Learning Translation Templates from Examples[1]

H. Altay Güvenir and Ilyas Cicekli

Department of Computer Engineering and Information Sciences
Bilkent University, Ankara 06533, Turkey
e-mail: {guvenir,ilyas}@cs.bilkent.edu.tr

Abstract

This paper proposes a mechanism for learning lexical level correspondences between two languages from a set of translated sentence pairs. The proposed mechanism is based on an analogical reasoning between two translation examples. Given two translation examples, the similar parts of the sentences in the source language must correspond the similar parts of the sentences in the target language. Similarly, the different parts should correspond to the respective parts in the translated sentences. The correspondences between the similarities, and also differences are learned in the form of translation templates. The approach has been implemented and tested on a small training dataset and produced promising results for further investigation.

1 Introduction

Traditional approaches to machine translation (MT) suffer from tractability, scalability and performance problems due to the necessary extensive knowledge of both the source and the target languages. Corpus-based machine translation is one of the alternative directions that have been proposed to overcome the difficulties of traditional systems. Two fundamental approaches in corpus-based MT have been followed. These are *statistical* and *example-based* machine translation (EBMT), also called *memory-based* machine translation (MBMT). Both approaches assume the existence of a bilingual parallel text (an already translated corpus) to derive a translation for an input. While statistical MT techniques use statistical metrics to choose the most probable structures in the target language, EBMT techniques employ pattern matching techniques to translate subparts of the given input [1].

Exemplar-based representation has been widely used in Machine Learning (ML). According to Medin and Schaffer [8], who originally proposed exemplar-based learning as a model of human learning, examples are stored in memory without any change in the representation. The characteristic examples stored in the memory are called exemplars. The basic idea in exemplar-based learning is to use past experiences or cases to understand, plan, or learn from novel situations [5, 7, 11]. EBMT has been proposed by Nagao [9] as *translation by analogy* which is in parallel with memory based reasoning [14], case-based reasoning [12] and derivational analogy [2]. Example-based translation relies on the use of past translation examples to derive a translation for a given input [3, 10, 13, 15]. The input sentence to be translated is compared with the example translations analogically to retrieve the *closest* examples. Then, the fragments of the retrieved examples are translated and recombined in the target language. Prior to the translation of an input

[1]This research has been supported in part by NATO Science for Stability Program Grant TU-LANGUAGE.

sentence, the correspondences between the source and target languages should be available to the system; however this issue has not been given enough consideration by the current EBMT systems. Kitano has adopted the manual encoding of the translation rules, however this is a difficult and an error-prone task for a large corpus [6]. This paper formulates the acquisition problem as a machine learning task in order to automate the process.

Our first attempt was to construct parse trees between the example translation pairs [4]. However, the difficulty was the availability of a reliable parser for both languages. In this paper, we propose a technique which stores exemplars in the form of *templates* that are *generalized exemplars*, rather than parse trees. A template is an example translation pair where some components (e.g., words stems and morphemes) are generalized by replacing them with variables in both sentences, and establishing bindings between the variables. We will refer this technique as GEBMT for *Generalized Exemplar Based Machine Translation*.

The algorithm we propose here, for learning such templates, is based on a heuristic to learn the correspondences between the patterns in the source and target languages, from two translation pairs. The heuristic can be summarized as follows: Given two translation pairs, if the sentences in the source language exhibit some similarities, then the corresponding sentences in the target language must have similar parts, and they must be translations of the similar parts of the sentences in the source language. Further, the remaining differing constituents of the source sentences should also match the corresponding differences of the target sentences. However, if the sentences do not exhibit any similarity, then no correspondences are inferred. Consider the following translation pairs given in English and Turkish to illustrate the heuristic:

I gave the ticket to Mary ↔ Mary'e bileti verdim
I gave the pen to Mary ↔ Mary'e kalemi verdim .

