Ordering Translation Templates by Assigning Confidence Factors*

Zeynep Öz and Ilyas Cicekli

Dept. of Comp. Eng. and Info. Sc., Bilkent University,
06533 Bilkent, Ankara, TURKEY,
{ozzey,ilyas}@cs.bilkent.edu.tr

Abstract. TTL (*Translation Template Learner*) algorithm learns lexical level correspondences between two translation examples by using analogical reasoning. The sentences used as translation examples have similar and different parts in the source language which must correspond to the similar and different parts in the target language. Therefore these correspondences are learned as translation templates. The learned translation templates are used in the translation of other sentences. However, we need to assign confidence factors to these translation templates to order translation results with respect to previously assigned confidence factors. This paper proposes a method for assigning confidence factors to translation templates learned by the TTL algorithm. Training data is used for collecting statistical information that will be used in confidence factor assignment process. In this process, each template is assigned a confidence factor according to the statistical information obtained from training data. Furthermore, some template combinations are also assigned confidence factors in order to eliminate certain combinations resulting bad translation.

1 Introduction

Traditional approaches to machine translation (MT) require detailed knowledge about languages and the world knowledge. Therefore corpus-based machine translation is a good alternative for avoiding them. *Example-based* machine translation (EBMT) is one of the main approaches of corpus-based machine translation and originally proposed by Nagao [12]. This approach is based on the idea of performing translation by imitating translation examples of similar sentences. It involves translating the source language into the target language via remindings from the previous translation cases as stated in Brona [5]. After this proposal several machine translation methods that utilize translation examples and bilingual corpora have been studied such as [13, 15, 16, 1]. EBMT is the marriage of the MT and *Case-based reasoning* techniques (CBR). Finding the

* 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.

correspondence of units in a bilingual text, retrieving the best matches from previous translation examples, and producing the translation of the given input by using these examples are the fundamental phases in EBMT. Brown [2] and Gale [6] have proposed methods for establishing correspondence between sentences in bilingual corpora. Brown [3], Sadler [14] and Kaji [9] have tackled the problem of establishing correspondences between words and phrases in bilingual texts.

Statistical machine translation is another approach of corpus-based machine translation. Statistical MT techniques use statistical metrics to choose the best structures in the target language among all possible candidates. These techniques are useful for retrieving the best matches from the previous translation examples, which is a vital issue in EBMT. This fact motivated us to develop a machine translation system that is a combination of statistical MT and EBMT.

Using previous examples for learning from new examples is the main idea behind exemplar-based learning which is originally proposed by Medin and Schaffer [11]. This way of learning stores the examples in memory without any change in the representation. The characteristic examples stored in the memory are called exemplars.

In the translation process, providing the correspondences between the source and target languages is a very difficult task in EBMT. Although, manual encoding of the translation rules has been achieved by Kitano [10], when the corpus is very large, it becomes a complicated and error-prone task. Therefore Cicekli and Güvenir [7, 4] offered a technique in which the problem is taken as a machine learning task. Exemplars are stored in the form of templates that are generalized exemplars. 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 variables. These templates are learned by using translation examples and finding the correspondences between the patterns in the source and target languages. The heuristic of the translation template learning (TTL) [7, 4] algorithm can be summarized as follows: Given two translation pairs, if there are some similarities in the source language, 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. Similar parts are replaced with variables to get a template which is a generalized exemplar by this method. Translation examples are stored as a list of string formed by strings of root words and morphemes. In other words, the lexical level representation of the sentences are used.This representation of translation examples is suitable for learning algorithm. If we used surface level representation, the number of correspondences would be decreased and we could learn less number of generalized exemplars. For example the sentence pair **i came from school⇔ben okuldan geldim** is stored as:

i come+p from school⇔ben okul+DAn gel+DH+m

where *i, come, from, school* denote root words and *+p* denotes the past tense morpheme in English sentence, and *ben, okul, gel* denote root words and *+DAn, +DH, +m* denote ablative, past tense and first singular person morphemes in

Turkish sentence. The following translation pairs given in English and Turkish illustrates the heuristic:

<u>I go+p to</u> school <u>by bus</u> ⇔ okul <u>+yA otobüs+ylA git+DH+m</u>
<u>I go+p to</u> city <u>by bus</u> ⇔ şehir <u>+yA otobüs+ylA git+DH+m</u>

The similarities between the translation examples are underlined. The similarities in English are represented as **I go+p to** X^{L_1} **by bus**, and the corresponding similarities in Turkish as X^{L_2}**+yA otobüs+ylA git+DH+m** by replacing differences by variables. According to the heuristic, these similarities should correspond to each other. Here, X^{L_1} denotes a component that can be replaced by any appropriate structure in English and X^{L_2} refers to its translation in Turkish. In addition to this, it is also inferred that *school* is the translation of *okul* and *city* is the translation of *şehir*. This shows that it is possible to learn more than one template by using two translation examples.

The order of the translation templates that will be used for the translation of new sentences is an important fact for the soundness of the outputs, however, the early versions of the algorithm uses a simple criterion for the order of the translation templates inferred. We need to assign confidence factors, i.e., weights, to these translation templates to have more accurate translations. Confidence factor assignment is done by using training data and collecting some statistical information. In the learning phase of the algorithm, each template is given a template number. Since translation is bidirectional, templates (specific templates without variables and generalized ones with variables) are assigned two weights, one for left to right usage and one for right to left usage of that template by using the translation examples. In addition to these, some template combinations are also assigned confidence factors in order to eliminate bad translation results. Translation accuracy is increased by using these weights. In the translation process, the output translations which have the highest weights are selected among all possibilities. Thus, it is ensured that the correct answer will be among these selected output.

The rest of the paper is organized as follows. Section 2 explains the confidence factor assignment process to the templates. Translation algorithm is described in Section 3. In Section 4 performance results of the system are provided. Section 5 concludes the paper and gives some future directions.

2 Methods for Assigning Confidence Factors

The translation templates are ordered according to the number of terminal symbols of the templates in the previous version of TTL algorithm [7]. However, this criteria is not sufficient for large systems, and we need another method where a statistical method is a powerful candidate, in order to improve the soundness of the translation process. Therefore, in the new version of the TTL algorithm, learning translation templates is followed by a confidence factor assignment process in which each rule and some rule combinations are assigned weights. Our main resource for assigning confidence factor is the training data that is used in the learning of translation templates. This process has three fundamental parts: Confidence factor assignment to facts (i.e. specific templates without variables), rules (i.e. generalized templates in which the similarities are replaced with variables) and rule combinations. These three parts are explained in detail in the following sections.

Our translation process is bidirectional. In other words, it is possible to give an input sentence in language L_1 and obtain a translation in language L_2 and vice versa. Therefore we have templates that will be used for translation from L_1 to L_2, (left to right) and from L_2 to L_1(right to left).

2.1 Method for Assigning Confidence Factors to Facts

In this section confidence factor assignment to facts, which are the simplest case of this process, are discussed. We do not need to consider any other rule during this process and we use only the translation examples.

Consider the case that, $rule_k$ is a fact which will be used for left to right translation. Assume that, it is in the form of $X \Leftrightarrow Y$ and we have training pairs in the form of $trainpair(X_i, Y_i)$ then the confidence factor of $rule_k$ for left to right translation is evaluated as follows:

- $N1$ denotes the number of training pairs where X is a substring of X_i and Y is a substring of Y_i
- $N2$ denotes the number of training pairs where X is a substring of X_i and Y is not a substring of Y_i
- $confidencefactor_{rule_k} = \frac{N1}{N1+N2}$

If $rule_k$ is a fact which will be used for right to left translation, everything will be the same except definition of $N2$

- $N2$ denotes the number of training pairs where X is not a substring of X_i and Y is a substring of Y_i

