Example-Based Machine Translation

Satoshi Sato

September 1991

Kyoto University

Abstract

In an *expert system*, the source of intelligence is the *knowledge base*. Almost all *knowledge* in knowledge bases is in the form of rules, as they are also known as *rule bases*. The performance of an expert system strongly depends on the quality of the knowledge base; in order to construct a good expert system, we have to construct a good knowledge base.

The process as constructing a knowledge base is called *knowledge acquisition*. The main task of knowledge acquisition is to encode expert knowledge in the form of rules. This task is very difficult, and it needs a great deal of effort over a long period to construct a good knowledge base. This difficulty, *the knowledge acquisition bottleneck*, prevents the rapid development of expert systems.

We need some new technique to overcome the knowledge acquisition bottleneck. There are two directions. One is *example-based rule learning*, the other is *example-based reasoning*.

This thesis describes both example-based learning and reasoning in machine translation.

First, the example-based rule learning method was investigated. Chapter 2 describes a method for learning a set of rules for language translation from positive and negative examples. Learning language translation is categorized as *learning to perform a multiple-step task*, which is the most difficult class of machine learning problems. The author shows that the formalism of *translation grammar* and its learning algorithm make it possible to learn language translation. The proposed method of learning automatically learns from positive and negative examples, and guarantees that the obtained translation grammar satisfies all given examples. The author has implemented a machine learning/translation system,

called Takuma II, which is the first learning system for natural language translation. Experiments in constructing English-Japanese translation grammars have shown that the system can discover the correspondences of words, words groups, and phrase structures between two languages, and represent them in a translation grammar.

Second, example-based reasoning methods were investigated. Example-based reasoning frees us from the need for rule acquisition, because it directly uses examples in the reasoning process. In this framework, we can construct translation systems simply by collecting translation examples, and improve them by adding appropriate translation examples.

Chapter 3 describes the first prototype of an example-based translation system, MBT1, which can solve the word selection problem for translation between verb frame instances. This method consists of three components: the translation database, the definition of metric, and the translation process. The translation database is the collection of translation examples. A translation example is a pair of verb-frame instances. A verb frame instance is one verb with several nouns as its arguments. The metric is defined, which measures the 'distance' between a translation candidate and a translation example in the database. In the translation process, MBT1 generates all candidate translations. For each candidate, MBT1 retrieves the most similar translation example and computes the score of the candidate based on the above metric. MBT1 uses the score to evaluate the correctness of the candidate. MBT1 has been implemented in English-Japanese translation and the experiments have shown how well MBT1 solves the word selection problem. The major limitation of MBT1 is that it requires a fixed-format database and can not manage free-form data like sentences which have optional elements. Still MBT1 can be applicable to other subtasks in machine translation.

Chapter 4 describes the second prototype of an example-based translation system, MBT2. It can transfer full sentences represented by word-dependency trees. A key problem in the implementation is how to utilize more than one translation example for translating a source sentence. This problem arises from the fact that a long sentence is too large to be matched by one translation example. It is a critical problem for example-based translation, and the author shows a solution for it in MBT2. The author introduces the representation, called the *matching expression*, which represents the combination of fragments of translation examples. The translation process consists of three steps: (1) Make the source matching expression from the source sentence. (2) Transfer the source matching expression into the target matching expression. (3) Construct the target sentence from the target matching expression. This mechanism generates some candidate translations. To select the best translation from them, the score of a translation was defined. MBT2 has implemented in English-Japanese translation and has demonstrated the ability of MBT2. Although MBT2 covers only the transfer phase, it can be extended to cover the whole translation process. The proposed method will be used as a basic method to implement a complete example-based translation system. MBT2 inherits some advantages from the example-based translation idea: it is easy to construct and upgrade the system, to produce high quality translation, and to produce an intuitive explanation why the system generates a translation output.

Chapter 5 first discusses the relations and differences between rule-based approach and example-based approach from the viewpoint of learning. The major differences are: (1) whether or not they use rules as an intermediate representation which holds results of generalization , and (2) whether they use exact match reasoning or best match reasoning. Rule learning corresponds to understanding or making explanations for some phenomena in a task, and example-based reasoning corresponds to constructing a task executor. The example-based approach seems more promising method than the rule-based approach for constructing machine translation systems. Second, the example-based translation family is discussed. It can be divided into three groups: translation aid systems, word selection systems, and fully translation systems. Their current status and future prospects are discussed.

Chapter 6 outlines the conclusions of this thesis.

Acknowledgments

I would like to acknowledge my enormous debt of gratitude to Professor Makoto Nagao of Kyoto University for supervision and continuous encouragement.

I also would like to thank Professor Jun-ichi Tsujii of the University of Manchester and Professor Jun-ichi Nakamura of Kyusyu Institute of Technology for constructive and fruitful discussions when they were at Kyoto University.

I am grateful to all previous and current members of Professor Nagao's laboratory, especially Professor Takashi Matsuyama of Okayama University, Professor Yuji Matsumoto and Professor Yuichi Nakamura of Kyoto University, Professor Tetsuya Takahama of Fukui University, and Mr. Itsuki Noda of Kyoto University.

I am also grateful to all participants in the Workshop of Learning '89 and '90, especially Dr. Hideyuki Nakashima and Dr. Hitoshi Matsubara of Electrotechnical Laboratory, for their constructive discussions.

I would like to thank Dr. Nigel Ward of the University of Tokyo for his helpful comments on a draft of this thesis.

Contents

Chapter 1

Introduction

1.1 Knowledge Acquisition Bottleneck

In an *expert system*, the source of intelligence is the *knowledge base*. Almost all *knowledge* in knowledge bases is in the form of rules, as they are also known as *rule bases*. The performance of an expert system strongly depends on the quality of the knowledge base; in order to construct a good expert system, we have to construct a good knowledge base.

The process of constructing a knowledge base is called *knowledge acquisition*. The main task of knowledge acquisition is to encode expert knowledge in the form of rules. This task is very difficult, and it needs a great deal of effort over a long period to construct a good knowledge base. This difficulty, *the knowledge acquisition bottleneck*, prevents the rapid development of expert systems.

Machine translation systems are among the largest and most complicated expert systems. For example, the Mu system [Nagao et al 85] [J. Nakamura 88] has a large knowledge base, with about 3000 rules for analysis, transfer and generation. In addition, there are a large number of special rules for individual words in the dictionary. Knowledge acquisition was very difficult task to the Mu system: a dozen *grammar writers* worked to do it. In this process, the most difficult problem was the debugging of the knowledge base. If the system outputs an incorrect translation, we must correct the faulty rules. But it is very difficult to find the rules at fault, because nobody knows all the rules and how these rules behave in certain specific situations. The larger the size of the knowledge base, the more difficult the debugging.

We need some new technique to overcome the knowledge acquisition bottleneck: it must cover not only constructing knowledge bases, but also debugging them. There are two directions. One is *example-based rule learning*, the other is *example-based reasoning*.

1.2 Rule Learning from Examples

Example-based rule learning is one way to overcome the knowledge acquisition bottleneck. *Learning from examples* allows one to make rules from the training examples. There are two major methods in learning from examples; *similarity-based learning* and *explanation-based learning*.

Similarity-based learning is a purely empirical, data-intensive method that relies on large numbers of training examples to constrain the search for the correct generalization. This method employs some kind of inductive bias to guide the inductive leap that it must make in order to infer a rule from only a subset of its input–output pairs.

On the other hand, explanation-based learning uses *domain knowledge* in order to constrain the search for a correct generalization. After analyzing a single training example in terms of this knowledge, this method is able to produce a valid generalization of the example along with a deductive justification of the generalization in terms of the system's knowledge.

Explanation-based learning is suited for learning more efficient ways to apply knowledge, but not for learning new domain knowledge itself. There is only one choice, i.e. similarity-based learning, for learning domain knowledge.

Almost all research on learning from examples have developed methods for learning a single concept. The task is not so difficult. But we have to develop a method for learning to perform multiple-step tasks, because expert systems perform multiple-step tasks. Research on *grammatical inference* contains some methods for learning a set of rules that perform multiple-step tasks; i.e. parsing and generation of sentences.

1.3 Example-Based Reasoning

In recent years, a new approach has gradually been developed. It has two historical sources. One is research on *analogical reasoning*, this has long history in artificial intelligence [Hall 89]. Another source is the research by Schank, on "*Dynamic Memory*"[Schank 82], which shows the importance of previous experiences and their memory organization. These sources have given rise to a new paradigm, called *case-based reasoning(CBR)*, *memory-based reasoning(MBR)* or *example-based reasoning*.

These methods are basically composed of the following process:

1. Store previous experiences (*cases*) into a *case database*.
2. When a problem arises, retrieve a case similar to that problem.
3. Answer the problem by adjusting the retrieved case.

These methods are not only reasoning methods but also learning methods. "*DARPA: Machine Learning Program Plan*"[DARPA 89] reports that CBR constitutes a fifth major paradigm of machine learning research.

Memory-based reasoning demonstrates the power of memories (case database) [Stanfill & Waltz 86]. It has been implemented on the Connection Machine [Hillis 85], a massively parallel computer. Parallel search of the database makes this method practical for large databases.

In the context of machine translation research, Nagao proposed the idea of *translation by analogy* [Nagao 84]. It is also a pioneering study in case-based reasoning. But it was a discussion on conceptual level, and was not implemented on a computer.

The major merit of these methods is that there is no need to construct rules for the task. All that the developer needs is to collect cases or examples, which is much easier than constructing rules. This solves the knowledge acquisition bottleneck completely.

1.4 Outline of the Thesis

This thesis describes both example-based learning and reasoning in machine translation.

Chapter 2 describes a method for learning a set of rules for language translation by using positive and negative examples. First, the author proposes a *translation grammar*, which can generate or accept pairs of bilingual sentences and perform bidirectional translation. Second, the author shows that a translation grammar can be learned by grammatical inference techniques. Two processes for learning a translation grammar are developed; the *batch learning process* and the *incremental learning process*. The former can generate a translation grammar which satisfies a given set of positive and negative examples. The latter can improve a translation grammar incrementally by making it handle a newly given positive or negative examples. The author has implemented a machine learning/translation system, called Takuma II, which has this learning ability. In the experiments of constructing English-Japanese translation grammars, the system found the correspondences of words, words groups, and phrase structures between the two languages, and represented them as a translation grammar.

Chapter 3 proposes a prototype example-based translation system, called MBT1, which is a method to select the best target words in the translation between verb frame instances. MBT1 consists of three components: the translation database, the definition of metric, and the translation algorithm. The translation database is the collection of translation examples. A translation example is a pair of verb-frame instances. A verb frame instance is one verb with several nouns as its arguments. The author defines a metric which measures the 'distance' between a translation candidate and a translation example in the database. In the translation process, MBT1 generates many candidate translations. For each candidate, MBT1 retrieves the most similar translation example and scores of the candidate based on the metric. MBT1 uses the value to evaluate the appropriateness of the candidate. MBT1 has been implemented in English-Japanese translation.

Chapter 4 describes a solution to a critical problem for example-based translation; how to utilize more than one translation example for translating a source sentence. This chapter introduces 'matching expressions', which represent the combination of fragments of translation examples. The translation process in this model consists of three steps: (1) Make the source matching expression from the source sentence. (2) Transfer the source matching expression into the target matching expression. (3) Construct the target sentence from the target matching expression. This mechanism generates some candidate translations. In order to select the best, the score of a translation is defined. MBT2 has been implemented for English-Japanese translation.

Chapter 5 first discusses the relations and differences between rule learning and example-based reasoning. Rule learning corresponds to the understanding or the making of explanations, and example-based reasoning corresponds to constructing an executor for a task. These are two different types of learning, and it seems that example-based reasoning is a more promising method than rule learning for constructing machine translation systems. Second, the comparative characteristics of various example-based translation approaches, namely, translation aid systems, word selection systems, and full translation systems, are discussed. Their current status and future prospects are discussed.

Chapter 6 outlines the conclusions of this thesis.

Chapter 2

Learning Translation Rules

2.1 Introduction

Learning from observation is the process of constructing descriptions, hypotheses or theories about a given collection of facts or observations. There are many difficult problems in implementing the learning ability on machines. For example, since a learning system has no a priori information exemplifying desired theories or structures, the system has to construct rules or theories that satisfy all given positive and negative examples [Ohsuga 86]. In most learning situations, nobody knows the desired goal of learning, so that it is impossible to presuppose something like an *oracle* which is used in Model Inference System [Shapiro 82].

This chapter describes an attempt to implement the learning ability on the domain of language translation. We assume that the system is given positive and negative translation examples (pairs of sentences). The system is required to construct a set of rules that satisfies all given examples without a teacher and to predict translation equivalents for *unknown* sentences.

Language translation is obviously a multiple-step task: a sentence is translated by applying a sequence of rules. Therefore learning language translation is learning a set of rules, not a rule. This type of learning is categorized as *learning a rule set* or *learning to perform a multiple-step task*. It is the most difficult class of learning, because the learner has to do the followings.

1. Divide a whole task into subtasks.

2. Infer examples of subtasks from given examples of the whole task, because examples of subtasks are not given explicitly.

3. Infer rules which perform subtasks.

4. Keep consistency of rules.

To make the problem tractable, the author made the following three assumptions.

- Many translation pairs with similar constructions are given.

- They are not so complicated (containing one or two predicates).

- The given examples have no noise. [1]

The following characterizes the approach the author adopted.

1. The idea of a system which acquires linguistic knowledge for translation from examples has already been suggested in [Nagao 84]. Following the same lines, the author devised a new formalism for representing *knowledge* of translation. The formalism is called *Translation Grammar*. A translation grammar is a set of rules for bidirectional translation, and it can generate or accept a set of translations (pairs of sentences).

2. The framework of grammatical inference [Gold 67] [Biermann & Feldman 72] [Fu 74] [Fu & Booth 75] [Dietterich et al 82] is used to learn a translation grammar.

3. Two learning processes for a translation grammar are developed. The batch learning process generates a translation grammar from a set of given examples. The incremental learning process improves a translation grammar to satisfy a newly given example.

A machine learning/translation system, called Takuma II[2], which has this learning ability was developed.

[1] There is no translation example that is positive *and* negative.

[2] This name comes from the Japanese phrase "Sessa Takuma", which means "improving oneself by competing with each other".

This chapter is organized as follows. The next section describes the outline of Takuma II. Section 2.3 defines the notation of translation grammar, and Section 2.4 describes a method for learning a translation grammar. Section 2.5 describes some experiments in constructing translation grammars between English and Japanese. The last section summarizes.

2.2 Outline of Takuma II

Figure 2.1 shows the outline of Takuma II. It consists of two major modules, the learning engine and the translation engine. The former acquires a translation grammar from examples, and the latter translates sentences using the translation grammar.

A translation grammar is obtained by Takuma II through the following steps.

1. A set of positive and negative examples is given.
2. Takuma II constructs a translation grammar which satisfies the given examples using the batch learning process.
3. A new training sentence is given to Takuma II.
4. If Takuma II cannot translate the sentence, then a correct translation of the sentence is given by the human trainer. If Takuma II outputs an incorrect translation, then the human trainer suggests to the system that it is wrong. Takuma II updates the translation grammar so as to satisfy the newly given example, by using the incremental learning process.
5. Go to 3.

Takuma II is implemented in Zetalisp and Flavors on a Symbolics 3600/3640.

2.3 Translation Grammar

In this section, we describe translation grammar, which is the basic representational framework of knowledge of Takuma II. We will first define *translation grammar* formally and then explain how the grammar performs translation.

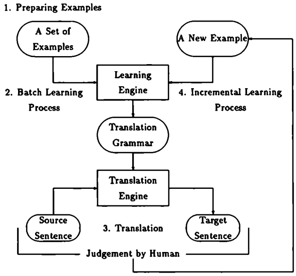


Figure 2.1: Outline of Takuma II

2.3.1 Definition

A Translation Grammar G between language A and language B is a 5-tuple,

$$G = < V_N, V_{TA}, V_{TB}, P, \sigma >$$

where

V_N: finite set of nonterminal symbols of language A and B

V_{TA}: finite set of terminal symbols of language A, $V_{TA} \cap V_N = \phi$

V_{TB}: finite set of terminal symbols of language B, $V_{TB} \cap V_N = \phi$

$\sigma \in V_N$: start symbol

P: finite set of translation rules (productions) of the form

$\xi_A \leftarrow X \rightarrow \xi_B$

where

$X \in V_N$: left hand side (LHS)

$\xi_A \in V_A^+$: language A's right hand side (RHS-A), where $V_A = V_N \cup V_{TA}$

$\xi_B \in V_B^+$: language B's right hand side (RHS-B), where $V_B = V_N \cup V_{TB}$

and satisfies the following condition.

If RHS-A has some nonterminal symbols, then RHS-B must have the same nonterminal symbols corresponding to the symbols in RHS-A.

If $\xi_A \leftarrow X \rightarrow \xi_B$ is a rule of P, α and β are any string of V_A^*, and γ and δ are any string of V_B^*, then the rule $\xi_A \leftarrow X \rightarrow \xi_B$ may be applied to the pair of strings $[\alpha X \beta,\ \gamma X \delta]$ to obtain $[\alpha \xi_A \beta,\ \gamma \xi_B \delta]$. This process is denoted as $[\alpha X \beta,\ \gamma X \delta] \Rightarrow [\alpha \xi_A \beta,\ \gamma \xi_B \delta]$. The reflexive transitive closure of $\Rightarrow$ is denoted $\Rightarrow^*$. For any translation grammar G, the set of translations (pairs of strings) $T(G)$ generated by G is defined by

$$T(G) = \{[a,\ b] \mid [\sigma,\ \sigma] \Rightarrow^* [a,\ b]\ ,\ a \in V_{TA}^+ \text{ and } b \in V_{TB}^+\}.$$

```
%0001 %0002 %0003  . ← %S → %0001 は %0003 を %0002 。   (R1)
                 I ← %0001 → 私                         (R2)
               you ← %0001 → あなた                     (R3)
             speak ← %0002 → 話す                       (R4)
             learn ← %0002 → 学ぶ                       (R5)
           English ← %0003 → 英語                       (R6)
          Japanese ← %0003 → 日本語                     (R7)
```

Figure 2.2: Example of Translation Grammar

Figure 2.2 shows an example of a translation grammar. Symbols starting with '%' are nonterminal symbols and other symbols are terminal symbols. A nonterminal symbol represents a syntactic or semantic group at translation. Two RHS's in a rule represent a correspondence between two languages.

2.3.2 Translation Method

A translation grammar can perform bidirectional translation between two languages. In the following, we mainly discuss on the translation from language A to language B. In principle, translating a source sentence a ($\in V_A^+$) is to find translations $[a,\ b_i]$ for $i = 1 \ldots n$, which are generated by a given translation grammar, and b_i for $i = 1 \ldots n$ are target sentences.

Bidirectional translation can be performed by a slightly changed CFG parser which outputs all parse trees. The difference from ordinary CFG parsing is that a source language's RHS is used for pattern matching and a target language's RHS is used for constructing a tree. For example, in the translation from language A to language B, the rule $\xi_A \leftarrow X \rightarrow \xi_B$ is interpreted as the following tree construction rule.

Condition: If the pattern ξ_A matches a subsequence of the input,

Action: then replace the subsequence by the tree whose root node is X and whose descendant nodes are trees of ξ_B. [3]

[3]Each nonterminal symbol in ξ_A and ξ_B is interpreted as a *pattern variable* which matches a

Target sentences are obtained as leaves (terminal symbols) of output trees with the initial symbols in their root nodes. Figure 2.3 shows an example of the translation process.

2.3.3 Desirable Characteristics for Learning

A translation grammar has some desirable characteristics for learning.

- A translation grammar can represent knowledge for bidirectional translation in a *uniform* style.

In the translation grammar formalism, there is only one type of rule: i.e. the *translation rule*. A translation rule is a combination of a parsing rule, a transfer rule, and a generation rule. A set of translation rules, i.e. a translation grammar, can translate an input sentence into a target sentence: it can perform the whole process of translation. By using this uniform style representation, we can concentrate on development of a learning engine for it. In contrast, if we use three representations for parsing, transfer and generation, we have to develop three learning engines for them.

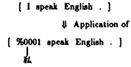
- The *single-representation trick* [Dietterich et al 82] can be used for learning a translation grammar.

A translation pair of sentences (a positive example) can be represented as an *instance translation rule*. A given set of positive examples can be represented as a translation grammar. Therefore, learning is done only on the *rule space*. We do not need to consider an *instance space* or interpretation of given instances. It simplifies the learning process.

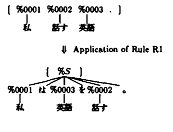
2.4 Learning a Translation Grammar

Learning a translation grammar can be viewed as grammatical inference, because a translation grammar is a *grammar* which generates or accepts a set of pairs of

tree whose root node is the nonterminal symbol, and corresponding nonterminal symbols in two RHS's are interpreted as the same variable.



