A Similarity-Driven Transfer System

Hideo Watanabe
IBM Research, Tokyo Research Laboratory
5-19 Sannancho, Chiyoda-ku, Tokyo 102 Japan
e-mail: watanabe@trl.vnet.ibm.com

Abstract

The transfer phase in machine translation (MT) systems has been considered to be more complicated than analysis and generation, since it is inherently a conglomeration of individual lexical rules. Currently some attempts are being made to use case-based reasoning in machine translation, that is, to make decisions on the basis of translation examples at appropriate points in MT. This paper proposes a new type of transfer system, called a \textit{Similarity-driven Transfer System (SimTran)}, for use in such case-based MT (CBMT).

1 Introduction

The transfer process in machine translation systems is, in general, more complicated than the processes of analysis and generation. One reason for this is that it relies heavily on human heuristic knowledge or the linguistic intuition of the rule writers. Unfortunately, linguistic intuition tends to be unable to control the process properly for a wide variety of inputs, because of the huge amount of data and the huge number of situations that need to be considered. However, rule writers must rely on their linguistic intuition to some extent, because there is no linguistic theory on lexical transfer \cite{7}. Another reason \cite{8,11} is that the transfer task is inherently a conglomeration of individual lexical rules. Therefore, the transfer process can be said to fall into a class of problem that cannot easily be controlled by the linguistic intuition of rule writers.

In accordance with these observations, various attempts have been made to overcome the problems of transfer; they include knowledge-based MT \cite{12}, bilingual signs \cite{13}, and Tags for MT\cite{1}. One such approach is case-based or example-based MT \cite{4} \cite{9} \cite{10} \cite{11}. The essential idea behind all case-based MT (CBMT) methods is that the system chooses the case (or example) most similar to the given input from the case base, and applies the knowledge attached to the chosen case to the input. \cite{1}

Supposing that there is a corpus of parsed translation examples in which corresponding parts are linked to each other, we can regard those parsed translation examples as translation rules. A promising approach is therefore to make a transfer process that (1) chooses a set of translation examples, each source part of which is similar to a part of the input, and all source parts of which overlap the whole input, and (2) constructs an output by combining the target parts of those translation examples chosen. However, this does not mean that existing transfer knowledge should be abandoned. Rather, such transfer knowledge should be used as a fail-safe mechanism if there are no appropriate examples. In the \textit{similarity-driven transfer system (SimTran)} we have developed, both translation examples and existing transfer knowledge are treated uniformly as translation patterns, and are called translation rules.

In Figure 1, for example, (a) is the parsed dependency structure of an input Japanese sentence, “kare ga kusuri wo nomu.” Suppose that (b) is selected as the most similar translation rule for the part “kare ga ... nomu” from the translation rule-base, and that (c) is selected as the most similar translation rule for the part “kusuri wo nomu,” even though there are several translation candidates for the Japanese verb “nomu.” This figure illustrates what we would like to do; that is, to construct (d), the translated structure by combining the target structures of the selected translation rules.

To develop this kind of system, we must consider the following issues:

(a) a metric for similarity,

(b) a mechanism for combining target parts of rules, and

(c) correspondence between the source part and the target part of a rule.

To handle the last two issues, I developed a model called \textit{Rules Combination Transfer (RCT)} \cite{14}. \textit{SimTran} is RCT coupled with a similarity calculation method. In this paper, I will introduce RCT and the similarity calculation method used in \textit{SimTran}.

The next section defines the data structure for graphs, and the format of a translation rule. Section 3 presents a method for calculating the similarity between an input and the source part of a translation rule. Section 4 describes the flow of the transfer process in RCT. Section 5 gives examples of translation using \textit{SimTran}, and Section 6 discusses related work. Some concluding remarks bring the paper to an end.

\footnote{This approach can be regarded as an application of case-based reasoning \cite{5} to natural language translation.}
2 Translation Rules

A basic type of graph used in this paper is a labeled directed graph, or an ldg. An ldg $G$ consists of a set of nodes $N$, and a set of arcs $A$. Further, each node and arc has a label. In particular, node labels are unique. Each node consists of features, each of which is a pair of a feature name and a feature value.

