Example-Based Translation of Technical Terms

Satoshi SATO
Japan Advanced Institute of Science and Technology, Hokuriku
Asahidai 15, Tatsunokuchi, Nomi, Ishikawa, 923-12, JAPAN
sato@jaist-east.ac.jp

Abstract

This paper describes MBT3, a new Example-Based Translation system. MBT3 is designed for translation of technical terms, a difficult part of translation. A technical term is a noun phrase with several words in most cases. For handling translation of noun phrases, we propose a new database format, a new flexible matching mechanism, and a new translation controller. In the preliminary evaluation in the domain of Computer Science, the translation accuracy is 99% for known terms, 78% for unknown terms, and 90% in total.

1 Introduction

Example-Based Translation (EBT) [Sato 91] is a new paradigm of machine translation. The basic idea, which was proposed by Nagao as *Translation by Analogy* [Nagao 84], is very simple; translate a sentence by using translation examples of similar sentences. Several prototype systems have been proposed in this five years [Sato & Nagao 89, Sadler 89, Sato & Nagao 90, Sumita et al 90, Watanabe 90, Sumita & Iida 91, Furuse & Iida 92, Watanabe 92].

This paper describes MBT3 [Sato 93a], which is the latest system in a series of MBT by the author. MBT3 is designed for translation of technical terms, a difficult part of translation. Since a technical term is a noun phrase with several words in most cases, the system must have the following components.

1. A database that can store any type of noun phrase examples; *word lengths*, i.e. numbers of words, of noun phrases are arbitrary.
2. A flexible matcher that can handle matching between two noun phrases that have different word lengths.
3. A translation controller that can utilize more than one translation example to translate an input term.

These components have been partly studied in the research of MBT2 [Sato & Nagao 90]. This paper presents *better* and *practical* solutions for these components.

2 Translation of Technical Terms

Translating technical terms is a difficult part of translation because of lack of systematic rules. Here are some examples of technical terms in Computer Science, which were extracted from Iwanami Encyclopedic Dictionary of Computer Science [Nagao et al 90]. The following two English terms have the same word 'source' in the first position, but the word is translated into two different words in Japanese.

source language → $\underset{\text{source}}{\text{原始}}\ \underset{\text{language}}{\text{言語}}$
source coding → $\underset{\text{source}}{\text{情報源}}\ \underset{\text{coding}}{\text{符号化}}$

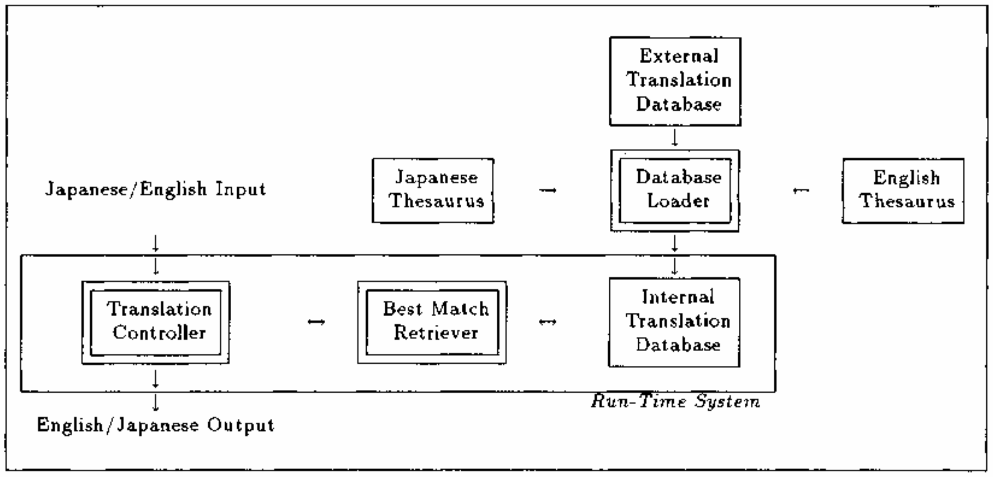

Figure 1: The Organization of MBT3

Japanese	*English*	*Correspondence*
上昇 型 構文 解析 法	bottom-up parsing method	1-5=1-3 1-2=1 3-5=2-3 3-4=2 5=3
木 の 走査	tree traversal	1-3=1-2 1=1 3=2

Figure 2: A Part of External Translation Database

In the reverse direction, although the following three Japanese terms have the same construction, their translations are different.

