Pergamon

Information Systems Vol. 23, No. 6, pp. 353–363, 1994

PII: S0306-4379(98)00017-9
0306-4379/98 $19.00 + 0.00

LEARNING TRANSLATION TEMPLATES FROM EXAMPLES†

H. ALTAY GÜVENIR and ILYAS CICEKLI

Bilkent University, Dept. of Computer Engineering and Information Science, Ankara 06533, Turkey

(Received 11 September 1997; in final revised form 12 June 1998)

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 to 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.

Key words: Machine learning, machine translation, translation templates

1. INTRODUCTION

Traditional machine translation (MT) systems require large-scale knowledge about both source and target languages in the form of computational grammars, lexicons and domain knowledge. In these traditional knowledge-based machine translation systems such as KBMT-89 [4], this large-scale knowledge is hand-coded and hand-crafted for specific domains. They make the use of this large-scale knowledge residing in different resources such as lexicons, grammar rules and mapping rules during translation. Since the acquisition of this kind of knowledge is time-consuming and expensive, researchers have been studying to the ways of automatically and semi-automatically acquiring some portions of the required knowledge from large corpora. For example, in the KANT [13] system which is an immediate descendant of the KBMT-89 system, some methods for automatic acquisition of lexicons from a large-corpus are used [11]. The work presented in this paper tries to automate the acquisition of the required knowledge for machine translation task (except for morphological rules) from a sentence-level aligned bilingual text corpora only.

Because of the large-scale acquisition problem in traditional approaches to machine translation, corpus-based approach is one of the alternative directions that have been proposed to overcome the difficulties of traditional systems. Two fundamental directions 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 [12], 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 [7, 10, 15]. The learning paradigm used in this paper is referred as *Exemplar-Based Generalization* (coined by Salzbeg [17]), where exemplars are not only simple examples, but templates that are obtained by generalizing appropriate components of examples [5].

†Recommended by Arun Sen

EBMT has been proposed by Nagao [14] as *translation by analogy* which is in parallel with memory based reasoning [20], case-based reasoning [16] and derivational analogy [2]. Example-based translation relies on the use of past translation examples to derive a translation for a given input [3, 9, 19, 21]. 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 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 [8]. Sato [18] also proposed an exemplar-based system with manually encoded matching expressions which are used as translation templates. This paper formulates the acquisition of translation rules as a machine learning task in order to automate the process.

Our first attempt was to construct parse trees between the example translation pairs [6]. However, the difficulty was the unavailability 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:

$\underline{\texttt{I took a}}$ $\texttt{ticket}$ $\underline{\texttt{from Mary}}$ $\leftrightarrow$ $\underline{\texttt{Mary'den bir}}$ $\texttt{bilet}$ $\underline{\texttt{aldım}}$
$\underline{\texttt{I took a}}$ $\texttt{pen}$ $\underline{\texttt{from Mary}}$ $\leftrightarrow$ $\underline{\texttt{Mary'den bir}}$ $\texttt{kalem}$ $\underline{\texttt{aldım}}$

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 took a` X^E `from Mary"`, and the corresponding similarities in Turkish as `"Mary'den bir` X^T `aldım"`. 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, and 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 following three translation templates that would be learned from the example translations given above are:

I took a X^1 from Mary $\leftrightarrow$ Mary'den bir X^2 aldım
    if $X^1 \leftrightarrow X^2$
ticket $\leftrightarrow$ bilet
pen $\leftrightarrow$ kalem

The first translation template is read as the sentence "I took a X^1 from Mary" in L^1 and the sentence "Mary'den bir X^2 aldım" 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 took a basket from Mary" can be translated into L^2 as "Mary'den bir sepet aldım". Each of the second and third templates represents an atomic correspondence of two strings in the languages L_1 and L_2 without any dependency. In fact, we also use given example translations as atomic translation templates.

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 the surface level representation is used, it is not possible to find a template from that translation. 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, these two translation pairs would be given as

I am come+ing $\leftrightarrow$ gel+Hyor+yHm
I am go+ing $\leftrightarrow$ gid+Hyor+yHm

The translation templates learned from these two translation pairs are

I am X^1+ing $\leftrightarrow$ X^2+Hyor+yHm
    if $X^1 \leftrightarrow X^2$
come $\leftrightarrow$ gel
go $\leftrightarrow$ gid

This lexical level 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_a, E_b) from a bilingual parallel corpus. Formally, a translation example $E_a : E_a^1 \leftrightarrow E_a^2$ is composed of a pair of sentences, E_a^1 and E_a^2, that are translations of each other in L_1 and L_2, respectively.