If an ldg has only one root node, then it is called a rldg, and if an ldg has no cyclic path, then it is called an ldag. Therefore, an ldag denotes an ldg that has only one root node and no cyclic path.

A translation rule $r$ consists of the following three components:

$$ r = (G_m, M, G_c) $$

where $G_m$ is a matching graph, $G_c$ is a construction graph, and $M$ is a set of mappings between $G_m$ and $G_c$.

A matching graph $G_m$ and a construction graph $G_c$ must be at least an rldg. Further, nodes in $G_m$ must be labeled uniquely; that is, each node in $G_m$ must have only one unique label, and the label of the node $n_m$ in $G_c$ is determined to be the label of the node $n_m$ in $G_m$ such that $n_c = M(n_m)$.

Mapping between $G_m$ and $G_c$ designates the correspondences between nodes in $G_m$ and $G_c$. For instance, in Figure 2, the Japanese word "nagai" ("long") should correspond to both of the English words "have" and "long," because if another word governs the word "nagai" then its English relation should be connected to the word "have." On the other hand, if the Japanese word "toton" ("very") modifies "nagai" then its English translation "very" should be connected to "long." This shows that for a node in a source language, two kinds of connection point, for translations of both governing structures and governed structures of the node, are needed in its translated structure. This implies that there should be two kinds of correspondence between $G_m$ and $G_c$, namely, (1) a mapping from a $G_m$ node $n_m$ to a $G_c$ node $n_c$ that is to be a node connected to translations...
of structures governing \( n_m \), and (2) a mapping from \( n_m \) to a \( G_e \) node \( n'_e \) that is to be a node connected to translations of structures governed by \( n_m \). We call the former an **upward mapping** and the latter a **downward mapping**, and denote these two kinds of mapping as follows:

\[
M = \langle M \uparrow, M \downarrow \rangle
\]

where \( M \uparrow \) is upward mapping, and \( M \downarrow \) is downward mapping.

Not all kinds of mapping should be permitted as \( M \uparrow \) and \( M \downarrow \). A translation rule \( r = \langle G_m, M, G_e \rangle \) must satisfy the following conditions:

1. \( M \uparrow \) and \( M \downarrow \) are both injections,
2. there are no two distinct nodes \( x \) and \( y \) in \( G_m \) such that \( M(x) = M(y) \),
3. \( M \downarrow (\text{root}(G_m)) = \text{root}(G_e) \).

Condition (1) ensures that there is only one connection point in \( G_e \) for each translation of governing structures and governed structures, condition (2) ensures that the label of a \( G'_m \) node is determined uniquely, and condition (3) ensures that the result of this transfer model becomes a rooted graph (see [15] for details). A rule satisfying these conditions is said to be **sound**.

### 3 Similarity Calculation

This section describes how a similarity is calculated.

#### 3.1 Graph Distance

The similarity between a \( G_m \) and an input graph \( G_{in} \) is defined as the inverse of the graph distance between them. First, the simple graph distance \( D_g \) between \( G_{in} \) and \( G_m \) is given as follows:

\[
D_g(G_{in}, G_m) = D_a(R_{in}, R_m) + \sum_{a \in m} \min(D_g(G(R_{in}, a_m), G(R_m, a_m)))
\]

where \( R_{in} \) and \( R_m \) are roots of \( G_{in} \) and \( G_m \), respectively, \( D_a \) is a node distance, \( a_m \) is an arc in \( G_m \) such that its source node is \( R_m \), and \( G(S(n, a)) \) denotes a subgraph that is related to an arc \( a \) from \( n \).

Briefly, a simple distance is the sum of the node distance between two roots and the sum of the minimal simple distances between \( G_{in} \) subgraphs and \( G_m \) subgraphs that, for each arc \( a \) outgoing from the \( G_m \) root node, are related to the all arcs \( a \) from the root nodes.

