LEARNING PART-OF-SPEECH TAGGED TRANSLATION TEMPLATES FROM BILINGUAL TRANSLATION EXAMPLES

Hande Doğan
Department of Computer Science, Bilkent University
handed@cs.bilkent.edu.tr

Abstract

In this paper, a method for learning translation templates is presented. A translation template is generalized form of sentences which gives the word order and determines the type phrases that are to be replaces for translation.Translation templates are learned using analogical reasoning. The correspondences in both languages are represented in form of translation templates. A translation template (similarity translation template) is learned from two pairs of translation examples by replacing the differences with PoS tagged variables.

Keywords: exemplar-based machine learning, translation templates with PoS tags, similarity translation templates

1. Introduction

Translation had always been a complex cognitive progress, so computational (automatic) translation does. Makato Nagao, who had first proposed, example based machine translation (he had actually proposed as *machine translation by analogy*), inspired this idea from the necessity to help Japanese people learn a second language like English. He had modeled the learning process as: a Japanese man is given short and simple English sentences with their Japanese correspondences; he memorizes these pairs and then becomes able to translate new sentences via these pairs in the memory. Actually this learning pattern summarizes the basic principles of example based machine translation (EBMT).

> *"Man does not translate a simple sentence by doing deep linguistic analysis, rather, man does translation, first, by properly decomposing an input sentence into certain fragmental phrases,…then by translating these phrases into other language phrases, and finally by properly composing these fragmental translations into one long sentence. The translation of each fragmental phrase will be done by the analogy translation principle with proper examples as its reference." (Nagao, 1984)*[3]

Nagao's this statement identifies the translation process using EBMT approach:

- Matching fragments in database of examples (translation pairs)
- Specifying corresponding translation fragments
- Recombine results from previous steps to get the target text.

In this work, I will propose an extension for an existing method, in order to learn translation templates from examples of translation pairs. Translation templates are generalized form of sentences that are used for translating new sentences.

The paper is organized as follows: In the next section, I will briefly discuss the method that my work is based on and some other previous works. In section 2, I will introduce the translation template term. The learning algorithm and details are given in section 3. After evaluating performance of the system in section 4, I will come up with a brief summary of example based machine translation and give the place of the learning algorithm in EBMT process.

2. Previous Work

As mentioned before, the extended algorithm is based on Çiçekli&Güvenir's work [1], which infers translation templates from bilingual examples. In this method, the translation templates are not part-of-speech [1](PoS) tagged, that means templates do not carry any information about the grammar. Consequently usage of templates that are not PoS tagged may cause the translator to produce incorrect sentences.

There is also another system called EDGAR [2] developed by Michael Carl which uses PoS tag information. I have inspired the representation of the PoS tagged translation templates from this system.

3. Translation Templates

In [1], a translation template is defined as generalized form of the translation pairs where some lexical items are replaced with variables in both sentences, in this work I will propose an extension where the variables that are used for replacing are PoS tagged. A translation template determines the word order for both source and target language, and in this work a template also determines the type of phrases. Let's take the example given in [1]:

I will drink orange juice ↔ portakal suyu içeceğim
I will drink coffee ↔ kahve içeceğim

Using these examples, we can infer the following translation templates:

$$\text{I will drink } X_1 \leftrightarrow X_2 \text{ içeceğim}$$

if X_1 and X_2 are translations of each other. Extending the variables with PoS tags will yield the following template:

$$\text{I will drink } X_1^{NP} \leftrightarrow X_2^{NP} \text{ içeceğim}$$

With the following atomic translation template[2]

$$(\text{orange juice})^{NP} \leftrightarrow (\text{portakal suyu})^{NP}$$

$$(\text{coffee})^{NP} \leftrightarrow (\text{kahve})^{NP}$$

In previous work [1], there are two different kinds of translation templates:

- Similarity templates: based on non-empty sequence of common items in both sentences.
- Difference templates: contains a pair of two sequences from language L1 and language L2 where these subsequences do not contain any common item.

In this work, I will only infer similarity templates from the examples.

4. Learning Similarity Templates

Let's take the following examples as translation pairs:

John gave me his book ↔ John bana kitabını verdi
John gave me his pencil ↔ John bana kalemini verdi

These examples are the surface level representations, if they are morphologically analyzed, we get the following pairs:

John give+PAST me his book ↔ John bana kitap+ACC+3 SG ver+PAST+3 SG
John give+PAST me his pencil ↔ John bana kalem+ACC+3 SG ver+PAST+3 SG

The underlined items are the similar parts in two examples, so when we replace the different part(s) with variables, we will be able to infer a template for that sentence pair:

John give+PAST me X ↔ John bana Y ver+PAST+3 SG
If $X^{NP} \leftrightarrow Y^{NP}$
book ↔ kitap
pencil ↔ kalem

General form of a template containing a single difference is as follows:

$$S_0^1\ D_0^1\ S_1^1 \leftrightarrow S_0^2\ D_0^2\ S_1^{2\,3}$$

[1] If you are unfamiliar with grammatical notation and abbreviations, please refer to the Appendix-I part of this paper.
[2] An atomic translation template is the translation template that does not contain variables.

