

A ranking method for example based machine translation results by learning from user feedback

Turhan Daybelge · Ilyas Cicekli

Published online: 24 March 2010
© Springer Science+Business Media, LLC 2010

Abstract Example-Based Machine Translation (EBMT) is a corpus based approach to Machine Translation (MT), that utilizes the translation by analogy concept. In our EBMT system, translation templates are extracted automatically from bilingual aligned corpora by substituting the similarities and differences in pairs of translation examples with variables. In the earlier versions of the discussed system, the translation results were solely ranked using confidence factors of the translation templates. In this study, we introduce an improved ranking mechanism that dynamically learns from user feedback. When a user, such as a professional human translator, submits his evaluation of the generated translation results, the system learns “context-dependent co-occurrence rules” from this feedback. The newly learned rules are later consulted, while ranking the results of the subsequent translations. Through successive translation-evaluation cycles, we expect that the output of the ranking mechanism complies better with user expectations, listing the more preferred results in higher ranks. We also present the evaluation of our ranking method which uses the precision values at top results and the BLEU metric.

Keywords Example-based machine translation · Translation template ranking

T. Daybelge · I. Cicekli (✉)
Department of Computer Engineering, Bilkent University,
06800 Bilkent, Ankara, Turkey
e-mail: ilyas@cs.bilkent.edu.tr

T. Daybelge
e-mail: daybelge@bilkent.edu.tr

1 Introduction

Example-Based Machine Translation (EBMT), which is regarded as an implementation of the case-based reasoning approach of machine learning, is a corpus-based approach to Machine Translation (MT). EBMT was first proposed by Nagao under the name *translation by analogy* [18]. Translation by analogy is a rejection of the idea that humans translate sentences by applying deep linguistic analyses on them. Instead, it is argued that, humans first decompose the sentence into fragmental phrases, then translates these phrases into phrases in the target language, and finally composes these fragmental translations into a sentence. The translation of fragmental phrases is done in the light of prior knowledge, acquired in the form of translation rules.

In this paper, we propose several improvements to an existing EBMT system [3, 4, 8]. We present here a new method for ranking the translation results generated by this system. Contrary to the previous versions, the results ranking mechanism is dynamically trained by the user. The user feedback is obtained in the form of an evaluation of the generated results. From the evaluation of the user, the system learns *context-dependent co-occurrence rules*, which are later consulted while ranking the results of the subsequent translations. Through successive translation-evaluation cycles, it is expected that the output of the ranking mechanism complies better with user expectations, listing the more preferred results in higher ranks.

When the translation system generates multiple results, either due to the morphological ambiguities or multiple translation template combination options for the translation, the results are presented to the user in descending order of confidence values. In the previous versions of the system, during the translation phase, the user had no effect on the confidence values assigned to each result, hence on

the presentation order of the results. In order to reflect his preference into the ordering of results, the user had to enter more translation examples and rerun the learning component, which consumes computation resources and takes time. Moreover, in a realistic situation, it will be impossible for a user to estimate the number of examples to add, that will adjust the ordering of the results to the desired configuration.

The confidence factors are calculated merely from the translation examples in the learning phase. A problem with this scheme of confidence factor assignment is that, it does not consider the co-occurrence of the translation templates. Certain templates may be assigned low confidence factors when considered individually, but their co-existence in a translation result may require a different treatment, as the combination deserves a higher confidence. The reverse can also be true. Oz and Cicekli [21] proposed a modification to the original scheme of confidence factor assignment, that takes template combinations into consideration. This method calculates the confidence factors for template combinations in the learning phase, and once this is done the factors are never updated.

Moreover, the confidence factors learned from the translation examples in the learning phase may not always overlap with the user expectations. The translation results that are correct for a given context, may be inappropriate for another context. A human translator can translate a phrase differently depending on the characteristics of the context of the text. Besides, different users may perceive the same translation result differently, depending on their background.

