_1$                                                                    (9)
            for each $n$ in ancestors of $t_2$ in $D_t$ begin                                       (10)
                if $n$ is a root node, or a parent node of $n$ or $n$ is related by $M\uparrow$ or $M\downarrow$ then begin   (11)
                        $t_3 \leftarrow n$                                                          (12)
                        break                                                                       (13)
                end                                                                                 (14)
            end                                                                                     (15)
            add $(s_1, t_3)$ to $M\uparrow$                                                         (16)
        end                                                                                         (17)
        $R_s \leftarrow root(D_s)$                                                                  (18)
        $R_t \leftarrow root(D_t)$                                                                  (19)
        if $M\uparrow(R_s) \neq R_t$ then begin                                                     (20)
            remove $(R_s, M\uparrow(R_s))$ from $M\uparrow$                                         (21)
            add $(R_s, R_t)$ to $M\uparrow$                                                         (22)
        end                                                                                         (23)
    end                                                                                             (24)
end
```

Figure 3: Algorithm for finding a structural mapping

Even though an example-based transfer system does not use structural mappings, at least it has a lexical mapping. Therefore, in such systems, a lexical mapping is used to find a correponding G_s node instead of structural mappings. In what follows, description about structural mappings can be ignored for such example-based transfer systems.

3 Mapping

This section describes how to find mappings described in the previous section.

Utsuro and et al. [9] and Kaji and et al. [2] proposed methods for finding the correspondences between translation pairs. The former method finds one-to-one correspondence between two feature structures of translation equivalent, and the latter method finds one-to-one correspondence between two phrase structures. This one-to-one mapping corresponds to a lexical mapping in *SimTran*. Therefore, the following proposed method uses Utsuro's method as a basic mechanism for finding a lexical mapping, and obtains a structural mapping based on the lexical mapping.

Our algorithm is shown in Figure 3. Briefly, this algorithm applies Utsuro's method first for the given translation pair to get a lexical mapping (line 1), then finds two correspondences (s_1, t_1)

and (s_2, t_2) such that s_1 is an ancestor of s_2 while t_1 is a descendant of t_2 (lines 4-7). If such node pairs are found, it searches ancestors of t_2 for a node t_3 corresponding to s_1 such that t_3 is a root node, or a parent node of t_3 or t_3 is related by other source node (line 11), and then makes (s_1, t_1) an downward mapping and (s_1, t_3) an upward mapping (lines 8-16). The last part (lines 20-23) checks whether both root nodes are related by an upward mapping, because a *SimTran* translation pattern must satisfy this condition. If they are not related, then it relates them.

For instance, in the case of Figure 2, the following lexical mapping is obtained by Utsuro's method:

M^L: {(nagai,long),(kanojo,she),(kami,hair)}

The corresponding node pairs (nagai,long) and (kami,hair) hold true for lines 4-7, and (nagai,have) is found as an upward mapping. Therefore, the following upward and downward mappings are obtained:

$M\uparrow$: {(nagai,have),(kanojo,she),(kami,hair)}
$M\downarrow$: {(nagai,long),(kanojo,she),(kami,hair)}

4 Finding Translation Patterns

This section describes how to find translation patterns.

Let T_p be a set of translation patterns used by TS to produce D_t from D_s, and let M_c be a mapping bewteen D_s and D_c obtained by the procedure described in the previous section.

Given a set of nodes in D_s, then there is a set of nodes in D_t or D_c projected by M_t or M_c from those nodes. We call a connected subgraph containing all of these projected nodes in a target structure a *projected subgraph*, and the minimum one of those subgraphs a *minimum projected subgraph*.