However, the larger \( G_m \) is, the larger this simple distance becomes. Therefore, when normalized by the number of nodes in \( G_m \), the graph distance \( D_g \) is given as follows:

\[
D_g(G_{in}, G_m) = \frac{D_g'(G_{in}, G_m)}{N}
\]

where \( N \) is the number of nodes in \( G_m \).

#### 3.2 Node Distance

When considering the distance between two words (nodes), we usually think of their semantic distance in a semantic hierarchy. In general, no matter what semantic hierarchy we use, it is inevitable that there will be some sort of distortion. Further, as stated before, a node consists of several features and may not have a lexical form that is a pointer to a semantic hierarchy. Therefore, a promising approach to calculating distances between nodes is to use both a semantic hierarchy and syntactic features, that is, to use syntactic features to correct the distortion contained in the semantic hierarchy to some extent.

The node distance between a \( G_{in} \) node \( n_i \) and a \( G_m \) node \( n_m \) is defined as follows:

\[
D_n(n_i, n_m) = \frac{D_f + D_s \cdot \delta_s}{N_f + \delta_s}
\]

where \( D_f \) is a feature node distance, \( D_s \) is a semantic node distance, \( N_f \) is the number of features in \( n_m \) for \( D_f \), and \( \delta_s \) is the weight of a semantic distance.

The semantic distance \( D_s \) between a \( G_{in} \) word \( w_{in} \) and a \( G_m \) word \( w_m \) is given by the following equation. In SimTrans, Bunrui Goi Hyou [5] code (or bghcode\(^6\)) is used for calculating the semantic distance between Japanese words.

\[
D_s(w_{in}, w_m) =
\begin{cases}
0 & \text{if } w_{in} = w_m \\
0.5 & \text{if } w_{in} \text{ or } w_m \text{ is unknown} \\
1 & \text{if } w_{in} \text{ and } w_m \text{ are unknown} \\
\frac{\text{bgh}(w_{in}) - \text{bgh}(w_m) + \delta_b}{\text{bghmax} + \delta_b} & \text{otherwise}
\end{cases}
\]

where \( \text{bgh}(w) \) is the fraction part of the bghcode of \( w \), \( \text{bghmax} \) is the maximal difference between two bghcode fraction parts, and \( \delta_b \) is a penalty incurred if two words are not identical.

The feature distance \( D_f \) between a \( G_{in} \) node \( n_i \) and a \( G_m \) node \( n_m \) is given as follows:

\[
d_f(n_i, f) =
\begin{cases}
0 & \text{if } f \text{ is consistent with } f_n \text{ for } f_n \\
\sum_{f \in f_n} d(f_{in}, f) & \text{otherwise}
\end{cases}
\]

where \( d(f_{in}, f_n) \) is the distance between \( f_{in} \) and \( f_n \) and \( f_n \) is consistent with \( f_{in} \).

---

\(^6\) This means that either \( M \uparrow (x) \) or \( M \downarrow (y) \) is equal to either \( M \uparrow (y) \) or \( M \downarrow (x) \),

\(^7\) Distances defined in this section are not actual distances in the mathematical sense.

---

**ACTES DE COLING-92, NANTES, 23-28 Août 1992**

**772**

**PROC. OF COLING-92, NANTES, AUG. 23-28, 1992**
Each matching pivot in a similar i-cover rule set must have $M \uparrow$ or $M \downarrow$, to ensure that the $G_c$s of the i-cover rule set produce a connected graph as a result. If there are rules in the given i-cover rule set that do not satisfy this condition, they are removed from the set of rule candidates, and the cover search method is executed until an i-cover rule set that satisfies this condition is found. Such an i-cover rule set is called a proper rule set.

Next, for each projection of the given i-cover, we must make a copy of its origin rule, or rule instance, because one rule may make more than one projection on $G_m$.