(a) A Example of Rule Application



(b) Another Example of Rule Application

Figure 2.3: Translation Process

A source language's RHS is used for pattern matching, and a LHS and a target language's RHS is used for constructing a tree. Each nonterminal symbol in RHS's is interpreted as a *pattern variable* which matches a tree whose root node is a nonterminal symbol, and corresponding nonterminal symbols in two RHS's are interpreted as the same variable.

strings. Therefore techniques of grammatical inference for context free grammars [Dietterich et al 82] [Knobe & Knobe 76] can be applied for acquiring translation grammars, with some extensions. In this section, we will first define seven operators to update a translation grammar, and then describe two learning processes.

2.4.1 Operators

The following seven operators are used in learning process to update a translation grammar.

1. Adding a new rule.
   Add a new rule to the grammar.

2. Deleting a rule.
   Delete a rule from the grammar.

3. Creating a new nonterminal symbol.
   Create a new nonterminal symbol X for some pairs of strings

   $$[\xi_{A1}, \xi_{B1}], [\xi_{A2}, \xi_{B2}], \ldots, [\xi_{An}, \xi_{Bn}]$$

   and add rules

   $$\xi_{A1} \leftarrow X \rightarrow \xi_{B1},$$
   $$\xi_{A2} \leftarrow X \rightarrow \xi_{B2},$$
   $$\ldots,$$
   $$\xi_{An} \leftarrow X \rightarrow \xi_{Bn}$$

   to the grammar.

4. Generalizing a rule.
   If a rule

   $$\xi_A \leftarrow X \rightarrow \xi_B$$

   is in the grammar, then replace a rule

   $$\alpha\xi_A\beta \leftarrow Y \rightarrow \gamma\xi_B\delta$$

in the grammar by a rule

$$\alpha X \beta \leftarrow Y \rightarrow \gamma X \delta.$$

5. Integration of rules.

   If some rules

   $$\alpha \xi_{A1} \beta \leftarrow Y \rightarrow \gamma \xi_{B1} \delta,$$
   $$\alpha \xi_{A2} \beta \leftarrow Y \rightarrow \gamma \xi_{B2} \delta,$$
   $$\ldots,$$
   $$\alpha \xi_{An} \beta \leftarrow Y \rightarrow \gamma \xi_{Bn} \delta$$

   are in the grammar, then replace these rules by a rule

   $$\alpha X \beta \leftarrow Y \rightarrow \gamma X \delta$$

   by applying an operator of creating a new nonterminal symbol X for pairs of strings

   $$[\xi_{A1}, \xi_{B1}], [\xi_{A2}, \xi_{B2}], \ldots, [\xi_{An}, \xi_{Bn}].$$

6. Merging nonterminal symbols.

   Create a new nonterminal symbol W and replace all occurrences of nonterminal symbols $X_1, X_2, ..., X_n$ in the grammar by W.

7. Expanding a nonterminal symbol.

   If a rule

   $$\alpha X \beta \leftarrow Y \rightarrow \gamma X \delta$$

   is in the grammar, and if rules with the nonterminal symbol X in the LHS are

   $$\xi_{A1} \leftarrow X \rightarrow \xi_{B1},$$
   $$\xi_{A2} \leftarrow X \rightarrow \xi_{B2},$$
   $$\ldots,$$
   $$\xi_{An} \leftarrow X \rightarrow \xi_{Bn},$$

   then replace the rule

$$\alpha X \beta \leftarrow Y \rightarrow \gamma X \delta$$

by the rules

$$\alpha \xi_{A1} \beta \leftarrow Y \rightarrow \gamma \xi_{B1} \delta,$$
$$\alpha \xi_{A2} \beta \leftarrow Y \rightarrow \gamma \xi_{B2} \delta,$$
$$\ldots,$$
$$\alpha \xi_{An} \beta \leftarrow Y \rightarrow \gamma \xi_{Bn} \delta.$$

These seven operators are divided into the following three groups according to the relationships between $T(G)$ and $T(G')$, where G is a grammar before application of an operator and G' is the grammar obtained by application of the operator.

a. Reformulation operators: $T(G) = T(G')$: 3, 5, 7.

b. Generalization operators: $T(G) \subseteq T(G')$: 1, 4, 6.

c. Specialization operators: $T(G) \supseteq T(G')$: 2.

To construct a translation grammar which satisfies all given examples, operators of generalization and specialization should be applied carefully. The following are the conditions of application of these operators.

- Generalization operators are applicable only if the updated grammar does not incorrectly cover any negative examples.
- Specialization operators are applicable only if the updated grammar covers all positive examples.

These applicability conditions guarantee that the application of these operators does not make a translation grammar inconsistent with the examples given.

2.4.2 Batch Learning Process

The purpose of the batch learning process is to generalize and simplify a given translation grammar, while maintaining consistency with the given negative examples. The specialization operators are not used in this process. In constructing a translation grammar from a given set of positive and negative examples, Takuma

II first creates an initial (instance) grammar from a given set of positive examples. An initial grammar is a set of instance rules which of the form

$$a_i \leftarrow \sigma \rightarrow b_i$$

where $[a_i, b_i]$ for $i = 1 \ldots n$ are given positive examples and σ is the initial symbol of the grammar. The initial grammar does not accept any given negative examples under the condition that given examples have no noise. Second, the following algorithm (batch learning algorithm) is used to generalize and simplify the initial grammar.

1. Compare all possible pairs of rules in the grammar, and get differences between the two rules. The difference between $\alpha\xi_{A1}\beta \leftarrow Y \rightarrow \gamma\xi_{B1}\delta$ and $\alpha\xi_{A2}\beta \leftarrow Y \rightarrow \gamma\xi_{B2}\delta$ is a 4-tuple $< \xi_{A1}, \xi_{A2}, \xi_{B1}, \xi_{B2} >$.

2. Categorize the differences obtained at step 1 into the following types.

   Types of Differences

   Type A: $|\xi_{A1}| = |\xi_{A2}| = |\xi_{B1}| = |\xi_{B2}| = 0$

   Type B: $\xi_{Ai} = \xi_{Bi} \in V_N$, $|\xi_{Aj}| \geq 1$ and $|\xi_{Bj}| \geq 1$ except $\xi_{Aj} = \xi_{Bj} \in V_N$, where $i = 1, j = 2$ or $i = 2, j = 1$

   Type C: $|\xi_{Ai}| = |\xi_{Bi}| = 0$, $|\xi_{Aj}| \geq 1$ and $|\xi_{Bj}| \geq 1$ except $\xi_{Aj} = \xi_{Bj} \in V_N$, where $i = 1, j = 2$ or $i = 2, j = 1$

   Type D: $\xi_{A1} = \xi_{B1} \in V_N$ and $\xi_{A2} = \xi_{B2} \in V_N$

   Type E: $1 \leq |\xi_{A1}| \leq ml$, $1 \leq |\xi_{A2}| \leq ml$, $1 \leq |\xi_{B1}| \leq ml$, $1 \leq |\xi_{B2}| \leq ml$, and

   $$\frac{|\xi_{A1}| + |\xi_{A2}| + |\xi_{B1}| + |\xi_{B2}|}{|\alpha\xi_{A1}\beta| + |\gamma\xi_{A2}\delta| + |\alpha\xi_{B1}\beta| + |\gamma\xi_{B2}\delta|} < N$$

   where ml and N are parameters. [4]

   Type F: Otherwise

3. The differences obtained by step 1 are ordered in such a way that the difference easiest to resolve comes first, the second easiest comes next and so

[4] $ml = 4$ and $N = 0.5$ are used in experiments.

on. The ordering is made based on their types.

Order of Difference

(a) The differences of Type A is taken as easiest to resolve and those of Type F as hardest.

$$\text{Type A} < \text{Type B} < \ldots < \text{Type F}$$

(b) In the same types, the total length of difference ($|\xi_{A1}| + |\xi_{A2}| + |\xi_{B1}| + |\xi_{B2}|$) is taken into account. The shorter differences precede the longer.

4. Update the grammar by applying one of the following heuristic rules to the least (minimum) difference obtained by step 3. The heuristic rules are associated with the types of differences of two rules. If a rule application succeeds, then go to next step. If not, then apply the rule to the next least difference.

Heuristic Rules

Type A: Delete one of these two rules.

Type B: Try to add the rule

$\xi_{Aj} \leftarrow \xi_{Ai} \rightarrow \xi_{Bj}$.

If it succeeds[5], delete the rule

$\alpha\xi_{Aj}\beta \leftarrow Y \rightarrow \gamma\xi_{Bj}\delta$.

If not, fail.

Type C: Create a new nonterminal symbol for the pair of strings $[\xi_{Aj}, \xi_{Bj}]$.

Type D: Try to merge ξ_{A1} and ξ_{A2}. If it fails, then integrate two rules.

Type E: Integrate two rules.

Type F: Terminate the batch learning process.

5. If new rules are added at step 4, try to generalize the grammar by applying the new rules to rules in the grammar. If a new rule is further added, repeat this step recursively.

[5]It satisfies the applicable conditions described in the previous subsection.

6. Go to step 1.

2.4.3 Incremental Learning Process

The purpose of the incremental learning process is to modify a translation grammar to satisfy a newly given positive or negative example. There are two cases; a new positive example is given, and a new negative example is given.

If a new positive example which cannot be translated by the current grammar is given, the system gets the most general partial parse[6] by applying the grammar to the positive instance, and tries to add a new rule which has the initial symbol in the LHS and the partial parse in the RHS's. If it succeeds, the system invokes the batch learning algorithm (Section 2.4.2) to generalize or simplify the grammar. If it fails, the system tries to add a new rule that is made from the next most general partial parse. This method is an extension of [Knobe & Knobe 76] for translation grammars.

If a new negative example which can be translated by the current grammar is given, the system finds overgeneralized rules and specializes the grammar by the following algorithm.

1. Find a nonterminal symbol to be expanded.
   The system constructs the lattice of derivation of a negative example. In the lattice, nodes represent pairs of strings and arcs represent applications of rules in the derivation. The system starts to search for a node which covers the given positive examples starting from the bottom node representing the negative instance by the breadth-first search. The node which is found first is called the *branching point*, and the LHS of the rule on the last scanned arc is the nonterminal symbol to be expanded.

2. Specify a rule which contains the nonterminal symbol obtained at Step 1.
   By breadth first search from the branching point to the root node $[\sigma, \sigma]$, the system finds a rule with the nonterminal symbol obtained at Step 1 in

[6] A partial parse is a pair of strings of terminals and nonterminals in which the original instance strings have been partly parsed into nonterminals; the more general partial parse are shorter, since most of strings has been successfully parsed.

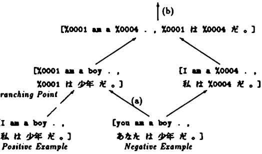


Figure 2.4: Incremental Learning Process for Negative Example

the RHS's among the rules on arcs. The rule which is found first is to be expanded.

3. Expand the nonterminal symbol.

   The system expands the nonterminal symbol obtained at Step 1 in the rule obtained at Step 2.

4. Delete the rules.

   The system tries to delete the rules which were added at Step 3 and the rules which have the nonterminal symbol obtained at Step 1 in their LHS's.

5. Check.

   The system check whether the grammar fails to generate the negative instance. If so, go to the next step. If not, go to Step 1.

6. Invoke the batch learning algorithm.

   The system invokes the batch learning algorithm to generalize and simplify the grammar.

Let's explain the execution of this algorithm by using an example (Figure 2.4). We assume that the translation grammar consists of the rules in Figure 2.2 and the following rules

```
%0001 am a %0004  . ← %S → %0001 は %0004 だ 。   (R8)
               boy ← %0001 → 少年                (R9)
              girl ← %0001 → 少女                (R10)
```

and the negative example

```
[you am a boy .] ↔ [あなた は 少年 だ 。]
```

is given. The system constructs the lattice of derivation of the negative instance (solid lines), and finds the branching point. In this example, the branching point is

```
[%0001 am a boy . , %0001 は 少年 だ 。]
```

and %0001, which is the LHS of the rule on the last scanned arc (a), is the nonterminal symbol to be expanded. The rule to be expanded is

```
%0001 am a %0004  . ← %S → %0001 は %0004 だ 。   (R8)
```

which is applied on the arc (b). Therefore the rule R8 is expanded to two rules

```
    I am a %0004  . ← %S → 私 は %0004 だ 。      (R11)
%0001 am a boy    . ← %S → %0001 は 少年 だ 。    (R12)
```

and deletions of R11, R12, R2 and R3 are tested, and R12 is deleted.

2.4.4 Characteristics

The proposed method of learning a translation grammar has the following characteristics.

- It is an automatic learning procedure.

The system acquires a translation grammar only using a set of examples. It does not require any other information sources. This is desirable since it is impossible to have a teacher or an oracle, because the system must construct a previously unknown grammar.

- The two learning processes are complementary to each other.

If the system has only the batch learning process, the system must repeatedly perform the whole learning process for a newly given example. This can be avoided by using the incremental learning process. In contrast, if the system has only the incremental learning process, one must carefully control the order of giving examples to get good learning results, because the incremental learning algorithm generally tends to be sensitive to the order of examples. This can be avoided by using the batch learning process on the first stage of learning.

2.5 Experiments

Experiments to construct simple translation grammars between English and Japanese from some sets of examples are performed. The author prepared elementary English sentences which appear in a standard English textbook used in the first grade of junior high school in Japan. The Japanese examples do not include free-word order sentences. We simply use the surface forms, which are strings of words with no additional information such as syntactic category or root-by-inflection breakdown. The size of training set is 208.

2.5.1 Trace of Experiment I31

This subsection shows the trace of an experiment (I31) in the construction of a translation grammar.

(1) Figure 2.5 shows the set of examples. The set consists thirty-one positive examples.

(2) The set of examples was given to the system. The system invoked the batch learning process (Figures 2.6–2.7), and constructed a translation grammar (Figures 2.8–2.9). The translation grammar has eleven nonterminal symbols and forty rules. We can read that %0003 is the group of nouns referring to *things*, %0006 is a group of nouns referring to *people*, %0008 is a group of noun phrases referring to *people*, and %0009 is the group of adjectives. The numbers after each rule are the ID's of the positive examples which the rule is used to translate.

(3) The system translated all given sentences of (2) (Figure 2.10–2.11). Since a translation grammar may output more than one target sentence, new translation

```
Positive instances : 31
  1 : [ I am Takuma . ] <-> [ 私 は 琢磨 だ 。 ]
  2 : [ I am Taro . ] <-> [ 私 は 太郎 だ 。 ]
  3 : [ you are Takuma . ] <-> [ あなた は 琢磨 だ 。 ]
  4 : [ you are Hanako . ] <-> [ あなた は 花子 だ 。 ]
  5 : [ I am a boy . ] <-> [ 私 は 少年 だ 。 ]
  6 : [ you are a girl . ] <-> [ あなた は 少女 だ 。 ]
  7 : [ you are a boy . ] <-> [ あなた は 少年 だ 。 ]
  8 : [ I am a tall boy . ] <-> [ 私 は 背 の 高い 少年 だ 。 ]
  9 : [ you are a small girl . ] <-> [ あなた は 小さな 少女 だ 。 ]
  10 : [ this is a book . ] <-> [ これ は 本 だ 。 ]
  11 : [ that is a book . ] <-> [ あれ は 本 だ 。 ]
  12 : [ this is an apple . ] <-> [ これ は りんご だ 。 ]
  13 : [ that is an orange . ] <-> [ あれ は オレンジ だ 。 ]
  14 : [ this is my apple . ] <-> [ これ は 私 の りんご だ 。 ]
  15 : [ this is my book . ] <-> [ これ は 私 の 本 だ 。 ]
  16 : [ that is your book . ] <-> [ あれ は あなた の 本 だ 。 ]
  17 : [ this is a dog . ] <-> [ これ は 犬 だ 。 ]
  18 : [ that is Taro's dog . ] <-> [ あれ は 太郎 の 犬 だ 。 ]
  19 : [ he is a boy . ] <-> [ 彼 は 少年 だ 。 ]
  20 : [ she is a girl . ] <-> [ 彼女 は 少女 だ 。 ]
  21 : [ he is my friend . ] <-> [ 彼 は 私 の 友達 だ 。 ]
  22 : [ she is a teacher . ] <-> [ 彼女 は 先生 だ 。 ]
  23 : [ she is Hanako's teacher . ] <-> [ 彼女 は 花子 の 先生 だ 。 ]
  24 : [ that is a tennis ball . ] <-> [ あれ は テニスボール だ 。 ]
  25 : [ it is your tennis ball . ] <-> [ それ は あなた の テニスボール だ 。 ]
  26 : [ I'm not Taro . ] <-> [ 私 は 太郎 で は ない 。 ]
  27 : [ she is a nurse . ] <-> [ 彼女 は 看護婦 だ 。 ]
  28 : [ you aren't a nurse . ] <-> [ あなた は 看護婦 で は ない 。 ]
  29 : [ he isn't my teacher . ] <-> [ 彼 は 私 の 先生 で は ない 。 ]
  30 : [ she isn't Hanako's mother . ] <-> [ 彼女 は 花子 の 母親 で は ない 。 ]
  31 : [ it isn't a tennis ball . ] <-> [ それ は テニスボール で は ない 。 ]
Negative instances : 0
```

Figure 2.5: Prepared Examples

```
[Batch learning loop : 1 ] ...
Change.
  [you are a %0001 girl .  <- %S -> あなた は %0001 少女 だ 。 : 9 ]
  + [small  <- %0001 -> 小さな : 9 ]
  <= [you are a small girl .  <- %S -> あなた は 小さな 少女 だ 。 : 9 ]
     ([you are a girl .  <- %S -> あなた は 少女 だ 。 : 6 ]).
[Batch learning loop : 2 ] ...
Change.
  [I am a %0002 boy .  <- %S -> 私 は %0002 少年 だ 。 : 8 ]
  + [tall  <- %0002 -> 背 の 高い : 8 ]
  <= [I am a tall boy .  <- %S -> 私 は 背 の 高い 少年 だ 。 : 8 ]
     ([I am a boy .  <- %S -> 私 は 少年 だ 。 : 5 ]).
[Batch learning loop : 3 ] ...
Integration.
  [this is my %0003 .  <- %S -> これ は 私 の %0003 だ 。 : 14 15 ]
  + [book  <- %0003 -> 本 : 15 ] + [apple  <- %0003 -> りんご : 14 ]
  <= [this is my book .  <- %S -> これ は 私 の 本 だ 。 : 15 ]
   + [this is my apple .  <- %S -> これ は 私 の りんご だ 。 : 14 ]
Checking the following generalization.
  [that is your %0003 .  <- %S -> あれ は あなた の %0003 だ 。 : ]
  <= [that is your book .  <- %S -> あれ は あなた の 本 だ 。 : 16 ] ... OK.
Checking the following generalization.
  [that is a %0003 .  <- %S -> あれ は %0003 だ 。 : ]
  <= [that is a book .  <- %S -> あれ は 本 だ 。 : 11 ] ... OK.
Checking the following generalization.
  [this is a %0003 .  <- %S -> これ は %0003 だ 。 : ]
  <= [this is a book .  <- %S -> これ は 本 だ 。 : 10 ] ... OK.
Checking the following generalization.
  [this is an %0003 .  <- %S -> これ は %0003 だ 。 : ]
  <= [this is an apple .  <- %S -> これ は りんご だ 。 : 12 ] ... OK.
[Batch learning loop : 4 ] ...
Checking the following change.
  [this is a %0003 .  <- %S -> これ は %0003 だ 。 : 10 ] (already exist)
  + [dog  <- %0003 -> 犬 : ]
  <= [this is a dog .  <- %S -> これ は 犬 だ 。 : 17 ] ...OK.
```

Figure 2.6: Trace of Batch Learning Process

```
Checking the following generalization.
  [that is Taro's %0003 .  <- %S -> あれ は 太郎 の %0003 だ 。 : ]
  <= [that is Taro's dog .  <- %S -> あれ は 太郎 の 犬 だ 。 : 18 ] ... OK.
[Batch learning loop : 5 ] ...
.
[Batch learning loop : 20 ] ...
Integration.
  [a %0009 %0006  <- %0008 -> %0009 %0006 : ]
  + [%0002  <- %0009 -> %0002 : ] + [%0001  <- %0009 -> %0001 : ]
  <= [a %0002 %0006  <- %0008 -> %0002 %0006 : 8 ]
   + [a %0001 %0006  <- %0008 -> %0001 %0006 : 9 ]
Checking the following merging.
  [ %0009 ] <= [ %0002 ] + [ %0001 ] ... OK.
[Batch learning loop : 21 ] ...
.
[Batch learning loop : 29 ] ...
Integration.
  [%0011 %0008 .  <- %S -> %0011 は %0008 で は ない 。 : 26 28 ]
  + [you aren't  <- %0011 -> あなた : 28 ] + [I'm not  <- %0011 -> 私 : 26 ]
  <= [you aren't %0008 .  <- %S -> あなた は %0008 で は ない 。 : 28 ]
   + [I'm not %0008 .  <- %S -> 私 は %0008 で は ない 。 : 26 ]
[Batch learning loop : 30 ] ...
Integration.
  [%0012 %0008 .  <- %S -> %0012 は %0008 だ 。 : 1 2 3 4 5 6 7 8 9 ]
  + [I am  <- %0012 -> 私 : 1 2 5 8 ] + [you are  <- %0012 -> あなた : 3 4 6 7 9 ]
  <= [I am %0008 .  <- %S -> 私 は %0008 だ 。 : 1 2 5 8 ]
   + [you are %0008 .  <- %S -> あなた は %0008 だ 。 : 3 4 6 7 9 ]
[Batch learning loop : 31 ] ... Terminate.
```