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

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

Here, S_k^1 represents a similarity (a sequence of common items) between E_a^1 and E_b^1. Similarly, $D_k^1 : (D_{k,a}^1, D_{k,b}^1)$ represents a difference between E_a^1 and E_b^1, where $D_{k,a}^1$ and $D_{k,b}^1$ are non-empty differing items between two similar constituents S_k^1 and S_{k+1}^1. Corresponding differing constituents do not contain common items. That is, for a difference D_k, $D_{k,a}$ and $D_{k,b}$ 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. Furthermore, at least one similarity on each side must be non-empty. Note that there exists either a unique match or no match between two example translation pairs.

For instance, let us assume that the following translation examples are given for translation pairs `"I gave the ticket to Mary"` $\leftrightarrow$ `"Mary'e bileti verdim"` and `"I gave the pen to Mary"` $\leftrightarrow$ `"Mary'e kalemi verdim"`:

```
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
```

For these translation examples, the following match sequence is obtained by our matching algorithm.

```
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.`

After a match sequence is found for two translation examples, we use a learning heuristic to infer translation templates from that match sequence. This heuristic tries to locate corresponding differences in the match sequence. If it can locate all corresponding differences, it can learn a new translation template by replacing all differences with variables.

If there exists 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. In other words, we are able to locate the corresponding differences in the match sequence. In this case, the match sequence must be in the following form.

$$S_0^1, D_0^1, S_1^1 \leftrightarrow S_0^2, D_0^2, S_1^2$$

Since D_0^1 and D_0^2 are the corresponding differences, the following translation template is inferred by replacing these differences with variables.

$$S_0^1\ X^1\ S_1^1 \leftrightarrow S_0^2\ X^2\ S_1^2$$
$$\quad \textbf{if } X^1 \leftrightarrow X^2$$

Furthermore, the following two translation templates are learned from the corresponding differences $(D_{0,a}^1, D_{0,b}^1)$ and $(D_{0,a}^2, D_{0,b}^2)$.

$D^1_{0,a} \leftrightarrow D^2_{0,a}$
$D^1_{0,b} \leftrightarrow D^2_{0,b}$

For example, since the match sequence of the example above contains a single difference in both sides, the following translation template and two additional translation templates from the corresponding differences `(ticket,pen)` and `(bilet,kalem)` can be inferred:

`I give+p the` X^1 `to Mary` $\leftrightarrow$ `Mary+'e` X^2`+yH ver+DH+m`
    if $X^1 \leftrightarrow X^2$
`ticket` $\leftrightarrow$ `bilet`
`pen` $\leftrightarrow$ `kalem`

On the other hand, if 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 in one side corresponds to which difference on the other side. Therefore, learning depends on previously acquired translation templates. Our learning algorithm tries to locate n–1 corresponding differences in the match sequence by checking previously learned translation templates. We say that the k^{th} difference $(D^1_{k,a}, D^1_{k,b})$ on the left side corresponds to the l^{th} difference $(D^2_{l,a}, D^2_{l,b})$ on the right side if the following two translation templates have been learned earlier.

$D^1_{k,a} \leftrightarrow D^2_{l,a}$
$D^1_{k,b} \leftrightarrow D^2_{l,b}$

We call this corresponding pair of differences as *Corresponding Difference Pair* (CDP). After finding $n-1$ CDPs, the last two unchecked differences, one at each side (language), should correspond to each other. Thus, for all differences in the match sequence, we determine which difference in one side corresponds to which difference on the other side.

Let us assume that the list