We could have encouraged the user, to add more translation examples in order to teach his preferences to the system. By adding enough new translation examples, the user could achieve adjusting the system to give the results that best match his expectations, at the top. The disadvantage of this approach lies in its complexity. An ordinary user would not be able to estimate the number of new examples to add, in order to fine-tune the confidence factors assigned to the templates.

Instead, we propose a different mechanism for incorporating useful *user feedback* into the translation result ordering mechanism, which is one of the new features of our translation system. After each translation, the user has the option of evaluating the translation results in terms of their correctness. The system, using the information gathered by user interactions, ensures that the results marked correct in the evaluation will be ranked above the results that are marked as incorrect, during the next translation of the same phrase. From the evaluation data, the system extracts template co-occurrence rules, which specify aggregate confidence factors for certain template configurations. The extracted rules are then kept in the file system to be used in later translations.

The user interface provides two methods for inputting the user feedback. The first one, *Shallow Evaluation*, lists the search results in their bare surface-level representations; and the user can either mark a result as correct or incorrect. The second type of analysis is *Deep Evaluation*, which is targeted for advanced users and provides the option of evaluating individual nodes of the parse trees built for each translation result. After inputting the user feedback, the system learns context-dependent co-occurrence rules from that information.

The context-dependent co-occurrence rules learned from the user feedback reflect the preferences of a particular user. The translation characteristics vary from one human translator to another, and usually there are numerous correct translations of a given text. Therefore, we use the concept of user profiles in our system. When a user evaluates the results of a translation, the co-occurrence rules learned from the evaluation are kept in his own user profile. Thus, other user profiles in the system are not affected. Also, a single user can create multiple profiles, each of which is used for a different text context—such as science, literature, law, etc.—that has a distinct characteristic.

The rest of this paper is organized as follows. We discuss other machine translation systems that use user feedbacks to improve their performances in Sect. 2. The components that were available at the earlier versions of the presented EBMT system, and its general architecture are presented in Sect. 3. We also discuss the learning algorithms and the induction of confidence factors for extracted translation templates. We discuss the structures of context dependent co-occurrence rules and their usage in the translation phase in Sect. 4. Section 5 discusses how context dependent co-occurrence rules are extracted from user feedbacks. The details of shallow evaluation and deep evaluation of the translation results are also given. In order to test the effects of the context dependent co-occurrence rules in the translation, we conducted a set of evaluations, and the results of these evaluations are presented in Sect. 6. We give the concluding remarks in Sect. 7.

2 Related work

Machine translation systems that acquire their translation rules from bilingual corpora can generate a lot of translation results for a given translation task, and most of the generated translation results can be incorrect. There can be different reasons for these incorrect translations. The acquired transfer rules may contain many redundant and incorrect rules because of the difficulties in the rule induction and the translation varieties in bilingual corpora. The same phrase may correspond to multiple phrases in corpora, and a machine translation system can induce all of these correspondences

although some of them are low frequency. In fact, a correspondence can be treated as a correct translation in certain contexts, and it can be treated as an incorrect translation in some other contexts. According to an experiment conducted by Imamura [11], most of induced transfer rules were low-frequency rules, and they are rarely used in many of the contexts.

Some of the machine translation systems that induce transfer rules can try to avoid the usage of incorrect rules. They mainly use two approaches to overcome this problem. The first approach is the detection and removal of the incorrect rules after the automatic acquisition of transfer rules [11, 12, 15, 16]. The second approach is the selection of appropriate transfer rules during the translation phase, or the ordering of the translation results according to certain metrics [17]. Our EBMT system can be treated as a system that uses the second approach.

Menzes and Richardson [16] uses a cutoff point to avoid the usage of the low-frequency rules. They employ a best-first alignment algorithm to determine high precision transfer mappings. They collect the required information from the training corpora.

Imamura et al. [11, 12] uses a feedback cleaning mechanism which selects and removes the extracted rules in order to increase the BLEU [22] score of an evaluation corpus. After they extract translation rules, examples in the evaluation corpus are translated using these transfer rules. They try to determine the removal of which transfer rules increases the BLUE score of the evaluation corpus. In other words, they clean the incorrect transfer rules according to an evaluation corpus.