情報 (information)	検索 (retrieval)	→ information retrieval
情報 (information)	公害 (pollution)	→ pollution by information
情報 (information)	流通 (distribution)	→ distribution of information

The traditional solution for the problem is to store literal translations of technical terms into the bilingual dictionary. In the traditional framework with exact match reasoning, this solution is just a patch because a system do not translate un-stored terms. The flexible matching and adjustment mechanism of Example-Based Translation makes it possible to translate unknown terms.

3 Organization of MBT3

Figure 1 shows the organization of MBT3. The knowledge source of MBT3 is an *external translation database* and two *thesauri*; a Japanese thesaurus and an English thesaurus. These three are loaded by a database loader program into the run-time system. The run-time system of MBT3 consists of three modules; an *internal translation database*, a *best match retriever* and a *translation controller*. The details will be described in the following sections.

4 Translation Database

External Translation Database The external translation database is a collection of translation examples that is given to the system as major knowledge source. Figure 2 shows a part of the external translation database. A record of the database is a translation example that consists of three

ID	Previous	Focus	Next
#1		[$_{s1}$ 上昇 型] [$_{s2}$ 構文 解析 法]	
		[$_{s1}$ bottom-up] [$_{s2}$ parsing method]	
#2		[上昇 型]	構文 解析 法
		[bottom-up]	parsing method
#3	上昇 型	[$_{s1}$ 構文 解析] [$_{s2}$ 法]	
	bottom-up	[$_{s1}$ parsing] [$_{s2}$ method]	
#4	上昇 型	[構文 解析]	法
	bottom-up	[parsing]	method
#5	上昇 型 構文 解析	[法]	
	bottom-up parsing	[method]	
#6		[$_{s1}$ 木] [$_{s2}$ の] [$_{s3}$ 走査]	
		[$_{s1}$ tree] [$_{s3}$ traversal]	
#7		[木]	の 走査
		[tree]	traversal
#8	木 の	[走査]	
	tree	[traversal]	

Figure 3: Internal Translation Database

parts; Japanese, English, and correspondence part. A Japanese part is a list of Japanese words, an English part is a list of English words, and a correspondence part is a list of correspondent pairs. A correspondent pair is written as A-B=C-D, which means that the segment from A to B in a Japanese part corresponds to the segment from C to D in an English part. When $A = B$ or $C = D$, we simply write A or C.

Internal Translation Database The internal translation database is generated from the external one by the database loader program. An internal record is generated per a correspondent pair of external records. Figure 3 shows the internal translation database generated from the external database of Figure 2.

A record of the internal database consists of a Japanese part and an English part, and each part consists of three sub-parts: *focus*, *previous* and *next*. A *focus* is a Japanese or an English part of a translatable unit. A *previous* is the preceding context of a *focus* and a *next* is the following context of a *focus*. For example, #4 record in Figure 3 means that

- '構文 解析 (parsing)' in '上昇 型 構文 解析 法 (bottom-up parsing method)' can be translated into 'parsing'.
- 'parsing' in 'bottom-up parsing method' can be translated into '構文 解析 (parsing)'.

A *focus* part consists of singular or plural segments, which are enclosed by square brackets in Figure 3. There are two types of segments: A *translatable* segment has the correspondent segment in the target part, and an *un-translatable* segment has no correspondent segment. Subscripts after open brackets indicate the correspondence between segments of two languages. For example, in #6 record of Figure 3, the Japanese *focus* part consists of three segments; [$_{s1}$ 木 (tree)], [$_{s2}$ の (of)] and [$_{s3}$ 走査 (traversal)]. The first or the last is translatable because it has the correspondent segment, [$_{s1}$ tree] or [$_{s3}$ traversal], in the English *focus* part. The second segment is un-translatable in this case.