Figure 2.7: Trace of Batch Learning Process (cont.)

```
Rules : 40
  book <- %0003 -> 本 : 10 11 15 16
  apple <- %0003 -> りんご : 12 14
  dog <- %0003 -> 犬 : 17 18
  tennis ball <- %0003 -> テニスボール : 24 25 31
  orange <- %0003 -> オレンジ : 13
  Taro's <- %0004 -> 太郎 : 18
  your <- %0004 -> あなた : 16 25
  my <- %0004 -> 私 : 14 15 21 29
  Hanako's <- %0004 -> 花子 : 23 30
  it <- %0005 -> それ : 25 31
  that <- %0005 -> あれ : 11 13 16 18 24
  this <- %0005 -> これ : 10 12 14 15 17
  nurse <- %0006 -> 看護婦 : 27 28
  teacher <- %0006 -> 先生 : 22 23 29
  girl <- %0006 -> 少女 : 6 9 20
  boy <- %0006 -> 少年 : 5 7 8 19
  mother <- %0006 -> 母親 : 30
  friend <- %0006 -> 友達 : 21
  he <- %0007 -> 彼 : 19 21 29
  she <- %0007 -> 彼女 : 20 22 23 27 30
  Hanako <- %0008 -> 花子 : 4
  Takuma <- %0008 -> 琢磨 : 1 3
  Taro <- %0008 -> 太郎 : 2 26
  a %0006 <- %0008 -> %0006 : 5 6 7 19 20 22 27 28
  a %0009 %0006 <- %0008 -> %0009 %0006 : 8 9
  %0004 %0006 <- %0008 -> %0004 の %0006 : 21 23 29 30
  tall <- %0009 -> 背 の 高い : 8
  small <- %0009 -> 小さな : 9
  is <- %0010 -> だ : 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27
  isn't <- %0010 -> で は ない : 29 30 31
  you aren't <- %0011 -> あなた : 28
  I'm not <- %0011 -> 私 : 26
  I am <- %0012 -> 私 : 1 2 5 8
  you are <- %0012 -> あなた : 3 4 6 7 9
```

Figure 2.8: Translation Grammar Obtained by Batch Learning Process

```
%0005 %0010 a %0003 .  <- %S -> %0005 は %0003 %0010 。 : 10 11 17 24 31
%0007 %0010 %0008 .  <- %S -> %0007 は %0008 %0010 。 : 19 20 21 22 23 27 29 30
%0005 %0010 an %0003 .  <- %S -> %0005 は %0003 %0010 。 : 12 13
%0005 %0010 %0004 %0003 .  <- %S -> %0005 は %0004 の %0003 %0010 。 : 14 15 16
                                                                    18 25
%0011 %0008 .  <- %S -> %0011 は %0008 で は ない 。 : 26 28
%0012 %0008 .  <- %S -> %0012 は %0008 だ 。 : 1 2 3 4 5 6 7 8 9
```

Figure 2.9: Translation Grammar Obtained by Batch Learning Process (cont.)

examples may appear. If the system finds a new translation example, it asks the user whether the translation is correct or not. The underlined characters are the user's inputs. In this experiment, the system found two unknown examples,

```
32 : [this is an book .] ↔ [これ は 本 だ 。]
33 : [this is a apple .] ↔ [これ は りんご だ。]
```

The user taught the system that these examples are incorrect. The system treated these as new negative examples, and invoked the incremental learning process to deal with them. Though the grammar in Figure 2.8–2.9 accepts the following negative examples,

```
[this is an dog .] ↔ [これ は 犬 だ 。]
[this is a orange .] ↔ [これ は オレンジ だ 。]
```

they are excluded by the incremental learning process for negative examples 32 and 33.

(4) The user gave some new sentences to the system in interactive mode (Figure 2.12). First, he gave

```
[I am a girl .]
```

This is an unknown sentence for the system, but the system output a correct translation. Second, he gave

```
[she is a pretty girl .]
```

The system could not translate it into Japanese, so he taught it a correct translation. The system then invoked the incremental learning process to satisfy the positive example. Finally, the rule

```
Start to check all translations.
Instance ID : 1
Translating positive instance 1 [ I am Takuma . ] -> ...
  English : [ I am Takuma . ] -> Japanese : [ 私 は 琢磨 だ 。 ]: correct.
Translating positive instance 1 [ 私 は 琢磨 だ 。 ] -> ...
  Japanese : [ 私 は 琢磨 だ 。 ] -> English : [ I am Takuma . ]: correct.
.
Instance ID : 10
Translating positive instance 10 [ this is a book . ] -> ...
  English : [ this is a book . ] -> Japanese : [ これ は 本 だ 。 ]: correct.
Translating positive instance 10 [ これ は 本 だ 。 ] -> ...
  Japanese : [ これ は 本 だ 。 ] -> English : [ this is a book . ]: correct.
  Japanese : [ これ は 本 だ 。 ] -> English : [ this is an book .]: correct?
(Yes or No) No
Specialization by expanding nonterminal %0003 ...
  Expanding rule [%0005 %0010 an %0003 .  <- %S -> %0005 は %0003 %0010 。 :
                 12 13 ] ...
    Rule [%0005 %0010 an %0003 .  <- %S -> %0005 は %0003 %0010 。 : 12 13 ]
         is deleted.
    Rule [%0005 %0010 an orange .  <- %S -> %0005 は オレンジ %0010 。 : 13 ]
         is added.
    Rule [%0005 %0010 an apple .  <- %S -> %0005 は りんご %0010 。 : 12 ]
         is added.
    Rule [orange  <- %0003 -> オレンジ : ]
         is deleted because it has no positive instances.
[Batch learning loop : 1 ] ...
Integration.
  [%0005 %0010 an %0013 .  <- %S -> %0005 は %0013 %0010 。 : 12 13 ]
  + [apple  <- %0013 -> りんご : 12 ] + [orange  <- %0013 -> オレンジ : 13 ]
  <= [%0005 %0010 an apple .  <- %S -> %0005 は りんご %0010 。 : 12 ]
   + [%0005 %0010 an orange .  <- %S -> %0005 は オレンジ %0010 。 : 13 ]
Checking the following generalization.
  [%0013  <- %0003 -> %0013 : ]
  <= [apple  <- %0003 -> りんご : 14 ] ... OK.
Checking the following generalization.
  [%0005 %0010 an %0003 .  <- %S -> %0005 は %0003 %0010 。 : ]
  <= [%0005 %0010 an %0013 .  <- %S -> %0005 は %0013 %0010 。 : 12 13 ] ... Fail.
   Negative instance [ this is an book . <-> これ は 本 だ 。 ] is coverd.
```

Figure 2.10: Translation of All Sentences and Incremental Learning Process

```
[Batch learning loop : 2 ] ...

.

[Batch learning loop : 6 ] ... Terminate.

.

Instance ID : 12
Translating positive instance 12 [ this is an apple . ] -> ...
  English : [ this is an apple . ] -> Japanese : [ これ は りんご だ 。 ]: correct.
Translating positive instance 12 [ これ は りんご だ 。 ] -> ...
  Japanese : [ これ は りんご だ 。 ] ->
    English : [ this is a apple . ]: correct? (Yes or No) No
  Japanese : [ これ は りんご だ 。 ] ->
    English : [ this is an apple . ]: correct.
Specialization by expanding nonterminal %0013 ...
  Expanding rule [%0013  <- %0003 -> %0013 : 14 17 18 24 26 31 ] ...
    Rule [%0013  <- %0003 -> %0013 : 14 17 18 24 25 31 ] is deleted.
    Rule [tennis ball  <- %0003 -> テニスボール : 24 25 31 ] is added.
    Rule [dog  <- %0003 -> 犬 : 17 18 ] is added.
    Rule [apple  <- %0003 -> りんご : 14 ] is added.
    Rule [tennis ball  <- %0013 -> テニスボール : ]
         is deleted because it has no positive instances.
    Rule [dog  <- %0013 -> 犬 : ] is deleted because it has no positive instances.
Specialization by expanding nonterminal %0003 ...
  Expanding rule [a %0003  <- %0014 -> %0003 : 10 11 17 24 31 ] ...
    Rule [a %0003  <- %0014 -> %0003 : 10 11 17 24 31 ] is deleted.
    Rule [a dog  <- %0014 -> 犬 : 17 ] is added.
    Rule [a tennis ball  <- %0014 -> テニスボール : 24 31 ] is added.
    Rule [a book  <- %0014 -> 本 : 10 11 ] is added.
[Batch learning loop : 1 ] ... Terminate.

.

Instance ID : 31
Translating positive instance 31 [ it isn't a tennis ball . ] -> ...
  English : [ it isn't a tennis ball . ] ->
    Japanese : [ それ は テニスボール で は ない 。 ]: correct.
Translating positive instance 31 [ それ は テニスボール で は ない 。 ] -> ...
  Japanese : [ それ は テニスボール で は ない 。 ] ->
    English : [ it isn't a tennis ball . ]: correct.
Checking all translations is complete.
```

Figure 2.11: Translation of All Sentences and Incremental Learning Process (cont.)

```
[Interactive learning loop : 1]
Please input an English sentence: (I am a girl .)
  English : [ I am a girl . ] ->
    Japanese : [ 私 は 少女 だ 。 ]: correct? (Yes or No) Yes
[Interactive learning loop : 2]
Please input an English sentence: (she is a pretty girl .)
I can't translate [ she is a pretty girl .] into Japanese.
Please translate [ she is a pretty girl . ] into Japanese:
      (彼女 は かわいい 少女 だ 。)
[ she is a pretty girl .] <-> [ 彼女 は かわいい 少女 だ 。] is OK? Yes
Checking the addition of the following rule.
  [%0007 %0010 a pretty %0006 .  <- %S -> %0007 は かわいい %0006 %0010 。 : ]
   ... OK.
    Rule [ %0007 %0010 a pretty %0006 .  <- %S -> %0007 は かわいい %0006 %0010 。
          : 35 ] is added.
[Batch learning loop : 1 ] ...
Checking the following change.
  [%0007 %0010 %0008 .  <- %S -> %0007 は %0008 %0010 。
   : 19 20 21 22 23 27 29 30 ] (already exist)
  + [a pretty %0006  <- %0008 -> かわいい %0006 : ]
  <= [%0007 %0010 a pretty %0006 .  <- %S -> %0007 は かわいい %0006 %0010 。
      : 35 ] ...OK.
[Batch learning loop : 2 ] ...
Checking the following change.
  [a %0009 %0006  <- %0008 -> %0009 %0006 : 8 9 ] (already exist)
  + [pretty  <- %0009 -> かわいい : ]
  <= [a pretty %0006  <- %0008 -> かわいい %0006 : 35 ] ...OK.
[Batch learning loop : 3 ] ... Terminate.
[Interactive learning loop : 3] ... Terminate.
```

Figure 2.12: Learning in Interactive Mode

Table 2.1: Summary of Experiments

Figures enclosed in parentheses are the numbers of negative examples which were used to restrain generalization of grammars.

ID			I31	I122	I530
Positive Examples	Number of Positive Examples		33	74	208
	Number of Terminals	English	39	66	159
		Japanese	31	48	145
	Average Length of Sentences	English	5.0	4.9	5.3
		Japanese	6.0	6.3	6.5
Negative Examples	Number of Negative Examples		2(2)	50(15)	345(61)
Translation Grammar	Number of Nonterminals		13	25	73
	Number of Rules		45	92	270
	Average Length of RHS	English	1.6	1.6	2.0
		Japanese	1.6	1.8	2.2

```
pretty ← %0009 → かわいい
```

was added.

(5) Figure 2.13-2.14 shows the final translation grammar of the experiment.

2.5.2 Results

Table 2.1 shows the summary of the experiments.

The experiments show:

- A translation grammar can be learned from translation examples by using simple grammatical inference techniques. The translation grammars obtained satisfy all given positive and negative examples and can produce some translations which are not explicitly given.
- The system can find correspondences of words, word classes and phrase structures between two languages.
- The proposed framework has several limitations; the grammar of each language must have a structure which can be described easily in a context free

```
Name : I31
Language-a : English
Language-b : Japanese
Positive Instances : 33
Negative Instances : 2
Rules : 45
  apple  <- %0003 -> りんご : 14
  dog  <- %0003 -> 犬 : 17 18
  tennis ball  <- %0003 -> テニスボール : 24 25 31
  book  <- %0003 -> 本 : 15 16
  Taro's  <- %0004 -> 太郎 : 18
  your  <- %0004 -> あなた : 16 25
  my  <- %0004 -> 私 : 14 15 21 29
  Hanako's  <- %0004 -> 花子 : 23 30
  it  <- %0005 -> それ : 25 31
  that  <- %0005 -> あれ : 11 13 16 18 24
  this  <- %0005 -> これ : 10 12 14 15 17
  nurse  <- %0006 -> 看護婦 : 27 28
  teacher  <- %0006 -> 先生 : 22 23 29
  girl  <- %0006 -> 少女 : 6 9 20 34 35
  boy  <- %0006 -> 少年 : 5 7 8 19
  mother  <- %0006 -> 母親 : 30
  friend  <- %0006 -> 友達 : 21
  he  <- %0007 -> 彼 : 19 21 29
  she  <- %0007 -> 彼女 : 20 22 23 27 30 35
  Hanako  <- %0008 -> 花子 : 4
  Takuma  <- %0008 -> 琢磨 : 1 3
  Taro  <- %0008 -> 太郎 : 2 26
  a %0006  <- %0008 -> %0006 : 5 6 7 19 20 22 27 28 34
  a %0009 %0006  <- %0008 -> %0009 %0006 : 8 9 35
  %0004 %0006  <- %0008 -> %0004 の %0006 : 21 23 29 30
  tall  <- %0009 -> 背 の 高い : 8
  small  <- %0009 -> 小さな : 9
  pretty  <- %0009 -> かわいい : 35
  is  <- %0010 -> だ : 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27 35
  isn't  <- %0010 -> で は ない : 29 30 31
```

Figure 2.13: Final Translation Grammar

```
you aren't  <- %0011 -> あなた : 28
I'm not  <- %0011 -> 私 : 26
I am  <- %0012 -> 私 : 1 2 5 8 34
you are  <- %0012 -> あなた : 3 4 6 7 9
orange  <- %0013 -> オレンジ : 13
apple  <- %0013 -> りんご : 12
an %0013  <- %0014 -> %0013 : 12 13
%0004 %0003  <- %0014 -> %0004 の %0003 : 14 15 16 18 25
a dog  <- %0014 -> 犬 : 17
a tennis ball  <- %0014 -> テニスボール : 24 31
a book  <- %0014 -> 本 : 10 11
%0007 %0010 %0008 .  <- %S -> %0007 は %0008 %0010 。 : 19 20 21 22 23 27
                                                        29 30 35
%0011 %0008 .  <- %S -> %0011 は %0008 で は ない 。 : 26 28
%0012 %0008 .  <- %S -> %0012 は %0008 だ 。 : 1 2 3 4 5 6 7 8 9 34
%0005 %0010 %0014 .  <- %S -> %0005 は %0014 %0010 。 : 10 11 12 13 14 15
                                                        16 17 18 24 25 31
```

Figure 2.14: Final Translation Grammar (cont.)

grammar, and two languages must have *similar* sentence structures. These limitations come from the use of translation grammar as the basic framework of representing knowledge.

- A translation grammar is not powerful enough to describe precise knowledge for natural language translation. Many nonterminal symbols and rules are needed to translate complex sentences. The mechanism needs to be extended to use feature bundles instead of nonterminal symbols.

- Many negative examples are needed, because generalization is restrained only by explicitly given negative examples. Restraint of generalization by implicit negative examples (namely generalization of negative examples) is needed in the future.

2.6 Summary

This chapter has described a procedure of learning language translation. Learning language translation is categorized as *learning to perform a multiple-step task*, which is the most difficult class of machine learning. We have showed that the formalism of translation grammar and its learning algorithm make it possible to learn language translation. The proposed method of learning is automatic. Major results of this chapter are:

- A translation grammar can represent knowledge for bidirectional translation in a uniform style. This formalism has some desirable characteristics for learning. They simplifies the learning process.

- A translation grammar can be learned from translation examples by using simple grammatical inference techniques. Obtained translation grammars satisfy all given positive and negative examples and can predict some translations which are not explicitly given.

- Two learning processes, batch learning process and incremental learning process, are complementary to each other.

- Experiments shows that the learning system can find correspondences of words, word classes and phrase structures between two languages.

There are many open problems to be solved before this method can be used for real application. They include:

- Extension of knowledge representation.
  Nonterminal symbols in the current framework should be replaced with bundles of features.
- Extension of learning engine.
  A mechanism for automatically generating a set of features is needed.
- User interface.
  A user interface through which the human and the machine can cooperate to construct machine translation systems is needed.

These problems are very interesting and worth addressing.

Chapter 3

Example-Based Word Selection

3.1 Introduction

In the previous chapter, the author proposed a method for learning translation rules. It is one direction for research towards overcoming the knowledge acquisition bottleneck. Another direction is to develop a translation mechanism which does not need rule acquisition, namely *example-based translation*.

The original idea of example-based translation was suggested by Nagao as *translation by analogy* [Nagao 84]. The basic idea is very simple: translate a source sentence by imitating a translation example of a similar sentence. If this can be implemented, it frees us from rule acquisition. All we need to do is to collect translation examples.

The main part of translation is the process that transfers or rewrites a fragment in source language into a corresponding fragment in target language. This process heavily depends on individual words and individual contexts, not on general principles or regularity. This characteristic suggests that example-based reasoning is suited for the translation task.

Moreover, recent progress in computer hardware makes it possible for us to use large size memories and massively parallel computing. These support practical example-based reasoning systems, as demonstrated by the *memory-based* *reasoning* of [Stanfill & Waltz 86].

In this chapter, we discuss two explorations of example-based translation — translation by analogy and memory-based reasoning — and propose a simplified version of example-based translation, called MBT1, which can solve the word selection problem.

3.2 Translation by Analogy

Nagao first suggested the basic idea of example-based translation in [Nagao 84]. Nagao's idea can be divided into two parts:

1. grouping word pairs and learning case frames
2. translating by using the analogy principle

In this section, we discuss these ideas and modify them.

3.2.1 Grouping Word Pairs and Learning Case Frames

The basic heuristics for grouping word pairs and learning case frames is *learning from near miss* [Winston 77]. Consider the following two translation examples. [1]

(3–1)	He eats vegetables.	彼は 野菜を 食べる。
(3–2)	He eats potatoes.	彼は じゃがいもを 食べる。

It is a plausible inference that we can extract the following corresponding relations by comparing (3–1) and (3–2).

(3–3)	He eats Y.	↔	彼は Y を 食べる。
(3–4)	vegetables	↔	野菜
(3–5)	potatoes	↔	じゃがいも

Moreover, let's consider another translation example.

[1] Although Japanese has no delimiter for word separation, we use a space as a delimiter for 'bunsetsu' separation. A 'bunsetsu' consists of a content word and some function words.

(3–6) She eats vegetables. 彼女は 野菜を 食べる。

From (3-3) and (3-6), we will extract the following corresponding relations.

(3–7)	*X* eats *Y*.	↔	*X* は *Y* を 食べる。
(3–8)	he	↔	彼
(3–9)	she	↔	彼女

This story indicates that we can formulate:

1. Correspondence between English and Japanese sentence frames. If we carefully choose a set of similar examples for a verb, we can obtain case frames for the verb.
2. A bilingual dictionary between English and Japanese.
3. A set of noun groups distinguished by the contexts in which they can appear. If this process is done for different kinds of verbs, the noun grouping will become finer, and more reliable.

This story is too naive to implement straightforwardly in a practical software system. If we use surface string matching mechanism for comparing two examples, some trivial problems prevent success. Consider the following translation example.

(3–10) I eat potatoes. 私は じゃがいもを 食べる。

From (3-2) and (3-10), the system cannot extract the corresponding relations 'he ↔ 彼' and 'I ↔ 私', because the system does not know any relation between 'eats' and 'eat'. Moreover, because Japanese has no delimiter for word separation, the system will often extract non-word strings.

To avoid these problems, we assume morphological and syntactic analysis. We give the system examples in the form of pairs of verb frame instances: e.g.

(3–11) (eat he vegetable) (食べる 彼 野菜)

This differs from Nagao's proposal for translation by analogy using non-preprocessed examples, and we cannot give the system new examples fully automatically. But the author thinks it is a practical answer.

3.2.2 Translation by Analogy

Nagao explains translation by analogy as follows. Let's assume that the system knows translation example (3-2). The system also has a word dictionary between English and Japanese, and a thesaurus. When the following sentence is given to the system to translate,

(3-12) John eats apples.

the system checks similarity or replacability between 'John' and 'he', and 'apples' and 'potatoes' by tracing the synonym and superordinate/subordinate concept relations in the thesaurus. Because these are similar word pairs, the system determines that the translation example (3-2) can be used for the translation of (3-12). As a result, the system outputs the following good translation.