An algorithm for finding translation patterns is shown in Figure 4. Lines 25-30 find new word-level translation patterns, and those patterns are stored in the new translation pattern list. Lines 31-36 find new structural translation patterns. For each translation pattern p, a target structure p_t of p and a minimum projected subgraph p_c in D_c are compared in line 35. If p_t is not same as p_c then it is stored in the new translation pattern list; otherwise, it is stored in the same translation pattern list. When p_t and p_c are compared, they are the same if they are structurally isomorphic and corresponding nodes are the same, where a p_t node n_t and a p_c node n_c are the same, (1) if parts-of-speech are equal when n_t is a non-lexical node, or (2) if both lexical forms and parts-of-speech are equal when n_t is a lexical node. After that, the translation patterns in the new translation pattern list are merged if they share any common nodes (lines 37-42). Further, for each subgraph g of D_c which is included in neither the new translation pattern list nor the same translation pattern list, a minimum subgraph g' such that g' contains both g and any leaf node of p_t in the new translation pattern list is found, and g' is merged with p_t (lines 44-48).

5 Example

This section gives an example of the method's use.

```
procedure find-trnpat($D_s$,$D_t$,$D_c$,$M_t$,$M_c$,$T_p$) begin
    $New \leftarrow \phi$, $Same \leftarrow \phi$                                                    (25)
    for each node $n$ in $D_s$ begin                                                                 (26)
        $n_t \leftarrow M_t^L(n)$                                                                    (27)
        $n_c \leftarrow M_c^L(n)$                                                                    (28)
        if $n_t \neq n_c$ then add $(n, n_c)$ to $New$                                               (29)
    end                                                                                              (30)
    for each pattern $p$ in $T_p$ begin                                                              (31)
        $p_s \leftarrow$ a source part of $p$                                                        (32)
        $p_t \leftarrow$ a target part of $p$                                                        (33)
        $p_c \leftarrow$ a minimum projected subgraph of $p_s$ in $D_c$                              (34)
        if same($p_t, p_c$) then add $(p_s, p_t)$ to $Same$ else add $(p_s, p_t)$ to $New$           (35)
    end                                                                                              (36)
    for each node $n$ in $D_s$ begin                                                                 (37)
        $P \leftarrow \{(p_s, p_t) | (p_s, p_t) \in New \wedge n \in p_s\}$                          (38)
        remove $P$ from $New$                                                                        (39)
        $(p'_s, p'_t) \leftarrow$ merge($P$)                                                         (40)
        add $(p'_s, p'_t)$ to $New$                                                                  (41)
    end                                                                                              (42)
    $D'_c \leftarrow$ merge(all $p_t$s in $New$ and $Same$)                                          (43)
    for each disconnected subgraph $g$ in $D_c - D'_c$ begin                                         (44)
        $g' \leftarrow$ minimum subgraph of $D_c$ which contains $g$ and any leaf node $x$ of $p_t$ in $New$   (45)
        remove $(p_s, p_t)$ from $New$                                                               (46)
        add ($p_s$,merge($p_t, g'$)) to $New$                                                        (47)
    end                                                                                              (48)
end
```

Figure 4: Algorithm for finding translation patterns

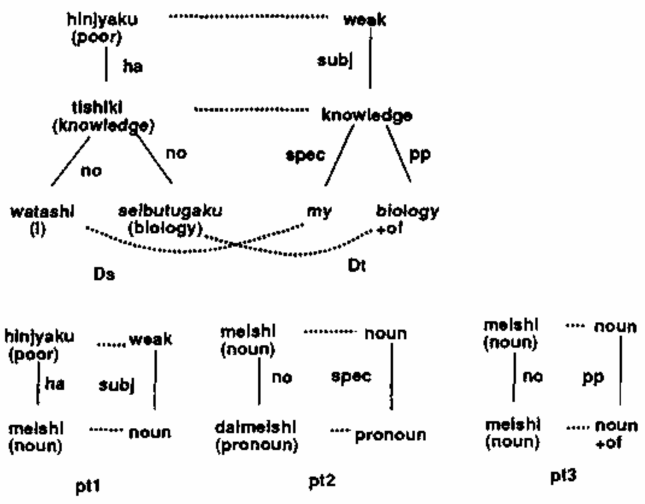

Figure 5: Translation patterns and dependency structures of input and translation by TS

In Figure 5, D_s is a dependency structure of the following Japanese sentence:

watashi no seibutugaku no tishiki ha hinjyakuda

(I have little knowledge of biology.)

