Similarities and Differences

Ilyas Cicekli
Dept. of Comp. Eng., Bilkent University, 06533 Bilkent, Ankara, Turkey
ilyas@cs.bilkent.edu.tr

Abstract

A match sequence between two sentences of a language consists of similarities and differences to represent similar parts and differing parts between those sentences. A translation example is a corresponding of two sentences in which one of them from the language $\mathcal{L}^a$ and the other one from the language $\mathcal{L}^b$. From given two translation examples we create a match sequence for sentences of the language $\mathcal{L}^a$, and another match sequence for sentences of the language $\mathcal{L}^b$. From these match sequences, general templates are learned using certain learning heuristics. These heuristics replace differences or similarities in the match sequences, and establish bindings between variables to create a general template. These learned translation templates can be used in the translation of other sentences in both directions.
Keywords: Example-Based Machine Translation, Natural Language Processing

1 Introduction

Traditional machine translation (MT) systems require the large-scale knowledge such as lexicons, grammar rules, mapping rules and an ontology. Acquiring these knowledge resources manually is a time consuming and expensive process. For this reason, researchers have been studying the ways of automatically acquiring some portions of the required knowledge. Even in some traditional systems [12], some portions of the required knowledge such as the lexicon are acquired automatically from a corpus [10]. The technique presented here aims at acquiring all required knowledge except morphological rules for the machine translation task from sentence-level aligned bilingual text corpora only.

EBMT, originally proposed by Nagao [13], is one of the main approaches of corpus-based machine translation. The main idea behind EBMT is that a given input sentence in the source language is compared with the example translations in the given bilingual parallel text to find closest matching examples so that these examples can be used in the translation of that input sentence. After finding the closest matchings for the sentence in the source language, parts of the corresponding target language sentence are constructed using structural equivalences and deviances in those matches. Following Nagoa's original proposal, several machine translation methods that utilize bilingual corpora have been studied [5, 9, 16, 17, 18, 19]. Some researchers [3, 20] only utilized bilingual corpora to create a bilingual dictionary and use it during the translation process. In other words, they aligned bilingual corpora at word level to figure out corresponding words in languages. Bilingual corpora is also aligned at phrase level by some other researchers [1, 2, 14]. But these correspondences between two languages are only accomplished at atomic level, and they are used in the translation of portions of sentences. Kaji [8] tried to learn correspondences of English and Japanese syntactic structures from bilingual corpora. This is similar to the work in [6] and it needs reliable parsers for both source and target languages. The technique described here not only learns atomic correspondences between two languages, it also learns general templates describing structural correspondence (not syntactic structure) from bilingual corpora. Learning technique described in [4, 7] is similar to the first learning heuristic described in this paper.

Researchers in Machine Learning (ML) community have widely used exemplar-based representation. Medin and Schaffer [11] were the first researchers who proposed exemplar-based learning as a model of human learning. The characteristic examples stored in the memory are called *exemplars*. In EBMT, translation examples should be available prior to the translation of an input sentence. In most of the EBMT systems, these translation examples are directly used without any generalization. Kitano [9] manually encoded translation rules, however this is a difficult and error-prone task for a large corpus. In this paper, we formulate the acquisition of translation rules, which are similar to exemplars, as a machine learning problem in order to automate this task.

The translation template learning framework presented in this paper is based on a heuristic to infer the correspondences between the patterns in the source and target languages from given two translation pairs. According to this heuristic, given two translation examples, 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 similarities, then no correspondences are inferred. Given a corpus of translation examples, our learning heuristics infer the correspondences between the source and target languages in the form of templates. These templates can be used for translation in both directions.

The rest of the paper is organized as follows. First, we explain how a match sequence, which represents similarities and differences in a pair of translation examples, can be found. Then, we describe our learning heuristics, and how our learning heuristics can be used an example-based machine translation system. Finally, we conclude the paper with pointers for further research.

2 Match Sequence

In this section we formally describe what is a match sequence. Before we do that, we give other required definitions first.

An *alphabet* $\mathcal{A}$ is a non-empty set of terminals. For our purposes, the set of all words, suffixes, and prefixes in a natural language is the alphabet of that natural language. Here, we treat terminals as the smallest indivisible meaningful units of a language.