Now, we illustrate how to find the confidence factor of a fact by giving an example. Let us assume that our all training pairs are as follows:

he come+s⇔gel+Hr
he go+s⇔git+Hr
book+s⇔kitap+lAr
pen+s⇔kalem+lAr

where +Hr and +lAr denote present tense and plural morphemes in Turkish. If we want to find the confidence factor of the rule **+s** $\rightarrow$ **Hr** (this rule is a fact and it will be used in left to right translation) which has been learned from these training pairs, we can find its confidence factor as follows:

$N1 = 2$ from pairs 1 and 2
$N2 = 2$ from pairs 3 and 4
$confidencefactor_{rule} = \frac{N1}{N1+N2} = \frac{2}{2+2} = 0.5$

If *rule* is **+s**$\leftarrow$ **+Hr**, (i.e. it is a fact and it will be used in right to left translation), we can find its confidence factor as follows:

$N1 = 2$ from pairs 1 and 2
$N2 = 0$ no such pair
$confidencefactor_{rule} = \frac{N1}{N1+N2} = \frac{2}{2+0} = 1.0$

It is possible to have the same confidence factor for left to right and right to left usage of the same rule, but it is more probable to have different values. For example, in the following example we have same confidence factors in both direction.

1) if *rule* is **come**$\rightarrow$ **gel** (i.e. it is a fact and it will be used in left to right translation), and our translation examples are the same with the previous example. Then we will find confidence factor of *rule* as:

$N1 = 1$ from pair 1
$N2 = 0$ no such pair
$confidencefactor_{rule} = \frac{N1}{N1+N2} = \frac{1}{1+0} = 1.0$

2) if *rule* is **come**$\leftarrow$ **gel** (i.e. it is a fact and it will be used in right to left translation), we will find confidence factor of *rule* as:

$N1 = 1$ from pair 1
$N2 = 0$ no such pair
$confidencefactor_{rule} = \frac{N1}{N1+N2} = \frac{1}{1+0} = 1.0$

2.2 Method for Assigning Confidence Factors to Rules

Assigning confidence factor to a rule, (a template that has variables in it) is a more complicated task if we try to find the confidence factor of that rule completely. Therefore, if $rule_k$ has variables which will be unified with other rules in the translation phase then we will assign a partial confidence factor to this rule by considering the parts which do not include variables according to the confidence factor formula used in the previous section. In the translation process, the variables are bound using some other rules or facts, and we find the whole confidence factor of this rule by multiplying the confidence factors of all rules which are used to bind the variables. The following is an example for this:

If $rule_k$ is

X^{L_1}**+s**$\Leftrightarrow$ X^{L_2}**+Hr** if $X^{L_1} \Leftrightarrow X^{L_2}$

and our training pairs are the same with the previous example. Since X^{L_1}**+s** can be a substring of left sides of all pairs and $\mathbf{X}^{L_2}$**+Hr** can be a substring of right sides of pairs 1 and 2 by assuming that the variables can match one or more tokens of the string (i.e. variables can not match empty string), we will get the following confidence factor for left to right usage:

$N1 = 2$ from pairs 1 and 2
$N2 = 2$ from pairs 3 and 4
$partialconfidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+2} = 0.5$

Since X^{L_2}**+Hr** can be a substring of right sides of pairs 1 and 2 and $\mathbf{X}^{L_1}$**+s** can be a substring of left sides of pairs all pairs by assuming that the variables can match one or more tokens of a string, we will find the following confidence factor for right to left usage:

$N1 = 2$ from pairs 1 and 2
$N2 = 0$ no such pair
$partialconfidencefactor_{rule_k} = \frac{N1}{N1+N2} = \frac{2}{2+0} = 1.0$

In the translation phase, these partial confidence factors are multiplied by the confidence factors of the rules replacing variables to calculate the real confidence factor of that translation output.

2.3 Method for Assigning Confidence Factors to Rule Combinations

The most complicated task of the procedure is the assignment confidence factors to rule combinations. The reason for considering these rule combinations is the following: Although some rules or facts are assigned high confidence factors when they are considered as single rules or facts, they may have a very low confidence factor when they are used with other rules or facts. The algorithm of this assignment process is given in Table 1. The algorithm in Table 1 is used only for left to right translation. This algorithm is repeated for right to left translation by replacing X^{L_1} with X^{L_2}.