Similarities between the translation examples are shown as underlined. The remaining parts are the differences between the sentences. We represent the similarities in English as [I gave the X^E to Mary], and the corresponding similarities in Turkish as [Mary'e X^T+i verdim]. According to our heuristic, these similarities should correspond each other. Here, X^E denotes a component that can be replaced by *any* appropriate structure in English and X^T refers to its translation in Turkish. This notation represents an *abstraction* of the differences "ticket" vs. "pen" in English and "bilet" vs. "kalem" in Turkish. Continuing even further, we infer that "ticket" should correspond to "bilet" and "pen" should correspond to "kalem"; hence learning further correspondences between the examples.

Our learning algorithm based on this heuristic is called TTL (for *Translation Template Learner*). Given a corpus of translation pairs, TTL infers the correspondences between the source and target languages in the form of templates. These templates can be used for translation in both directions. Therefore, in the rest of the paper we will refer these languages as L^1 and L^2. Although the examples and experiments herein are on English and Turkish, we believe the model is equally applicable to other language pairs.

The rest of the paper is organized as follows. Section 2 explains the representation in the form of translation templates. The TTL algorithm is described in Section 3. Section 4 illustrates the TTL algorithm on some example translation pairs. Section 5 describes how these translation templates can be used in translation. Section 6 concludes the paper.

2 Translation Templates

A template is a generalized translation exemplar pair, where some components (e.g., word stems and morphemes) are generalized by replacing them with variables in both sentences, and establishing bindings between these variables. For example, the translation template that would be learned from the example translations given above is:

$$\begin{aligned} \texttt{[I gave the } X^1 \texttt{ to Mary]} &\leftrightarrow \texttt{[Mary'e } X^2\texttt{+i verdim] if} \\ [X^1] &\leftrightarrow [X^2] \texttt{ .} \end{aligned}$$

This translation template is read as the sentence "I gave the X^1 to Mary" in L^1 and the sentence "Mary'e X^2+i verdim" in L^2 are translations of each other, given that X^1 in L^1 and X^2 in L^2 are translations of each other. Therefore, for example, if it has already been acquired that "basket" in L^1 and "sepet" in L^2 are translations of each other, i.e., `[basket]` $\leftrightarrow$ `[sepet]` then the sentence "I gave the basket to Mary" can be translated into L^2 as "Mary'e sepeti verdim".

Since the TTL algorithm is based on finding the similarities and differences between translation examples, the representation of sentences plays an important role. As it is, the TTL algorithm may use the sentences exactly as they can be found in a regular text. That is, no grammatical information or no preprocessing, e.g. bracketing, on the bilingual parallel corpus is needed. Therefore, it is a grammarless extraction algorithm for phrasal translation templates from bilingual parallel texts.

For agglutinative languages such as Turkish, this surface level representation of the sentences limits the generality of the templates to be learned. For example, the translation of the sentence "I am coming" in Turkish is a single word "geliyorum". When surface level representation is used, it is not possible to find a template from that translation and "I am going" $\leftrightarrow$ "gidiyorum". Therefore, we will represent a word in its lexical level representation, that is, its stem and its morphemes. For example, the translation pair "I am coming" $\leftrightarrow$ "geliyorum" will be represented as `i am come+ing` $\leftrightarrow$ `gel+Hyor+yHm`. Similarly, the pair "I am going" $\leftrightarrow$ "gidiyorum" will be represented as `i am go+ing` $\leftrightarrow$ `gid+Hyor+yHm`. Here, the `+` symbol is used to mark the beginning of a morpheme, and the letter `H` in the morphemes represents a vowel whose surface level realization is determined according to vowel harmony rules of the Turkish language. According to this representation, the first two translation pairs would be given as

```
i give+p the ticket to mary  ↔ mary+'e bilet+yH ver+DH+m
   i give+p the pen to mary  ↔ mary+'e kalem+yH ver+DH+m .
```

The translation template learned from these two translation pairs is

$$\begin{aligned} \texttt{[i give+p the } X^1 \texttt{ to Mary]} &\leftrightarrow \texttt{[mary+'e } X^2\texttt{+yH ver+DH+m] if} \\ [X^1] &\leftrightarrow [X^2] \texttt{ .} \end{aligned}$$

This representation allows an abstraction over technicalities such as vowel and/or consonant harmony rules, as in Turkish and also, different realizations of the same verb according to tense, as in English. We assume that the generation of surface level representation of words from their lexical level representations is trivial.