5 Best Match Retriever

The best match retriever finds the most similar records to the given retrieval query from the internal database. A retrieval query is given as a word sequence that is divided into three parts, *focus*, *previous* and *next*. At first, the system examines matching between the given retrieval query and every record

	Previous	Focus	Next
Query # 3	下降型 top-down 上昇型 bottom-up	構文解析 プログラム parsing program [$_{s1}$ 構文解析] [$_{s2}$ 法] parsing method	
B-list		[$_{s1}$ 構文解析] = [構文解析], [$_{s2}$ 法] $\approx$ [プログラム] parsing parsing method program	

	Previous	Focus	Next
Query #6		木 構造 tree structure [$_{s1}$ 木] [$_{s2}$ の] [$_{s3}$ 走査] tree of traversal	
		Fail.	

Figure 4: Examples of Matching

in the database. If the matching succeeds, the matching score is calculated for the next step. Finally the system selects the best N records that took the highest scores. Currently the value $N = 10$ is used.

5.1 Matching

The matcher examines whether a record matches a retrieval query or not by using matching constraints. In the matching process, only *focus* parts are considered, and *previous* and *next* parts are ignored. Because a focus part of a record consists of singular or plural segments, the matcher determines a *binding* of every segment in the focus. A binding of a segment indicates which words in the focus of the query the segment corresponds to. There are two types of bindings: exact match binding and replacement binding. The former indicates that correspondent segments are the literally same, and the latter indicates other cases.

The matching constraints are given as restriction of segment binding. They are:

1. A segment of a record must be bound to a non-empty segment of a query.
2. If a focus part of a record consists of singular segment, only exact match binding is permitted for the segment.
3. Only exact match binding is permitted for un-translatable segments.

The first constraint is introduced for the purpose of forbidding deletion of segments in the matching process; only replacement is permitted. The second constraint is needed to guarantee termination of translation. In the bottom level of translation, only exact matched examples can be used. The last constraint is needed to guarantee production of translation. The system cannot produce a translation when un-translatable segments are mismatched, because un-translatable segments have no information on correspondence or translation.

Figure 4 shows examples of matching. The matcher produces a list of bindings (b-list in short) when matching succeeds. In this figure, exact match binding is written as '=' and replacement binding is written as '$\approx$'.

5.2 Matching Score

When the matching succeeds, the matching score is calculated for the next step. In the matching process described in the previous subsection, only *focus* parts are considered. However, in the calculation of the matching score, *previous* and *next* parts (contexts) are also considered. The matching score S is the sum of the *focus* matching score S_F and the context matching score S_C.

$$S = S_F + S_C \tag{1}$$

Focus Matching Score The *focus* matching score is defined as follows. It is calculated from the b-list produced by the matcher.

$$S_F = f\left(\sum_k S_{b_k}\right) \tag{2}$$

where S_{b_k} is the score of a binding b_k in the b-list and f is the function given as follows.

$$f(x) = \begin{cases} x^2/100 & \text{if } x < 300 \\ 3x & \text{otherwise} \end{cases} \tag{3}$$

The function f is introduced for emphasizing better matching.

The score of a binding S_{b_k} is defined as follows.

$$b_k = [w_1 \dots w_n] \approx [v_1 \dots v_m] \tag{4}$$

$$S_{b_k} = dp_{n,m} \times \min(n,m)/\max(n,m) \tag{5}$$

$$dp_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \max\begin{pmatrix} dp_{i-1,j}, \\ dp_{i,j-1}, \\ dp_{i-1,j-1} + sim(w_i, v_j) \end{pmatrix} & \text{otherwise} \end{cases} \tag{6}$$

where $sim(w_i, v_j)$ is the word similarity between w_i and v_j. It is calculated by using Japanese or English thesaurus. The best score 100 is given when two words are exactly the same, while the worst score zero is given when two words have no relation[1]. In the above definition, $dp_{n,m}$ is calculated by a standard dynamic programming method. The factor $\min(n,m)/\max(n,m)$ is introduced to consider the difference between segment lengths, n and m. The maximum value is given when two lengths are the same.

Context Matching Score The context matching score is the sum of the *previous* matching score S_P and the *next* matching score S_N.

$$S_C = S_P + S_N \tag{7}$$

We only describe the *next* matching score here, because two matching scores are symmetry.