A *language* $\mathcal{L}$ on an alphabet $\mathcal{A}$ is a non-empty finite set of non-empty strings on $\mathcal{A}$ with finite length. In other words, $\mathcal{L} \subset \mathcal{A}^+$ such that $\mathcal{L}$ is a finite non-empty set, and if $\alpha \in \mathcal{L}$ then the length of α is finite and greater than 0. So, we treat a natural language as a finite set of its sentences and sentence chunks. Of course, we make an assumption that the set of all sentences of a natural language is finite.

A *string* on $\mathcal{A}$ is a member of $\mathcal{A}^*$. If a string α is a member of $\mathcal{L}$, α is called a *sentence* of $\mathcal{L}$. The reader will notice that a sentence is a non-empty string with finite length according to the definitions above.

A *similarity* between α_1 and α_2, where α_1 and α_2 are two sentences of a language $\mathcal{L}$, is a non-empty string β such that $\alpha_1 = \alpha_{1,1}\beta\alpha_{1,2}$ and $\alpha_2 = \alpha_{2,1}\beta\alpha_{2,2}$. A similarity represents a similar part between two sentences.

A *difference* between α_1 and α_2, where α_1 and α_2 are two sentences of a language $\mathcal{L}$, is a pair of two strings (β_1, β_2) where β_1 is a substring of α_1 and β_2 is a substring of α_2, the same terminal cannot occur in both β_1 and β_2, and at least one of them is not empty. A difference represents a pair of differing parts between two sentences. Sometimes we will insist that both constituents of a difference are not empty.

A *match sequence* between two sentences α_1 and α_2 of a language $\mathcal{L}$ is a sequence of similarities and differences between α_1 and α_2 such that the following conditions must be satisfied by this match sequence:

1. Concatenation of similarities and the first constituents of differences must be equal α_1.
2. Concatenation of similarities and the second constituents of differences must be equal α_2.

The order of operands in the concatenation operations are same as their order in the match sequence. So, a match sequence M between α_1 and α_2 is in the following form:

$M = P_1...P_n$ where each P_i is a similarity S_i or a difference $D_i = (D_{i,1}, D_{i,2})$, and $n \geq 1$.

If we define two constituent functions as follows:

$$C_{i,1} = \begin{cases} S_i & \text{if } P_i \text{ is a similarity } S_i \\ D_{i,1} & \text{if } P_i \text{ is a difference } (D_{i,1}, D_{i,2}) \end{cases}$$

$$C_{i,2} = \begin{cases} S_i & \text{if } P_i \text{ is a similarity } S_i \\ D_{i,2} & \text{if } P_i \text{ is a difference } (D_{i,1}, D_{i,2}) \end{cases}$$

then

$\alpha_1 = C_{1,1}...C_{n,1}$

$\alpha_2 = C_{1,2}...C_{n,2}$

An *instance of a match sequence* $M = P_1...P_n$ is another match sequence $M' = P'_1...P'_m$ where $m \geq n$ such that $C_{1,1}...C_{n,1} = \alpha_1 = C'_{1,1}...C'_{m,1}$ and $C_{1,2}...C_{n,2} = \alpha_2 = C'_{1,2}...C'_{m,2}$. An instance is created by separating similarities and differences in a match sequence. A similarity can be separable into two similarities if its length is greater than 1. A difference can be separable into two differences if the length of one of its constituent is greater than 1. But a difference can be separable into two differences without empty constituents if the lengths of both of its constituents are greater than 1. For our purposes, we create instances by just separating similarities or by just separating differences but not both of them at the same time. An instance is called *similarity instance* if its created by just separating differences, and it is called as *difference instance* if it is created by just separating similarities. A similarity instance will have exactly the same similarities as the original match sequence, and a difference instance will have exactly the same differences as the original match sequence.