4.2 Pre-Lexicalization

It may happen that a lexical form of a $G_c$ in the given rule instance is not a candidate translation word of its corresponding word in the input, because a lexical form in a matching node in its $G_m$ is not necessarily the same as the input word. In this case, such a node is lexicalized by candidate translation words.

4.3 Node Labeling

The label of a $G_m$ node becomes the label of its matching node in $G_{in}$. Since $G_{in}$ nodes are labeled uniquely, $G_m$ nodes are also labeled uniquely. On the other hand, the label of a $G_c$ node $n_c$ becomes the label of a $G_m$ node $(n_m)$ such that $n_c = M \uparrow(n_m)$ or $n_c = M \downarrow(n_m)$.

There may, however, be two nodes in $G_c$ in a rule instance that are mapped by a node in $G_m$ with $M \uparrow$ and $M \downarrow$, respectively. In the succeeding process, $G_c$ nodes with the same label are merged into one node in order to generate an output structure. In this phase, the transferred labels of these two nodes should be different, because the two nodes should not be merged for this rule. We must therefore relabel $G_c$ nodes of rule instances as follows:

$G_c$ Node Relabeling: for any label $l$ in $G_c$, if $l$ is distributed to two distinct nodes of $G_c$ by both $M \uparrow$ and $M \downarrow$ from a node of $G_m$, then a label $l$ in a $G_c$ node, which is mapped only by $M \uparrow$, or is mapped by both $M \uparrow$ and $M \downarrow$ and has no descendants, is changed to $l'$.

4.4 Gluing

Unification is a well-known computational tool for connecting graphs, and is widely used in natural language processing. Usually, unification uses two func-
4.5 Post-Lexicalization

The constructed target structure is still imperfect; there might be a \( G_c \) node that has no lexical-form, because there are some rules made from transfer knowledge that have no lexical-forms. Therefore, as in the pre-lexicalization phase, non-lexical \( G_c \) nodes are lexicalized.

5 Examples

This section gives examples of translation by SimTran. Figure 5 shows how the Japanese sentence “Kanojo no me ga totemo kireina no wo sitteiri” is translated into the English sentence “(1) know that she has very beautiful eyes.” In this figure, (a) is an input sentence structure, (b), (c), and (d) are rules (precisely, rule instances), and (e) is the output structure produced. In these rules, a mapping line not marked \( M \) or \( \bar{M} \) has both \( M \) and \( \bar{M} \). Dotted lines designate matching or gluing correspondences between rule nodes and input or output nodes, respectively. Further, numbers prefixed by \( \ast \) denote node labels. In this example, we assume type hierarchies in which, for instance, ‘yogen(predicate)’ is a super-category of ‘keiynu(axlj)’, and “kanojo(she)” is an instance of ‘human’. Note that the node labels of both “have” in rule instance (c) and lower ‘pred’ in rule instance (b) are changed from that of the corresponding Japanese word “kirei(beautiful)” by the \( G_c \) node relabeling procedure.

Another example is shown in Figure 6, which shows how the Japanese sentence “US ga ... wo fusegu tame ni buhin ni kanzei wo kakeru” is translated into the English sentence “US imposes tax on parts in order to blockade .... ”. In this example, (a) is an input structure, (b), (c) and (d) are matched rules, and (e) is the output structure produced. The Japanese verb “kakeru” has several translation candidates associated with different governing words, as shown in the following table:

| Similarity | Japanese English |
|------------|------------------|
| 5.988      | (meishi) ni zeiken wo kakeru |
| 3.077      | (meishi) wo saiban ni kakeru |
| 2.717      | (meishi) wo mado ni kakeru |
| 2.545      | (meishi) wo suzou ni kakeru |
| 2.040      | hankati ni kousai wo kakeru |

This table lists the top five similar rules for the part “buhin ni kanzei wo kakeru” of the input. As shown...
Figure 5: Example 1 of translation by SimTran

Figure 6: Example 2 of translation by SimTran
in this table, rule (c) is the most similar one. Note that this similarity calculation was done for all rules, including non-lexical translation rules. There were no appropriate example rules for the part “US ga kakeru,” and a non-lexical rule (b) was therefore selected. Further, note that the lexical forms in *3 nodes of (c) and (e) are different, and that *4 node of (c) has no lexical form other than a preposition, whereas *4 node of (e) has a lexical form. The former was obtained by pre-lexicalization, and the latter by post-lexicalization.