3 Learning Translation Templates

The TTL algorithm infers translation templates using similarities and differences between two example translation pairs (E_i, E_j) from a bilingual parallel corpus. Formally, a translation example $E_i : E_i^1 \leftrightarrow E_i^2$ is composed of a pair of sentences, E_i^1 and E_i^2, that are translations of each other in L_1 and L_2, respectively.

Given two translation examples (E_i, E_j), we try to find similarities between the constituents of E_i and E_j. A sentence is considered as a sequence of lexical items (i.e., words or morphemes). If no similarities can be found, then no templates from these examples is learned. If there are similar constituents then a *match sequence* in the following form is generated.

$$S_0^1, D_0^1, S_1^1, \cdots, D_{n-1}^1, S_n^1, \leftrightarrow S_0^2, D_0^2, S_1^2, \cdots, D_{m-1}^2, S_m^2 \qquad \text{for } 1 \leq n, m.$$

Here, S_k^1 represents a similarity (a sequence of common items) between E_i^1 and E_j^1. Similarly, $D_k^1 : (D_{i,k}^1, D_{j,k}^1)$ represents a difference between E_i^1 and E_j^1, where $D_{i,k}^1$ and $D_{j,k}^1$ are non-empty differing items between two similar constituents S_k^1 and S_{k+1}^1. Corresponding differences do not contain common items. That is, for a difference D_k, $D_{i,k}$ and $D_{j,k}$ do not contain any common item. Also, no lexical item in a similarity S_i appears in any previously formed difference D_k for $k < i$. Any of S_0^1, S_n^1, S_0^2 or S_m^2 can be empty, however, S_i^1 for $0 < i < n$ and S_j^2 for $0 < j < m$ must be non-empty. Note that there exists either a unique match or no match between two example translation pairs. For instance, the match sequence obtained for the translation examples given above is

```
i give+p the (ticket,pen) to mary
                    ↔ mary'e (bilet,kalem)+yH ver+DH+m .
```

That is, S_0^1="`i give+p the`", D_0^1=("`ticket`","`pen`"), S_1^1="`to mary`", S_0^2= "`mary'e`", D_0^2=("`bilet`","`kalem`"), S_1^2= "`+yH ver+DH+m`".

If there exist only a single difference in both sides of a match sequence, i.e., $n = m = 1$, then these differing constituents must be the translations of each other. Therefore, from the match sequence given above the following translation template can be inferred:

$$\begin{aligned} \texttt{[i give+p the } X^1 \texttt{ to mary]} &\leftrightarrow \texttt{[mary+'e } X^2\texttt{+yH ver+DH+m] if} \\ [X^1] &\leftrightarrow [X^2] \ . \end{aligned}$$

If, on the other hand, the number of differences are equal on both sides, but more than one, i.e., $1 < n = m$, without prior knowledge, it is impossible to determine which difference pairs in one side correspond to which difference pairs on the other side. Therefore, learning depends on previously acquired translation templates. For example, the following translation examples have two differences on both sides.

```
  i give+p the book  ↔ kitab+yH ver+DH+m
you give+p the pen   ↔ kalem+yH ver+DH+n .
```