Table 1. Algorithm for assigning confidence factor to rule combinations

---

For each training pair $X^{L_1} \Leftrightarrow X^{L_2}$
- Find all corresponding $\mathbf{Xs}^{L_2}$ for X^{L_1} from training pairs
- Find all translations (**Ts**) with their proofs (**Ps**) of X^{L_1} from translation templates where proofs show the rules used in the translation
- For each $T_i \in \mathbf{Ts}$ do the following steps
  If $T_i \in \mathbf{Ts}$ is the same as $X_j \in \mathbf{Xs}^{L_2}$
    - Assign confidence factor of the rule combination $P_i \in \mathbf{Ps}$ as 1

  else
    - Find distances between T_i and each $X_j \in \mathbf{Xs}^{L_2}$
    - Choose the minimum distance **d** among these distances
    - Assign confidence factor of this rule combination $\mathrm{P}_i \in \mathbf{Ps}$ as
      $confidencefactor_{P_i} = \frac{1}{1+d}$

At this point, calculation of the minimum distance between a translation result, T_i, and a part of training pair, $X_j \in \mathbf{Xs}^{L_2}$, needs more explanation. First of all, X_j and T_i are assumed to be points whose coordinates are (Length of X_j, 0) and (Length of Similarities between X_j and T_i, Length of Differences between X_j and T_i) in a two-dimensional space, respectively. Then the distance is calculated by using the Euclidean formula for calculating the distance between two points:

$$distance = \sqrt{(LengthofX_j - LengthofSimilarities)^2 + (LengthofDifferences)^2}$$

Assume that we have X^{L_1}=`you come+p` and we obtained $\mathbf{Xs}^{L_2}$= {`gel+DH+n, siz gel+DH+nHz`} and **Ts**={`gel+Hr+DH+n, gel+DH+nHz`} then confidence factors for rule combinations used to find translations in **Ts** are computed as follows:

1) For T_1=`gel+Hr+DH+n` where T_1 is found by using n rules $i_1, \cdots, i_n$, the confidence factor of the rule combinations $i_1, \cdots, i_n$ is calculated as:

- Find the distance between T_1 and X_1:
  - Since similarities between T_1 and X_1 are [gel,+DH,+n], the length of similarities is 3.
  - Differences between T_1 and X_1 are [[],[+Hr]], and the length of differences is 1, since length of [+Hr] is 1.
  - **d1**=$\sqrt{(3-3)^2+(1)^2} = 1$
- Find the distance between T_1 and X_2:
  - Since similarities between T_1 and X_2 are [gel,+DH], the length of similarities is 2.
  - Differences between T_1 and X_2 are [([siz],[]),([],[+Hr]),([+nHz],[+n])] and the length of differences is 3, since length of [siz] is 1, length of [+Hr] is 1 and length of [+nHz] or [+n] is 1, giving a total of 3.
  - **d2**=$\sqrt{(4-2)^2+(3)^2} = \sqrt{13}$
- **min(d1,d2)=d1**=1 and confidence factor$_{[i_1,\cdots,i_n]}$= $\frac{1}{1+1}$=0.5

2) For T_2=`gel+DH+nHz` where T_2 is found by using m rules $j_1, \cdots, j_m$, the confidence factor of the rule combinations $j_1, \cdots, j_m$ is calculated as:

- Find the distance between T_2 and X_1:
  - Since similarities between T_2 and X_1 are [gel,+DH] and the length of similarities is 2.
  - Differences between T_2 and X_1 are [([+n],[+nHz])] and the length of differences is 1, since length of [+n] or [+nHz] is 1.
  - **d1**=$\sqrt{(3-2)^2+(1)^2} = \sqrt{2}$
- Find the distance between T_2 and X_2:
  - Since similarities between T_2 and X_2 are [gel,+DH,+nHz]] and length of similarities is 3.
  - Differences between T_2 and X_2 are [([siz],[])] and the length of differences is 1, since length of [siz] is 1.
  - **d2**=$\sqrt{(4-3)^2+(1)^2} = \sqrt{2}$

– **min(d1,d2)=d1** or **d2** and confidence factor$_{[j_1,\cdots,j_m]} = \frac{1}{1+\sqrt{2}}$

Note that, the length of differences is calculated by choosing the maximum of lengths in difference pairs.