There can be more than one match sequence for a pair of sentences. Now, we will put extra restrictions on match sequences by the following definition to ensure that two sentences have a unique match sequence or no match sequence. A *minimal match sequence* is a match sequence satisfying the following conditions:

1. A similarity cannot follow another similarity, and a difference cannot follow another difference in a minimal match sequence. This means that if P_i is a similarity in a minimal match sequence, P_{i+1} must be a difference, and if P_i is a difference in a minimal match sequence, P_{i+1} must be a similarity.
2. If a terminal occurs in a similarity, it cannot occur in any difference.
3. If a terminal occurs in the first constituent of a difference, it cannot occur in the second constituent of a prior difference. In other words, if a terminal occurs in $D_{i,1}$, it cannot occur in $D_{j,2}$ where $j<i$.
4. If a terminal occurs in the second constituent of a difference, it cannot occur in the first constituent of a prior difference. In other words, if a terminal occurs in $D_{i,2}$, it cannot occur in $D_{j,1}$ where $j<i$.

Although a terminal can appear in more than one similarity according to the conditions above, we can observe the following facts.

- If a terminal appears in both α_1 and α_2, it must be appear n times, where $n \geq 1$, in both sentences. Otherwise, they cannot have a minimal match sequence.
- If a terminal appears more than once in both α_1 and α_2, its i^{th} occurrence in α_1 and its i^{th} occurrence in α_2 must end up in the same similarity of their minimal match sequence.

For example, the minimal match sequence of the sentences $abcbd$ and $ebfbg$ will be $(a,e)b(c,f)b(d,g)$. But, sentences $abcbd$ and ebf cannot have a minimal match sequence because b occurs twice in the first sentence and b occurs only once in the second sentence.

Now we give an important theorem about the minimal match sequences.

Theorem 1 *. If two sentences α_1 and α_2 have a minimal match sequence, it is unique. (i.e. if it exists,there is only one minimal match sequence for α_1 and α_2.)*

Proof. Let us assume that M_1 be a minimal match sequence for two sentences α_1 and α_2, and M_2 be another minimal match sequence (different from M_1) for these sentences. Since M_1 and M_2 are different minimal match sequences, at least one of two following cases must be true:

1. A terminal appearing in a similarity of M_1 must appear in a difference of M_2 or,
2. A terminal appearing in a difference of M_1 must appear in a similarity of M_2.

In the first case, a same terminal must be appear in both a similarity and a difference of M_2, or it must appear in both the first constituent of a difference and the second constituent of a difference of M_2. But, both of them are not possible according to the definition of minimal match sequence. The second case is also impossible, because the terminal in question does not appear in both sentences. So, two sentences cannot have two different minimal match sequences. □

Now, we give an algorithm to compute a minimal match sequence for given two sentences in Figure 1 and 2. This algorithm finds a minimal sequence or it indicates that there is no minimal match sequence for given two sentences. Figure 1 contains the main function, and Figure 2 contains an auxiliary function of the algorithm. Functions *minimalM* with 3 arguments and *minimalMD* with 5 arguments are recursive functions, and they call each other in the computation of a minimal match sequence. The reader will notice that the algorithm tries to satisfy the conditions which are required for minimal match sequences.

3 Learning Heuristics

A translation template is an *atomic* or *general* translation template. A *atomic translation template* between languages $\mathcal{L}^a$ and $\mathcal{L}^b$ is a pair of two sentences $\alpha \leftrightarrow \beta$ where $\alpha \in \mathcal{L}^a$ and $\beta \in \mathcal{L}^b$. A given *translation example* will be an atomic translation template.

A *general translation template* between languages $\mathcal{L}^a$ and $\mathcal{L}^b$ is an if-then rule in the following form:

$T^a \leftrightarrow T^b$ **if** $X_1 \leftrightarrow Y_1$ **and** ... **and** $X_n \leftrightarrow Y_n$

where $n \geq 1$, T^a is a string of terminals in the alphabet of the language $\mathcal{L}^a$ and variables $X_1, ..., X_n$, T^b is a string of terminals in the alphabet of the language $\mathcal{L}^b$ and variables $Y_1, ..., Y_n$, and both T^a and T^b must contain at least one terminal.