(3-13) ジョンは りんごを 食べる。

Let's consider another input (3-14).

(3-14) Acid eats metal.

In this case, the similarity check of 'acid ~ he', and 'metal ~ vegetable' fails in the thesaurus, and no translation is produced. If this is an example sentence in the entry of 'eat', and has the Japanese translation (3-15), then the input sentence (3-16) can be translatable as (3-17).

(3-15) 酸は 金属を 侵す。

(3-16) Sulfuric acid eats iron.

(3-17) 硫酸は 鉄を 侵す。

This explanation has two problems. The first comes from the concept 'replacablity'. Let's assume that the system knows the translation example (3-18).

(3-18) This is soup. これは スープ です。

According to Nagao's explanation, it is impossible to translate (3-19) into (3-20) using (3-18).

(3-19) This is iron.

(3-20) これは 鉄 です。

Obviously replacability depends on its context. Can we make a system that can translate (3-19) into (3-20) using (3-18) and does not output (3-21) as the translation of (3-16)?

(3-21) 硫酸は 鉄を 食べる。

The answer is that it is almost impossible. The reasons are:

- If the system uses context independent measure of word similarity or replacability, it is impossible because the system cannot obtain any context dependent information from the thesaurus and examples.
- It is almost impossible to obtain context dependent measure of word similarity or replacability, because there are a huge number of context variations. Even if we can collect a huge number of translation examples, it will not be enough to obtain context dependent word similarity.

A practical solution is:

- If the system does not know the translation example (3-14) - (3-15), the system outputs (3-21) as a translation candidate of (3-16).
- If the system knows the translation example (3-14) - (3-15), the system outputs two translation candidates (3-17) and (3-21) for (3-16), and prefers (3-17) rather than (3-21).

In short, the system outputs some translation candidates and their preference scores. It is implementable in a software system.

Another problem with Nagao's explanation is that it cannot select preferable targets for nouns. Let's assume that the system knows the translation examples (3-2) and (3-14)-(3-15), and the sentence (3-22) is given to be translated.

(3-22) She eats vegetables.

If the system knows two translation candidates, '野菜' and '植物人間', for 'vegetable', then it produces four translation candidates for (3-22):

(3-23) 彼女は 野菜を 食べる。

(3-24) 彼女は 植物人間を 食べる。

(3-25) 彼女は 野菜を 侵す。

(3-26) 彼女は 植物人間を 侵す。

The system can prefer (3-23) and (3-24) over (3-25) and (3-26), because 'vegetable' is more similar to 'potato' than 'iron'. But the system cannot prefer (3-23) over (3-24), because similarity is calculated on only the source (English) side in Nagao's proposal. This problem can be solved by calculating similarity on both side, i.e. using the similarity of word pairs. If 'potato - じゃがいも' is more similar to 'vegetable - 野菜' than 'vegetable - 植物人間', the system can prefer (3-23) over (3-24).

3.2.3 Summary of Modification

As a result of the discussion, we have modified Nagao's idea as follows:

1. We give the system examples in the form of pairs of verb frame instances.

2. The system outputs some translation candidates and their preference scores.

3.3 From Memory-Based Reasoning to Translation

We cannot simply apply Memory-based reasoning (MBR) [Stanfill & Waltz 86] to translation, because translation task is not so simple. In this section, we discuss the memory-based reasoning framework, and will modify it to suit the word selection task.

	Predictor fields				Goal fields			
	f_1	f_2	$\dots$	f_n	g_1	g_2	$\dots$	g_m
ρ_1								
ρ_2								
$\cdot$								

Figure 3.1: Database for MBR

3.3.1 Memory-Based Reasoning

Conceptually, memory-based reasoning consists of three components:

1. Database
2. Metric
3. Evidence-combining rule

Database

A database is a set of records. Each record has a fixed set of fields. The field containing the "answer" to a problem is called the *goal field*, and the other fields are *predictor fields*. Figure 3.1 shows the form of MBR database. Novel records which are to be classified are *target records*. The reasoning task is to infer a value of the goal fields of the target records.

Notation

We use Greek letters (τ, ρ) for records, and italics for field names (f, g). Field f of a record ρ is written $\rho.f$. The set of possible values for a field f is written V_f. A value is represented by an italic letter v. A *database* is written D. The set of goal field is written G_τ, and the set of predictor fields is written P_τ.

A *feature* is a combination of a field and a value, such as $[f = v]$. We use features to restrict a database to a subset, as in $D[f = v]$. We can count the number of items in the full database, as $|D|$, or in a restricted database, as in $|D[f = v]|$

Value Difference Metric

Stanfill and Waltz [Stanfill & Waltz 86] have tested three metrics, the *overlap metric*, the *weighted feature metric*, and the *value difference metric*, and they have concluded that the value difference metric is the best for the task of pronouncing English words. Suppose we are given a target record τ, a record ρ, a goal filed g, and a database D. The value difference metric is:

$$\Delta^g(D,\tau,\rho) = \sum_{F \in P_\rho} \delta^g_f(D,\tau.f,\rho.f) \tag{3.1}$$

$$\delta^g_f(D,\tau.f,\rho.f) = d^g_f(D,\tau.f,\rho.f)\, w^g_f(D,\tau.f) \tag{3.2}$$

$$w^g_f(D,\tau.f) = \sqrt{\sum_{v \in V_g} \left(\frac{|D[f=\tau.f][g=v]|}{D[f=\tau.f]|}\right)^2} \tag{3.3}$$

$$d^g_f(D,\tau.f,\rho.f) = \sum_{v \in V_g} \left(\frac{|D[f=\tau.f][g=v]|}{D[f=\tau.f]|} - \frac{|D[f=\rho.f][g=v]|}{D[f=\rho.f]|}\right)^2 \tag{3.4}$$

This metric is very sensitive to *context* : Formula 3.3 indicates that weight of a field $w^g_f(D,\tau.f)$ depends on an individual value $\tau.f$ and a goal field g, and Formula 3.4 indicates that distance between two value $d^g_f(D,\tau.f,\rho.f)$ depends on the field f and the goal field g. This context sensitive metric shows good performance when the size of the database is very large.

Evidence-Combining Rule

In the reasoning process, MBR first computes distance values between the target record and individual records in the database. Second, MBR selects the top ten records in similarity ranking. Finally MBR sums evidence scores for each candidate value for the goal field, and outputs them. The evidence score of a record is $\frac{1}{\Delta+1}$.

3.3.2 Toward Memory-Based Translation

Before discussing a modification of MBR suited for translation task, we define the form of an example and an input informally. Suppose we are given an example in the following form.

	Source	Target
Head	eat	食べる
Slot$_1$	he	彼
Slot$_2$	potato	じゃがいも

And suppose we are given a input in the following form.

	Source	Target
Head	eat	?
Slot$_1$	John	?
Slot$_2$	apple	?

The translation task is to fill the empty fields (indicated with '?') in the table.

Now we start to discuss the application of MBR to translation. The simplest example is the following.

	f_1	f_2	f_3	g_1	g_2	g_3
p_1	eat	he	potato	食べる	彼	じゃがいも

We soon find two problems with this method:

- The number of fields is not fixed. For example, some verbs have three argument slots.
- The similarity or distance calculation is computed on only source side.

These leads us to make the following modifications.

- We divide the database into some subdatabases which have fixed number of fields. Moreover, because it is very difficult to obtain the similarity between two verbs mechanically [Nagao 84], we have to make a subdatabase for each verb frame pair.
- We have to use word pairs as values in the fields.

As a result, a subdatabase is in the following form.

Subdatabase for (eat 食べる 2)

f_1	f_2
(he 彼)	(potato じゃがいも)

In the subdatabase, there are no target fields. But it is not serious, because: if the system has a dictionary of word pairs and verb frame pairs, the system can produce some translation candidates from the given input. And the system can compute the preference score for each candidate using a method like MBR, and select the best one.

We now have to define metric for the similarity between a translation candidate and an example in a database. We cannot use the value difference metric, because:

- There are no target fields.
- In almost all cases, the value $|D[f = r.f]|$ will be zero.

We have to use a simpler and more context independent metric.

First we consider the distance between values of fields: i.e. the distance between two word pairs. A promising hint is in Nagao's paper. It is the use of external similarity: if two word pairs appear in similar contexts, they will be considered similar. To quantify this, first we count the number of appearances of each word pair in each context. In our situation, a context is a slot of a verb frame pair. As a result, we obtain the following matrix.

	(eat 食べる 2)		(eat 侵す 2)		
Word Pair	Slot_1	Slot_2	Slot_1	Slot_2	...
(he 彼)	2	0	0	0	...
(potato じゃがいも)	0	1	0	0	...
(acid 酸)	0	0	1	0	
			.	.	

Using the matrix, we define the distance between word pairs $r.f$ and $\rho.f$ as the following: [2]

$$d(D, r.f, \rho.f) = \frac{1}{similarity(D, r.f, \rho.f)} - 1 \qquad (3.5)$$

$$similarity(D, r.f, \rho.f) = \frac{L_{r.f}\, {}^{t}L_{\rho.f}}{\|L_{r.f}\|\,\|L_{\rho.f}\|} \qquad (3.6)$$

where $L_{r.f}$ means a row vector for a word pair $r.f$ in the matrix. Formula 3.5 is one of the transformations from similarity to distance, which is developed by Maruyama and Watanabe [Maruyama & Watanabe 87]. Formula 3.6 is one of the standard definitions of similarity between two vectors [Nagao 83]. We employ Formula 3.6 as a measure of the similarity because two independent works, [Maruyama & Watanabe 87] and [Sato & Nagao 87] show that it is suited for calculating similarity between words.

Next, we consider the weights of fields. When the number of fields is one, the weight has no role. When there is only one candidate target verb, the weights are not important. When there are some candidates of target verbs, the weights are important. We compute the weights by judging how strongly individual fields affect the selection of target verb.

We adopt the method used in ID3 for feature selection [Quinlan 83] for our purpose. Suppose we are given the input (eat man vegetable) to translate. And suppose the system has two verb frames (eat 食べる 2) and (eat 侵す 2). In this case, the system merges the two subdatabases for (eat 食べる 2) and (eat 侵す 2), and creates the following database.

[2]Strictly, this is a pseudo-distance, because it does not satisfy

$$d(a, b) + d(b, c) \geq d(a, c)$$

when $(d(a, b) \leq \infty) \wedge (d(b, c) \leq \infty) \wedge (d(a, c) = \infty)$. This is a special case, and not important for our purpose.

Database for (eat * 2)

f_1	f_2	Target Head(g)
(he 彼)	(potato じゃがいも)	食べる
.	.	.
(acid 酸)	(metal 金属)	侵す
.	.	.

We call it 'the database for (eat * 2)' and abbreviate it a X. Using this database X, we define the weight of field f_i as follows.

$$w_{f_i}(D, \mathtt{eat}, 2) = \frac{G(X, f_i)}{\sum_{f_j \in P} G(X, f)} \tag{3.7}$$

$$G(X, f) = I(X) - J(X, f) \tag{3.8}$$

$$I(X) = -\sum_{v \in V_g} (q_v \log_2 q_v) \quad \text{where } q_v = \frac{|X[g = v]|}{|X|} \tag{3.9}$$

$$J(X, f) = \sum_{v \in V_f} (r_f\, I(X[f = v])) \quad \text{where } r_h = \frac{|X[f = v]|}{|X|} \tag{3.10}$$

If we know that a value of field f is the value v, we can gain some information about the selection of the target verb. $G(X, f)$ means the expected information gain. These formulas means that the weights of fields are computed by judging how much information gain individual fields carry.

In this section, we have informally discussed the modifications of MBR for the word selection task. The next section describes it formally as MBT1.

3.4 MBT1

In this section, we define MBT1 formally. MBT1 consists of three components:

1. Translation database
2. Metric
3. Translation process

3.4.1 Translation Database

Translation Example

A translation example is given in the following form.

	Source	Target
Head	eat	食べる
Slot_1	he	彼
Slot_2	potato	じゃがいも

A translation example consists of:

1. A *Head.* A head is a *translation frame.*
   A translation frame consists of:
   (a) A *source head.*
   (b) A *target head.*
   (c) *Arity*, the number of slots that a translation frame has.
2. Some *arguments* of the translation frame. An argument is a *word pair.* A word pair consists of a source word and a target word.

We use new notation in this section. We use Greek letters (ϵ, τ) for translation examples or translation candidates. Translation candidates are in the same form as translation examples. We use italics for subcomponents of a translation example or a translation candidate: h for a head, and s_i for a argument in slot_i. We also use italics f for a translation frame, and p for a word pair. We use superscripts (^{s}, ^{t}) for source or target sides of a translation frame or word pair, and ^{a} for arity of a translation frame. Using the notation, we introduce the formal definition of a translation example.

$$\epsilon = < f, p_1, p_2, \ldots, p_{f^a} > \tag{3.11}$$

$$f = < f^s, f^t, f^a > \tag{3.12}$$

$$p = < p^s, p^t > \tag{3.13}$$

The above translation example is written as the following formally.

$$\epsilon = << \text{eat}, \text{食べる}, 2 >, < \text{he}, \text{彼} >, < \text{potato}, \text{じゃがいも} >> \tag{3.14}$$

Subcomponents of the example are written with the following 'path' representations.

$$\begin{aligned}
\epsilon.h &= <\mathtt{eat},\text{食べる},2> && (3.15)\\
\epsilon.h^s &= \mathtt{eat} && (3.16)\\
\epsilon.h^t &= \text{食べる} && (3.17)\\
\epsilon.h^a &= 2 && (3.18)\\
\epsilon.s_1 &= <\mathtt{he},\text{彼}> && (3.19)\\
\epsilon.s_1^s &= \mathtt{he} && (3.20)\\
\epsilon.s_1^t &= \text{彼} && (3.21)
\end{aligned}$$

Translation Database

We use E as a set of translation examples. We use F as a set of translation frames which appeared in E, and P as a set of word pairs which appeared in E. A translation database D is 3-tuple of E, F and P.

$$\begin{aligned}
E &= \{e_i\}_{1\le i\le N_E} && (3.22)\\
F &= F(E) = \{f \mid \exists e \in E,\ e.h = f\} && (3.23)\\
P &= P(F) = \{p \mid \exists e \in E,\ e.s_k = p\\
&\qquad \text{where } 1 \le k \le e.h^a\} && (3.24)\\
D &= < E, F, P > && (3.25)
\end{aligned}$$

A *feature* is a combination of a subcomponent name and a value, such as $[h = f]$ or $[s_i = p]$. We use features to restrict the set of translation examples, as in $E[h = f]$. We can count the number of items in a set of examples, such as $|E|$ or $|E[h = f]|$. We use the same notation for a set of translation frames or a set of word pairs, such as $|F[s = x]|$ or $|P[t = y]|$.

3.4.2 Metric

The distance between a translation candidate τ and a translation example ε is defined as follows:

$$\Delta(D,\tau,\varepsilon) = \begin{cases} \sum_{k=1}^{\tau.h^{a}} \delta_k(D,\tau.h,\tau.s_k,\varepsilon.s_k) & \text{if } \tau.h = \varepsilon.h \\ \infty & \text{otherwise} \end{cases} \tag{3.26}$$

$$\delta_k(D,f,p_i,p_j) = d(D,p_i,p_j)\, w_k(D,f^{s},f^{a}) \tag{3.27}$$

Distance between word pairs

In order to define the distance between word pairs, we introduce the following vector.

$$V_i = (v_{i,11}, v_{i,12}, \ldots, v_{i,1f_1^{a}}, v_{i,21}, \ldots, v_{i,nf_n^{a}}) \tag{3.28}$$

$$v_{i,jk} = |E[h = f_j][s_k = p_i]| \tag{3.29}$$

$E[h = f_j][s_k = p_i]$ is a subset of E: a set of translation example which has f_j in the head and p_i in slot$_k$. So, $v_{i,jk}$ is the number of appearance of p_i in slot$_k$ of translation frame f_j. Using these vectors, the distance between word pairs is defined as the following.

$$d(D,p_i,p_j) = \frac{1}{similarity(D,p_i,p_j)} - 1 \tag{3.30}$$

$$similarity(D,p_i,p_j) = \frac{V_i\ {}^tV_j}{\|V_i\|\,\|V_j\|} \tag{3.31}$$

Weight of Slot

The weight of a slot is defined as the following.

$$w_k(D,f^{s},f^{a}) = \begin{cases} 1 & \text{if } f^{a} = 1 \\ \frac{1}{f^{a}} & \text{if } |F[s = f^{s}][a = f^{a}]| = 1 \\ \frac{G(X,k)}{\sum_{j=1}^{f^{a}} G(X,j)} & \text{otherwise} \\ & \text{where } X = E[h^{s} = f^{s}][h^{a} = f^{a}] \end{cases} \tag{3.32}$$

$$G(X,k) = I(X) - J(X,k) \tag{3.33}$$

$$I(X) = -\sum_{f\in F(X)} (q_f \log_2 q_f) \quad \text{where } q_f = \frac{|X[h=f]|}{|X|} \tag{3.34}$$

$$J(X,k) = \sum_{p\in P(X)} (r_p\, I(X[s_k = p])) \quad \text{where } r_p = \frac{|X[s_k = p]|}{|X|} \tag{3.35}$$

3.4.3 The Translation Process

The translation process consists of two steps: generation of translation candidates and calculation of their preference scores.

An example of an input for translation is the following:

	Source	Target
Head	eat	?
Slot_1	John	?
Slot_2	apple	?

Formally, an input for translation is written as

$$< f^s, p_1^s, \ldots, p_{j^s}^s >$$

For example, above input is written:

$$< \texttt{eat}, \texttt{John}, \texttt{apple} >$$

The procedure to generate candidates for a given input $< f^s, p_1^s, \ldots, p_{j^s}^s >$ is:

1. Find a set of translation frames $F[s = f^s][a = f^a]$.
2. For each slot, find a set of word pair $P[s = p_i^s]$.
3. Make a set of translation candidates C by combining 1. and 2.

Formally, a set of translation candidates C is represented as the following.

$$C = \bigcup_{f\in F[s=f^s][a=f^a]} \; \bigcup_{q_1\in P[s=p_1^s]} \cdots \bigcup_{q_{j^s}\in P[s=p_{j^s}^s]} \{< f, q_1, \ldots, q_{j^s} >\} \tag{3.36}$$

For each candidate, the system retrieves the nearest (most similar) translation examples. The score of a candidate $\tau \in C$ is defined as the following.

$$score(D,\tau) = \min_{\epsilon \in E} \Delta(D,\tau,\epsilon) \tag{3.37}$$

In practice, the system does not need to calculate $\Delta(D,\tau,\epsilon)$ for all ϵ in E, because $\Delta(D,\tau,\epsilon)$ is infinity when the head of τ is not same as the head of ϵ (See Formula 3.26). The system uses the following formula.

$$score(D,\tau) = \min_{\epsilon \in E[h=\tau.h]} \Delta(D,\tau,\epsilon) \tag{3.38}$$

Finally, the system outputs all candidates ordered by the score. The translation candidate which has the smallest score is the best translation.

3.5 Experiments

3.5.1 MBT1 System

The author has implemented the mechanism described above as the system *MBT1*, written in Symbolics Common Lisp on Symbolics 3600 series computers.

It was very easy to implement MBT1's mechanism. Therefore the main effort for constructing an MBT1 system is collecting translation examples and making the translation database. It corresponds to writing rules or making dictionaries in the traditional framework, but it is much easier than writing rules. For experiments, the author made a small English-Japanese translation database, containing translation examples for basic English verbs. This database was made by the following process.

1. Extract non-processed examples from some English-Japanese dictionaries and other sources.

2. Transform them into pairs of verb frame instances.

For example, the non-processed example (3-27) is transformed into (3-28).

(3-27) He bought a new book. 彼は 新しい 本を 買った。

Number of translation examples	1403
Number of translation frames	364
Average arity per translation frame	1.88
Number of English verb frames	175
Number of Japanese verb frames	299
Number of word pairs	429
Number of English words	396
Number of Japanese words	408
Total count of word pairs	2680

Table 3.1: Translation Database

(3–28) (buy he book) (買う 彼 本)

As in the above example, some modifiers of nouns are omitted. When the arity of an English verb is not same as the arity of the corresponding Japanese verb, we use a special symbol '*dummy*', e.g.

(3–29) There is a book. 本が ある。

(3–30) (be there book) (ある *dummy* 本)

Figure 3.2 - 3.4 shows translation examples for the verbs 'be', 'play' and 'write' in the database. Table 3.1 shows the size of the database.

Typical Outputs

Figure 3.6 shows some typical translation outputs of MBT1 [3]. The first shows a case whose a noun has several target candidates. In this case, the noun 'paper' has three candidates. The second shows a case whose a verb has several target candidates. In this case, the verb 'be' has four candidates. The last shows a case whose both the verb and nouns have several target candidates. MBT1 can select the correct translation in these cases.