$$CDP_1, CDP_2, \cdots, CDP_n$$

represents the list of all CDPs, where CDP_n is the pair of the two unchecked differences. A CDP_i is the pair of two differences in the form $(D^1_{k_i}, D^2_{l_i})$. For each CDP_i, we replace $D^1_{k_i}$ with a variable X^1_i and $D^2_{l_i}$ with a variable X^2_i in a match sequence $M_{a,b}$. The newly formed match sequence is called as $M_{a,b}DV$ for *match sequence with difference variables*. As a result, the following translation template can be inferred.

$M_{a,b}DV$
    if $X^1_1 \leftrightarrow X^2_1$ **and** $\cdots$ **and** $X^1_n \leftrightarrow X^2_n$

In addition, the following translation templates are learned from the last corresponding differences $(D^1_{k_n,a}, D^1_{k_n,b})$ and $(D^2_{l_n,a}, D^2_{l_n,b})$.

$D^1_{k_n,a} \leftrightarrow D^2_{l_n,a}$
$D^1_{k_n,b} \leftrightarrow D^2_{l_n,b}$

For example, the following translation examples, which are lexical forms of translation pairs `"I gave the book"` $\leftrightarrow$ `"Kitabı verdim"` and `"You gave the pen"` $\leftrightarrow$ `"Kalemi verdin"`, have two differences on both sides.

`i give+p the book` $\leftrightarrow$ `kitab+yH ver+DH+m`
`you give+p the pen` $\leftrightarrow$ `kalem+yH ver+DH+n`

The following match sequence is obtained for these examples.

`(i,you) give+p the (book,pen)` $\leftrightarrow$ `(kitab,kalem) +yH ver+DH (+m,+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 translation template and two additional translation templates can be inferred.

procedure TTL($M_{a,b}$)
begin
- Let the match sequence $M_{a,b}$ be:
  $S_0^1, D_0^1, \cdots, D_{n-1}^1, S_n^1, \leftrightarrow S_0^2, D_0^2, \cdots, D_{m-1}^2, S_m^2$

if $n = m = 1$ **then**
- return the following rules:
  $S_0^1\ X^1\ S_1^1 \leftrightarrow S_0^2\ X^2\ S_1^2$
  if $X^1 \leftrightarrow X^2$
  $D_{0,a}^1 \leftrightarrow D_{0,a}^2$
  $D_{0,b}^1 \leftrightarrow D_{0,b}^2$

else if $1 < n = m$ **and** $n-1$ corresponding differences can be found in $M_{a,b}$ **then**
- Assume that the unchecked corresponding differences is $((D_{k_n,a}^1, D_{k_n,b}^1), (D_{l_n,a}^2, D_{l_n,b}^2))$.
- Let the list of corresponding differences be $(D_{k_1}^1, D_{l_1}^2) \cdots (D_{k_n}^1, D_{l_n}^2)$ including unchecked ones.
- For each corresponding difference $(D_{k_i}^1, D_{l_i}^2)$ replace $D_{k_i}^1$ with X_i^1 and $D_{k_i}^2$ with X_i^2 to get the new match sequence $M_{a,b}DV$.
- return the following rules:
  $M_{a,b}DV$
  if $X_1^1 \leftrightarrow X_1^2$ and $\cdots$ and $X_n^1 \leftrightarrow X_n^2$
  $D_{k_n,a}^1 \leftrightarrow D_{l_n,a}^2$
  $D_{k_n,b}^1 \leftrightarrow D_{l_n,b}^2$

end

Fig. 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. A formal description of the TTL algorithm is summarized in Figure 1.

The TTL algorithm described in this section can only be applied to the match sequences with the same number of differences on both sides. If we get a match sequence with different number of differences, we try to equate the number of differences by separating differences before we apply our TTL algorithm. If the number of differences in one side is less than the number of differences in the other side, we try to divide the differences in the side with less differences to increase their number. If both constituents of a difference contain more than one morpheme (or root word), we may divide that difference into two differences by separating both constituents of that difference from morpheme boundaries.

In each pass of the learning phase, the TTL algorithm is applied to each pair of examples in the training set to infer translation templates. The learning phase continues until no new translation template is learned. In other words, when the number of new learned translation templates in a pass is zero, the learning process stops. The maximum number of passes of the learning phase is theoretically $n-1$ when the number of examples in the training set is n. However, in usual training sets, the number of passes needed is much less. For example, only 4 passes were executed for a sample training set of 112 example pairs.