For example, if the alphabet of $\mathcal{L}^a$ is $\mathcal{A} = \{a, b, c, d, e, f, g, h\}$ and the alphabet of $\mathcal{L}^b$ is $\mathcal{B} = \{t, u, v, w, x, y, z\}$, the followings are some examples of general templates between $\mathcal{L}^a$ and $\mathcal{L}^b$.

- $abX_1c \leftrightarrow uY_1$ **if** $X_1 \leftrightarrow Y_1$
- $aX_1bX_2c \leftrightarrow Y_2uvY_1$ **if** $X_1 \leftrightarrow Y_1$ **and** $X_2 \leftrightarrow Y_2$
- $aX_1X_2b \leftrightarrow Y_2vY_1$ **if** $X_1 \leftrightarrow Y_1$ **and** $X_2 \leftrightarrow Y_2$

A general template is a generalization of translation examples, where certain components are generalized by replacing them with variables and establishing bindings between these variables. For example, in the first example above, abX_1c represents all sentences of $\mathcal{L}^a$ starting with ab and ending with c where X_1 represents a non-empty string on $\mathcal{A}$, and uY_1 represents all sentences of $\mathcal{L}^b$ starting with u where Y_1 represents a non-empty string on $\mathcal{B}$. That general template says that a sentence of $\mathcal{L}^a$ in the form of abX_1c corresponds to a sentence of $\mathcal{L}^b$ in the form of uY_1 given that X_1 corresponds to Y_1. If we know that the correspondence $de \leftrightarrow vyz$, the correspondence $abdec \leftrightarrow uvyz$ can be inferred from that general template.

A *minimal match sequence between two translation examples* $\alpha_1 \leftrightarrow \beta_1$ and $\alpha_2 \leftrightarrow \beta_2$ is a pair of two minimal match sequences $M^a \leftrightarrow M^b$ where M^a is the minimal match sequence of α_1 and α_2 and M^b is the minimal match sequence of β_1 and β_2. If a minimal match sequence exists for a pair of translation examples, it will be unique. An instance of $M^a \leftrightarrow M^b$ is a pair of an instance of M^a and an instance of M^b. If both instances are similarity instances, we call the resulting instance as a similarity instance of $M^a \leftrightarrow M^b$, and as a difference instance of $M^a \leftrightarrow M^b$ if both of them are difference instances.

Our learning heuristics learn translation templates from given two translation examples. To learn translation templates from given two examples, first a minimal match sequence of these examples is found; then learning heuristics are applied to the instances of this minimal match sequence. A learning heuristic learns a general template by replacing similarities or differences with variables in a match sequence, and establishing bindings between these variables. In addition, a learning heuristic can also learn atomic templates. Of course, if there is no minimal match sequence for examples, learning heuris-

```
minimalM(α1, α2) { % where α1 and α2 are two sentences
  return (minimalM(α1, α2, ϵ)) }
minimalM(α1, α2, M) {
  if ((α1 = ϵ) and (α2 = ϵ)) return(M);
  else if ((α1 = ϵ))
    if (a terminal in α2 appears in a Ci,1 of M)
      exit(no-minimal-match-sequence);
    else return(M ‖ (ϵ, α2));
  else if ((α2 = ϵ))
    if (a terminal in α1 appears in a Ci,2 of M)
      exit(no-minimal-match-sequence);
    else return(M ‖ (α1, ϵ));
  else {
    ft1 ← first terminal of α1;
    ft2 ← first terminal of α2;
    if (ft1 = ft2) {
      • find the longest prefix between α1 and α2,
        and call it similarity S;
      • drop S from both α1 and α2;
      if (a terminal in S appears in a difference of M)
        exit(no-minimal-match-sequence);
      else return(minimalM(α1, α2, M ‖ S)); }
    else if (ft1 appears in α2) {
      • let α2 = α2,1 ft1 α2,2
        where α2,1 does not contain ft1;
      if (a terminal in α2,1 appears in a Ci,1 of M)
        exit(no-minimal-match-sequence);
      else return(minimalM(α1, ft1α2,2, M ‖ (ϵ, α2,1))); }
    else if (ft2 appears in α1) {
      • let α1 = α1,1 ft2 α1,2
        where α1,1 does not contain ft2;
      if (a terminal in α1,1 appears in a Ci,2 of M)
        exit(no-minimal-match-sequence);
      else return(minimalM(ft2α1,2, α2, M ‖ (α1,1, ϵ))); }
    else {
      • drop ft1 from α1;
      • drop ft2 from α2;
      return(minimalMD(α1, α2, M, ft1, ft2)); } } }
```