In the calculation of the *next* matching score, the first three words in the *next* context are considered, which is called *restricted next context.* For example, the restricted *next* context of #2 record is [構文 解析 法 (parsing method)]. When the *next* context is less than three words, it becomes the restricted *next* context automatically. For example, the restricted *next* context of #4 record is [法 (method)], and it of #1 is [].

The *next* matching score between two restricted *next* contexts, $[w_1 \dots w_n]$ and $[v_1 \dots v_m]$, is defined as follows.

$$S_N = \widehat{dp}_{n,m} \tag{8}$$

$$\widehat{dp}_{i,j} = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ \max\begin{pmatrix} \widehat{dp}_{i-1,j}, \\ \widehat{dp}_{i,j-1}, \\ \widehat{dp}_{i-1,j-1} + sim(w_i, v_j)/weight(i,j) \end{pmatrix} & \text{otherwise} \end{cases} \tag{9}$$

$$weight(i,j) = 2^{\max(i,j)} \tag{10}$$

The factor $weight(i,j)$ is introduced to change weights according to closeness to the *focus* part. The first word in the restricted *next* context is the most important of all, and the second word is secondarily important and so on. The best of the *next* matching score 87 is given when the two restricted *next* contexts are exactly the same; i.e. $w_i = v_j \quad (i = 1, 2, 3)$.

[1]See [Sato 93a] in details.

Rank	ID / Score	Previous	Focus	Next
1	5927 / 900 (0+900+0)		[$_{s1}$ 構文 解析] [$_{s2}$ プログラム] [$_{s1}$ syntactic analysis] [$_{s2}$ program]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ プログラム] = [プログラム]		
2	9108 / 576 (0+576+0)		[$_{s1}$ 構文 解析] [$_{s2}$ 表] [$_{s1}$ parsing] [$_{s2}$ table]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 表] ≈ [プログラム]		
3	8447 / 559 (75+484+0)	下降 型 top-down	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
4	8475 / 554 (70+484+0)	上昇 型 bottom-up	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
5	8554 / 536 (52+484+0)	予測 型 predictive	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
6	7679 /499 (15+484+0)	演算 子 順位 operator precedence	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
7	7690 / 499 (15+484+0)	再帰 的 下向き recursive descent	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
8	10649 / 489 (5+484+0)	ＳＬＲ SLR	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
9	10654 / 489 (5+484+0)	ＬＲ LR	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		
10	10654 / 489 (5+484+0)	ＬＡＬＲ LALR	[$_{s1}$ 構文 解析] [$_{s2}$ 法] [$_{s1}$ parsing] [$_{s2}$ method]	
	B-List =	[$_{s1}$ 構文 解析] = [構文 解析], [$_{s2}$ 法] ≈ [プログラム]		

Figure 5: Example of Best Match Retrieval

5.3 Examples of Best Match Retrieval

Figure 5 shows the result of the best match retrieval for the following query.

$$previous = [\underset{\text{top-down}}{\text{下降 型}}],\quad focus = [\underset{\text{parsing}}{\text{構文 解析}}\ \underset{\text{program}}{\text{プログラム}}],\quad next = []$$

6 Translation Controller

The translation controller controls the overall translation process that is execution of the translation procedure given as follows.

1. Receive a translation input.
2. Send it to the best match retriever as a retrieval query.
3. Receive the retrieval result from the best match retriever.
4. Make candidate target patterns from the retrieval result.
5. Select the best target pattern from the candidates.
6. Call itself recursively if the selected target pattern is partial.

Rank	Target Pattern Candidates	
	[[syntactic analysis] [program]]	[[parsing] [$\underset{\text{program}}{\text{プログラム}}$]]
1	900	-
2	-	576
3	-	$559 \times 0.1 = 55$
4	-	$554 \times 0.1 = 55$
5	-	$536 \times 0.1 = 53$
6	-	$499 \times 0.1 = 49$
7	-	$499 \times 0.1 = 49$
8	-	$489 \times 0.1 = 48$
9	-	$489 \times 0.1 = 48$
10	-	$489 \times 0.1 = 48$
Total	900	981

Figure 6: Accumulation of Evidence of Figure 5

7. Produce the final translation output.

We describe the details of step 4, 5, 6, and 7 in the rest of this section.