Font-Llitjos et al. [15] present a framework that automatically refines transfer rules using the user feedbacks provided by bilingual speakers. In order to get machine translation error information from bilingual speakers, a translation correction tool [14] is used. While Menzes and Richardson [16] and Imamura et al. [12] try to remove redundant and incorrect translation rules after the rule acquisition, Font-Llitjos et al. [15] try to refine translation rules by editing incorrect translations that are determined with the user feedbacks.

Gough and Way [9] used a marker-based sub-sentential alignment algorithm in order to reduce the size of the marker-based lexicon. With this approach, they try to avoid the learning of incorrect translation rules and increase the precision of their system.

In statistical machine translation, variations of probabilistic synchronous context-free grammars (CFGs) are used in order to sort the translation results [1, 7, 19, 24, 26]. The conditional probabilities of the target language rules with respect to the source language rules in probabilistic synchronous CFG production rules are used in order to find the probabilities of the translation results. Our previous works in [2–4, 10, 21] may have similarity with the methods based on

probabilistic CFGs. However, to the best of our knowledge there does not exist any statistical machine translation system based on the probabilistic synchronous CFG formalism that uses a user-feedback mechanism to rearrange the probability values of the rules. The main contribution of this paper is the introduction of a user-feedback mechanism in order to capture user preferences in the forms of context-dependent co-occurrence rules, and use these rules in the ordering of the translation results to accommodate user preferences. We discuss the similarities between our previous work and probabilistic synchronous CFGs in Sect. 3.4.

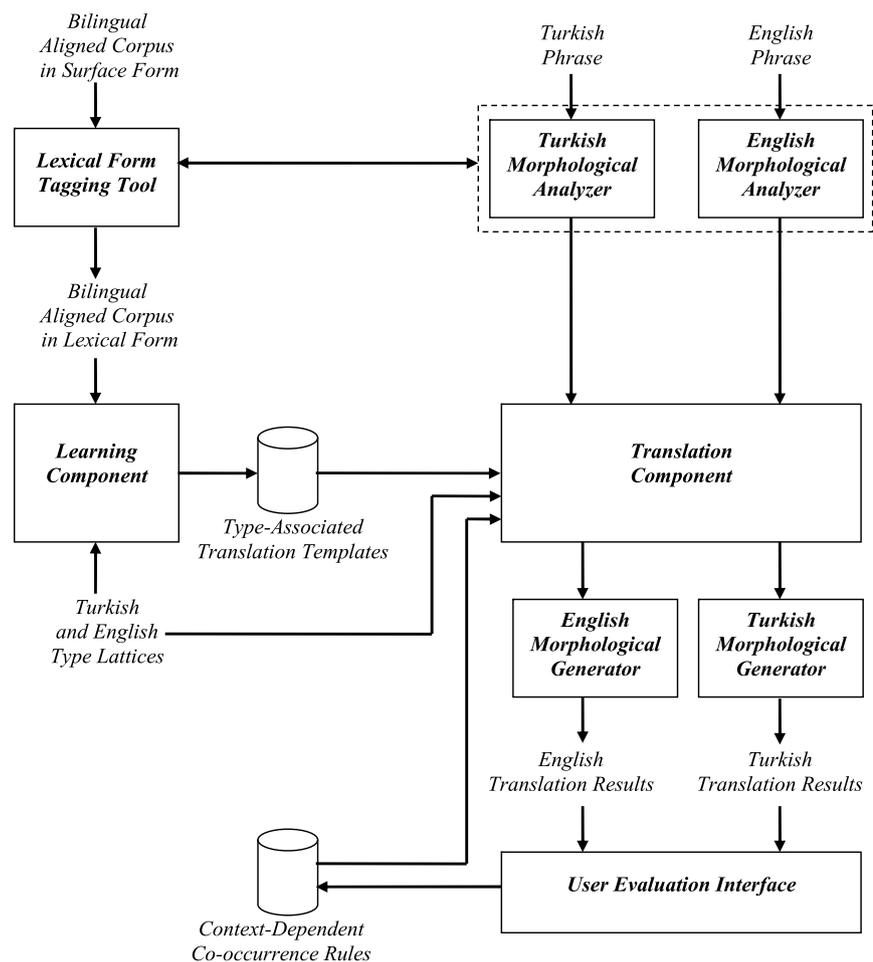
In our EBMT system, we also use a graphical user interface to get the user feedbacks from bilingual speakers, and the user feedbacks are used to determine the correct and incorrect transfer rules. Although we do not remove or refine the determined incorrect rules, we decrease their confidence factors so that they cannot generate higher confidence values than the confidence values generated by the determined correct rules during translation.