[3] Here S_0^1 denotes the similar subsequence number 0 in language 1, D_0^1 different subsequence number 0 in language 2.

There are two main points that must be taken in care: firstly the number of differences in both pairs is equal and secondly for now the number of different constituents is one.

As Çiçekli & Güvenir stated in their paper [1], when the number of differences is equal but greater than one (m = n > 1), there should exist prior knowledge to infer templates for that kind of sentences. So at this point, the learning procedure depends on translation templates that have been learned previously. In other words, if we have n differences on both side and if we have learned the templates for (n-1) differences, we can infer a similarity template for that sentence.

I gave her a flower ↔ Ona çiçek verdim
You gave her a present ↔ Ona hediye verdin

For the sake of full understanding of similarities, if we take the sentences with morphological analysis:

I give+PAST her a flower ↔ O+DAT çiçek ver+PAST+1 SG
You give+PAST her a present ↔ O+DAT hediye ver+PAST+2 SG

Now we have two differences, at this point if we do not have atomic templates for one of the difference pairs, we cannot infer whether "I" is the translation of "çiçek" or "+1 SG". But if we have learned from the previous examples, that "I" is the translation of "+1 SG", then it would be no hard work to infer that "flower" is the translation of "çiçek". The translation template will look like:

X_1^{NP} give+PAST her a Y_1^{NP} ↔
Ona Y_2^{NP} ver+PAST+X_2^{NP}
if X_1^{NP} ↔ X_2^{NP}
and
Y_1^{NP} ↔ Y_2^{NP}
$(\text{flower})^{NP}$ ↔ $(\text{çiçek})^{NP}$
$(\text{present})^{NP}$ ↔ $(\text{hediye})^{NP}$

5. Modification of the Algorithm

As I have stated before, this work is built on the algorithms developed in [1], and offers an enhancement with PoS tagging to carry the template learning to a point that takes the grammatical correctness into account.

As explained in the previous section, for similarity template generation the different part(s) in the sentences are replaced with variables and the important point is that while replacing these variables, the PoS tag information must be attached to the variable. This brings a modification for both the parallel corpus and template learning algorithm. The changes in the corpus representation will be explained later.

With the modification for the PoS tagged variables, the algorithm is given in Figure 1.

The match sequence ($M_{a,b}$) that is given as input is in the form:

$S_0^1, D_0^1, \ldots, D_{n-1}^1, S_n^1 \leftrightarrow S_0^2, D_0^2, \ldots, D_{m-1}^2, S_m^2$

6. Performance and Evaluation

The original algorithm with no PoS tag information was implemented in Prolog, but I begin to implement this modified version in Java. As the implementation is still in progress, for the time being I cannot give performance results, unfortunately. It is not wrong to say that PoS tagging phase will not change the learner's efficiency dramatically.

The original algorithm's complexity is $O(n^3)$ at worst case, in practice it is $O(n^2)$ [1] with the modification that I am making for PoS tagging, the in practice complexity will be $O(n^2)$+PoS tagger's complexity (constant), so overall complexity will be $O(n^2)$.

In [1] there are two kinds of translation templates inferred from the example translation pairs, similarity and difference templates. In this work, only similarity templates are inferred from the example pairs, also difference templates can be inferred as an extension.

From another point of view, one can use semantic information along with PoS tag information for translation. As known PoS tagging prevents the production of grammatically wrong sentences, with semantic

procedure SimilarityTTL_POS($M_{a,b}$)
begin

if (n = m = 1) **then**

- Infer the templates:
  Find the PoS tags of X and Y
  $S_0^1 (X)_{POS} S_1^1 \leftrightarrow S_0^2 (Y)_{POS} S_1^2$ **if** $(X)_{POS} \leftrightarrow (Y)_{POS}$
  $(D_{0,a}^1)_{POS} \leftrightarrow (D_{0,a}^2)_{POS}$
  $(D_{0,b}^1)_{POS} \leftrightarrow (D_{0,b}^2)_{POS}$