Without prior information, we cannot determine if `i` corresponds to `kitab` or `+m`. However, if it has already been learned that `i` corresponds to `+m` and `you` corresponds to `+n`, then the following three translation templates can be inferred:

procedure TTL(*Training_Set*)
begin
for each pair of translation examples E_i and E_j in *Training_Set* **do begin**
Let the match sequence be
$M_{i,j} = S_0^1, D_0^1, \cdots, D_{n-1}^1, S_n^1, \leftrightarrow S_0^2, D_0^2, \cdots, D_{m-1}^2, S_m^2$
if $n = m = 1$ **then** generate the following rules:
$[S_0^1, \ X^1, \ S_1^1] \leftrightarrow [S_0^2, \ X^2, \ S_1^2]$ `if` $[X^1] \leftrightarrow [X^2]$
$[D_{0,i}^1] \leftrightarrow [D_{0,i}^2]$
$[D_{0,j}^1] \leftrightarrow [D_{0,j}^2]$
else if $1 < n = m$ **and** for all differences in $M_{i,j}$ except one,
D_k^1 and D_l^2 the differences can be reduced **then**
generate the following rules:
$[S_0^1, \ \cdots \ X^1 \cdots \ S_n^1] \leftrightarrow [S_0^2, \ \cdots \ X^2 \cdots \ S_m^2]$ `if` $[X^1] \leftrightarrow [X^2]$ `and` $\mathcal{D}$
$[D_{k,i}^1] \leftrightarrow [D_{l,i}^2]$
$[D_{k,j}^1] \leftrightarrow [D_{l,j}^2]$
Here, $\mathcal{D}$ stands for the list of conditions for known differences
end if
end for
sort the templates by their *specificities*
end.

Figure 1: The TTL algorithm.

$[X_1^1$ `give+p the` $X_2^1]$ $\leftrightarrow$ $[X_2^2$`+yH ver+DH`$X_1^2]$
`if` $[X_1^1] \leftrightarrow [X_1^2]$ `and` $[X_2^1] \leftrightarrow [X_2^2]$
`[book]` $\leftrightarrow$ `[kitab]`
`[pen]` $\leftrightarrow$ `[kalem]` .

In general, when the number of differences in both sides of a match sequences is greater than or equal to 1, e.i., $1 \leq n = m$, the TTL algorithm learns new translation templates only if at least $n - 1$ of the differences have already been learned. Otherwise, the current version of the algorithm cannot learn new rules.

The last step of the algorithm is to order templates according to their specificities. Given two templates, the one that has a higher number of terminals is more specific than the other. Note that, the specificity is defined according to the source language. For two way translation, the templates are ordered once for each language as the source. A formal description of the TTL algorithm is summarized in Figure 1.

4 Examples

In order to evaluate the TTL algorithm we have implemented it in PROLOG and tested on a sample bilingual parallel text. Training set contained 112 training pairs.

In the first pass, the TTL algorithm learned 320 translation templates. In the second pass, using

the initial training pairs and these new translation templetes, the TTL algorithm inferred 29 new translation templates. In the third pass, the algorithm learned 2 more templates in 3.24 seconds. No new templates were learned in the fourth pass. On a SPARC 20/61 workstation, each pass took about 3.8 seconds real time. Using these 463 templates, translation of a new sentence, took about 16 miliseconds on the average.

In this section, we will illustrate the behavior of TTL on that sample training text.

Example 1: Given the example translations "I saw you at the garden" ↔ "Seni bahçede gördüm" and "I saw you at the party" ↔ "Seni partide gördüm", their lexical level representations are

```
i see+p you at the garden  ↔ sen+yH bahçe+DA gör+DH+m
 i see+p you at the party  ↔ sen+yH parti+DA gör+DH+m .
```

From these examples with one pair of differences in both sides, the following translation templates are learned:

```
[i see+p you at the X1]  ↔ [sen+yH X2+DA gör+DH+m]
                           if [X1]↔[X2]
              [garden]   ↔ [bahçe]
               [party]   ↔ [parti].
```