After receiving the retrieval result, the controller makes *candidate target patterns* from the result. A *target pattern* is a complete or partial translation produced by a retrieved record and its b-list. If all bindings in the b-list are exact match bindings, the target *focus* part of the record becomes the target pattern automatically. In this case, the target pattern will be complete. Otherwise the target *focus* part in which mismatched segments have been replaced by variables becomes the target pattern. In this case, the target pattern will be partial. In Figure 5, for example, the second retrieved record produces the target pattern '$[_{s1}$ parsing$]$ $[_{s2}$ $*s2]$' where $*s2$ is the variable to be filled by a translation of '[プログラム (program)]'. We also write it as '[[parsing] [プログラム (program)]]' in short. All target patterns produced by retrieved records become candidate target patterns.

The next step determines the best target pattern from the candidates. In case all retrieved records produce the same target pattern, it becomes the best target pattern automatically. Otherwise, the controller have to select the best one from the candidates. As a mean of the selection, we use *accumulation of evidence*. The *evidence* of each target pattern is accumulated by the following accumulation rules.

1. The best score of each target pattern is accumulated as the *evidence* of the target pattern.
2. 10% of the not-best score of each target pattern is accumulated as the *evidence* of the target pattern.

The value 10% (= 1/10) was determined by the value 10 that is the number of selected records in the best match retrieval. Figure 6 shows how records contribute to each target pattern in Figure 5. In this case, the target pattern '[[parsing] [プログラム (program)]]' is the winner, because the total *evidence* is larger, although it is not produced by the best retrieved record.

After the best target pattern is selected, the controller works for making the translation output. If the selected target pattern is complete, then it becomes the translation output automatically. Otherwise, the controller calls itself recursively to translate remaining segments in the target pattern. In the examples mentioned before, because the selected target pattern '[[parsing] [プログラム (program)]]' is partial, the controller works for translating 'プログラム (program)'; the next input of translation is

$$previous = [\underset{\text{top-down}}{\text{下降 型}}\ \underset{\text{parsing}}{\text{構文 解析}}],\quad focus = [\underset{\text{program}}{\text{プログラム}}],\quad next = []$$

Note that the segment '構文 解析 (parsing)' is used as a part of the *previous* context in the recursive call.

Figure 7 shows the overall translation process of '下降 型 構文 解析 プログラム (top-down parsing program)'. In this process, the translation procedure is called 4 times and the *focus* parts of their inputs are enclosed by square brackets. As a result of the first call, the system gets to know that a translation of '下降 型 構文 解析 プログラム (top-down parsing program)' can be constructed by a translation of the segment '下降 型 (top-down)' and a translation of the segment '構文 解析 プログラム (parsing program)'. The second and third are called to translate these segments, and the last is called as a subcall of the third.

7 Implementation

Software Implementation There are currently two implementations. This paper is based on the original MBT3 system that is written by C and runs on SparcStation2. There is a parallel version, MBT3n, which is written by C and runs on nCUBE2. The details of the MBT3n system is presented in [Sato 93b].

Database Implementation We collected 7057 bilingual technical terms from Iwanami Encyclopedic Dictionary of Computer Science [Nagao et al 90] and stored them into the external translation database. Its Japanese vocabulary is 3502 words and English vocabulary is 4086 words.

There are two methods for linking segments of bilingual examples; *head oriented linking* and *naive linking*. The former is based on pairings of dependency trees and it was used in MBT2. In MBT3, however, we use *naive linking* in which we link correspondent segments of bilingual examples intuitively. The reason why we use *naive linking* is that *head oriented linking* is too strict; only linguists can link correctly. *Naive linking* is easy enough for non-linguistic experts. Figure 8 shows some examples of linking by two methods. We linked 7057 bilingual examples by hand[2] and those linked examples produce 19667 internal records.

8 Translations by MBT3

This section demonstrates sophisticated translation ability of MBT3. Figure 9 presents several translations produced by MBT3 and a commercial MT system using the traditional framework. The first two source terms have the same construction, but the translation results by MBT3 are different; both are good translations. This kind of selection is very difficult for the traditional MT framework. While the third and fourth source terms have the similar constructions, the word '法 (method)' is explicitly translated into 'methods' in the third translation and is not explicitly translated in the fourth translation. This kind of selection is also very difficult for the traditional MT framework. The fifth is a translation example of a long term and the last is a translation example in the reverse direction.