All of the systems that try to avoid the usage of incorrect translation rules keep a single set of translation rules either after cleaning or refining the incorrect translation rules. This single set of the resulting transfer rules is used in all contexts. In our system, the user has an option to create more than one set of rules, and each set may reflect a different context such as science, law, etc. We call each set of rules as user profile. Thus, the behavior of our system can be different in each user profile since each of them represents a different set of rules.

3 Architecture of EBMT system

This section describes the interactions among various components that make up our EBMT system. The components that were available in the earlier versions of the system [2–4, 10, 21] are summarized in the rest of this section. The major contribution of this paper, namely user evaluation process, is discussed in Sects. 4 and 5.

Figure 1 depicts the interactions among the components of our system. Tasks of the system can be divided into two phases. In the first phase, the system is trained using a bilingual aligned corpus. The training corpus contains bilingual translation examples in their lexical form. Training of the system finishes when the *Learning Component* writes the translation templates it has learned, into a file. In the first phase, the user is passive, i.e., the learning process is completed without any user interaction. The second phase uses the translation templates extracted in the first phase, in order to translate the natural language phrases that are taken from the user. Unlike the previous phase, the translation phase is interactive, i.e., the *Translation Component* asks the user to enter a phrase in either Turkish or English, and after performing the translation, returns the results back to the user,

Fig. 1 A detailed view of the system components

and waits for the next input. Now, the user has an option to evaluate the translation results. As a result of the user evaluation process, context-dependent co-occurrence rules are extracted, and these co-occurrence rules are used in the ordering of the subsequent translation results.

In order to extract some translation templates, the learning component takes a bilingual corpus as input. This corpus has to be in the lexical-form, since using the lexical-forms of the translation examples enables the system to learn more useful templates when compared to using the surface-forms. Manually converting translation examples in a corpus, from their more natural surface-forms into lexical-forms, without using any software tool, would be an inefficient and error-prone task. Therefore, we have developed a tagging tool [5, 6] that simplifies the conversion process. Thus, all translation examples at the surface level are converted into translation examples at lexical level with the help of this tagging tool.

We used a Turkish morphological analyzer and an English morphological analyzer in the translation phase and during the tagging of translation examples. The Turkish morphological analyzer is a re-implementation of the morphological analyzer described in [20]. Several modifications

have been also introduced to this version of the morphological analyzer, such as re-organizing the output into a more standard format and changing the internal encoding to cover Turkish specific letters [13]. We have also developed an English morphological analyzer in order to get a morpheme representation similar to the morpheme representation used in our Turkish morphological analyzer. Although there are some changes, the parse formatting of our analyzer is very similar to that of the online Xerox morphological analyzer [25] and our Turkish morphological analyzer [13].

In the following subsections, we discuss the details of the learning component. We also show how the extracted translation templates are used in the translation phase. Thus, we review the parts of our EBMT system that were developed at the earlier versions of the system. In addition, we also compare the earlier version of our EBMT system with the statistical machine translation systems based on the probabilistic synchronous CFG formalism in Sect. 3.4.

3.1 Learning translation templates

The learning component [2, 3, 10] infers the translation templates from the set of translation examples. Each transla-

tion example is a pair of an English sentence and a Turkish sentence. The lexical level representations of the examples are used in order to learn more useful translation templates.