Figure 1: Minimal Match Sequence Algorithm

```
minimalMD(α1, α2, M, d1, d2) {
  if ((α1 = ε) and (α2 = ε))
    if (a terminal in d2 appears in a Ci,1 of M
        or a terminal in d1 appears in a Ci,2 of M)
      exit(no-minimal-match-sequence);
    else return(M ‖(d1, d2));
  else if ((α1 = ε))
    if (a terminal in d2 ‖α2 appears in a Ci,1 of M
        or a terminal in d1 appears in a Ci,2 of M)
      exit(no-minimal-match-sequence);
    else return(M ‖(d1, d2 ‖α2));
  else if ((α2 = ε))
    if (a terminal in d1 ‖α1 appears in a Ci,2 of M
        or a terminal in d2 appears in a Ci,1 of M)
      exit(no-minimal-match-sequence);
    else return(M ‖(d1 ‖α1, d2));
  else {
    ft1 ← first terminal of α1;
    ft2 ← first terminal of α2;
    if (ft1 appears in α2) {
      • let α2 = α2,1 ft1 α2,2
        where α2,1 does not contain ft1;
      if (a terminal in d2 ‖α2,1 appears in a Ci,1 of M
          or a terminal in d1 appears in a Ci,2 of M)
        exit(no-minimal-match-sequence);
      else return(minimalM(α1, ft1α2,2, M ‖(d1, d2 ‖α2,1))); }
    else if (ft2 appears in α1) {
      • let α1 = α1,1 ft2 α1,2
        where α1,1 does not contain ft2;
      if (a terminal in d1 ‖α1,1 appears in a Ci,2 of M
          or a terminal in d2 appears in a Ci,1 of M)
        exit(no-minimal-match-sequence);
      else return(minimalM(ft2α1,2, α2, M ‖(d1 ‖α1,1, d2))); }
    else {
      • drop ft1 from α1;
      • drop ft2 from α2;
      return(minimalMD(α1, α2, M, d1 ‖ft1, d2 ‖ft2)); } } }
```

Figure 2: Minimal Match Sequence Algorithm (cont.)

tics cannot be applied to those examples. In addition, each learning heuristic may insist extra conditions on minimal match sequences.

3.1 Replacing Differences with Variables

Our first learning heuristic tries to learn new translation templates by replacing differences with variables in match sequences, and establishing bindings between these variables. To able to apply this heuristic to the minimal match sequence of two translation examples $E_1 = \alpha_1 \leftrightarrow \beta_2$ and $E_2 = \alpha_2 \leftrightarrow \beta_2$, both the minimal match sequence of α_1 and α_2 and the minimal match sequence of β_1 and β_2 must contain at least one similarity and one difference, and both of them cannot contain a difference with empty constituent.

This heuristic can learn new templates from each similarity instance $M^a \leftrightarrow M^b$ of the minimal match sequence of translation examples E_1 and E_2 satisfying the following conditions:

1. Both M^a and M^b must contain at least one similarity and one difference. This condition will be automatically satisfied by the instance, because we insist that the original minimal match sequence must satisfy this condition.
2. Both M^a and M^b cannot contain a difference with empty constituent.
3. Both M^a and M^b must contain n differences where $n \geq 1$. In other words, they must contain equal number of differences.
4. Each difference in M^a must correspond to a difference in M^b, and a difference cannot correspond to more than one difference in other side. Thus, we will have n difference correspondings.

If we a match sequence satisfying the first three conditions, n difference correspondings must be found to satisfy the fourth condition. For example, if there are two differences D_1^a and D_2^a in M^a, and two differences D_1^b and D_2^b in M^b; we cannot determine whether D_1^a corresponds to D_1^b or D_2^b without using prior knowledge. Now, let us assume that the corresponding of D_1^a to D_1^b has been learned earlier; in this case D_2^a must correspond to D_2^b. In general, if $n-1$ correspondings of differences have been learned earlier, the last two differences must correspond to each other. We say that the corresponding of differences $D^a = (D_1^a, D_2^a)$ and $D^b = (D_1^b, D_2^b)$ has been learned, if the following two atomic translation template have been learned earlier.