Example 2: Given the example translations "It falls" ↔ "Düşer", "I will take the car" ↔ "Arabayı alacağım", "If a pen is dropped then it falls" ↔ "Bir kalem bırakılırsa, düşer" and "If he brought the keys then I will take car" ↔ "anahtarları getirdiyse arabayı alacağım", their lexical level representations are

```
              it fall+s  ↔ düş+Ar
     i will take the car  ↔ araba+yH al+yAcAk+yHm
     if a pen is drop+pp  ↔ bir kalem bırak+Hl+Hr+ysA,
          then it fall+s    düş+Ar
  if he bring+p the keys  ↔ anahtarlarI getir+DH+ysA,
then i will take the car    araba+yH al+yAcAk+yHm.
```

The match sequence between the last two example translations contains two similatities for `if` and `then`, and two differences. Since there are more than one differences, no translations templates can be learned directly. However, with the help of the first two example pairs, the following translation templates are learned:

```
   [a pen is drop+pp]  ↔ [bir kalem bırak+Hl+Hr]
[he bring+p the keys]  ↔ [anahtarlarI getir+DH]
     [if X1 then X2]   ↔ [X1+ysA, X2]
                         if [X1]↔ [X1] and [X2]↔[X2].
```

Example 3: Given the example translations "I would like to look at it" ↔ "Ona bakmak isterim" and "Do not look at it" ↔ "Ona bakma" their lexical level representations are

```
i would like to look at it  ↔ o+nA bak+mAk iste+Hr+yHm
         do not look at it  ↔ o+nA bak+mA .
```

Even from these structurally different translation examples, the following translation templates are learned:

```
  [X1 look at it]  ↔ [o+nA bak X2] if [X1] ↔ [X2]
[i would like to]  ↔ [+mAk iste+Hr+yHm]
         [do not]  ↔ [+mA] .
```

Example 4: Given the example translations "he can write ↔ "yazabilir", "do not talk" ↔ "konuşma", "he can write while he is reading" ↔ "okurken yazabilir" and "do not talk while you are eating" ↔ "yemek yerken konuşma", their lexical level representations are

```
                    he can write  ↔ yaz+yAbil+Hr
                     do not talk  ↔ konuş+mA
he can write while he is read+ing  ↔ oku+Hr+yken yaz+yAbil+Hr
do not talk while you are eat+ing  ↔ yemek ye+Hr+yken konuş+mA .
```

From these translations examples the following useful translation templates are learned:

```
[X1_1 while X1_2+ing]  ↔ [X2_2+Hr+yken X2_1]
                           if [X1_1]↔[X2_1] and [X1_2]↔[X2_2]
     [he is read]  ↔ [oku]
    [you are eat]  ↔ [yemek ye] .
```

The last two translation templates may be used to fill in more complex translation templates.

Example 5: Natural languages are full of idiomatic expressions. For example, in Turkish, "kafayı yediler" is such an expression meaning "they have got crazy", while its literal translation would be "they ate the head". Since, the TTL algorithm sorts the templates according to their specificities, such idiomatic expressions can be handled easily. Consider that the following examples are provided to learn the templates: "we have got crazy" ↔ "kafayı yedik", "this is an apple" ↔ "bu bir elmadır, "this is an orange" ↔ "bu bir portakaldır", "i ate the apple" ↔ "elmayı yedim", and "you ate the orange" ↔ "portakalı yedin". The lexical level representations are

```
      this is an apple  ↔ bu bir elma+DHr
     this is an orange  ↔ bu bir portakal+DHr
they have get+p crazy   ↔ kafa+yH ye+DH+lAr
  we have get+p crazy   ↔ kafa+yH ye+DH+k
    i eat+p the apple   ↔ elma+yH ye+DH+m
 you eat+p the orange   ↔ portakal+yH ye+DH+n .
```