In order to induce the translation templates from a given pair of translation examples, we find their match sequence whose first part is a match sequence between English sentences of the examples and the second part is a match sequence of Turkish sentences. A match sequence between two sentences is a sequence of similarities and differences. A similarity is a non-empty sequence of common items in both sentences. A difference is a pair of two subsequences where the first one from the first sentence, the second one is from the second sentence, and they do not have any common items.

One of the learning heuristics described in [2, 3, 10], which is called *similarity template learning*, can induce the translation templates from match sequences by replacing the differences with variables, and setting the correspondences between the differences. When there is only one difference on each side of the match sequence, the translation templates are derived without any prior knowledge. On the other hand, when there are several differences, the correspondences between differences except one of them must be known in order to induce the translation templates. Let us assume that we have the match sequence in (2), which is extracted from the translation examples¹ in (1).

$$\begin{aligned} & \text{I+Pron } \underline{\text{drink+Verb +Past}} \text{ tea+Noun } \underline{\text{+Sg}} \\ & \Leftrightarrow \underline{\text{çay+Noun +A3sg +Pnon +Nom iç+Verb +Past}} \\ & \quad \text{+1Sg} \end{aligned} \quad (1)$$

$$\begin{aligned} & \text{you+Pron } \underline{\text{drink+Verb +Past}} \text{ coffee+Noun } \underline{\text{+Sg}} \\ & \Leftrightarrow \underline{\text{kahve+Noun +A3sg +Pnon +Nom iç+Verb +Past}} \\ & \quad \text{+2Sg} \end{aligned}$$

$$\begin{aligned} & (\text{I+Pron, you+Pron}) \text{ drink+Verb} \\ & \quad \text{+Past (tea+Noun, coffee+Noun) +Sg} \\ & \Leftrightarrow (\text{çay+Noun, kahve+Noun}) \text{ +A3sg} \\ & \quad \text{+Pnon +Nom iç+Verb +Past (+1Sg, +2Sg)} \end{aligned} \quad (2)$$

¹The examples in (1) are the lexical forms of the translation examples “I drank tea \leftrightarrow çay içtim” and “You drank coffee \leftrightarrow kahve içtin”. In the lexical forms of English words, “+Verb”, “+Noun”, and “+Pron” are part of speech tags; “+Sg” indicates that the noun is singular; “+Past” indicates that the verb is in past tense. In the lexical forms of Turkish words, “+Verb”, and “+Noun” are part of speech tags; “+A3sg” indicates that the noun is singular; “+Pnon” indicates that the noun does not have a possessive marker; “+Nom” indicates that the noun is in nominative case; “+Past” indicates that the verb is in past tense.

In order to be able to learn any translation templates, at least one of the correspondence pairs between differences should be known. Assuming that the correspondences “I+Pron \leftrightarrow +1Sg” and “you+Pron \leftrightarrow +2Sg” are known a priori, the translation template learning algorithm extracts the three templates given in (3). One should note that the corresponding variables, namely (X^1, Y^1) , and (X^2, Y^2) , are marked with identical superscripts.

$$\begin{aligned} & X^1 \text{ drink+Verb +Past } X^2 \text{ +Sg} \\ & \Leftrightarrow Y^2 \text{ iç+Verb +Past } Y^1 \text{ +A3sg +Pnon +Nom} \\ & \text{tea+Noun } \Leftrightarrow \text{çay+Noun} \\ & \text{coffee+Noun } \Leftrightarrow \text{kahve+Noun} \end{aligned} \quad (3)$$

The translation templates above are induced by replacing differences with variables. Another learning heuristic, which is called *difference template learning*, replaces the similarities in match sequences in order to deduce translation templates [2, 3, 10]. If there is a single similarity on both sides of the match sequence, then that pair of similarities should be the translations of each other.