9 Preliminary Evaluation

This section describes the result of preliminary evaluation, which has demonstrated good tractability and performance of MBT3. Table 1 shows the translation accuracy of 195 technical terms in Computer Science. This set includes 114 known terms stored in the translation database of the system. In this table, 'Error' means the case that incorrect translations are produced and 'Fail' means the case that no translation are produced. The result shows

- All known terms except one term are correctly translated. Note that the system does not guarantee correct translations of known terms, because it uses *accumulation of evidence*.
- The system shows good performance: The translation accuracy is
  - 78% for unknown terms.
  - 90% in total.

[2]It took about 12 hours.

```
MBT>下降型構文解析プログラム
*** Input ***
                                   [ 下降 型 構文 解析 プログラム ]
Retrieve...
*** Target Pattern Ranking ***
 1 : ( 1448) : [ [ 下降 型 ] [ 構文 解析 プログラム ] ]
 2 : ( 1371) : [ [ top-down ] [ 構文 解析 プログラム ] ]
 3 : (  749) : [ [ 下降 ] [ 型 ] [ 構文 解析 プログラム ] ]
*** Input ***
                                   [ 下降 型 ] 構文 解析 プログラム
Retrieve...
*** Target Pattern Ranking ***
 1 : (  561) : [ [ top-down ] ]
 2 : (  252) : [ [ 下降 ] [ type ] ]
 3 : (  158) : [ [ 下降 ] [ 型 ] ]
*** Input ***
                              下降 型 [ 構文 解析 プログラム ]
Retrieve...
*** Target Pattern Ranking ***
 1 : (  981) : [ [ parsing ] [ プログラム ] ]
 2 : (  900) : [ [ syntactic analysis ] [ program ] ]
*** Input ***
                 下降 型 構文 解析 [ プログラム ]
Retrieve...
*** Target Pattern Ranking ***
 1 : (  276) : [ [ program ] ]
 2 : (  117) : [ [ programs ] ]

*** Final Result ***
                                   [ 下降 型 構文 解析 プログラム ]
                                   [ top-down parsing program ]
```

Figure 7: Translation Process

	アクセス 時間 access time	計算 の 理論 theory of computation	拡張 文脈 自由 文法 augmented context free grammar
Head oriented linking	1-2=1-2 1=1	1-3=1-3 1-2=2-3 1=3	1-4=1-4 1=1 2-3=2-3 2=2
Naive linking	1-2=1-2 1=1 2=2	1-3=1-3 1=3 3=1	1-4=1-4 1=1 2-4=2-4 2=2 3=3 4=4

Figure 8: Head Oriented Linking vs. Naive Linking

Source Term	*Translation by MBT3*	*Translation by a MT*
プログラム の 設計 program of design	program design	a design of a program
構文 解析 プログラム の 設計 parsing program of design	design of parsing program	a design of a sentence structure analytic program
問題 解決 法 problem solving method	problem solving methods	a way of a problem solution
オブジェクト 指向 設計 object oriented design 法 method	object-oriented design	a way of a オブジェクト pointing towards design
神経 回路 網 の 並列 neural network of parallel 情報 処理 information processing	parallel information processing of neural network	parallel information processing of nervous circuit net
design of parsing programs for natural language	自然 言語 の 構文 解析 natural language of parsing プログラム の 設計 program of design	

Figure 9: Translations Produced by MBT3

Table 1: Translation Accuracy of 195 Terms

	Good	Error	Fail	Total
Known Terms	114 (99%)	1 (1%)	0	115
Unknown Terms	62 (78%)	16 (20%)	2 (3%)	80
Total	176 (90%)	17 (9%)	2 (1%)	195

Table 2: Improvement of Translation Accuracy

Adding 10 Examples				
	Good	Error	Fail	Total
Known Terms	119 (99%)	1 (1%)	0	120
Unknown Terms	69 (92%)	6 (8%)	0	75
Total	188 (96%)	7 (4%)	0	195

Adding 6 More Examples				
	Good	Error	Fail	Total
Known Terms	124 (98%)	2 (2%)	0	126
Unknown Terms	69 (100%)	0	0	69
Total	193 (99%)	2 (1%)	0	195

The system improves by adding new translation examples into the database. After finding lack of several translation units by the analysis of failures, we added new translation examples (bilingual terms) into the database. Adding 10 new examples corrected 12 failures out of 18 of unknown terms, and adding 6 more corrected 5 failures out of the rest 6 (Table 2). These improvement could be done by a few hours' work. Note that there were no side-effects of adding new examples; good translation remained as it was.