[3] In this experiment, we used the value 999 instead of infinity in Formula 3.26.

```
;;; (be ある 2)
((be there water) (ある *dummy* 水))
((be there air) (ある *dummy* 空気))
((be there doll) (ある *dummy* 人形))
((be there desk) (ある *dummy* 机))
((be there book) (ある *dummy* 本))
((be there apple) (ある *dummy* りんご))
((be there building) (ある *dummy* ビル))
((be there dish) (ある *dummy* 皿))
((be there plant) (ある *dummy* 植物))

;;; (be いる 2)
((be there american) (いる *dummy* アメリカ人))
((be there boy) (いる *dummy* 少年))
((be there girl) (いる *dummy* 少女))
((be there cat) (いる *dummy* ねこ))
((be there dog) (いる *dummy* 犬))
((be there actress) (いる *dummy* 女優))
((be there doctor) (いる *dummy* 医者))
((be there student) (いる *dummy* 生徒))
((be there teacher) (いる *dummy* 先生))
((be there japanese) (いる *dummy* 日本人))

;;; (be ここにある 2)
((be here orange) (ここにある *dummy* オレンジ))
((be here book) (ここにある *dummy* 本))
((be here apple) (ここにある *dummy* りんご))

;;; (be です 2)
((be he policeman) (です 彼 警察官))
((be he director) (です 彼 取締役))
((be you writer) (です あなた 作家))
((be you teacher) (です あなた 先生))
((be you takuma) (です あなた 琢磨))
((be you student) (です あなた 生徒))
((be you nurse) (です あなた 看護婦))
((be you japanese) (です あなた 日本人))
((be you hanako) (です あなた 花子))
((be you girl) (です あなた 少女))
((be you friend) (です あなた 友達))
```

Figure 3.2: Translation Examples (Part I)

```
((be we student) (です 私たち 生徒))
((be we nurse) (です 私たち 看護婦))
((be we friend) (です 私たち 友達))
((be those banana) (です あれら バナナ))
((be this room) (です ここ 部屋))
((be this piano) (です これ ピアノ))
((be this dog) (です これ 犬))
((be this car) (です これ 車))
((be this book) (です これ 本))
((be this apple) (です これ りんご))
((be they tulip) (です それら チューリップ))
((be they teacher) (です 彼ら 先生))
((be they student) (です 彼ら 生徒))
((be they dog) (です それら 犬))
((be they cat) (です それら ねこ))
((be they banana) (です それら バナナ))
((be these tulip) (です これら チューリップ))
((be that piano) (です あれ ピアノ))
((be that organ) (です あれ オルガン))
((be that dog) (です あれ 犬))
((be that car) (です あれ 車))
((be that book) (です あれ 本))
((be that ball) (です あれ ボール))
((be teacher japanese) (です 先生 日本人))
((be she teacher) (です 彼女 先生))
((be she nurse) (です 彼女 看護婦))
((be she mother) (です 彼女 母))
((be she girl) (です 彼女 少女))
((be i taro) (です 私 太郎))
((be i takuma) (です 私 琢磨))
((be i boy) (です 私 少年))
((be i american) (です 私 アメリカ人))
((be he teacher) (です 彼 先生))
((be he friend) (です 彼 友達))
((be he boy) (です 彼 少年))
((be friend american) (です 友達 アメリカ人))
((be horse animal) (です 馬 動物))
((be banana plant) (です バナナ 植物))
((be canary bird) (です カナリア 鳥))
((be banana fruit) (です バナナ くだもの))
((be fruit plant) (です くだもの 植物))
```

Figure 3.3: Translation Examples (Part II)

```
;;; (play する 2)
((play you tennis) (する あなた テニス))
((play you basketball) (する あなた バスケットボール))
((play we tennis) (する 私たち テニス))
((play we baseball) (する 私たち 野球))
((play tom tennis) (する トム テニス))
((play they tennis) (する 彼ら テニス))
((play they chess) (する 彼ら チェス))
((play they card) (する 彼ら トランプ))
((play taro tennis) (する 太郎 テニス))
((play she volleyball) (する 彼女 バレーボール))
((play she tennis) (する 彼女 テニス))
((play she baseball) (する 彼女 野球))
((play i tennis) (する 私 テニス))
((play i game) (する 私 試合))
((play i baseball) (する 私 野球))
((play bill baseball) (する ビル 野球))
((play they ball) (する 彼ら 野球))

;;; (play ひく 2)
((play you violin) (ひく あなた バイオリン))
((play you piano) (ひく あなた ピアノ))
((play you guitar) (ひく あなた ギター))
((play they piano) (ひく 彼ら ピアノ))
((play she piano) (ひく 彼女 ピアノ))
((play she flute) (ひく 彼女 フルート))
((play i violin) (ひく 私 バイオリン))
((play i piano) (ひく 私 ピアノ))
((play i organ) (ひく 私 オルガン))
((play he organ) (ひく 彼 オルガン))

;;; (play 演じる 2)
((play you hamlet) (演じる あなた ハムレット))
((play she juliet) (演じる 彼女 ジュリエット))
((play he romeo) (演じる 彼 ロメオ))
```

Figure 3.4: Translation Examples (Part III)

```
;;; (write 書く 2)
((write you name) (書く あなた 名前))
((write she program) (書く 彼女 プログラム))
((write she paper) (書く 彼女 論文))
((write she letter) (書く 彼女 手紙))
((write i letter) (書く 私 手紙))
((write you book) (書く あなた 本))
((write he book) (書く 彼 本))
((write you address) (書く あなた 住所))
((write he address) (書く 彼 住所))
```

Figure 3.5: Translation Examples (Part IV)

```
Source = (WRITE HANAKO PAPER)   Weight-List = ( 0.5 0.5)
Rank   Target                  Distance          Most Similar Translation
   1   (書く 花子 論文)          1.33(2.67  0.0)   (WRITE SHE PAPER) -> (書く 彼女 論文)
   2   (書く 花子 手紙)          3.57(2.67 4.48)   (WRITE SHE LETTER) -> (書く 彼女 手紙)
   3   (書く 花子 紙)            5.15(3.42 6.89)   (WRITE HE BOOK) -> (書く 彼 本)
```

```
Source = (BE THERE NOTEBOOK)         Weight-List = (.564 .436)
Rank   Target                        Distance          Most Similar Translation
   1   (ある *DUMMY* ノート)          1.81( 0.0 4.16)   (BE THERE BOOK) -> (ある *DUMMY* 本)
   2   (いる *DUMMY* ノート)          435.( 0.0 999.)   (BE THERE JAPANESE) -> (いる *DUMMY* 日本人)
   3   (です *DUMMY* ノート)          565.(999. 4.16)   (BE THAT BOOK) -> (です あれ 本)
   3   (ここにある *DUMMY* ノート)    565.(999. 4.16)   (BE HERE BOOK) -> (ここにある *DUMMY* 本)
```

```
Source = (PLAY JAPANESE CARD)    Weight-List = (.211 .789)
Rank   Target                    Distance          Most Similar Translation
   1   (する 日本人 トランプ)      1.26(4.34 .429)   (PLAY TARO TENNIS) -> (する 太郎 テニス)
   2   (ひく 日本人 トランプ)      6.05(18.4 2.74)   (PLAY YOU VIOLIN) -> (ひく あなた バイオリン)
   3   (ひく 日本人 カード)        6.72(18.4 3.58)   (PLAY YOU VIOLIN) -> (ひく あなた バイオリン)
   4   (する 日本語 トランプ)      211.(999.  0.0)   (PLAY THEY CARD) -> (する 彼ら トランプ)
   5   (する 日本語 カード)        212.(999. 1.45)   (PLAY THEY CARD) -> (する 彼ら トランプ)
   5   (する 日本人 カード)        212.(999. 1.45)   (PLAY THEY CARD) -> (する 彼ら トランプ)
   7   (ひく 日本語 トランプ)      213.(999. 2.74)   (PLAY I VIOLIN) -> (ひく 私 バイオリン)
   8   (ひく 日本語 カード)        214.(999. 3.58)   (PLAY I VIOLIN) -> (ひく 私 バイオリン)
   9   (演じる 日本人 トランプ)    792.(18.4 999.)   (PLAY YOU HAMLET) -> (演じる あなた ハムレット)
   9   (演じる 日本人 カード)      792.(18.4 999.)   (PLAY YOU HAMLET) -> (演じる あなた ハムレット)
  11   (演じる 日本語 トランプ)    999.(999. 999.)   (PLAY HE ROMEO) -> (演じる 彼 ロメオ)
  11   (演じる 日本語 カード)      999.(999. 999.)   (PLAY HE ROMEO) -> (演じる 彼 ロメオ)
```

Figure 3.6: Translation Output

Group	Word selection	Count
S	Success	122
F	Fail	20
N	(No correct translation in candidates)	5
O	(One candidate)	41

Table 3.2: Result of Experiment 1

Experiment 1

Experiment 1 investigated the success rate of word selection by MBT1. 188 novel inputs were given to MBT1. In order to reduce the effort to make novel inputs, the author made them as the following procedure.

1. MBT1 generated 300 plausible novel inputs using the translation database by the following mechanism.
   (a) Select a translation example randomly.
   (b) Replace each word pair in the example by a similar word pair. The similar word pair is selected randomly from the top ten word pairs in similarity ranking.
   (c) Extract English side of it.

   This generation of plausible translations can be done automatically.

2. The author checked their validity, and collected valid English inputs.

Table 3.2 shows the result of the experiment. In the table, 'S(Success)' means the case that the correct translation is the top of the candidates.[4] 'F(Fail)' means the case that the correct translation is not the top of the candidates. 'N' means the case that MBT1 cannot output any correct translations because of the lack

[4]If there are two or more correct translations, they have to be ranked higher among other candidates.

Group	Reason	Count
Fe	Lack of translation examples	7
Fd	Wrong distance between word pairs	11
Fw	Wrong weight	2

Table 3.3: Analysis of Failures

of translation frames or word pairs. 'O' means the cases that MBT1 outputs only one (correct) translation. In this case, word selection is not needed.

The success rate of the word selection task is:

$$\frac{S}{S+F} = \frac{122}{122+20} = 85.9\%$$

Table 3.3 shows the result of the analysis of failures of word selection in the experiment. We can correct failures in the type 'Fe' and 'Fw' by adding a few translation examples in the database. But failures in the type 'Fd' are not so simple. To correct these failures completely, we need some hundreds of translation examples.

Discussion

MBT1 shows good performance on the word selection task in Experiment 1, though the mechanism of MBT1 is very simple. The main problem is not in the mechanism; it is how to construct a good database.

First, we discuss the relation between the size of translation database and the quality of the thesaurus. In MBT1, the thesaurus is constructed from the database, but conceptually the thesaurus is independent of the database; these are independent knowledge sources. The system can rely primarily on either the thesaurus or the database. If the system can use a good thesaurus, the system will show good performance using a relatively small database. In contrast, if the system has a large database, it will be able to make up for a weak thesaurus.

The method which MBT1 employs to construct a thesaurus needs a large number of translation examples. Therefore, MBT1 needs a large database even if the system mainly depends on a thesaurus. Can we construct a good thesaurus without a large database? Do we have another candidate for a thesaurus? Another choice is the use of an existing thesaurus created by human. If it is suited for our purpose, we can construct a system with a small database. The next subsection describes an experiment based on this idea.

3.5.2 MBT1b System

MBT1b is a slightly changed version of MBT1. The differences between MBT1b and MBT1 are:

1. MBT1b uses an existing thesaurus created by hand. The distance between word pairs is defined based on thesaurus codes.
2. MBT1b does not use weights of slots.

MBT1b is implemented in Sicstus Prolog on UNIX workstations.

Distance Based on Thesaurus Codes

The thesaurus which used in the experiments were made from the online version of "*Word List by Semantic Principles(WLSP)*" [NLRI], a thesaurus of Japanese words, by adding corresponding English words to Japanese words.[5]

WLSP has the following thesaurus code for each entry.

Major code, Minor code, Serial Number

For example, "野菜 (vegetable)" has the following thesaurus code.

15510,09,10

Each figure in a major code corresponds to a node of the semantic hierarchy. A minor code corresponds to a subgroup of a major code. We use the code which has six figures: five from major code and one from minor code. For example, the code of "野菜 (vegetable)" is the following:

[5]The author did not add English words to all entries, but only to those for words which appear in the database.

	ml (number of matching figures of the thesaurus codes)							exact match
	0	1	2	3	4	5	6	(same words)
similarity	0.00	0.02	0.08	0.20	0.44	0.70	0.96	1.00
distance	99.0	49.0	11.5	4.00	1.27	0.43	0.04	0.00

Table 3.4: Similarity and Distance Based on Thesaurus Code

1,5,5,1,0,09

We use the following notation for a thesaurus code of a word pair p_i:

$$WLSP(p_i) \quad = \quad < z_{i,1},\ z_{i,2},\ z_{i,3},\ z_{i,4},\ z_{i,5},\ z_{i,6} > \tag{3.39}$$

For example,

$$WLSP(<\text{野菜},\texttt{vegetable}>) \quad = \quad <1,5,5,1,0,09> \tag{3.40}$$

The similarity between two word pairs, p_i and p_j, is calculated as follows. First, the number of matching figures of the two thesaurus codes, $WLSP(p_i)$ and $WLSP(p_j)$, is calculated. This number $ml(p_i, p_j)$ is defined as the maximum value of l satisfying

$$z_{i,k} \quad = \quad z_{j,k} \quad \text{where} \quad 1 \leq k \leq l.$$

Second, the similarity is calculated by Table 3.4 from $ml(p_i, p_j)$.[6] The value 99 is used as the maximum value of distance in the experiments. For example, the similarity between <野菜,vegetable> and < じゃがいも,potato> is:

$$\begin{aligned}
WLSP(<\text{野菜},\texttt{vegetable}>) &= <1,5,5,1,0,09> \\
WLSP(<\text{じゃがいも},\texttt{potato}>) &= <1,5,5,2,0,07> \\
ml(<\text{野菜},\texttt{vegetable}>,<\text{じゃがいも},\texttt{potato}>) &= 3 \\
similarity(<\text{野菜},\texttt{vegetable}>,<\text{じゃがいも},\texttt{potato}>) &= 0.20
\end{aligned}$$

[6]The distance is calculated by Formula (3.30).

We use Formulae (3.26) and (3.27) to calculate the distance between two translation examples, and we use the same weight values for all slots in a translation example: i.e.

$$w_k(D, f^s, f^a) = \frac{1}{f^a} \qquad (3.41)$$

because the number of examples is too small to set weights automatically by statistical methods.

Experiment 2

Experiment 2 investigated whether the use of modified WLSP can make the size of database smaller. The experiment was done using the following procedure.

1. Prepare a training set of translation examples. The same set as in Experiment 1 is used. The size is 1403.
2. For each translation frame, store a translation example which has the translation frame. This is the initial translation database. The size is 364.
3. Extract source (English) sides of all the remaining translation examples, and translate them. If the output is correct[7], do nothing. If the output is incorrect, store the translation example.
4. If some translation examples were stored in Step 3, repeat Step 3 for unstored translation examples.

In the loop of the procedure, the size of translation examples increased in the sequence 364 → 490 → 507. Finally, the system with 507 translation examples can correctly translate all 1403 translation examples.

Experiment 3

In Experiment 3, the inputs that MBT1 failed to translate in Experiment 1 were translated by MBT1b. The set of 507 translation examples obtained in Experiment 2 was used as the run time database.

[7]If the training example is the top of translation candidates, the system is considered to be correct. The condition does not strictly imply correctness of the word selection, but we employ it because it can be checked automatically.

Word selection	Group in Table3.3		
	Fe	Fd	Fw
Success	0	8	1
Fail	7	3	1

Table 3.5: Result of Experiment 3

Table 3.5 shows the result of Experiment 3. MBT1b succeeds in translating eight out of the eleven inputs in Fd. This shows that the use of an existing thesaurus created by human is effective.

Discussion

Experiments 2 and 3 show that the combination of the small database and an existing thesaurus is effective for example-based word selection. Because it is very difficult to create a large translation database in a short period, the use of an existing thesaurus is very convenient in the early stages of system construction: even if the size of the database is small, the system is usable. This is a major advantage of using an existing thesaurus.

3.6 Discussion and Related Work

3.6.1 Advantages and Disadvantage

Although MBT1 is not a full realization of the example-based translation concept, it retains the following advantages of the original concept.

1. We can easily construct and upgrade the system.
   The main knowledge source of the system is translation examples. The system interprets examples directly: i.e. applies examples to the given input with best match. Therefore the system needs no rules. This frees us from rule acquisition: we can easily construct the system and upgrade it by adding appropriate translation examples.

2. Best match produces its robustness.
   Because traditional rule-based systems work on exact-match reasoning, they fail to translate when they have no knowledge that matches the input exactly. On the other hand, because MBT1 works with best-match reasoning, it intrinsically works in a fail-safe way.

3. MBT1 produces not only a translation output but also its score, i.e. its reliability factor.

4. The knowledge of the system has a long life cycle.
   The knowledge of a traditional rule-based system is in the form of rules, which are strongly dependent on the system and the particular linguistic theory. Therefore the knowledge cannot be transferred to other systems. The knowledge of MBT1 is in the form of translation examples and thesaurus, which are independent of the system and useful for a long time. Therefore the knowledge can be used in other systems and has a long life cycle.

A disadvantage is:

1. Best match is a time consuming task on sequential computers. It intrinsically involves the exhaustive search, which needs a great deal of computation.

MBR, one origin of MBT1, is implemented on a massively parallel computer. The author hopes that parallel computation will overcome the disadvantage for machine translation also.

3.6.2 Applicability and Restriction of MBT1

MBT1 is a general framework of translation between n-tuples as follows.

$$< \mathrm{Head}^s, \mathrm{Argument}^s_1, \ldots, \mathrm{Argument}^s_n > \leftrightarrow < \mathrm{Head}^t, \mathrm{Argument}^t_1, \ldots, \mathrm{Argument}^t_n >$$

Therefore, we can apply MBT1 to other word selection tasks; e.g. the translation between simple noun phrases, by encoding the following.

(3–31) a good example よい 例

(3–32) (example good) (例 よい)

However, there are some limitations on MBT1, which are:

1. We have to encode examples in the form of records which have a fixed set of fields.
2. The main process of MBT1 is to calculate the preference score of a translation candidate. In the process, the system utilizes only examples that have the same format as the translation candidate; the system cannot utilize other examples.

Because of these limitations, MBT1 cannot manage sentences that have optional elements.

In summary, MBT1 can apply to tasks which have fixed-formatted input and output. Many subtasks in machine translation can be encoded into a fixed-format: so we can employ MBT1 for submodules of machine translation systems. But MBT1 cannot handle translation of full sentences, because sentences have some optional elements and thus cannot be encoded into the fixed-format.

3.6.3 Related Work

After MBT1 was proposed, ATR has developed EBMT [Sumita et al 90]. They applied an MBT1-like method to translate Japanese "noun_1 NO(の) noun_2" into English. The translation of "noun_1 NO(の) noun_2" is one of the most difficult tasks in Japanese–English translation. EBMT can select the best translation pattern of "noun_1 NO noun_2": e.g. "noun_2 of noun_1" or "noun_1's noun_2". EBMT demonstrated how well an MBT1-like method works on the task.

3.7 Summary

In this chapter, the author has proposed MBT1, which is the first prototype of example-based translation. MBT1 has shown that example-based reasoning is applicable to language translation, and that it is a promising approach. The major results are:

- MBT1 has the following advantages:
  - MBT1 frees us from rule acquisition. We can easily construct the system by collecting translation examples, and upgrade it by adding appropriate translation examples.
  - Best match means it is robust.
  - MBT1 produces not only a translation output but also its reliability factor.
  - The knowledge of MBT1 has a long life cycle.
- A disadvantage of MBT1 is that it needs a great deal of computation.
- Experiments demonstrate how well MBT1 handles the word selection task in translation between verb frame instances.
- The use of existing thesauri is very convenient in the early stages of the system construction.
- MBT1 is applicable to other subtasks in machine translation, but it is not applicable to the translation of full sentences.

Chapter 4

Example-Based Transfer

4.1 Introduction

In the previous chapter, the author proposed MBT1, which can solve the word selection problem in translation between verb frame instances. The main restriction of MBT1 is the form of examples: an example has to be in the form of a fixed record. This obstructs the application of MBT1 to full sentence translation. In this chapter, we concentrate on the problem of overcoming this restriction.

First, we need a more flexible way to represent translation examples. Sentences are not represented in the form of fixed records, because some sentences have some optional fragments. We know that a sentences has some structures. Tree structure is often used to represent the syntactic structure or semantic structure of a sentence. Tree structures are flexible and recursive and so suited for our purpose.

Second, we have to solve a new critical problem for example-based translation: how to utilize more than one example for translating a sentence. This issue arises when an input sentence is too long to match to a single translation example. In almost all cases, we cannot translate a sentence without utilizing several translation examples.

This chapter describes a solution to this problem: MBT2. MBT2 can do bidirectional translation between an English word-dependency tree and a Japanese word-dependency tree. It covers the whole transfer process of a sentence.