3.1.1 Type associated template learning

Although learning by substituting similarities or differences with variables yields templates that can be successfully used by the translation component, the templates are usually over generalized [4]. When the algorithm replaces some parts of the examples with variables, the type information of the replaced parts are lost. When used in translation, such a template may yield unwanted results, since the variables can represent any word or phrase. In order to overcome this problem, each variable is associated with a type information. The template in (3) can be marked with type information as follows

$$\begin{aligned} & X^1_{\text{Pron}} \text{ drink+Verb +Past } X^2_{\text{Noun}} \\ & \Leftrightarrow Y^2_{\text{Noun}} \text{ iç+Verb +Past } Y^1_{\text{VerbAgr}} \end{aligned} \quad (4)$$

In this example, the variable X^1_{Pron} can only be replaced by a pronoun and Y^1_{VerbAgr} can only be replaced by a verb agreement suffix.

In order to assign a type label to a variable that substitutes a difference, the learning component must inspect the constituents of this difference. In general, the type of a root word is its part-of-speech category. For example, the type label of “book+Noun” would be simply “Noun”. On the other hand, the type label of any morpheme that is not a root word would be its own name.² For example, the type label

²In lexical representations, although a root word together with its part of speech tag is treated as a single token, a morpheme itself is treated as a single token.

of “+A1sg”, which is the first person noun agreement morpheme in Turkish, is merely its own name, that is “A1sg”. Let us assume that the learning algorithm tries to replace the difference in (5) with a variable.

$$(come+Verb, go+Verb) \quad (5)$$

Observing that there is a single token in both of the constituents and the types of the tokens match, the variable with type label would be X_{Verb} .

In some cases all of the type labels of tokens in the difference constituents match, however most of the time the situation can be different. Assume that the learning algorithm aims to replace the difference below with a variable.

$$(book+Noun +Sg, house+Noun +Pl) \quad (6)$$

In this case, the first pair of tokens of the difference constituents match in terms of type, but the second pair of tokens, “+Sg” and “+Pl”, which are the singular and plural markers, do not match. In such situations, the learning algorithm should be able to identify the supertype of “+Sg” and “+Pl”. Given that the supertype of “+Sg” and “+Pl” is “NounSufCount”, the variable that replaces the difference in (6) would be $X_{Noun NounSufCount}$.

The hierarchical structure that represents the subtype-supertype relations between the type labels is modeled as a lattice in our system [4, 8]. There are two such lattices, one for *English* and one for *Turkish*. The lattice can be regarded as a directed acyclic graph, if each connection from a subtype to a supertype is a one directional arrow. In the lattice there is a single node at the top of the hierarchy labeled “ANY”. The leaf nodes are tokens that appear in the lexical-level form of the translation examples. The use of the lattice instead of a tree allows situations where a node has multiple parents such as the case of “+A3sg” which can both appear as the singular noun agreement and the 3rd person singular verb agreement. Using the lattice structure, the learning algorithm can assign a label to token pairs by finding the nearest common parent of the two tokens.

3.1.2 Learning from previously learned templates

Although, extracting translation templates from translation example pairs provides an effective learning method, the generality of the learned templates are usually limited. In order to increase the learning effectiveness, we learn from not only example pairs, but also the pairs of the previously learned templates [4].

For example, assume that the translation templates in (7) have been learned earlier from some translation examples. In order to learn new translation templates from these templates, the first thing to do is to extract a match sequence

from them as if they were translation examples. This match sequence is given in (8).

$$\begin{aligned} &at+Adv \text{ least}+Adv X_{NumCard}^1 \text{ book}+Noun \\ &\leftrightarrow en+Adv \text{ az}+Adv Y_{NumCard}^1 \text{ kitap}+Noun \end{aligned} \quad (7)$$

$$\begin{aligned} &at+Adv \text{ least}+Adv \text{ one}+Num+Card X_{Noun}^1 \\ &\leftrightarrow en+Adv \text{ az}+Adv \text{ bir}+Num+Card Y_{Noun}^1 \end{aligned}$$

$$\begin{aligned} &at+Adv \text{ least} \\ &\quad +Adv (X_{NumCard}^1 \text{ book}+Noun, \\ &\quad \text{one}+Num+Card X_{Noun}^1) \\ &\leftrightarrow en+Adv \text{ az} \\ &\quad +Adv (Y_{NumCard}^1 \text{ kitap}+Noun, \text{bir}+Num \\ &\quad +Card Y_{Noun}^1) \end{aligned} \quad (8)$$