and D_t is a dependency structure produced by TS as an output by using translation patterns pt_1, pt_2, and pt_3. Dotted lines denote mappings between two structures. For convenience, an unmarked dotted line is equivalent to a line marked M^L, $M \uparrow$, and $M \downarrow$.

Suppose that D_c in Figure 6 is a correct English translation of D_s. Then the first step is to find mappings from D_s to D_c. By using a procedure described in Section 3, the mappings expressed by dotted lines in Figure 6 are obtained. In these mappings, note that the Japanese word "hinjyaku" ("poor") is related to the English word "have" even though it is not a translation word. This can be done in the last part of the mapping algorithm.

The next step is to find structural differences between translation patterns for $\langle D_s, D_t\rangle$ and translation patterns for $\langle D_s, D_c\rangle$. Translation patterns corresponding to pt_1, pt_2, and pt_3 are shown as pt'_1, pt'_2, and pt'_3 in Figure 6. Comparison of these corresponding translation patterns shows that pt'_1 and pt'_2 are different, and they are stored in the *new* tranlsation pattern list. D_c contains a portion, "little (mod)" that is not covered by translation patterns pt'_1, pt'_2, and pt'_3. This portion is attached to pt'_1, and becomes pt''_1 in Figure 6. Finally, a new translation pattern pt_4 is obtained by merging pt''_1 and pt'_2 in the new translation pattern list.

6 Discussion

Strong recent interest in corpus-based processing has produced some results on extracting relevant information from bilingual corpora. There has been some research [2, 9] on finding correspondences between translation pair sentences, and extracting translation patterns from them. The method proposed in this paper differs from previous ones in that it extracts new relevant translation

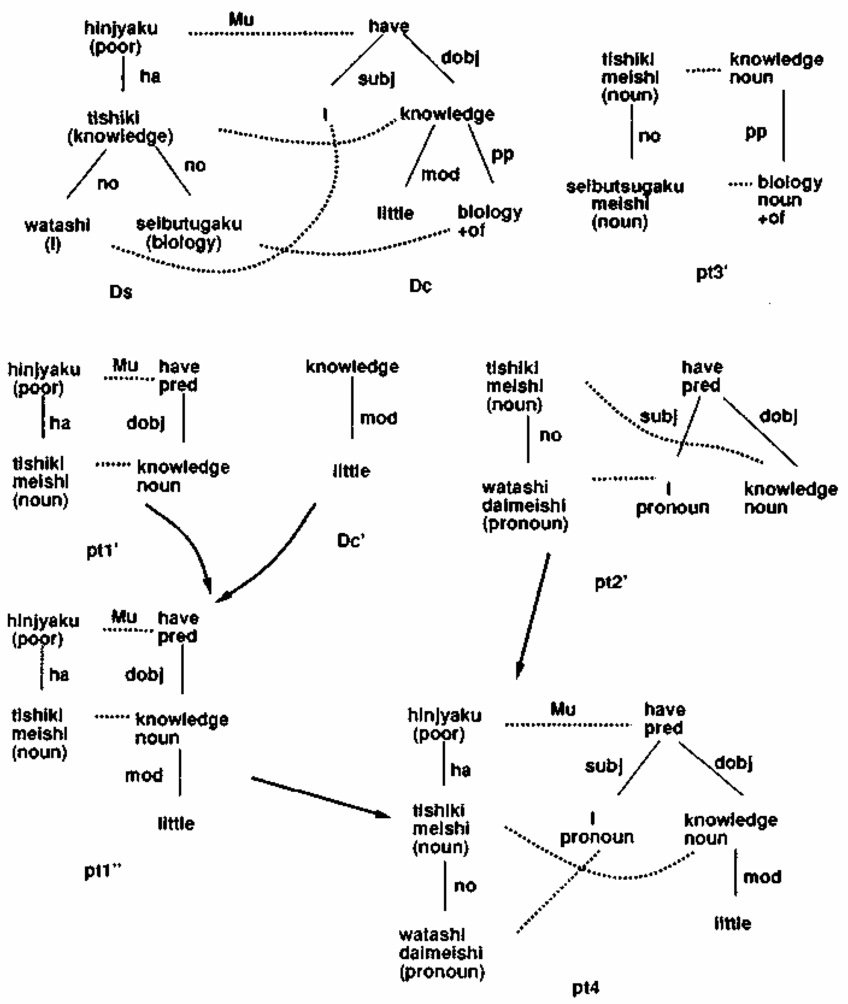

Figure 6: Translation patterns and dependency structures of input and correct translation

patterns (which are not contained in the current translation pattern base) by comparing the result output by a translation system with a correct translation.