These rule combinations are represented as tree structures. For example if $rule_i$ has two variables that are bound to $rule_j$ and $rule_k$, then the root of the tree is assumed to be $rule_i$ and its children are $rule_j$ and $rule_k$. If $rule_j$ or $rule_k$ has variables then they become the root of that subtree and their children become the numbers of the rules that are used in the binding of their variables. This tree structure is formed recursively. The tree structure will be helpful during the translation process and its usage will be explained in the next section.

3 Translation Process by Using Confidence Factors

Translation process can be summarized by the four steps given in Table 2. We find all possible translations by using the templates obtained in our learning phase. Then these results are evaluated according to their weights. These weights come either directly from the weights of rules or rule combinations. After the evaluation of the results, the ones that have the highest weights are given as the output, and the ones with lowest weights are eliminated. Therefore the correct output is ensured to be among these selected outputs, and hopefully will be on the top of the selected outputs.

The second step of the algorithm is the most important part of the translation process. Finding the confidence factors of these results is not as simple as it seems. We need both the confidence factors of the rules and rule combinations which are calculated in the learning process. The details of these calculations are given in Table 3. The rules that are pertaining to the result are found and a tree structure is obtained from these rules as explained in Section 2.3. Then this tree structure is used for comparison. If the result does not match a rule combination that is assigned a weight in the learning phase, then the comparison continues among the subtrees.

Table 2. Translation Algorithm

- Find all possible translations and their proofs
- Find confidence factors of these results by using the confidence factors assigned in the confidence factor assignment process.
- If one result is found more than once with different weights use the average of all possibilities for confidence factor.
- Sort results according to the calculated confidence factors in descending order by using a sort algorithm

Table 3. Algorithm for calculating confidence factors of the translations

Find the translation output's confidence factor by using the previously calculated confidence factors of rule combinations

- Find the set of rule combinations ($\mathbf{R}$) which are assigned confidence factors
- If $rp = R_i \in \mathbf{R}$ then cf_{result} =cf$_{R_i}$ where rp is the resulting proof
  else cf_{result} =cf$_{rp_{root}}$* cf$_{rp_{child_1}}$ * cf$_{rp_{child_2}}$ $* \cdots *$ cf$_{rp_{child_n}}$
  where if child$_k$ is a fact ($fact_m$), then cf$_{child_m}$ =cf$_{fact_m}$
  else calculate recursively cf$_{child_k}$ as a tree

4 Performance Results

In this section, the results of the simulation on small corpora are summarized. A training set of examples has contained 488 sentences. Total number of the translation templates that are learned in the learning phase is 4723. In the confidence factor assignment process 4723 templates for left to right usage (from English to Turkish), and 4723 templates for right to left usage (from Turkish to English) are assigned confidence factors. 55845 rule combinations for left to right usage and 53676 rule combinations for right to left usage are assigned confidence factors. Therefore, we obtained a total of 118967 confidence factor assignments.

Table 4. Performance Results

Type of data	Percentage of correct results in translations	Percentage of incorrect results in translations	Percentage of correct results in top 5 without weights	Percentage of correct results in top 5 with weights
Sentences selected from training data	42.0	58.0	44.0	80.0
New sentences not appearing in training data	33.0	67.0	40.0	60.0

In the translation process, we used two groups of sentences to evaluate the performance of the results. The first group of sentences are randomly selected from training data and the second group of sentences are the new sentences which do not occur in the training data. The results are obtained by using the previously assigned weights and they are sorted in ascending order according to these weights. We also produced the outputs without using the weights of the templates for comparison purposes. Then they are sent to the generator to obtain surface forms from the lexical forms. In Table 4 the results with weights and without weights are summarized. The columns denote the percentage of the correct translations among all the results, percentage of the incorrect translations, and percentage of the correct translations seen in the top five results, respectively.