4.2 Need to Combine Fragments

4.2.1 Need to Combine Fragments

The basic idea of example-based translation is very simple: translate a source sentence by imitating a translation example of a similar sentence in the database. But in almost all cases, it is necessary to decompose an input sentence into several smaller fragments and to find a suitable translation example for each of them.

Suppose the following sentence (4-1) is given the system to translate.

(4-1) He buys a book on international politics.

If the system knows the same sentence and its translation equivalent, the system can output the translation equivalent as the answer. But this is unlikely. In almost all cases, the system cannot find the same sentence and its translation equivalent in the database. Therefore the system utilizes some examples similar to the given input. If the system knows the following translation example (4-2), it will be used to translate (4-1).

(4-2) He buys a notebook. 彼は ノートを 買う。

If the system knows the following correspondence between fragments in (4-2),

(4-3) a notebook $\leftrightarrow$ ノート

the system can infer to translate (4-4) into (4-5).

(4-4) He buys X.

(4-5) 彼は X を 買う。

But the system cannot extract any information about the translation of the fragment (4-6) from (4-2).

(4-6) a book on international politics

If the given sentence is more simple, e.g.

(4-7) He buys a book.

the system may translate the difference between (4-7) and (4-2), i.e. 'a book', using a dictionary. But it is not a complete answer, because it is applicable only when the difference is one or two words. When the difference is larger, i.e. 'a book on international politics', it cannot be translated only by using a dictionary. The complete answer is to use another example that contains the difference. Suppose the system knows the following example.

(4-8)	I read a book on international politics.	私は 国際政治について 書かれた 本を 読む。

The system will be able to translate (4-1) into (4-9) by imitating (4-2) and (4-9) and combining fragments of them.

(4-9) 彼は 国際政治について 書かれた 本を 買う。

If the system does not know (4-8), the system may use a similar example with (4-6), for example,

(4-10)	a book on economics	経済学について 書かれた 本

and translate the difference between (4-6) and (4-10), i.e. 'international politics' by using another example.

It is easy for a human to do this, but not so for a machine. The ability to combine some fragments of translation examples is essential to example-based translation. A lack of this ability restricts the power of example-based translation. In this chapter, we concentrate on the implementation of this ability in a machine.

4.2.2 Towards Implementation

First, we have to define what is the fragment to be combined. In a translation example of two natural language sentences, there are some correspondences between fragments. For example, there are the following correspondences between the fragments in (4-2).

(4-11)	he	↔	彼
(4-12)	a notebook	↔	ノート

A fragment which has a correspondence is a partially translatable unit in a translation example. A fragment in which some partially translatable units are removed is also translatable, e.g.

(4-13) X buy Y ↔ X は Y を 買う。

Sadler calls these fragments *translation units* [Sadler 89]. Using the concept of translation units, we will be able to implement translation by combining some fragments as follows.

1. Find a combination of translation units which covers a given input.
2. Transfer translation units in the combination according to correspondences in the translation examples.
3. Generate the output from the transferred combination of translation units.

There are another problem in this approach. For example, suppose the system knows another example for 'a book on ~',

(4-14) a book on the desk 机の 上の 本

In this case, the system has to determine which example, (4-10) or (4-14), it should use to translate 'a book on ~' in (4-1). Generally, there are some candidates of combinations of translation units which covers a given input, and they produce different outputs. Therefore, we need a way to determine the best combination.

In MBT1, the score of a translation candidate is defined based on the distance between the translation candidate and a translation example in the database. But in this case, we cannot define the score based on distance, because the system uses multiple examples in order to translate one sentence. We have to define the score of a translation candidate based on the score of a combination of translation units, because different combinations produce different outputs.

4.3 Matching Expression

MBT2 translates a source word dependency tree into a target word dependency tree. This section will define the terms *translation example*, *translation unit*, and

[Translation Example 1]

```
%% He buys a notebook.
ewd_e([e1,[buy,v],
          [e2,[he,pron]],
          [e3,[notebook,n],
              [e4,[a,det]]]]).

%% 彼はノートを買う。
jwd_e([j1,[買う,v],
          [j2,[は,p],
              [j3,[彼,pron]]],
          [j4,[を,p],
              [j5,[ノート,n]]]]).

%% e1 <-> j1, e2 <-> j3, e3 <-> j5
clinks([[e1,j1],[e2,j3],[e3,j5]]).
```

Figure 4.1: Translation Example 1

matching expression. MBT2 is implemented in Sicstus Prolog. We will use Prolog syntax in some representations.

4.3.1 Translation Database

The translation database is the collection of translation examples. A translation example consists of three parts:

- an English word-dependency tree (EWD)
- a Japanese word-dependency tree (JWD)
- correspondence links

Figure 4.1 and 4.2 show translation examples in Prolog.

[Translation Example 2]

```
%% I read a book on international politics.
ewd_e([e11,[read,v],
          [e12,['I',pron]],
          [e13,[book,n],
               [e14,[a,det]],
               [e15,[on,p],
                    [e16,[politics,n],
                         [e17,[international,adj]]]]]]).

%% 私は国際政治について書かれた本を読む。
jwd_e([j11,[読む,v],
          [j12,[は,p],
               [j13,[私,pron]]],
          [j14,[を,p],
               [j15,[本,n],
                    [j16,[た,aux],
                         [j17,[れる,aux],
                              [j18,[書く,v],
                                   [j19,[について,p],
                                        [j20,[国際政治,n]]]]]]]]]).

clinks([[e11,j11],[e12,j13],[e13,j15],[e16,j20]]).
```

Figure 4.2: Translation Example 2

In these figures, each number with prefix 'e' or 'j' in a word-dependency tree represents the ID of the subtree. Each node in a tree contains a word (in root form) and its syntactic category. A correspondence link is represented as a pair of IDs.

4.3.2 Translation Unit

In translation examples, we define a translatable tree as follows.

Translatable Tree A translatable tree is a tree or subtree which has a correspondence link in a translation example.

In Translation Example 1 (Figure 4.1), there are three translatable trees in each side.

English	Japanese
e1	j1
e2	j3
e3	j5

Next, we introduce the notion of translation unit [Sadler 89]. In short, a translation unit is a translatable fragment in a translation example. A translation unit is defined as follows.

Translation Unit A translation unit is:

- a translatable tree, or
- a translatable tree in which some translatable subtrees are removed

In Translation Example 1 (Figure 4.1), there are six translatable trees in each side.

English	Japanese
e1	j1
e2	j3
e3	j5
e1-e2	j1-j3
e1-e3	j1-j5
e1-e3-e5	j1-j3-j5

4.3.3 Matching Expression

Next we will introduce the concept *matching expression*. A matching expression represents a word dependency tree as a combination of translation units. A matching expression (ME) is defined as

```
<ME> ::= [<ID>|<ME-Commands>]
<ME-Commands> ::= []
               or [<ME-Command>|<ME-Commands>]
<ME-Command> ::= [d,<ID>]
              or [r,<ID>,<ME>]
              or [a,<ID>,<ME>]
```

A matching expression (`<ME>`) consists of a translatable tree (`<ID>`) and some transformational commands (`<ME-Commands>`). There are three commands:

1. `[d,<ID>]` : delete a translatable tree (`<ID>`).
2. `[r,<ID>,<ME>]` : replace a translatable tree (`<ID>`) with a matching expression(`<ME>`).
3. `[a,<ID>,<ME>]` : add a matching expression (`<ME>`) as a child of a root node of a translatable tree (`<ID>`).

For example, matching expression (a) represents word-dependency tree (b).

```
(a) [e1,[r,e3,[e13]]]
(b) [[buy,v],
[[he,pron]],
[[book,n],
 [[a,det]],
 [[on,p],
  [[politics,n],
   [[international,adj]]]]]]
```

The matching expression (a) consists of two translation units: e1-e3, and e13. And it has the information to combine them: replace e3 with e13.

If a matching expression is given, we can easily compose a word-dependency tree for it. [1] And we can easily transfer a matching expression into a corresponding matching expression of another language, because all ID's in a matching expression are translatable trees. There remains the problem of obtaining a matching expression from a given word-dependency tree. The next section will show an algorithm for this.

4.4 Translation via Matching Expression

Translation is done via two matching expressions: a source matching expression and a target matching expression. Figure 4.3 shows the flow of the translation process. The translation process consists of three steps: decomposition, transfer, and composition. This process generates all candidate translations using Prolog's backtrack mechanism.

4.4.1 Decomposition

In decomposition, the system decomposes a source word-dependency tree (SWD) into translation units, and makes a source matching expression (SME). For example,

[1] A delete command [d,<ID>] and a replace command [r,<ID>,<ME>] can be executed deterministically. But an add command [a,<ID>,<ME>] is ambiguous, because it does not specify the position of <ME> in the list of child trees under the root node of <ID>. This problem will be solved in Section 4.4.3.

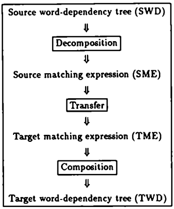

Source word-dependency tree (SWD)
Decomposition
Source matching expression (SME)
Transfer
Target matching expression (TME)
Composition
Target word-dependency tree (TWD)

Figure 4.3: Flow of Translation Process

```
SWD = [[buy,v],
       [[he,pron]],
       [[book,n],
        [[a,det]],
         [[on,p],
          [[politics,n],
           [[international,adj]]]]]]
SME = [e1,[r,e3,[e13]]]
```

The skeleton of algorithm to do this is shown in Figure 4.4 as a Prolog program. In this program, there are three points of nondeterminism.

1. **translatable_tree([ID,Node|Children1])** in (C-4-1). This term retrieves translatable trees which have the same root node as the root node of the given word-dependency tree.

```
% decomp(+WD,-ME)
decomp([Node|Children2],[ID|DifList]) :-
  translatable_tree([ID,Node|Children1]),
  decomp1(Children1,Children2,ID,DifList).          (C-4-1)

decomp1([],[],_,[]).                                 (C-4-2)

decomp1([X|Children1],[Y|Children2],P,DifList) :-
  decomp2(X,Y,DifList1),
  decomp1(Children1,Children2,P,DifList2),
  append(DifList1,DifList2,DifList).                 (C-4-3)

decomp1([[ID|_]|Children1],Children2,P,[[d,ID]|DifList]) :-
  translatable_tree([ID|_]),
  decomp1(Children1,Children2,P,DifList).           (C-4-4)

decomp1(Children1,[C|Children2],P,[[a,P,ME]|DifList]) :-
  translatable_tree([P|_]),
  decomp(C,ME),
  decomp1(Children1,Children2,P,DifList).           (C-4-5)

decomp2([ID,N|Children1],[N|Children2],DifList) :-
  decomp1(Children1,Children2,ID,DifList).          (C-4-6)

decomp2([ID|_],Y,[[r,ID,ME]]) :-
  translatable_tree([ID|_]),
  decomp(Y,ME).                                      (C-4-7)
```

Figure 4.4: Skeleton of the Decomposition Algorithm (in Prolog)

2. (C-4-6) and (C-4-7). This program produces some needless commands like `[r,e2,[e2]]`.

3. (C-4-3), (C-4-4) and (C-4-5). A replace command may be represented as a combination of a delete command and an add command.

The first use of nondeterminism is essential. But the second and the third are not essential. The second can be cut off easily. To cut off the third, we can use the following heuristics.

- Define replaceability between syntactic categories. A tree X can be replaceable with a tree Y, if the syntactic category of the root node of X is replaceable with the syntactic category of the root node of Y.

- If two trees are replaceable, the system produces only a replace command.

For example, suppose that we define the replaceablility between syntactic categories as follows.

1. Each syntactic category is replaceable with the same one.

2. The category `pron`(pronoun) is replaceable with the category `n`(noun).

3. The category `det`(determiner) is replaceable with the category `adj`(adjective).

Then, the system does not produce the matching expression

```
[e1,[d,e3],[a,e1,[e13]]]
```

because the syntactic category n of the root node of the tree `e3` is replaceable with the syntactic category n of the root node of the tree `e13`.

The modified program is shown in Appendix A. This program is for English word-dependency trees. The system has another program to decompose Japanese word-dependency trees. In comparison of Japanese word-dependency trees, the order of subtrees is not important.

4.4.2 Transfer

In the transfer step, the system replaces every ID in a source matching expression with its corresponding ID. For example,

```
SME = [e1,[r,e3,[e13]]]
TME = [j1,[r,j5,[j15]]]
```

4.4.3 Composition

In the composition step, the system composes a target word-dependency tree according to a target matching expression. For example,

```
TME = [j1,[r,j5,[j15]]]
TWD = [[買う,v],
        [[は,p],
         [[彼,pron]]],
        [[を,p],
         [[本,n],
          [[た,aux],
           [[れる,aux],
            [[書く,v],
             [[について,p],
              [[国際政治,n]]]]]]]]]]
```

This step is divided into two sub-steps; the main composing step and validity checking. In the main composing step, there is no ambiguity with one exception: because an add-command `[a,<ID>,<ME>]` specifies only the parent node (`<ID>`) to add the tree (`<ME>`), there are some choices in composing English word-dependency trees. In this step, all possibilities are generated.

Validity of the composed word-dependency trees are checked using syntactic categories. Validity is checked in every parent-children unit. For example, in the above target word-dependency tree,

```
[v,[p,p]], [p,[pron]], [p,[n]], [n,[aux]], ...
```

are checked. A unit is valid if there is a unit which has the same category pattern in the database. A word-dependency tree is valid if all parent-children units are valid.

4.5 Score of Translation

To select the best translation out of all candidates generated by the system, we introduce the score of a translation. It is defined based on the score of the matching expression, because the matching expression determines the translation output. The scores of the source matching expression and the target matching expression are calculated separately.

4.5.1 Score of Translation Unit

First, we will define the score of a translation unit. The score of a translation unit should reflect the correctness of the translation unit. Which translation unit is better? The two main factors are:

1. A larger translation unit is better.
2. A translation unit in a matching expression is a fragment of a source (or target) word-dependency tree, and also a fragment of a translation example. There are two environments of a translation unit; in a source (or target) tree and in a translation example. The more similar these two environments are, the better the result is.

To calculate 1, we define the size of a translation unit (TU).

$$size(TU) = \text{"the number of nodes in TU"} \quad (4.1)$$

To calculate 2, we need a measure of the similarity between two environments of a translation unit, i.e. the external similarity. To estimate the external similarity, we restrict these environments as follows. In the simplest case, the *restricted environment* of a translation unit consists of the nodes only a single link away from a node of the translation unit. If the corresponding nodes are the same in the two

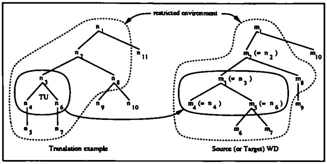


Figure 4.5: Restricted Environments of TU

environments, those environments are extended one more link outside. Figure 4.5 illustrates the restricted environments of a translation unit. External similarity is estimated as the best matching of the two restricted environments. To find the best matching, we first determine the correspondences between nodes in the two restricted environments. Some nodes have several candidates for correspondence. For example, n_7 corresponds with either m_6 or m_7. In this case, we select the most similar node. To do this, we assume that similarity values between nodes (words) are defined as numeric values between 0 and 1 in a thesaurus, explained in the next subsection. When the best match is found, we can calculate the matching point between two environments, $mpoint(TU, WD)$.

$$mpoint(TU, WD) = \text{"summation of similarity values between corresponding nodes in two restricted environments at the best matching"} \quad (4.2)$$

This value is used as a measure of similarity between two environments.

Finally, we define the score of a translation unit, $score(TU, WD)$.

$$score(TU, WD) = size(TU) \times (size(TU) + mpoint(TU, WD)) \quad (4.3)$$

For example, we assume that the following similarity values are defined in a thesaurus. [2]

```
ew_sim([book,n],[notebook,n],0.60).
ew_sim([buy,v],[read,v],0.00).
jw_sim([本,n],[ノート,n],0.70).
jw_sim([買う,v],[読む,v],0.08).
```

Then the scores of translation units in the previous section are as follows.

TU	size	mpoint	score
e1-e3	2	0.60	5.20
e13	5	0.00	25.00
j1-j5	4	0.70	18.80
j15	6	1.08	42.48

4.5.2 Score of Matching Expression

The score of a matching expression is defined as follows.

$$score(ME, WD) = \frac{\sum_{TU \in ME} score(TU, WD)}{size(WD)^2} \tag{4.4}$$

For example,

ME	score
[e1,[r,e3,[e13]]	0.616
[j1,[r,j5,[j15]]	0.613

4.5.3 Score of Translation

Finally, we define the score of a translation as follows.

$$\begin{aligned} score(SWD, SME, TME, TWD) &= \\ min(score(SME, SWD), score(TME, TWD)) \end{aligned} \tag{4.5}$$

For example, the score of the translation in the previous section is 0.613.

[2]These similarity values are calculated by the method described Section 4.5.4.

4.5.4 Thesaurus: Similarity Between Words

Similarity between words is calculated based on thesaurus codes of existing thesauri. *"Word List by Semantic Principle (WLSP)"*[NLRI] is used for Japanese and *"Longman Lexicon of Contemporary English (LLCE)"*[McArthur 81] is used for English. The use of WLSP was described in Section 3.5.2.

In LLCE, each entry word has a thesaurus code. For example, word 'apple' has 'A150'. LLCE has three level in the hierarchy. For example, 'A150' is in the following position.

A Life and Living Things
A150 – 158 Plants Generally
A150 Kinds of Fruits

We represent this as[3]

a,150,150

The following notation is used for a thesaurus code of a English word w_i:

$$LLCE(w_i) = < z_{i,1}, z_{i,2}, z_{i,3} > \qquad (4.6)$$

For example,

$$LLCE(\mathbf{apple}) = < a, 150, 150 > \qquad (4.7)$$

The method for caluculating the similarity based on WLSP, which was described in Section 3.5.2, is applicable to calculating the similarity based on LLCE. Although, we have to define another 'ml[4] – *similarity* table' for LLCE's code, because the length (the number of symbols or figures) of a LLCE's code is different from the one of a WLSP's code. The table is shown in Table 4.1.

4.6 Examples

The English verb "*eat*" corresponds to some Japanese verbs, e.g. "食べる" and "侵す". For example,

[3]The second level is described by the starting number of the group.

[4]The value ml is the number of matching symbols of the thesaurus codes, which was defined in Section 3.5.2.

	ml (number of matching symbols of thesaurus code)				exact match
	0	1	2	3	(same words)
similarity	0.00	0.10	0.60	0.96	1.00

Table 4.1: Similarity Based on English Thesaurus Code

(4–15) The man eats vegetables. 人は 野菜を 食べる。

(4–16) Acid eats metal. 酸は 金属を 侵す。

Figure 4.6 shows the prepared translation database, and Figure 4.7 shows the thesaurus. And Figures 4.8 and 4.9 show outputs by MBT2. MBT2 chooses '食べる' for 'he eats potatoes' and '侵す' for 'sulphuric acid eats iron'. These are correct outputs.

4.7 Discussion

The transfer mechanism of MBT2 can be characterized using the two terms: example-based transfer and parallel non-destructive transfer.

Example-based transfer means that it works by using examples directly, whereas traditional *rule-based transfer* works by using rules. In other words, example-based transfer works using best-match reasoning, whereas rule-based transfer works using exact-match reasoning. An example-based transfer has two major advantages. First, we do not need to acquire rules: we can easily construct and upgrade the system by adding several translation examples to the database. Second, the system can produce high quality translations, because it sees as wide a scope as possible in a sentence, and uses the best combination of translation units.