If we consider human processes for acquiring translation patterns, we can categorize them into two types: finding translation patterns by viewing many similar translations, and finding translation patterns by comparing a correct translation and a wrong translation. An example of the former is that, given many translation instances of the Japanese word "toru" ("take"), one can obtain some translation patterns (or case frames) that are relevant for translating sentences containing "toru." An example of the latter type has been given in this paper. The above-mentioned studies [2, 9] deal with the former type of process, whereas the method proposed in this paper deals with the latter.

The former type of method is necessary for creating new translation pattern base, but when translation patterns are added to existing translation pattern base, the efficiency of coverage or the reusability of translation patterns might not be good, because many similar translation patterns, some of which might not be used, are added to the translation pattern base. On the other hand, the efficiency of the latter method's coverage is good, because it adds only new translation patterns to the translation pattern base.

Some may feel that the proposed method is peculiar to *SimTran*. Actually, a method for finding structural mappings is unique for *SimTran*, but a method for finding new relevant translation patterns is applicable for any type of example-based transfer systems, if you change structural mappings to lexical mapping in the descriptions above.

7 Conclusion

In this paper, I have proposed a method that compares a wrong translation and a correct translation in order to extract relevant translation patterns that can be added to a current translation pattern base. To finding mappings between two parsed structures, I extended an existing method that finds one-to-one mapping [9], so that structural mapping employed in *SimTran* can be obtained. To find new translation patterns, I proposed a method for finding translation patterns from the differences between translation patterns used in a wrong translation and those used in a correct translation. This method is useful for extending or enhancing a current translation pattern base efficiently.

References

[1] Furuse, O. and Iida, H., "Cooperation between Transfer and Analysis in Example-Based Framework," *Proc. of Coling 92*, Vol. 2, pp. 645-651, 1992.

[2] Kaji, H., Kida, Y., and Morimoto, Y., "Learning Translation Templates from Bilingual Text," *Proc. of Coling 92*, Vol. 2, pp. 672-678, 1992

[3] Nagao, M., "A Framework of a Mechanical Translation between Japanese and English by Analogy Principle," Elithorn, A. and Banerji, R. (eds.): *Artificial and Human Intelligence*, NATO 1984.

[4] Sadler, V., "Working with Analogical Semantics: Disambiguation Techniques in DLT," FORIS Publications, 1989.

[5] Sato, S., and Nagao, M., "Toward Memory-based Translation," Proc. of Coling '90, 1990

[6] Sato, S., "Example-Based Translation Approach," Proc. of Int. Workshop on Fundamental Research for the Future Generation of NLP, July 23-24, 1991

[7] Sato, S., "Memory-based Translation," *Doctor Thesis*, 1992.

[8] Sumita, E., Iida, H., and Kohyama, H., "Translating with Examples: A New Approach to Machine Translation," 3rd Int. Conf. on Theoretical and Methodological Issues in Machine Translation, 1990.

[9] Utsuro, T., Matsumoto, Y., and Nagao, M., "Lexical Knowledge Acquisition from Bilingual Corpora," Proc. of Coling '92, Vol. 2, pp. 581-587, 1992.

[10] Watanabe, H., "A Model of a Transfer Process Using Combinations of Translation Rules," *Proc. of Pacific Rim of Int. Conf. on AI '90*, 1990.

[11] Watanabe, H., "A Similarity-Driven Transfer System," Proc. of Coling '92, Vol. 2, pp. 770-776, 1992.