$D_1^a \leftrightarrow D_1^b$
$D_2^a \leftrightarrow D_2^b$

Now, let us assume that the differences in M^a are D_1^a,....,D_n^a and the differences in M^b are D_1^b,....,D_n^b where D_i^a corresponds to D_i^b. In this case, first n–1 correspondings have been learned earlier, and the corresponding of D_n^a and D_n^b is inferred now. The learning heuristic replaces each D_i^a with the variable X_i to create M^aDVars from M^a, and each D_i^b with the variable Y_i to create M^bDVars from M^b. Then, it learns the following general template.

$M^aDVars \leftrightarrow M^bDVars$
  if $X_1 \leftrightarrow Y_1$ **and** ... **and** $X_n \leftrightarrow Y_n$

In addition, the following two atomic templates are learned from the inferred corresponding of $D_n^a = (D_{n,1}^a, D_{n,2}^a)$ and $D_n^b = (D_{n,1}^b, D_{n,2}^b)$.

$D_{n,1}^a \leftrightarrow D_{n,1}^b$
$D_{n,2}^a \leftrightarrow D_{n,2}^b$

3.2 Replacing Similarities with Variables

Our second learning heuristic tries to learn new translation templates by replacing similarities with variables in match sequences, and establishing bindings between these variables. To able to apply this learning heuristic to difference instances of the minimal match sequence of two translation examples, the minimal match sequence has to contain at least one similarity and one difference at both sides.

A difference instance $M^a \leftrightarrow M^b$ of the minimal match sequence must satisfy the following conditions so that this learning heuristic can be applied to it.

1. Both M^a and M^b must contain at least one similarity and one difference. This condition will be automatically satisfied by the instance, because we insist that the original minimal match sequence must satisfy this condition.
2. Both M^a and M^b must contain n similarities where $n \geq 1$. In other words, they must contain equal number of similarities.

3. Each similarity in M^a must correspond to a similarity in M^b, and each similarity cannot correspond to more than one similarity in other side. Thus, we will have n similarity correspondings.

If we a match sequence satisfying the first three conditions, n similarity correspondings must be found to satisfy the fourth condition. For example, if there are two similarities S_1^a and S_2^a in M^a, and two similarities S_1^b and S_2^b in M^b; we cannot determine whether S_1^a corresponds to S_1^b or S_2^b without using prior knowledge. Now, let us assume that the corresponding of S_1^a to S_1^b has been learned earlier; in this case S_2^a must correspond to S_2^b. In general, if $n-1$ correspondings of similarities have been learned earlier, the last two similarities must correspond to each other. We say that the corresponding of similarities S^a and S^b has been learned, if the following atomic translation template has been learned earlier.

$S^a \leftrightarrow S^b$

Now, let us assume that the similarities in M^a are S_1^a,....,S_n^a and the similarities in M^b are S_1^b,....,S_n^b where S_i^a corresponds to S_i^b. In this case, first $n-1$ correspondings have been learned earlier, and the corresponding of S_n^a and S_n^b is inferred now. The learning heuristic replaces each S_i^a with the variable X_i to create M^aSVars from M^a, and each S_i^b with the variable Y_i to create M^bSVars from M^b. Then, M_1^aSVars and M_2^aSVars are created from M^aSVars by replacing differences with their first constituents and with their second constituents, respectively. Similarly, M_1^bSVars and M_2^bSVars are created from M^bSVars. The learning heuristic will infer the following first general template if both M_1^aSVars and M_1^bSVars contain at least one terminal, and it will infer the second one if both M_2^aSVars and M_2^bSVars contain at least one terminal.