The term *parallel non-destructive transfer* was introduced by Watanabe [Watanabe 90]. *Parallel* [5] means that the whole input structure is rewritten by one application of rules, and *non-destructive* means that the input structure is

[5] This term comes from *parallel production* in the area of Graph Grammars.

```
e(1,"A man eats vegetables.").
j(1,"人は野菜を食べる。").
ewd(1,[1,[eat,v],
        [2,[man,n],
           [3,[a,det]]],
        [4,[vegetable,n]]]).
jwd(1,[1,[食べる,動詞],
        [2,[が,助詞],
           [3,[人,名詞]]],
        [4,[を,助詞],
           [5,[野菜,名詞]]]]).
clist(1,[[1,1],[3,2],[5,4]]).

e(2,"Acid eats metal.").
j(2,"酸は金属を侵す。").
ewd(2,[1,[eat,v],
        [2,[acid,n]],
        [3,[metal,n]]]).
jwd(2,[1,[侵す,動詞],
        [2,[が,助詞],
           [3,[酸,名詞]]],
        [4,[を,助詞],
           [5,[金属,名詞]]]]).
clist(2,[[1,1],[3,2],[5,3]]).

e(3,"He likes potatoes.").
j(3,"彼はじゃがいもが好きだ。").
ewd(3,[1,[like,v],
        [2,[he,pron]],
        [3,[potato,n]]]).
jwd(3,[1,[好きな,形容動詞],
        [2,[が,助詞],
           [3,[彼,代名詞]]],
        [4,[を,助詞],
           [5,[じゃがいも,名詞]]]]).
clist(3,[[1,1],[3,2],[5,3]]).

e(4,"Sulphuric acid is dangerous.").
j(4,"硫酸は危険だ。").
ewd(4,[1,[be,v],
        [2,[acid,n],
           [3,[sulphuric,adj]]],
        [4,[dangerous,adj]]]).
jwd(4,[1,[危険な,形容動詞],
        [2,[が,助詞],
           [3,[硫酸,名詞]]]]).
clist(4,[[1,1],[3,2]]).

e(5,"Iron is the most useful metal.").
j(5,"鉄は最も有効な金属だ。").
ewd(5,[1,[be,v],
        [2,[iron,n]],
        [3,[metal,n],
           [4,[the,det]],
           [5,[useful,adj],
              [6,[most,adv]]]]]).
jwd(5,[1,[だ,助動詞],
        [2,[が,助詞],
           [3,[鉄,名詞]]],
        [4,[金属,名詞],
           [5,[有効な,形容動詞],
              [6,[最も,副詞]]]]]).
clist(5,[[1,1],[3,2],[4,3],[5,5],[6,6]]).
```

Figure 4.6: Translation Database

```
ew_sim([he,pron],[man,pron],0.000000).
ew_sim([he,pron],[acid,n],0.000000).
ew_sim([potato,n],[vegetable,n],0.100000).
ew_sim([potato,n],[metal,n],0.000000).
ew_sim([iron,n],[vegetable,n],0.000000).
ew_sim([iron,n],[metal,n],0.600000).

jw_sim([彼,代名詞],[人,名詞],0.200000).
jw_sim([彼,代名詞],[酸,名詞],0.020000).
jw_sim([硫酸,名詞],[人,名詞],0.020000).
jw_sim([硫酸,名詞],[酸,名詞],0.960000).
jw_sim([じゃがいも,名詞],[野菜,名詞],0.200000).
jw_sim([じゃがいも,名詞],[金属,名詞],0.080000).
jw_sim([鉄,名詞],[野菜,名詞],0.080000).
jw_sim([鉄,名詞],[金属,名詞],0.700000).
```

Figure 4.7: Thesaurus

never destroyed during a transfer process. Watanabe's Rule Combination Transfer (RCT) and MBT2 are parallel non-destructive transfers. In contrast, almost all transfer models are sequential destructive transfers. Grade [Nagao and Tsujii 86] is a typical example of this: a part of the input structure is destructively rewritten and this process is continued until all parts are rewritten. One of the problems of sequential destructive transfer systems is that they produce a data structure that holds features and grammatical relations of both source and target language in the course of the transfer process. This makes the transfer process very complicated. In contrast, since a parallel non-destructive transfer does not produce such a data structure, we can intuitively *understand* the transfer process. Let's consider how the system produce a translation output. A parallel non-destructive transfer system can produce not only the output but also an explanation why the system produces the output: the explanation is a combination of rules in RCT or a matching expression (i.e. a combination of translation units) in MBT2. If the system produces an incorrect translation output, we can easily find out the wrong rules or translation units. On the other hand, in a destructive sequential

```
*** Translation Source ***
[[eat,v],
 [[he,pron]],
 [[potato,n]]]
*** Translation Results ***
No. 1  (Score = 0.3444)
[[食べる,動詞],
 [[が,助詞],
  [[彼,代名詞]]],
 [[を,助詞],
  [[じゃがいも,名詞]]]]
SME = [e_1_1,[r,e_1_2,[e_3_2]],
             [r,e_1_4,[e_3_3]]]
      (Score = 0.3444)
TME = [j_1_1,[r,j_1_3,[j_3_3]],
             [r,j_1_5,[j_3_5]]]
      (Score = 0.5680)

No. 2  (Score = 0.3333)
[[侵す,動詞],
 [[が,助詞],
  [[彼,代名詞]]],
 [[を,助詞],
  [[じゃがいも,名詞]]]]
SME = [e_2_1,[r,e_2_2,[e_3_2]],
             [r,e_2_3,[e_3_3]]]
      (Score = 0.3333)
TME = [j_2_1,[r,j_2_3,[j_3_3]],
             [r,j_2_5,[j_3_5]]]
      (Score = 0.5320)
```

Figure 4.8: Output for "*He eats potatoes*"

```
*** Translation Source ***
[[eat,v],
 [[acid,n],
  [[sulphuric,adj]]],
 [[iron,n]]]
*** Translation Results ***
No. 1  (Score = 0.4750)
[[侵す,動詞],
 [[が,助詞],
  [[硫酸,名詞]]],
 [[を,助詞],
  [[鉄,名詞]]]]
SME = [e_2_1,[r,e_2_2,[e_4_2]],
             [r,e_2_3,[e_5_2]]]
      (Score = 0.4750)
TME = [j_2_1,[r,j_2_3,[j_4_3]],
             [r,j_2_5,[j_5_3]]]
      (Score = 0.6792)

No. 2  (Score = 0.3750)
[[食べる,動詞],
 [[が,助詞],
  [[硫酸,名詞]]],
 [[を,助詞],
  [[鉄,名詞]]]]
SME = [e_1_1,[r,e_1_2,[e_4_2]],
             [r,e_1_4,[e_5_2]]]
      (Score = 0.3750)
TME = [j_1_1,[r,j_1_3,[j_4_3]],
             [r,j_1_5,[j_5_3]]]
      (Score = 0.4920)
```

Figure 4.9: Output for *"Sulphuric asid eats iron"*

transfer, the explanation is a long chain of applications of rules: it is difficult for us to locate the incorrect rules.

MBT2 has the above desirable characteristics. But the mechanism of MBT2 is too general [6] and too simple to apply practical language transfer tasks. We have to solve the following problems.

1. Which representation should be used for examples?

2. How much information do we encode into examples?

There are several candidates for internal representation of sentences: e.g. syntactic representation, semantic representation, and intermediate or mixed representation of syntax and semantics. MBT2 uses word-dependency trees (i.e. syntactic representation) as an internal representation of sentences. Sadler uses word-dependency trees in which links have semantic labels [Sadler 89]. These are semantic-oriented syntactic structures.[7] Syntax is important in representing some constraints. Therefore, we need to represent at least some syntactic information. On the other hand, semantic information is useful for selecting the preferable output from some candidates. In MBT2, there is no explicit semantic information about sentences. Only word semantics exists, in the thesaurus. Therefore, MBT2 can use only syntax-oriented analogy. In order to make a system that can use semantics-oriented analogy, we have to give the system semantic information about sentences.

3. How to handle syntactic transformations.

Passive voice and relative clause are well known as syntactic transformations. Because there is no relation between normal forms and transformational forms in MBT2, MBT2 cannot utilize normal forms to translate transformational forms. There are two possible ways to solve this problem:

[6] MBT2 is a general mechanism for translating a tree into other tree based on pair of tree examples. It can used for any task that requires tree-to-tree transformation.

[7] Personal contact with T. Witkam at his seminar of *MT based on Analogy* at ATR, Kyoto, Japan on July 1990.

- Employ transformational rules that bridge the gap between normal forms and transformational forms.
- Introduce a new representation (e.g. a semantic representation) on which transformations are not important in the matching process.

4. Appropriate grain size of translation units.

It is not practical to store all translation units in translation examples into the database. Large translation units can produce better translations, but they have little chance to be used. In contrast, small translation units have much greater chance to be used, but they produce literal translations. We have to determine an appropriate grain size of translation units to be stored.

5. Computation problem.

MBT2 needs a great deal of computation. In order to overcome this disadvantage, we need parallel computation.

4.8 Summary

In this chapter, the author has discussed an implementation of a fully example-based transfer system. An critical problem in the implementation is how to utilize more than one translation example to translate one sentence. The author has shown a solution for it in MBT2. The major results are:

- The matching expression, which represents how to combine some translation fragments, is introduced. This makes it possible to utilize more than one translation examples.
- A translation mechanism, which produces some translation candidates, is introduced. This mechanism translates a sentence via two matching expression; i.e. a source matching expression and a target matching expression.
- The score of translation is introduced. It can determine the best translation output out of some translation candidates.

- The proposed framework, MBT2, has the following advantage:
  - We can easily construct and upgrade the system by adding several translation examples into the database, because the system uses examples directly, not rules.
  - MBT2 can produce high quality translation, because MBT2 sees as wide as a scope as possible in a sentence and uses the best combination of translation units.
  - MBT2 can produce not only a translation output but also the explanation why it produced the output.
- MBT2 is too general and naive to apply to real transfer tasks. We have to address the issues of: representation level, grammatical transformations, grain size of translation units, and computation problem.

MBT2 is the first prototype example-based transfer system: it does not include parsing and generation. However, generation is not serious, because the system can easily generate a target sentence by collecting nodes in a word-dependency tree. And parsing will be implemented using a modified version of MBT2's mechanism. We can modify the system to find a source matching expression satisfying the word order constraint of the given input word sequence [Sato 91].

The next step of research for example-based translation is to construct a model for parallel (distributed) example-based translation. In example-based translation, the knowledge source is distributed into individual translation examples or translation units; i.e. each translation example or unit is an agent for translation. A translation process will be implemented as a cooperative problem solving process by such agents [Sato 91].

Chapter 5

Discussion

5.1 The Rule-Based Approach versus the Example-Based Approach

Before the *Example-Based Approach* (EBA) was proposed, the *Rule-Based Approach* (RBA) was the only approach to constructing expert systems like machine translation systems. The major characteristic of RBA is use of *rules*. A typical rule-based system consists of a set of rules (if A, do B) and an inference engine, which invokes the appropriate rules for achieving a specified goal and then executes them. The process of acquiring rules is called *knowledge acquisition* or *rule acquisition*. Automatic acquisition of rules from examples is called *learning* or *rule learning*, and a module to do this task is called a *learning engine*. Figure 5.1(a) shows the basic diagram of RBA including the rule learning process.

On the other hand, the major characteristic of EBA is the direct use of *examples*; no use of rules. A typical example-based system consists of a set of examples (input-output pairs) and an inference engine, which retrieves the appropriate examples for achieving a specified goal and then adapts them. There is neither explicit knowledge acquisition process nor a learning engine in EBA, but the system can learn in the sense that it can handle for novel inputs. Figure 5.1(b) shows the basic diagram of EBA.

In this section, we will compare these two approaches from the viewpoint of

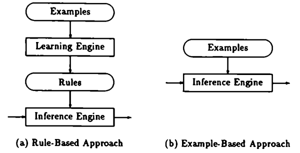


Figure 5.1: Basic Diagrams of Rule-Based Approach and Example-Based Approach

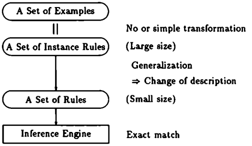
learning, and try to clarify the difference.

5.1.1 Explicit or Implicit Generalization

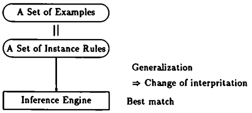
In RBA, learning is equivalent to the automatic generation of rules from examples. In a typical rule-based system, rules are acquired in the following steps.

1. A set of examples is given.
2. Make a set of instance rules from the set of examples. An instance rule is either the same as an example or a simple transformation of an example.
3. The learning engine *generalizes* a set of rules from the set of instance rules. Obtained rules are more general than instance rules, and the size of the learned set of rules is much smaller than the size of the set of instance rules.

The key operation of rule learning is generalization: it relaxes the applicability condition of a rule, and extends the coverage. Generalization in rule learning is done with change of the description of a rule, because the applicability condition of a rule is explicitly represented on the rule. In other words, a rule is an intermediate



(a) Rule-Based Approach



(b) Example-based Approach

Figure 5.2: Explicit and Inplicit Generalization

representation which holds a result of a generalization. The use of rules makes the inference engine simple: it works with exact matching (Figure 5.2(a)).

Generalization is done also in example-based reasoning, but it is done implicitly: i.e. without change of description. It is done as part of the matching process by the inference engine, which adapts the best matched example to the given input. In other words, generalization of an example is done with change of interpretation (Figure 5.2(b)).

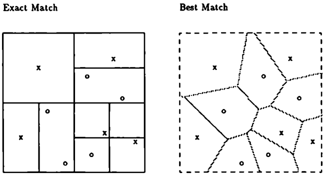
5.1.2 Exact Match versus Best Match

The use of exact match versus best match in interpretation is one of the major differences between RBA and EBA.

In exact matching, the interpreter determines whether the condition of the rule matches the given input or not: the output of the matching is '1(match)' or '0(not match)'. The description for conditions of rules usually has some variables. These variables have also explicit boundaries to match or not. Roughly speaking, the input space is divided into some subspaces by explicit boundaries. Individual subspaces are coverage of individual rules, and their boundaries are explicitly described on the individual rules. (Figure 5.3(a))

In best match, the interpreter first computes how well each example matches the given input: the output of the matching is a numerical value between '0(not match)' and '1(exact match)'. Then the interpreter adapts the best matched example to the given input. In the sense that the system determines an example to adapt, each example applies to a subset of the space of possible inputs. But its boundary is implicit: how large a subspace an example covers is not represented on the individual example, it depends on the whole examples in the database (Figure 5.3(b)).

This difference leads to the difference of efficiency on learning and task execution. In RBA, the efficiency of execution is relatively good, but the efficiency of learning is not so good. In the incremental learning process, the system has to change some existing rules in order to cover a new given example, and it needs much computation. In contrast, the efficiency of learning is very good in EBA: the system just stores a new given example in the database in the incremental learning



- Output of matching : $\{0, 1\}$
- Explicit boundaries
- Movement of boundaries with change of descriptions of rules

- Output of matching : $[0, 1]$
- Implicit boundaries
- Automatic movement of boundaries by adding new examples

Figure 5.3: Exact Match vs Best Match

In this figure, o meens a positive example and x meens a negative example.

process. The boundaries of coverage of individual examples move automatically.

5.1.3 Rule Learning as Compilation of Similarity into Rules

What is it in EBA that corresponds to *rule learning*?

Case 1: Guide information for generalization is given

In some cases, guide information or bias for generalizing rules is given to the system. Typical guide information is a generalization tree or conceptual hierarchy. In this case, the diagram of rule learning is the following.

$$\text{Examples} + \text{Generalization Tree} + \text{Heuristic} \ \Rightarrow\ \text{Rules} \qquad (5.1)$$

What information does the generalization tree bring to the system? From the viewpoint of EBA, it is some kind of similarity measure. In typical generalization trees, mother-daughter relations represent *is-a* relations between concepts, and sister relations represent similarity relations between concepts. We can easily define the distance between concepts on the generalization tree.

This discussion brings us the fact that we can construct example-based reasoning system with the combination of examples and a generalization tree. Now, we can clearly find out what is rule learning: it is compilation of similarity or distance measure into rules. Although a given set of examples is independent of a generalization tree, the learned set of rules is not independent of the generalization tree. Typically, the generalization tree becomes the hierarchy of variables which are used for describing the rules.

Case 2: Guide information for generalization is not given

In another case, guide information for generalizing rules is not given to the system. Takuma II falls under this case. The diagram of this case is the following.

$$\text{Examples} + \text{Heuristic} \ \Rightarrow\ \text{Rules} \qquad (5.2)$$

This learning process can be divided into the following two steps.

$$\text{Examples} + \text{Heuristic 1} \ \Rightarrow\ \text{Generalization Tree} \qquad (5.3)$$

$$\text{Examples} + \text{Generalization Tree} + \text{Heuristic 2} \Rightarrow \text{Rules} \qquad (5.4)$$

In these diagrams, Heuristic 1 is used to calculate similarity. In short, the similarity or distance on the task domain is not given in the case. Only general heuristic to calculate similarity or distance is given. The system first calculates the similarity or distance on the given task domain, and second compiles it into rules.

5.1.4 Discussion

Both rule learning and example-based reasoning extract the same information from the examples given. The major difference is whether it is represented it a explicitly with a symbol description or not. Now, we have reached the final question: Is it easy or useful to describe such information explicitly as symbol descriptions?

The ease of this depends on the task domain. It is easy in artificial domains; e.g. mathematics. These domains consist of clearly defined concepts and clear relations between concepts, therefore we can find the regularity on them and construct the theory for them. But it is difficult on natural domains; e.g. language translation. These domains have no clearly defined concepts and no clear relations between concepts, therefore it is very difficult for us to find the regularity on them and construct the theory for them.

We introduce two terms, *strength of regularity* and *abstraction level*, to distinguish domains. In some domains, there are a few principles which cover the whole domain, and few exceptions. In other domains, there are no principles which cover the whole domain, and many individual rules or exceptions. We call the former domain one with *strong regularity*, and the latter domain one with *weak regularity*.

The strength of regularity also depends on abstraction level. Abstraction is done to the direction in which regularity becomes stronger. The purpose is to find the major principles or regularities over the domain by ignoring unimportant details. From this viewpoint, abstraction is to make explanations for phenomena in the domain.

An importance advantage of describing the regularity explicitly is to make easy for human to understand it. We have worked to capture the regularity

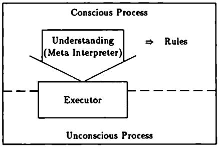


Figure 5.4: Executing and Understanding

of phenomena and to describe them by a few rules under the name of *science*. From this viewpoint, to give up describing regularities explicitly is to give up the scientific approach. From the viewpoint of software engineering, however, if we can construct an executor for the task without representing abstractions, it is useful. The task executor need not always use explicitly represented rules. For example, does human being use explicitly represented for all tasks in the brain?

Based on the discussion, the author draws the diagram in Figure 5.4. To make explicit rules is done by the meta interpreter: it is a conscious process which serves to understand phenomena and to make explanations. On the other hand, task execution is done as an unconscious process. They are independent of each other.

The author thinks that traditional rule learning roughly corresponds to the former. This explains why rule learning does not succeed well in natural domains. Rule learning may be powerful in the domains which have strong regularity, but is not powerful for domains which have weak regularity. Example-based reasoning is better suited to such domains. The author concludes that example-based reasoning is more promising method than rule learning for constructing machine translation systems.

Table 5.1 shows a summary of this section.

Approach	Rule-Based	Example-Based
General features	Explicit rules Explicit learning	Implicit rules Implicit learning
Learning		
Input	Examples	Examples
Main engine	Heuristic	(Organization of examples)
Purpose	Make explanation	Make executor
Method	Compile similarity into rules	Store examples
Output	Rules	—
Efficiency	Low	High
Type	Conscious learning	Unconscious learning
Execution		
Interpreter	Exact match	Best match
Knowledge source	Rules	Examples (+ Similarity)
Efficiency	High	Low (on traditional sequential machine)

Table 5.1: Summary: The Rule-Based Approach vs the Example-Based Approach

5.2 Example-Based Translation Family

In the previous section, the author concluded that the example-based approach is more promising than the rule-based approach for constructing machine translation systems. Two example-based translation systems, MBT1 and MBT2, have been presented, and some other example-based translation systems have been proposed by other researchers. They form the *Example-based Translation Family*, and can be divided into three groups: *Translation Aid Systems*, *Word Selection Systems*, and *Full Translation Systems*. In this section, we discuss each of them and clarify the current status and the future direction.

5.2.1 Translation Aid Systems

The first group in the example-based translation family is *Translation Aid Systems*: they are tools to assist human translators. Suppose that we have to translate some sentences or some texts. If we can obtain similar sentences or texts and their translations which can be a reference to the sentence or text we are going to translate, we can easily translate new sentences by editing them. Figure 5.5 shows the basic configuration of an example-based translation aid system. It is a kind of retrieval system: the task of the system is to retrieve texts similar to the texts given by the user.

There are some variations in the size of the text. The smallest size of the text is a word. In this case, the system is a standard retrieval system for a dictionary between two languages. But in case the text is a phrase or a sentence, we need a flexible retrieval system, because the given input may not exactly match any entries in the database. The system has to retrieve most similar entry to the given input.

ETOC [Sumita & Tsutsumi 88] is the first implemented system in this group. An entry (example) in ETOC is a pair of a Japanese sentence and an English sentence. Japanese sentences are analyzes morphologically. The retrieval mechanism is based on syntax-matching driven by generalization rules. ETOC first analyzes the given Japanese sentence morphologically. Next, ETOC retrieves some sentences which match the analyzed sentence. If it succeeds, ETOC shows

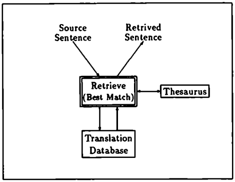


Figure 5.5: Basic Configuration of Example-based Translation Aid System

the sentences and their translations. If it fails, ETOC generalizes the sentence by applying generalization rules, which typically replace some words with variables, and retrieves some sentences which are match the generalized sentence pattern.

Nakamura implemented another Japanese-English example-based translation aid system [N. Nakamura 89]. In his system, Japanese sentences are analyzed morphologically and context words are extracted. Extracted context words from a sentence are used as retrieval keys for the sentence. In the retrieval phase, the system first extracts context words from a given input phrase or sentence. The best match is judged by the size of the intersection between context words extracted from the given input and keys of the example.

These two approaches show that there are two best match methods: one is syntax-oriented best match, another is semantic-oriented best match. How to determine the best match seems to be an open problem. We have to study it for not only Japanese but also the other language.