From these translation examples the following useful translation templates are learned, in the order of specificity:

$$
\begin{aligned}
[X_1^1 \texttt{ have get+p crazy}] &\leftrightarrow [\texttt{kafa+yH ye+DH}X_1^2] \texttt{ if } [X_1^1] \leftrightarrow [X_1^2] \\
[\texttt{this is an } X_1^1] &\leftrightarrow [\texttt{bu bir } X_1^2\texttt{+DHr}] \texttt{ if } [X_1^1] \leftrightarrow [X_1^2] \\
[X_1^1 \texttt{ eat+p the } X_2^1] &\leftrightarrow [X_2^2\texttt{+yH ye+DH}X_1^2] \\
&\quad \texttt{if } [X_1^1] \leftrightarrow [X_1^2] \texttt{ and } [X_2^1] \leftrightarrow [X_2^2] \\
[\texttt{they}] &\leftrightarrow [\texttt{+lAr}] \\
[\texttt{we}] &\leftrightarrow [\texttt{+k}] \\
[\texttt{apple}] &\leftrightarrow [\texttt{elma}] \\
[\texttt{orange}] &\leftrightarrow [\texttt{portakal}] \\
[\texttt{i}] &\leftrightarrow [\texttt{+m}] \\
[\texttt{you}] &\leftrightarrow [\texttt{+n}].
\end{aligned}
$$

5 Translation

The templates learned by the TTL algorithm can be used in the translation directly. These templates can be used for translation in both directions. The outline of the translation process is given below:

1. First, the lexical level representation of the input sentence to be translated is derived.
2. The most specific translation templates matching the input are collected. These templates are those that are most similar to the sentence to be translated.
3. For each selected template, its variables are instantiated with the corresponding values in the source sentence. Then, templates matching these bound values are sought. If they are found successfully, their values are replaced in the variables corresponding to the sentence in the target language.
4. The surface level representation of the sentence obtained in the previous step is generated.

For instance, after learning the templates in Example 5, if the input is given as "kafayı yedim", firts its lexical level represetation, which is `kafa+yH ye+dH+m`, is derived. Although there are two matching templates (the first and the third), the most specific template matching this is

$$[X_1^1 \texttt{ have get+p crazy}] \leftrightarrow [\texttt{kafa+yH ye+DH}X_1^2] \texttt{ if } [X_1^1] \leftrightarrow [X_1^2].$$

The variable X_1^2 is instantiated with `+m`. Then, the translation of `+m` is found to be `i` using

$$[\texttt{i}] \leftrightarrow [\texttt{+m}].$$

Therefore, replacing the value of `i` for X_1^1 in the template, the lexical level representation `i have get+p crazy` is obtained. Finaly, the surface level representation "I have got crazy" is derived easily.

On the other hand, if the input sentence is "portakalı yedim", only the third template can be used, and the correct translation "I ate the orange" is obtained.

Note that, if the sentence in the source language is ambiguous, then templates corresponding to each sense will be retrieved, and the sentences for each sense will be generated. Among the possible translations, a human user can choose the right one according to the context.

6 Conclusion

In this paper, we have presented a model for learning translation templates between two languages. The model is based on a simple pattern matcher. We integrated this model with an example-based translation model into Generalized Exemplar-Based Machine Translation. We have implemented this model as the TTL (Translation Template Learner) algorithm. The TTL algorithm is illustrated in learning translation templates between Turkish and English. The approach is applicable to any pair of languages.

The major contribution of this paper is that the proposed TTL algorithm eliminates the need for manually encoding the translations, which is a difficult task for a large corpus. The TTL algorithm can work directly on surface level representation of sentences. However, in order to generate useful translation patterns, it is helpful to use the lexical level representations. It is usually trivial, at least for English and Turkish, to obtain the lexical level representations of words.

Our main motivation was that the underlying inference mechanism is compatible with one of the ways humans learn languages, i.e. learning from examples. We believe that in everyday usage, humans learn general sentence patterns, using the similarities and differences between many different example sentences that they are exposed to. This observation led us to the idea that a computer can be trained similarly, using analogy within a corpus of example translations.