$M_1^aSVars \leftrightarrow M_1^bSVars$
  if $X_1 \leftrightarrow Y_1$ **and** ... **and** $X_n \leftrightarrow Y_n$
$M_2^aSVars \leftrightarrow M_2^bSVars$
  if $X_1 \leftrightarrow Y_1$ **and** ... **and** $X_n \leftrightarrow Y_n$

In addition, the following atomic template is learned from the inferred corresponding of S_n^a and S_n^b.

$S_n^a \leftrightarrow S_n^b$

4 Application to Example-Based Machine Translation

Our learning heuristics can be used in the learning of translation templates from a given bilingual corpus for two natural languages. To learn translation templates, the learning heuristics should be applied to every pair of atomic translation templates in the system. Given translation examples are also treated as atomic translation templates, in fact learning starts from those examples. Learning should continue until no more new templates can be learned from atomic translation templates. The learned translation templates can be used in the translation of other sentences in both directions.

The learning heuristics can work on surface level representation of sentences. However, in order to generate useful templates, it is helpful to use the lexical representation. In this case, the set of all root words, all prefixes, and all suffixes in a natural language is treated as the alphabet of that language for our purposes. So, a natural language is treated as the set of all meaningful strings on that alphabet. Normally, given translation examples should be sentences of those natural languages, but they can also be phrases in those languages. Of course, morphological analyzers will be needed for both languages to compose the lexical forms of sentences.

An example-based machine translation system using our learning heuristics can work as follows:

1. Sets of bilingual translation examples should be collected. Since sentences should be in the lexical representations for meaningful translation templates, the lexical representations of words in examples should be found using morphological analyzers.

2. Translation templates should be learned from these sets of bilingual translation examples using our learning heuristics. The learning heuristics should be applied for every pair of atomic translation templates until no more translation templates can be learned. At the end of this step, we will have a set of translation templates. Words in those templates will be in the lexical form, and these templates can be used in both directions of translations. In this step, a confidence factor can also be assigned to each translation template to indicate how good is that translation template. To able to assign these confidence factors [15], statistical techniques based on the information available in the sets of translation examples can be used.

3. To translate a sentence from one language to another, first the lexical representation of that sentence is created using a morphological analyzer of the source language. Using the learned translation templates, possible translations of this sentence are found. Using confidence factors assigned to templates, solutions are sorted with respect to calculated confidence factors of solutions. At the end, we hope that the top results contain good translations and the correct translation is among the top results. After solutions are converted into surface level representations by using the morphological analyzer of the target language, a human expert can choose the correct solution by just looking the top results.

To explain the behavior of our learning heuristics on the actual natural language sentences, we give a simple learning example for translation examples between English and Turkish. Assume that we have the translation examples '`I will drink water` ↔ `su içeceğim`' and '`I will drink tea` ↔ `çay içeceğim`' between English and Turkish. Their lexical representations are '`I will drink water` ↔ `su iç`+FUT+1SG' and '`I will drink tea` ↔ `çay iç`+FUT+1SG' where +FUT and +1SG denote future tense and first singular agreement morphemes in Turkish, respectively. For these two examples, the minimal match sequence will be '`I will drink (water,tea)` ↔ `(su,çay) iç`+FUT+1SG'. From this match sequence the first learning heuristic learns the following 3 templates.

`I will drink` $X_1 \leftrightarrow Y_1$ `iç`+FUT+1SG
  if $X_1 \leftrightarrow Y_1$
`water` ↔ `su`
`tea` ↔ `çay`

In addition, the second heuristic learns the following 3 templates.

X_1 `water` $\leftrightarrow$ `su` Y_1
  if $X_1 \leftrightarrow Y_1$
X_1 `tea` $\leftrightarrow$ `çay` Y_1
  if $X_1 \leftrightarrow Y_1$
`I will drink` $\leftrightarrow$ `iç`+FUT+1SG

5 Conclusion

In this paper, we presented a model for learning translation templates between two languages. The most important part of this model is two learning heuristics that are based on a simple pattern matcher. Translation templates are directly learned from sets of translation examples without using other knowledge resources such as lexicons, grammars, and ontology. The knowledge resources required for this technique are sets of translation examples and morphological processors for the languages.