Regardless of the fact that the differences in the match sequence contain variables, we can learn the templates given below by running the similarity translation template learning algorithm.

$$\begin{aligned} &at+Adv \text{ least}+Adv X_{NumCard}^1 \text{ Noun} \\ &\leftrightarrow en+Adv \text{ az}+Adv Y_{NumCard}^1 \text{ Noun} \\ &X_{NumCard}^1 \text{ book}+Noun \leftrightarrow Y_{NumCard}^1 \text{ kitap}+Noun \\ &\text{one}+Num+Card X_{Noun}^1 \leftrightarrow \text{bir}+Num+Card Y_{Noun}^1 \end{aligned} \quad (9)$$

Note that learning translation templates from previously learned ones may yield three non-atomic templates. This was not possible when templates were extracted from translation examples.

3.2 Confidence factor assignment

The translation templates generated during the learning phase are stored in the file system in order to be later used in the translation phase. Although the translations of some sentences submitted by the user can be given using a single template, vast amounts of the translations are done using a combination of more than one translation template. During the translation phase, in order to translate a given sentence from the source language to the target language, a parse tree of templates are generated by the translation algorithm.

For most of the inputs, there will be multiple translation results. This is due to the fact that if the learned templates are general enough and numerous, there may exist multiple parse trees that can be used to translate the input phrase. Another factor that increases the number of results is the morphological ambiguities faced when converting the input

from the surface-level to an equivalent lexical-level representation. In our EBMT system each translation result is assigned a confidence value and the results are then sorted in decreasing order of these values. The confidence value of a translation result is calculated as the multiplication of the confidence factors assigned to each template which corresponds to a node in the parse tree built in that particular translation [21].

Since the translation is bidirectional, each translation template is associated with a pair of confidence factors. The first confidence factor is used for the translations from *English to Turkish*, and the second one is used for the translations in the reverse direction. A confidence factor is calculated as

$$\text{confidence factor} = N1 / (N1 + N2) \quad (10)$$

where

- N1 is the number of translation examples containing substrings on both sides that matches the template.
- N2 is the number of translation examples containing a substring only on the source language side that matches the template.

For example, assume that the translation examples file contains only the four examples below.³

1. red+Adj hair+Noun+Sg ↔ kızıl+Adj saç+Noun +A3sg +Pnon +Nom
2. red+Adj house+Noun+Sg ↔ kırmızı+Adj ev+Noun +A3sg +Pnon +Nom
3. red+Adj ↔ kırmızı+Adj
4. long+Adj red+Adj hair+Noun+Sg ↔ uzun+Adj kızıl +Adj saç+Noun+A3sg +Pnon +Nom

In order to assign the confidence factor, which is to be used in English to Turkish translations, to a translation template such as

$$\text{red+Adj } X_{\text{Noun}}^1 \leftrightarrow \text{kırmızı+Adj } Y_{\text{Noun}}^1 \quad (11)$$

each translation example has to be evaluated individually. Initially both N1 and N2 are initialized to 0. The example (1) has a substring on its left side, “red+Adj hair+Noun”, that matches the left hand side of the translation template. But there is no substring on the right that matches the template. So, N2 is incremented by 1. Similarly, the example (2) matches the translation template on the left hand side and it also has a substring on the right, “kırmızı+Adj ev+Noun”, that matches the right hand side of the template. So N1 is 1. The example (3) does not match the template on either side,

³These examples are the lexical forms of the translation examples “red hair ↔ kızıl saç”, “red house ↔ kırmızı ev”, “red ↔ kırmızı”, and “long red hair ↔ uzun kızıl saç”.

so N1 and N2 remain unchanged. The example (4), like the first one, matches only the left hand side; therefore, N2 is incremented to 2. As a result, the English to Turkish confidence factor becomes $1 / (1 + 2) = 0.33$. The reader can verify that the Turkish to English confidence factor becomes 1.0 using the same approach since $N1 = 1$ and $N2 = 0$ for that case.