The accuracy of the translation templates learned by this approach is quite high with ensured grammaticality. Given that a translation is carried out using the rules learned, the accuracy of the output translation critically depends on the accuracy of the rules learned.

We do not require an extra operation to maintain the grammaticality and the style of the output, as in Kitano's EBMT model [6]. The information necessary to maintain these issues is directly provided by the translation templates.

The learning and translation times on the small training set are quite reasonable, and that indicates the program will scale up real large training corpora. Note that this algorithm is not specific to English and Turkish languages, but should be applicable to the task of learning machine translation between any pair of languages. Although the learning process on a large corpus will take a considerable amount of time, it is only one time job. After learning the translation templates, the translation process is fast.

The model that we have proposed in this paper may be integrated with other systems as a Natural Language Front-end, where a small subset of a natural language is used. This algorithm can be used to learn to translate user queries to the language of the underlying system.

This model may also be integrated with an intelligent tutoring system (ITS) for second language learning. The template representation in our model provides a level of information that may help in error diagnosis and student modeling tasks of an ITS. The model may also be used in tuning the teaching strategy according to the needs of the student by analyzing the student answers analogically with the closest cases in the corpus. Specific corpora may be designed to concentrate on certain topics that will help in student's acquisition of the target language. The work presented by this paper provides an opportunity to evaluate this possibility as a future work.

References

[1] Arnold D., Balkan L., Humphreys R. Lee, Meijer S., Sadler L.: *Machine Translation*, NCC Blackwell (1994).

[2] Carbonell, J.G.: Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition. In Jude W. Shavlik and Thomas G. Dietterich (eds), *Readings in Machine Learning*, Morgan Kaufmann (1990) 636–646.

[3] Furuse, O. & Iida, H.: Cooperation between Transfer and Analysis in Example-Based Framework, *Proceedings of COLING-92* (1992).

[4] Güvenir, H.A. & Tunç, A.: Corpus-Based Learning of Generalized Parse Tree Rules for Translation. In Gord McCalla (Ed). *New Directions in Artificial Intelligence: Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Springer-Verlag, LNCS 1081, Toronto, Ontario, Canada, (May 22-24, 1996), 121-131.

[5] Hammond, K.J.: (Ed.) *Proceedings: Second Case-Based Reasoning Workshop*. Pensacola Beach, FL:Morgan Kaufmann (1989).

[6] Kitano, H.: A Comprehensive and Practical Model of Memory-Based Machine Translation. In Ruzena Bajcsy (Ed.) *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann V.2 (1993) 1276-1282.

[7] Kolodner, J.L.: (Ed.) *Proceedings of a Workshop on Case-Based Reasoning*. Clearwater Beach, FL: Morgan Kaufmann (1988).

[8] Medin, D.L. & Schaffer, M.M.: Context theory of classification learning. *Psychological Review, 85* (1978) 207-238.

[9] Nagao, M. A.: Framework of a Mechanical Translation between Japanese and English by Analogy Principle (1985).

[10] Nirenburg, S., Beale, S. & Domashnev, C.: A Full-Text Experiment in Example-Based Machine Translation. *Proceedings of the International Conference on New Methods in Language Processing, NeMLap* Manchester, UK (1994) 78-87.

[11] Ram, A.: Indexing, Elaboration and Refinement: Incremental Learning of Explanatory Cases. In Janet L. Kolodner (ed.), *Case-Based Learning*, Kluwer Academic Publishers (1993).

[12] Reisbech, C. & Schank, R.: *Inside the Case-Based Reasoning*, Lawrence Elbaum Associates (1990).

[13] Sato, S. & Nagao, M.: The Memory-Based Translation, *Proceedings of COLING-90* (1990).

[14] Stanfill, C. & Waltz, D.: Toward Memory-Based Reasoning. *CACM*, Vol.29, No.12 (1991) 185-192.

[15] Sumita, E. & Iida, H.: Experiments and Prospects of Example-Based Machine Translation, *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics* (1991).