5 Conclusion and Future Work

In this paper, we have presented a statistical model for assigning confidence factors to the translation templates learned by the translation model offered in Cicekli [7, 4]. This translation model learns general translation patterns from the given translation examples by using analogy principle.

The early versions of the algorithm, translation templates are sorted according to their specificities (i.e., the number of terminals in templates). Although this way of sorting gives correct results, the accuracy was not high enough.The major contribution of this paper is assigning confidence factors to templates in order to improve the accuracy. Assigning confidence factor to these rules depends on the statistical data collected from translation examples which are assumed to be grammatically correct. As mentioned before, in the translation process, the output translations which have the highest weights are selected among all possibilities. Thus, it is ensured that the correct answer will be among these selected output and at the top of the list.

The algorithm is tested on Turkish and English for illustration purposes, but it is applicable to any pair of languages. On a small set of data, learning and translation times are reasonable enough. The accuracy of the results are promising. We need to test it on very large corpora. Thus, we are trying to form a large corpus for this purpose. The learning process on a large corpus will take a considerable amount of time, but it can be tolerated since it will be done only once and increase the translation accuracy.

In the future, the system accuracy can be increased by using a human assistance for the verification of the templates, morphological analysis etc. However, in order to fully automate the system, it will be better to use some additional reliable tools for parallel text alignment, disambiguation, etc.

References

1. R. D. Brown. Example-Based Machine Translation in the Pangloss System. In *Proceedings of COLING-96*, 1996.
2. P. F. Brown. Aligning Sentences in Parallel Corpora. In *Proceedings of the 29th Annual Meeting of th ACL*, pp:169-176, 1991.
3. P. F. Brown. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, pp:233-311, 1993.
4. I. Cicekli, and H. A. Güvenir. Learning Translation Rules From A Bilingual Corpus. *Proceedings of the 2nd International Conference on New Methods in Language Processing (NeMLaP-2)*, Ankara, Turkey, September 1996, pp:90-97.

5. B. Collins, and P. Cunningham. A Methodology for Example-Based Machine Translation. *Trinity College*, Dublin, 1995
6. W. A. Gale, and K.W. Church. A Program for Aligning Sentences in Bilingual Corpora. In *Proceedings of the 29th Annual Meeting of th ACL*, pp:177-184, 1991.
7. H. A. Güvenir, and I. Cicekli. Learning Translation Templates from Examples. *Information Systems* (accepted to be published).
8. H. A. Güvenir, and A. Tunc. 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.
9. H. Kaji, Y. Kida, and Y. Morimoto. Learning Translation Templates from Bilingual Text. In *Proceedings of COLING-92*, pp:672-678, 1992.
10. H. Kitano. 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 Volume 2, 1993, pp: 1276-1282.
11. D. L. Medin, and M. M. Schaffer. Context Theory of Classification Learning. *Psychological Review*, 85, 1978, pp:207-238.
12. M. A. Nagao. Framework of a Mechanical Translation between Japanese and English by Analogy Principle. *Artificial and Human Intelligence*, A. Elithorn and R. Banerji (eds.), NATO Publications, 1984.
13. V. Sadler. Working with Analogical Semantics: Disambiguation Techniques in DLT. *Foris Publications*, Dodrecht, Netherlands, 1989.
14. V. Sadler, and R. Vendelmans. Pilot Implementation of a Bilingual Knowledge Bank. In *Proceedings of COLING-90*, pp:449-451, 1990.
15. E. Sumita, H. Iida, and H. Kohyama. Translating with Examples: A New Approach to Machine Translation. In *Proceedings Third International Conference on Theoretical and Methodological issues in Machine Translation of Natural Language*, 1990.
16. E. Sumita, and Y. Tsutsumi. A Translation Aid System using flexible Text Retrieval Based on Syntax Matching. *TRL Res. Report TR-87-1019*, Tokyo Research Laboratory, IBM, Tokyo, Japan, 1988.