else if (n = m > 1**) then**

if (n-1) difference pairs can be found **then**

- Let the unpaired correspondence be
  $(D_{kn,a}^1, D_{kn,b}^1), (D_{ln,a}^2, D_{ln,b}^2)$
- Assume that known corresponding list is (already have POS tags)
  $(D_{k1}^1, D_{l1}^2)^{POS} \ldots\ldots (D_{kn}^1, D_{ln}^2)^{POS}$
- For each corresponding difference $(D_{ki}^1, D_{li}^2)^{POS}$ replace D_{ki}^1 with X_i^1 and D_{ki}^2 with X_i^2 to get the new match sequence $M_{a,b}^{new}$
- Infer the following templates:
  Find the PoS tags of X_n^1 and X_n^2
  $M_{a,b}^{new}$ **if** $(X_1^1)^{POS} \leftrightarrow (X_1^2)^{POS}$ and …… and $(X_n^1)^{POS} \leftrightarrow (X_n^2)^{POS}$
  $(D_{kn,a}^1)^{POS} \leftrightarrow (D_{ln,a}^2)^{POS}$
  $(D_{kn,b}^1)^{POS} \leftrightarrow (D_{ln,b}^2)^{POS}$

end

Figure 1- Similarity Template Learning Algorithm with PoS tagged variables

I
information we can also prevent the production of meaningless sentences.

7. Example Based Machine Translation

The algorithm that I have described so far is the analysis part of the known Example Based Machine Translation process. As seen in Figure 2, the whole process is composed of matching, alignment and recombination steps.

Matching phase is the most important step of translation. In this step, the corpus is searched for the source sentence, to find the best match example for it. In the transfer phase, the source sentence is translated via selected template to the target language. The last step is the recombination phase, as the transferred sentence is in lexical form, in this step the target sentence is generated in surface form.

Figure 3 summarizes the whole example based machine translation process and place of the template learning in the process.

APPENDIX-I

Grammatical Representation

For the sake of completeness, in this part I will give the grammatical notation that will help the reader to understand the grammatical representation of the words, morphemes etc.

There are two representations for a word:

- Surface level: is the representation that we are used to see a word.
- Lexical level: is the morphological representation of a word.

Figure 4 shows both the surface and lexical level representation of the Turkish word "geldim".

Surface level	**geldim**
Lexical level	**gel+PAST+1 SG**

Figure 4-Surface and lexical level representations

Table 1 gives the abbreviations for the affixes that are used in this paper:

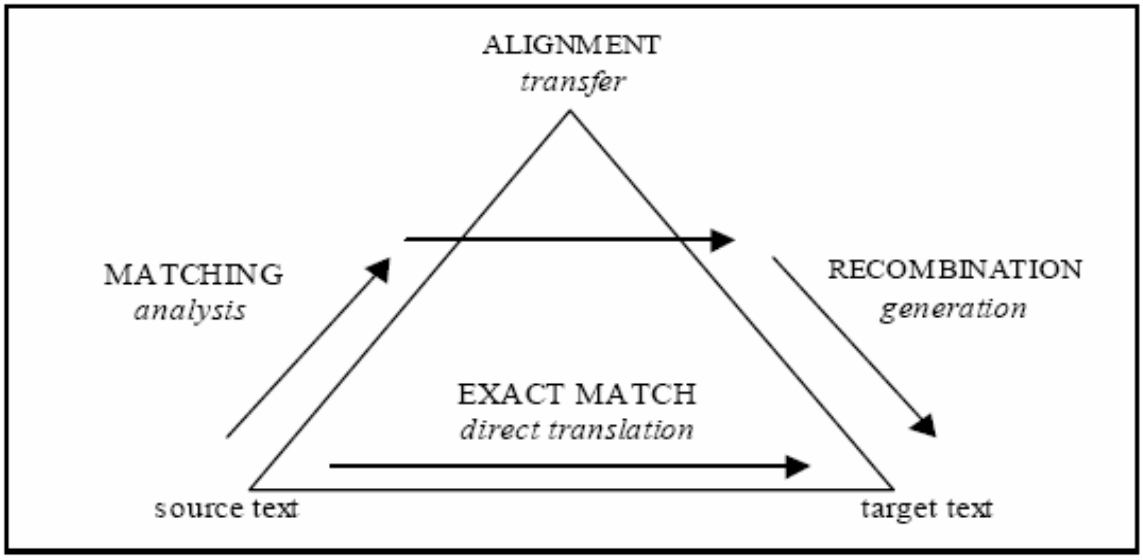

Figure 2 - "Vauquois Pyramid" represents the EBMT process

PAST	past tense morpheme
1 SG / 2 SG / 3 SG	1./ 2./ 3. singular agreement morphemes
1 PL / 2 PL / 3 PL	1./ 2./ 3. plural agreement morphemes
ACC	accusative case morpheme
DAT	dative case morpheme

Table 1 - Abbreviation list for used morphemes

Finally, the part-of-speech tagging (PoS) term refers to assigning a part-of-speech (verb, noun etc.) to each word in a corpus. For example for the phrase "the book on the table" can be PoS-tagged as follows:

((The(**det**) book(**noun**))**NP**
((on(**preposition**)) (the(**det**)
table(**noun**))**NP**)**PP**)**NP**

det: determiner
NP: noun phrase
PP: prepositional phrase

References

[1] Çiçekli I., Güvenir H.A., "Learning Translation Templates from Bilingual Translation Examples"

[2]Carl M, "Inducing Translation Templates for Example Based Machine Translation"

[3]Somers H, "An Overview of EBMT"

[4]Turcato D., Popowich F., "What is Example-Based Machine Translation"

[5] Somers H., Collinc B., "EBMT seen as Case-Based Reasoning"

[6]Jurafsky&Martin, "Speech and Language Processing", Prentice-Hall 2000

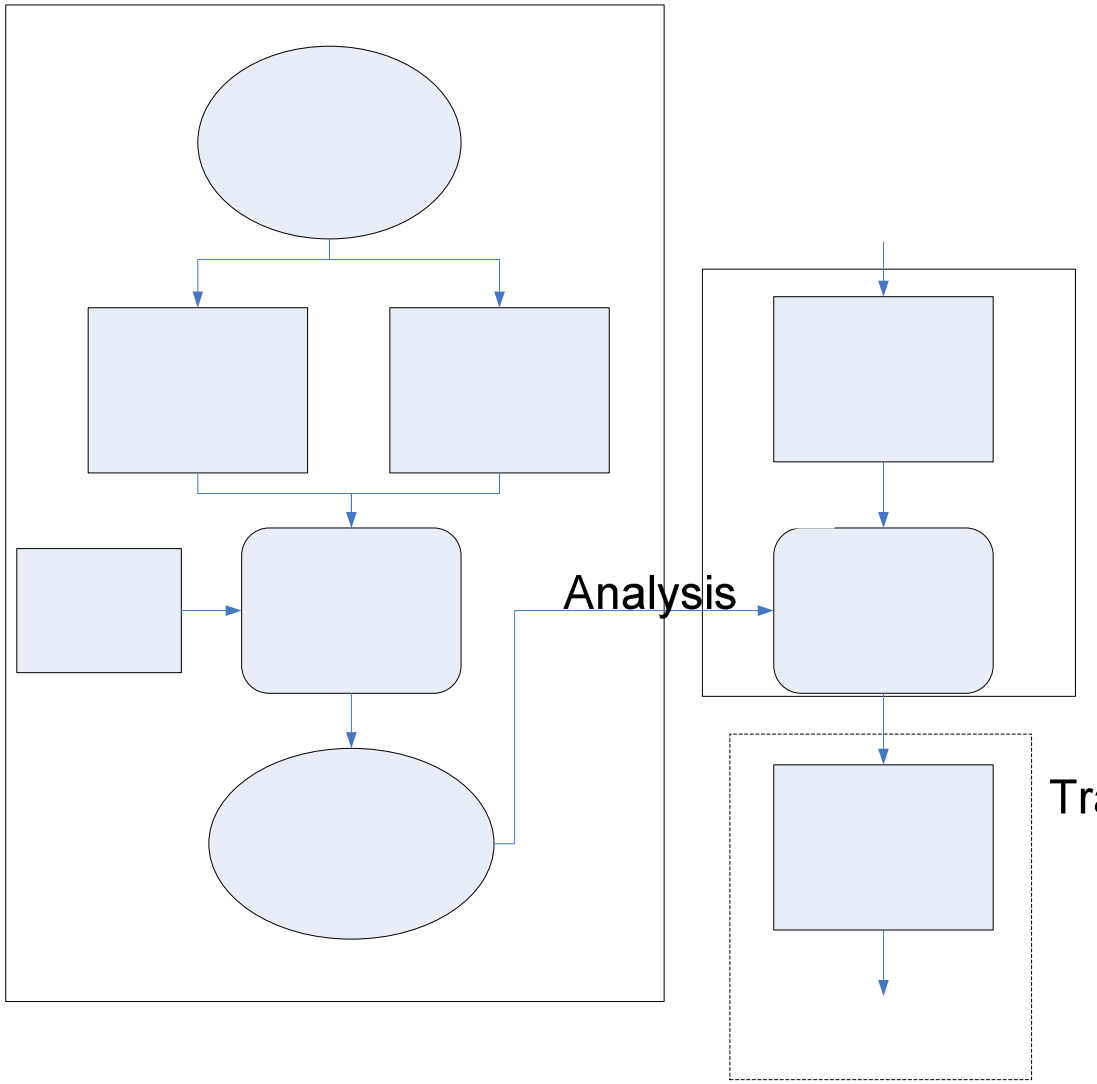

Figure 3 - Summary of EBMT process and Learning Algorithm's place in EBMT

Morphological Analyser for Language L_1