Another problem is the form of examples. ETOC and Nakamura's system store non-structured sentences. But sentences have structure. The best match between flat sentences has some limitations. To overcome these limitations, we

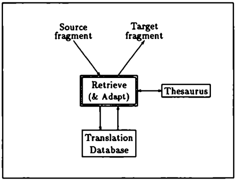


Figure 5.6: Basic Configuration of Example-based Word Selection System

need structured sentence examples.

Bilingual Knowledge Bank (BKB) is a database for bilingual structured texts [Sadler 89]. ATR is developing another structured database [Ehara et al 90]. These databases and their retrieval systems can be used as translation aid systems.

It seems likely that example-based translation aid systems will be practically usable in few years. But to find some better methods for best match and for organization of the database are still open problems.

5.2.2 Word Selection System

The second group in the example-based translation family is *Word Selection Systems*: they perform word selection tasks or target pattern selection tasks in the transfer phase as subsystems in whole translation systems. Figure 5.6 shows the basic configuration of an example-based word selection system.

There are currently two systems in this group, MBT1 (Chapter 3) and EBMT [Sumita et al 90]. MBT1 performs the word selection task in the translation between verb frame instances. EBMT selects the best English translation pattern

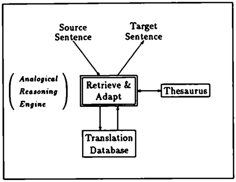


Figure 5.7: Basic Configuration of Full Example-based Translation System

for the Japanese "$noun_1$ NO $noun_2$" construction. The main problem is not on the mechanism side, but on the application side; i.e. how to cut off a subtask from the whole translation task and how to construct a database suited for the subtask.

ATR has some plans to use example-base word selection systems in its translation system for spoken telephone dialogue [Sumita et al 90]. How to combine example-based word selection subsystems with a traditional machine translation system is a new interesting problem.

5.2.3 Full Translation System

The third type of the example-based translation family is *Full Translation System*, which covers whole translation process. It is not fully implemented. MBT2 (Chapter 4) is nearest to this type, but it performs only the transfer task; it does not cover parsing and generation phases. Sadler presents a translation simulation, but it is not implemented [Sadler 89]. ATR is also studying this type of system [Furuse et al 90]. Figure 5.7 shows a basic configuration of a full example-based translation system.

The most difficult problem in implementing this type of system is how to adapt some similar translation examples to a given source sentence and make a appropriate target sentence. This is a general problem in analogical reasoning. It is partially solved in MBT2, but we need to study more.

The author plans to implement a system which covers the whole translation process [Sato 91]. The key technology in realizing it is parallel processing: i.e. cooperative problem solving by distributed agents. In example-based translation, knowledge source is distributed into individual translation examples or translation units. The author believes that example-based translation is suited for parallel processing. This will be able to solve the computation problem of example-based translation.

Chapter 6

Conclusions

Machine translation systems are among the largest and most complicated expert systems. In order to construct a machine translation system we need to encode dictionaries and many rules for parsing, transfer and generation. This is very difficult and time-consuming task, and it is a serious bottleneck in constructing and improving machine translation systems. There are two direction to solve this problem: example-based learning and example-based reasoning. This thesis has described both example-based learning and example-based reasoning for machine translation.

First, the example-based rule learning method was investigated. Chapter 2 described a method for learning a set of rules for language translation from positive and negative examples. Learning language translation is categorized as *learning to perform a multiple-step task*, which is the most difficult class of machine learning problem. The author showed that the formalism of *translation grammar* and its learning algorithm make it possible to learn language translation. The proposed method of learning automatically learns from positive and negative examples, and guarantees that the obtained translation grammar satisfies all given examples. The author implemented a machine learning/translation system, called Takuma II, which is the first learning system for natural language translation. Experiments in constructing English-Japanese translation grammars showed that the system can discover the correspondences of words, words groups, and phrase structures between two languages, and represent them in a translation grammar. However, this also showed that there are some open problems precluding real application.

Second, example-based reasoning methods were investigated. Example-based reasoning frees us from the need for rule acquisition, because it directly uses examples in the reasoning process. In this framework, we can construct translation systems simply by collecting translation examples, and improve them by adding translation examples.

Chapter 3 described the first prototype of an example-based translation system, MBT1, which can solve the word selection problem for translation between verb frame instances. This method consists of three components: the translation database, the definition of metric, and the translation process. The translation database is the collection of translation examples. A translation example is a pair of verb-frame instances. A verb frame instance is one verb with several nouns as its arguments. The metric was defined, which measures the 'distance' between a translation candidate and a translation example in the database. In the translation process, MBT1 generates all candidate translations. For each candidate, MBT1 retrieves the most similar translation example and computes the score of the candidate based on the above metric. MBT1 uses the score to evaluate the correctness of the candidate. MBT1 was implemented in English-Japanese translation and the experiment showed how well MBT1 solves the word selection problem. The major restriction of MBT1 is that it requires a fixed-format database and can not manage free-form data like sentences which have optional elements. Still MBT1 can be applicable to other subtasks in machine translation. After MBT1 was proposed, Sumita [Sumita et al 90] applied an MBT1-like method to the translation of the Japanese "$noun_1$ NO(の) $noun_2$" construction. These systems can be combined into whole machine translation systems, by serving as sub-systems for word selection tasks.

Chapter 4 described the second prototype of an example-based translation system, MBT2. It can transfer full sentences represented by word-dependency trees. A key problem in the implementation is how to utilize more than one translation example for translating a source sentence. This problem arises from the fact that a long sentence is too large to be matched by one translation example. It is a critical problem for example-based translation, and the author showed a solution for it in MBT2. The author introduced the representation, called the *matching expression*, which represents the combination of fragments of translation examples. The translation process consists of three steps: (1) Make the source matching expression from the source sentence. (2) Transfer the source matching expression into the target matching expression. (3) Construct the target sentence from the target matching expression. This mechanism generates some candidate translations. To select the best translation from them, the score of a translation was defined. The author implemented MBT2 for English-Japanese translation and demonstrated its ability. Although MBT2 covers only the transfer phase, it can be extended to cover the whole translation process. The proposed method will be used as a basic method to implement a complete example-based translation system. MBT2 inherits some advantages from the example-based translation idea: it is easy to construct and upgrade the system, to produce high quality translation, and to produce an explanation why the system generates a translation output. The major disadvantage of MBT2 is that it needs a great deal of computation. But parallel computation will overcome this problem.

Chapter 5 discussed the relations and differences between rule-based approach and example-based approach from the viewpoint of learning. The major differences are: (1) whether or not they use rules as an intermediate representation which holds the results of generalization, and (2) whether they use exact match reasoning or best match reasoning. Rule learning corresponds to understanding or making explanations for some phenomena in a task, and example-based reasoning corresponds to constructing a task executor. The example-based approach seems more promising method than the rule-based approach for constructing machine translation systems. Second, the example-based translation family was discussed. It can be divided into three groups: translation aid systems, word selection systems, and fully translation systems. Their current status and future prospects were discussed.

Appendix A

Program for Decomposition of MBT2

```
% decomp(+EWD,+ME)
decomp([Node|Children2],[ID1|DifList]) :-
  translatable_tree([ID1,Node|Children1]),
  decomp_sub(Children1,Children2,ID1,[],XDifList),
  diflist_simplify(XDifList,DifList).

decomp_sub([],[],_,_,[]) :-!.

decomp_sub([],[X|Rest],P,DelList,DifList) :-
  !,translatable_tree([P|_]),
  decomp_sub_add([X|Rest],P,DelList,DifList).

decomp_sub_add([],_,_,[]).
decomp_sub_add([X|Rest],P,DelList,[[a,P,ME]|DifList]) :-
  decomp_sub_add_check_dellist(X,DelList),
  decomp(X,ME),
  decomp_sub_add(Rest,P,DelList,DifList).
```

```
decomp_sub_add_check_dellist(_,[]).
decomp_sub_add_check_dellist([Node2|_],[Node1|_]) :-
  node_replaceable(Node1,Node2),!,fail.
decomp_sub_add_check_dellist(X,[_|Y]) :-
  decomp_sub_add_check_dellist(X,Y).

decomp_sub([X|Rest],Children2,P,DelList,DifList) :-
  decomp_sub2(X,Children2,P,DelList,DifList1,XDelList,C2),
  decomp_sub(Rest,C2,P,XDelList,DifList2),
  append(DifList1,DifList2,DifList).

decomp_sub2([ID1,Node|Rest1],Children2,P,DelList,DifList,[],C2) :-
  \+ translatable_tree([ID1|_]),!,
  decomp_sub21([ID1,Node|Rest1],Children2,P,DelList,DifList,C2).

decomp_sub21([ID1,Node|Rest1],[[Node|Rest2]|Rest3],_,_,
                DifList,Rest3) :-
  decomp_sub(Rest1,Rest2,ID1,[],DifList).
decomp_sub21(X,[Y|Rest3],P,DelList,[[a,P,ME]|DifList],C2) :-
  decomp_sub_add_check_dellist(Y,DelList),
  decomp(Y,ME),
  decomp_sub21(X,Rest3,P,DelList,DifList,C2).

decomp_sub2([ID,Node|_],Children2,P,DelList,DifList,[],C2) :-
  decomp_sub_replace(ID,Node,Children2,P,DelList,DifList,C2).

decomp_sub_replace(ID1,Node1,[[Node2|Rest2]|Rest3],_,_,
                        [[r,ID1,ME]],Rest3) :-
  node_replaceable(Node1,Node2),
  decomp([Node2|Rest2],ME).

decomp_sub_replace(ID1,X,[Y|Z],P,DelList,[[a,P,ME]|DifList],A) :-
```

```
  translatable_tree([P|_]),
  decomp_sub_add_check_dellist(Y,DelList),
  decomp(Y,ME),
  decomp_sub_replace(ID1,X,Z,P,DelList,DifList,A).

decomp_sub2([ID,Node|_],Children2,_,DelList,[[d,ID]],
             [Node|DelList],Children2).

node_replaceable([_,Cat1],[_,Cat2]) :-
  cat_replaceable(Cat1,Cat2).

cat_replaceable(X,X) :-!.
cat_replaceable(X,Y) :-
  cat_rep_data(X,Y),!.
cat_replaceable(X,Y) :-
  cat_rep_data(Y,X).

cat_rep_data(n,pron).
cat_rep_data(adj,det).

diflist_simplify([],[]).
diflist_simplify([[r,ID1,[ID1|DifList]]|Rest],NewDifList) :-
  !,
  diflist_simplify(Rest,DifList2),
  append(DifList,DifList2,NewDifList).
diflist_simplify([X|Rest],[X|Rest2]) :-
  diflist_simplify(Rest,Rest2).
```

Bibliography

[Biermann & Feldman 72] Biermann, A. W. and Feldman, J. A.: A Survey of Results in Grammatical Inference, in Watanabe, S. (Eds.), *Frontiers of Pattern Recognition*, Academic Press, pp31–54, 1972.

[DARPA 89] Case-based Reasoning from DARPA: Machine Learning Program Plan, *Proc. of Case-based Reasoning Workshop 89*, Morgan Kaufmann Publishers, pp1–13, 1989.

[Dietterich et al 82] Dietterich, T. G., London, B., Clarkson, K. and Dromey, G: Learning and Inductive Inference, in Cohen, P. and Feigenbaum, E. (Eds.), *Handbook of Artificial Intelligence Vol. III*, Pitman, pp323–512, 1982.

[Ehara et al 90] Ehara, T., Inoue, N., Kohyama, H., Hasegawa, T., Shohyama, S. and Morimoto, T.: Contents of the ATR Dialogue Database, ATR Technical Report, TR-I-0186, ATR interpreting Telephony Research Laboratories, 1990.

[Fu 74] Fu, K. S.: *Syntactic Methods in Pattern Recognition*, Academic Press, 1974.

[Fu & Booth 75] Fu, K. S. and Booth, T. L.: Grammatical Inference: Introduction and Survey, *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-5, pp95–111, 409–423, 1975.

[Furuse et al 90] Furuse, O., Sumita, E. and Iida, H.: A method for realizing Transfer-Driven Machine Translation, (in Japanese), *IPSJ-WGNL*, 90-8, 1990.

[Gold 67] Gold, E. M.: Language Identification in the Limit, *Information and Control*, Vol. 10, pp447–474, 1967.

[Hall 89] Hall, R.H.: Computational Approaches to Analogical Reasoning: A Comparative Analysis, *Artificial Intelligence*, Vol. 39, pp39–120, 1989.

[Hillis 85] Hillis, D.: *The Connection Machine*, The MIT Press, Cambridge Massachusetts, 1985.

[Knobe & Knobe 76] Knobe, B. and Knobe, K.: A Method for Inferring Context-free Grammars, *Information and Control*, Vol. 31, pp129–146, 1976.

[Maruyama & Watanabe 87] Maruyama, H. and Watanabe, H.: Integrating Natural Language Interface with Existing Menu Systems, (in Japanese), *Proc. of 4th Conference of JSSST*, B-1-5, pp63–66, 1987.

[McArthur 81] McArthur, T.: *Longman Lexicon of Contemporary English*, Longman Group Limited, 1981.

[Nagao 83] Nagao, M.: *Methods of Image Pattern Recognition*, (in Japanese), Corona Publishing Co., Ltd., 1983.

[Nagao 84] Nagao, M.: A Framework of a Mechanical Translation between Japanese and English by Analogy Principle, in Elithorn, A. and Barnerji, R. (Eds.), *Artificial and Human Intelligence*, North-Holland, pp173–180, 1984.

[Nagao and Tsujii 86] Nagao, M. and Tsujii, J.: The Transfer Phase of the Mu Machine Translation System, *Proc of COLING-86*, pp97–103, 1986.

[Nagao et al 85] Nagao, M., Tsujii, J. and Nakamura, J.: The Japanese Government Project for Machine Translation, *Computational Linguistics*, Vol. 11, No. 2-3, pp91–110, 1985.

[J. Nakamura 88] Nakamura, J.,: A Software System for Machine Translation, Doctoral Thesis, Kyoto University, 1988.

[N. Nakamura 89] Nakamura, N.: Translation Support by Retrieving Bilingual Texts, (in Japanese), *Proc. of 38th Convention of IPSJ*, pp357-358, 1989.

[NLRI] National Language Research Institute: *Word List by Semantic Principles*, (in Japanese), Syuei Syuppan, 1964.

[Ohsuga 86] Ohsuga, S.: Knowledge Acquisition and Learning, (in Japanese), *Information Processing*, Vol. 26, No. 12, pp1520-1528, 1985.

[Quinlan 83] Quinlan, J. R.: Learning Efficient Classification Procedures and their Application to Chess End Games, in Michalski, Carbonell & Mitchell (Eds.), *Machine Learning*, Tioga Publishing Company, pp463-482, 1983.

[Schank 82] Schank, R.C.: *Dynamic Memory*, Cambridge University Press, 1982.

[Sadler 89] Sadler, V.: *Working with Analogical Semantics: Disambiguation Techniques in DLT*, Foris Publications, Dordrecht Holland, 1989.

[Sato 91] Sato, S.: Towards Massively Parallel Implementation of Example-Based Translation, (in Japanese), *IPSJ-WGAI*, 77-16, pp139-147, 1991.

[Sato & Nagao 87] Sato, S. and Nagao, M.: A Method for Categorizing Words and Constructing a Grammar by Adjacency Matrix, (in Japanese), *Proc. of 1st Convention of JSAI*, pp89-92, 1987.

[Shapiro 82] Shapiro, E. Y.: *Algorithmic Program Debugging*, The MIT Press, 1982.

[Stanfill & Waltz 86] Stanfill, C. and Waltz, D.: Toward Memory-based Reasoning, *Communications of the ACM*, Vol.29, No.12, pp1213-1228, 1986.

[Sumita & Tsutsumi 88] E. Sumita and Y. Tsutsumi: A Translation Aid System Using Flexible Text Retrieval Based on Syntax-Matching, TRL Research Report, TR-87-1019, Tokyo Research Laboratory, IBM, 1988.

[Sumita et al 90] Sumita, E., Iida, H. and Kohyama, H.: Translating with Examples: A New Approach to Machine Translation, *Proc of the Third International Conference on Theoretical and Methodological Issues in Machine Translation of Natural Language*, 1990.

[Watanabe 90] Watanabe, H.: A Model of a Transfer Process Using Combinations of Translation Rules, *Proc. of PRICAI '90*, pp215–220, 1990.

[Winston 77] Winston, P.H.: *Artificial Intelligence*, Addison-Wesley Publishing Company, 1977.

List of Major Publications

[1] Sato, S. and Nagao, M.: A Method for Learning Language Translation Grammar based on Grammatical Inference, (in Japanese), *Computer Software*, Vol. 4, No. 4, pp56-68, 1987.

[2] Sato, S. and Nagao, M.: Toward Memory-based Translation, *Proc. of COLING-90*, Vol. 3, pp247-252, 1990.

[3] Sato, S.: MBT1: Example-Based Word Selection, (in Japanese), *Journal of Japan Society for Artificial Intelligence*, Vol.6, No.4, pp592-600, 1991.

[4] Sato, S.: MBT2: A Method for Combining Fragments of Examples in Example-Based Translation, (in Japanese), *Journal of Japan Society for Artificial Intelligence*, to appear.

List of Other Publications

[1] Sato, S. and Nagao, M.: Learning of Mal-Rules in Intelligent Tutoring System, (in Japanese), *Proc. of 30th Convention of IPSJ*, pp1437-1438, 1985.

[2] Sato, S. and Nagao, M.: A Method for Learning Rules for Machine Translation, (in Japanese), *Proc. of 32th Convention of IPSJ*, pp1273-1274, 1986.

[3] Sato, S. and Nagao, M.: A Method for Learning Language Translation Grammar based on Grammatical Inference, (in Japanese), *IPSJ-WGAI*, 46–11, 1986.

[4] Ikeda, Y., Sato, S., Tsujii, J. and Nagao, M.: Unification based Parser (KGW), (in Japanese), *Proc. of 34th Convention of IPSJ*, pp1181-1182, 1987.

[5] Sato, S.: A Method for Clustering Words and Constructing a Grammar by Adjacency Matrix, (in Japanese), *JCSS-SIGLAL*, 87-1-(3), pp12-19, 1987.

[6] Sato, S. and Nagao, M.: A Method for Categorizing Words and Constructing a Grammar by Adjacency Matrix, (in Japanese), *Proc. of 1st Convention of JSAI*, pp89-92, 1987.

[7] Sato, S. and Nagao, M.: A Method for Categorizing Words and Constructing a Grammar by Adjacency Matrix, (in Japanese), *Proc. of 35th Convention of IPSJ*, pp1867–1868, 1987.

[8] Sato, S. and Nagao, M.: Memory-based Translation, (in Japanese), *IPSJ-WGNL*, 70-9, 1989.

[9] Sato, S.: Memory-based Translation vs Rule Learning, (in Japanese), *Proc. of Workshop on Learning '89*, pp74–86, 1989.

[10] Sato, S. and Nagao, M.: Memory-based Translation, (in Japanese), *Proc. of 38th Convention of IPSJ*, pp333–334, 1989.

[11] Sato, S.: Toward New Generation Intelligent System, (in Japanese), *Proc. of First Sony Award for New Computers*, pp13–27, 1989.

[12] Sato, S.: Another Choice for Machine Learning, (in Japanese), *IPSJ-WGAI*, 65–1–3–3, 1989.

[13] Sato, S.: Memory-based Approach versus Rule Learning, (in Japanese), *Proc. of IPSJ Symposium of Learning Paradigm and its Application*, pp69–77, 1989.

[14] Sato, S.: Reconsideration of Symbolic Learning, (in Japanese), *Proc. of IPSJ Symposium of Learning Paradigm and its Application*, pp137–138, 1989.

[15] Sato, S.: Memory-based Translation II, (in Japanese), *Proc. of Workshop on Learning '91*, pp59–76, 1990.

[16] Sato, S.: Memory-based Translation II, (in Japanese), *IPSJ-WGAI*, 70-3, 1990.

[17] Sato, S.: Is It Essential for Machine Learning Systems to Interact Environments?, (in Japanese), *IPSJ-WGAI*, 71-1-15, 1990.

[18] Sato, S.: Challenge of Memory-based Reasoning, (in Japanese), *Proc. of JCSS Symposium '90*, pp1–10, 1990.

[19] Sato, S.: Analogical Reasoning in Example-Based Translation, (in Japanese), ICOT Technical Memorandum, TM-1049, 1991.

[20] Sato, S.: Towards Massively Parallel Implementation of Example-Based Translation, (in Japanese), *IPSJ-WGAI*, 77-16, pp139-147, 1991.

[21] Sato, S.: Example-Based Translation Approach, *Proc. of International Workshop on Fundamental Research for the Future Generation of Natural Language Processing*, ATR Interpreting Telephony Research Laboratories, pp1-16, 1991.

Abbreviations

COLING International Conference on Computational Linguistics

ICOT Institute for New Generation Computer Technology

IPSJ Information Processing Society of Japan

- **WGAI** Artificial Intelligence
- **WGNL** Natural Language

PRICAI Pacific Rim International Conference on Artificial Intelligence

JCSS Japan Cognitive Science Society

- **SIGLAL** Learning and Language

JSAI Japan Society for Artificial Intelligence

JSSST Japan Society for Software Science and Technology