10 Concluding Remarks

This paper proposed a new Example-Based Translation system, MBT3, which is designed for translation of technical terms (noun phrases), a difficult part of translation. We implemented the system with the translation database of 7000 technical terms in Computer Science on SparcStation2. In the preliminary evaluation, the translation accuracy was 99% for known terms, 78% for unknown terms, 90% in total. By adding a dozen new translation examples into the database, we could easily improve the translation accuracy; 92% for unknown terms, 96% in total. The result demonstrates good tractability and high translation accuracy of MBT3.

MBT3 has several advantages. They are:

- No linguistic expert is required; we can make large translation databases with low cost. Systems such as MBT2 [Sato & Nagao 90], RCT [Watanabe 90], SimTran [Watanabe 92] and TDMT [Furuse & Iida 92] require linguistic experts, because these systems use pairs of parse trees or translation patterns as their knowledge sources, which are expensive.
- MBT3 requires far less time to implement. It took only 12 hours to make the database of 7000 terms and only a few hours to improve the translation accuracy.
- Translation speed of MBT3 can be accelerated by use of parallel machines. MBT3n, a parallel version of MBT3 on nCUBE2, has shown high performance and good scalability [Sato 93b].

An open problem in example-based translation of noun phrases is article selection. Selection of proper articles is a difficult problem in generation of English, because Japanese has no articles. The traditional framework with grammatical rules has not solved this problem completely.

The next step of our EBT research is translation of technical papers' titles. The titles are complicated noun phrase because they sometimes contain verbs as past participles, gerunds, and to-infinitives. A method for handling verbs in the EBT framework has to be studied.

References

[Furuse & Iida 92] Furuse, O. and Iida, H.: Cooperation between Transfer and Analysis in Example-Based Framework, *Proc. of COLING-92*, Vol. II, pp.645–651, 1992.

[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, pp.173–180, 1984. (*Proc. of the International NATO Symposium on Artificial & Human Intelligence*, 1981, Lyon, France.)

[Nagao et al 90] Nagao, M. et al (Eds.): *Iwanami Encyclopedic Dictionary of Computer Science*, (in Japanese), Iwanami Shoten, 1990.

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

[Sato 91] 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, pp.1–16, 1991.

[Sato 93a] Sato, S.: *Example-Based Translation of Technical Terms*, JAIST Research Report, IS-RR-93-4I, School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku, 1993.

[Sato 93b] Sato, S.: *MIMD Implementation of MBT3*, JAIST Research Report, IS-RR-93-5I, School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku, 1993. (Also in *Proc. of the Second International Workshop on Parallel Processing for Artificial Intelligence*, Chambery, France, to appear.)

[Sato & Nagao 89] Sato, S. and Nagao, M.: Memory-Based Translation, (in Japanese), IPSJ SIG Notes, SIGNL-70-9, Information Processing Society of Japan, 1989. (English version is in Sato, S.: *Example-Based Machine Translation*, Ph.D. Thesis, Kyoto University, 1991.)

[Sato & Nagao 90] Sato, S. and Nagao, M.: Toward Memory-Based Translation, *Proc. of COLING-90*, Vol. 3, pp.247–252, 1990.

[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*, Texas, pp.203–212, 1990.

[Sumita & Iida 91] Sumita, E., and Iida, H.: Experiments and Prospects of Example-Based Machine Translation, *Proc. of ACL-91*, 1991.

[Watanabe 90] Watanabe, H.: A Method for a Transfer Process Using Combinations of Translation Rules, *Proc. of PRICAI '90*, pp.215–220, 1990.

[Watanabe 92] Watanabe, H.: A Similarity-Driven Transfer System, *Proc. of COLING-92*, Vol. II, pp.770–776, 1992.