6 Related Work

Although there were several early experimental projects on CBMT [4][9][11], MBT-II [10] is the first working prototype of a case-based transfer system, and demonstrates the promise of the CBMT approach. It uses Japanese-to-English translation examples as translation rules, chooses the source trees of examples that are most similar to the input tree from the root node down to the leaves, and assembles those target trees to produce an output tree. With respect to the transducing mechanism, MBT-II is a tree-to-tree transducer adopting one-to-one correspondence.

MT by LTAGs [1], although it is not an attempt of CBMT, proposed a similar mechanism to RCT described in this paper. It uses paired derivation trees of English and French as translation rules. An input sentence is parsed by the source grammar, and at the same time, its output tree is generated by derivation pairs of trees used in the parsing. As a transducer, this mechanism is also a tree-to-tree transducer adopting one-to-one correspondence.

In contrast, the RCT employed in SimTran is a rldag-to-rldag transducer adopting upward and downward correspondences. These extended correspondences are desirable for expressing the structural discrepancies that often occur in translation. Moreover, this transducing model is a parallel production system [2] that can produce an output structure in one execution of gluing if all the G_r's required to produce an output are supplied.

7 Conclusion

In this paper, I described a case-based transfer system, SimTran, which combines RCT with a similarity calculation method for graphs whose nodes consist of syntactic and semantic features, and show that a translation rule that has no lexical forms can be used as a default rule, that is, that such rules can provide a fail-safe mechanism if there are no appropriate translation examples.

References

[1] Abeillé, A., Schabes, Y., and Joshi, A. K., “Using Lexicalized Tags for Machine Translation,” Proc. of Coling 90, 1990.
[2] Ehrig, H., “Introduction to the Algebraic Theory of Graph Grammars,” Proc. of International Workshop on Graph Grammars, LNCS 73, 1-65, 1979.
[3] Kolodner, J. and Riesbeck, C., “Case-Based Reasoning,” tutorial textbook of 11th IJCAI, 1989.
[4] 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.
[5] National Language Research Institute: Bunrui Goi Hyou (in Japanese), Syuusi Syuppan, 1964.
[6] Maruyama, H. and Watanabe, H., “Tree Cover Search Algorithm for Example-Based Translation,” Proc. of 4th Int. Conf. on Theoretical and Methodological Issues in Machine Translation, 1992.
[7] Melby, A. K., “Lexical Transfer: A Missing Element in Linguistic Theories,” Proc. of Coling 86, pp. 104-106, 1986.
[8] Nitta, Y., “Idiosyncratic Gap: A Tough Problem to Structure-bound Machine Translation,” Proc. of Coling 86, pp. 107-111, 1986.
[9] Sadler, V., “Working with Analogical Semantics,” Foris Publications, 1989.
[10] Sato, S. and Nagao, M., “Toward Memory-based Translation,” Coling 90, 1990.
[11] Sumita, E., Iida, H., and Kohyama, H., “Translating with Examples: A New Approach to Machine Translation,” Proc. of Info Japan 90, 1990.
[12] Tomita, M. and Carbonell, J. G., “Another Stride Towards Knowledge-Based Machine Translation,” Proc. of Coling 86, 1986.
[13] Ushijima, J. and Fujita, K., “Lexical Transfer based on bilingual signs: Towards interaction during transfer,” Proc. of Seoul Int. Conf. on NLP, 1990.
[14] Watanabe, H., “A Model of a Transfer Process Using Combinations of Translation Rules,” Proc. of Pacific Rim of Int. Conf. on AI ’90, 1990.
[15] Watanabe, H., “A Formal Model of Transfer Using Rules Combination,” submitted to Machine Translation, 1991.