After all translation templates are learned, they are sorted 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.

4. LEARNING EXAMPLES

For an empirical evaluation of the TTL algorithm, we have implemented it in PROLOG and tested on sample bilingual parallel texts. One of artificially created training sets contained 747 training pairs. In the first pass, the TTL algorithm learned 642 translation templates. No new templates were learned in the second pass. On a SPARC 20/61 workstation, each pass took about 44 seconds real time. Using these 1389 templates including 747 translation examples, translation of a new sentence, took about 85 milliseconds on the average.

In this section, we illustrate the behavior of TTL algorithms on some sample training example pairs.

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

i see+p you at the garden $\leftrightarrow$ sen+yH bahçe+DA gör+DH+m
i see+p you at the party $\leftrightarrow$ 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 X^1 $\leftrightarrow$ sen+yH X^2+DA gör+DH+m
    if $X^1 \leftrightarrow X^2$
garden $\leftrightarrow$ bahçe
party $\leftrightarrow$ parti.

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

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

The match sequence between the last two example translations contains two similarities 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 $\leftrightarrow$ bir kalem bırak+Hl+Hr
he bring+p the keys $\leftrightarrow$ anahtarlarI getir+DH
if X_1^1 then X_2^1 $\leftrightarrow$ X_1^2+ysA, X_2^2
    if $X_1^1 \leftrightarrow X_1^2$ **and** $X_2^1 \leftrightarrow X_2^2$.

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

i would like to look at it $\leftrightarrow$ o+nA bak+mAk iste+Hr+yHm
do not look at it $\leftrightarrow$ o+nA bak+mA .

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

X^1 `look at it` $\leftrightarrow$ `o+nA bak` X^2
    if $X^1 \leftrightarrow X^2$
`i would like to` $\leftrightarrow$ `+mAk iste+Hr+yHm`
`do not` $\leftrightarrow$ `+mA` .

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

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

The following useful translation templates are learned, from these translations examples:

X_1^1 `while` X_2^1`+ing` $\leftrightarrow$ X_2^2`+Hr+yken` X_1^2
    if $X_1^1 \leftrightarrow X_1^2$ **and** `and` $X_2^1 \leftrightarrow X_2^2$
`he is read` $\leftrightarrow$ `oku`
`you are eat` $\leftrightarrow$ `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" $\leftrightarrow$ "kafayı yedik", "this is an apple" $\leftrightarrow$ "bu bir elmadır", "this is an orange" $\leftrightarrow$ "bu bir portakaldır", "i ate the apple" $\leftrightarrow$ "elmayı yedim", and "you ate the orange" $\leftrightarrow$ "portakalı yedin". The lexical level representations are

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

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

X_1^1 `have get+p crazy` $\leftrightarrow$ `kafa+yH ye+DH`X_1^2
    if $X_1^1 \leftrightarrow X_1^2$
`this is an` X_1^1 $\leftrightarrow$ `bu bir` X_1^2`+DHr`
    if $X_1^1 \leftrightarrow X_1^2$
X_1^1 `eat+p the` X_2^1 $\leftrightarrow$ X_2^2`+yH ye+DH`X_1^2
    if $X_1^1 \leftrightarrow X_1^2$ **and** $X_2^1 \leftrightarrow X_2^2$
`they` $\leftrightarrow$ `+lAr`
`we` $\leftrightarrow$ `+k`
`apple` $\leftrightarrow$ `elma`
`orange` $\leftrightarrow$ `portakal`
`i` $\leftrightarrow$ `+m`
`you` $\leftrightarrow$ `+n` .

Example 6: Our example pairs do not need to be pairs of sentences. They can also be pairs of expressions. First two example pairs of the following four examples are pairs of expressions:

```
red car ↔ kırmızı araba
red truck ↔ kırmızı kamyon
he buy+p a pen ↔ bir kalem satın al+DH
he buy+p a book ↔ bir kitap satın al+DH.
```

From these examples, the following translation templates are learned:

red X^1 ↔ kırmızı X^2
    if $X^1 \leftrightarrow X^2$
car ↔ araba
truck ↔ kamyon
he buy+p a X^1 ↔ bir X^2 satın al+DH
    if $X^1 \leftrightarrow X^2$
pen ↔ kalem
book ↔ kitap.

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. This process continues recursively for the variables which may be introduced in this step.
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", first its lexical level representation, 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 have get+p crazy ↔ kafa+yH ye+DHX_1^2
    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

i ↔ +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. Finally, 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. We expect that the correct answer will be among the first translations because of the usage of the specificity rule.

The translation phase is a recursive process for replacing variables in the templates. For example, to translate English sentence "If he bought a red pen then he can write" into Turkish, first we get its lexical form `"if he buy+p a red pen then he can write"`. This sentence will match with the left side of the third translation template given in Example 2 by binding X_1^1 to `"he buy+p a red pen"` and X_2^1 to `"he can write"`. Then, we will seek translations of these portions. The second expression `"he can write"` will be translated into `"yaz+yAbil+Hr"` because their correspondence was directly given in Example 4. On the other hand, the first one will match with the fourth translation template in Example 6 by binding X^1 to `"red pen"`. When we seek the translation of `"red pen"`, it will match with the first translation template in Example 6 by binding X^1 to `"pen"`. Then, `"pen"` will be translated into `"kalem"` by using the fifth translation template in Example 6. Thus,

`"pen"` is translated into `"kalem"`,
`"red pen"` is translated into `"kırmızı kalem"`,
`"he buy+p a red pen"` is translated into `"bir kırmızı kalem satın al+DH"`,
`"he can write"` is translated into `"yaz+yAbil+Hr"`,
Finally, `"if he buy+p a red pen then he can write"` is translated into
`"bir kırmızı kalem satın al+DH+ysA, yaz+yAbil+Hr"`.

We get the surface form `"bir kırmızı kalem satın aldıysa, yazabilir"` for the lexical form of Turkish sentence.

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 [8]. 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.

Acknowledgements — This research has been supported in part by NATO Science for Stability Program Grant TU-LANGUAGE and The Scientific and Technical Council of Turkey Grant EEEAG-244.

REFERENCES

[1] Doug Arnold, Lorna Balkan, R. Lee Humphreys, and L. Sadler S. Meijer. *Machine Translation: An Introductory Guide*. NCC Blackwell (1994).

[2] Jaime G. Carbonell. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*, pp. 247–288. Morgan Kaufmann Publishers, Inc. (1990).

[3] Osamu Furuse and Hitoshi Iida. Cooperation between transfer and analysis in example-based framework. In *Proc. of COLING-92*, pp. 645–651, Nantes, France (1992).

[4] Kenneth Goodman and Sergei Nirenburg. *KBMT-89: A Case Study in Knowledge Based Machine Translation*. Morgan Kaufmann (1992).

[5] H. Altay Güvenir and Izzet Sirin. Classification by feature partitioning. *Machine Learning*, **23**(1):47–67 (1996).

[6] H. Altay Güvenir and Aysegül Tunç. Corpus-based learning of generalized parse tree rules for translation. In Gordon McCalla, editor, *Proceedings of the Eleventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, volume 1081 of *LNAI*, pp. 121–132, Berlin. Springer (1996).

[7] Kristian J. Hammond. *Proceedings of the Second Case-Based Reasoning Workshop*. Morgan Kaufmann (1989).

[8] Hiroaki Kitano. A comprehensive and practical model of memory-based machine translation. In *IJCAI-93* (1993).

[9] Hiroaki Kitano. A full-text experiment in example-based machine translation. In *Proceedings of the International Conference on New Methods in Language Processing, NeMLap* (1994).

[10] Janet L. Kolodner, editor. *Proceedings of a Workshop on Case-Based Reasoning (DARPA)*. Morgan Kaufmann (1988).

[11] Deryle Lonsdale, Teruko Mitamura, and Eric Nyberg. Acquisition of large lexicons for practical knowledge-based mt. *Machine Translation*, **9**(3):251–283 (1994).

[12] Douglas L. Medin and M. M. Schaffer. Context theory of classification learning. *Psychological Review*, **85**:207–238 (1978).

[13] Teruko Mitumura and Eric Nyberg. The KANT system: Fast, accurate, high-quality translation in practical domains. In *Proc. of COLING-92*, pp. 1069–1073, Nantes, France (1992).

[14] Makoto Nagao. A framework of a mechanical translation between english and japanese by analogy principle. In A. Elithorn and R. Banerji, editors, *Artifical and Human Intelligence*, pp. 173–180. North-Holland (1984).

[15] Ashwin Ram. Indexing and elaboration and refinement: Incremental learning of explanatory cases. *Machine Learning*, **10**:201–248 (1993).

[16] Christopher K. Reisbeck and Roger C. Schank. *Proceedings of a Workshop on Case-Based Reasoning (DARPA)*. Lawrence Elbaum Associates (1989).

[17] Steven Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, **6**:251–276 (1991).

[18] Satoshi Sato. MBT2: a method for combining fragments of examples in example-based translation. *Artificial Intelligence*, **75**:31–50 (1995).

[19] Satoshi Sato and Makoto Nagao. Toward memory-based translation. In *Proc. of COLING-90*, pp. 247–252, Helsinki, Finland (1990).

[20] Craig Stanfill and David Waltz. Toward memory-based reasoning. *Communications of the ACM*, **29**(12):1213–1228 (1986).

[21] Eiichiro Sumita and Hitoshi Iida. Experiments and prospects of example-based machine translation. In *29th Annual Meeting of the Association for Computational Linguistics. Proceedings of the Conference. 18–21 June 1991. University of California, Berkeley, California, USA*, pp. 185–192 (1991).