We believe that humans learn general sentence patters using similarities and differences between many different example sentences that they are exposed to. This observation led us to idea that general sentence patterns can be thought to a computer using learning heuristics based on similarities and differences in sentence pairs. In the sense that our mechanism is close to how the humans learn languages from examples.

The learning technique described in this paper can be used in an incremental manner. Initially, a set of translation templates is learned from a set of translation examples. When a new set of translation examples is available, new translation templates can be learned from this new set together with previously learned atomic translation templates.

The learning heuristics are used in a example-based machine translation system between English and Turkish. In this system, we got very promising results. Although in that system examples between English and Turkish, we believe that the techniques are also applicable for other language pairs. In fact, to test this claim we have also applied this technique between English and French using a small set of translation examples. We got comparable results that we got for the system of English and Turkish.

References

[1] Brona, C., and Padraig, C., Translating Software Documentation by Example: An EBMT Approach to Machine Translation, in: *Proceedings of Int. ECAI Workshop: Multilinguality in the Software Industry*, 1996.

[2] Brown, R. D., Example-Based Machine Translation in the Pangloss System, in: *Proceedings of COLING-96*, 1996.

[3] Brown, R. D., Automated Dictionary Extraction for "Knowledge-Free" Example-Based Translation, in: *Proceedings of TMI'97*, 1997.

[4] Cicekli, I., and Güvenir, H. A., Learning Translation Rules From A Bilingual Corpus, in: *Proceedings of the 2nd International Conference on New Methods in Language Processing (NeMLaP-2)*, Ankara, Turkey, September 1996, pp:90-97.

[5] Furuse, O., and Iida, H., Cooperation between Transfer and Analysis in Example-Based Framework, in: *Proceedings of COLING-92*, Nantes, France, 1992, pp:645-651.

[6] Güvenir, H.A., and 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 1996, pp:121-131.

[7] Güvenir, H. A, and Cicekli, I., Learning Translation Templates from Examples, in: *Information Systems*, Vol. 23, No. 6, 1998, pp: 353-363

[8] Kaji, H., Kida Y., and Morimoto, Y., Learning Translation Templates from Bilingual Text, in: *Proceedings of COLING-92*, 1992, pp:672-678.

[9] Kitano, H., A Comprehensive and Practical Model of Memory-Based Machine Translation, in: *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1993, pp:1276-1282.

[10] Lonsdale, D., Mitamura, T., and Nyberg, E., Acquisition of Large Lexicons for Practical Knowledge-Based MT, in: *Machine Translation*, Vol 9(3), Kluwer Academic Publishers, 1994, pp:251-283.

[11] Medin, D.L., and Schaffer, M.M. Context Theory of Classification Learning, *Psychological Review, 85*, 1978, pp:207-238.

[12] Mitumura, T., and Nyberg, E., The KANT System: Fast, Accurate, High-Quality Translation in Practical Domains, in: *Proceedings of COLING-92*, Nantes, France, 1992, pp:1069-1073.

[13] Nagao, M. A., Framework of a Mechanical Translation between Japanese and English by Analogy Principle, in: *Artificial and Human Intelligence*, A. Elithorn and R Banerji (eds.), NATO Publications, 1984.

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

[15] Öz, Z., and Cicekli, I., Ordering Translation Templates by Assigning Confidence Factors, in: *Lecture Notes in Computer Science 1529*, Springer Verlag, 1998, pp:51-61.

[16] Sato, S., and Nagao, M., The Memory-Based Translation, in: *Proceedings of COLING-90*, 1990.

[17] Sato, S., MBT2: A Method for Combining Fragments of Examples in Example-Based Translation, *Artificial Intelligence*, Vol 75, Elsevier Science, 1995, pp: 31-30.

[18] Smadja, F., McKeown, K. R., and Hatzivassiloglou, V., Translating Collocation for Bilingual Lexicons: A Statistical Approach in: *Computational Linguistics*, Vol 22(1), The MIT Press, 1996, pp:1-38.

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

[20] Wu, D., Xia, X., Large-Scale Automatic Extraction of an English-Chinese Translation Lexicon in: *Machine Translation*, Vol 9, Kluwer Academic Publishers, 1995, pp:285-313.