While we are assigning a confidence factor to a template, we are actually approximating the ratio of the number of times a phrase matched with the source language side of the template is translated to a phrase matching the target language side of the template, to the total number of times such a phrase in the source language is ever translated.

3.3 Using templates in translation

Once the templates are learned from the translation examples in the bilingual corpus file, the learning process is terminated. When the user enters a phrase to the system in order to retrieve its translation, the translation component is responsible for handling this task. The translation component first parses the input phrase using a slightly modified implementation [8] of the Earley parsing algorithm. The Earley parser uses the learned translation templates as its grammatical rules. Since the templates are type associated, type checking is also performed by the translation component.

Parsing becomes successful if at least one parse tree can be built using a subset of the translation templates in the system. Usually, the parsing algorithm produces multiple parse trees, each representing a translation result. Then a translation result is produced merely by substituting each child template with the corresponding variable in the parent template, in a recursive fashion. The generated results may be identical, as there may be multiple ways of reaching the same translation result, or they may be distinct. Some of these results will be incorrect semantically or syntactically due to the inappropriate generalizations during template learning. But, hopefully some correct translation results will also be generated.

For example, assume that the user wants to translate the phrase

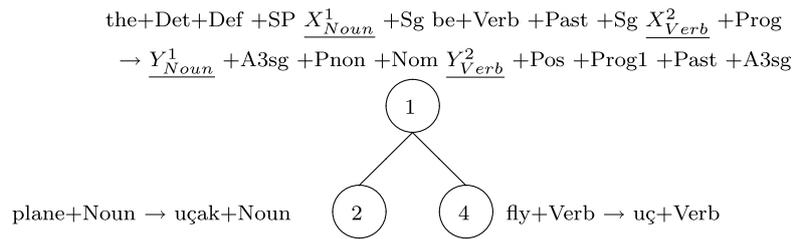
“the plane was flying”, (12)

which can be represented in the lexical form as

$$\begin{aligned} &\text{the+Det+Def +SP plane+Noun} \\ &\quad +\text{Sg be+Verb +Past +Sg fly+Verb +Prog.} \end{aligned} \quad (13)$$

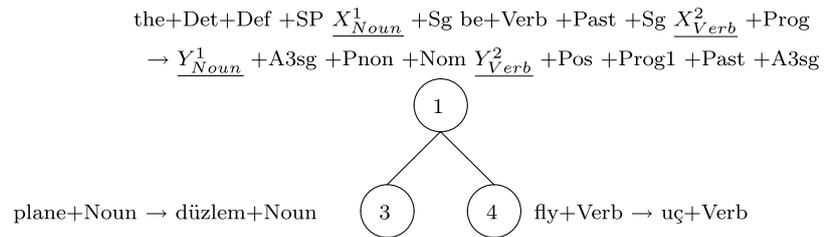
Let us assume that the known translation templates are as follows:

Fig. 2 Translation results for the English phrase “the plane was flying” (the+Det+Def +SP plane+Noun +Sg be+Verb +Past +Sg fly+Verb +Prog) in (12)



(a) uçak uçuyordu

(uçak+Noun +A3sg +Pnon +Nom uç+Verb +Pos +Prog1 +Past +A3sg)



(b) düzlem uçuyordu

(düzlem+Noun +A3sg +Pnon +Nom uç+Verb +Pos +Prog1 +Past +A3sg)

- 1: the+Det+Def +SP X_{Noun}^1
 +Sg be+Verb +Past +Sg X_{Verb}^2 +Prog
 $\leftrightarrow Y_{Noun}^1 + \text{A3sg +Pnon +Nom } Y_{Verb}^2$
 +Pos +Prog1 +Past +A3sg (14)
- 2: plane+Noun \leftrightarrow uçak+Noun
- 3: plane+Noun \leftrightarrow düzlem+Noun
- 4: fly+Verb \leftrightarrow uç+Verb

where the associated English to Turkish confidence factors are 0.9, 0.8, 0.2 and 1.0, respectively. In (14), there are two different translations in Turkish for the noun “plane”. The first meaning is “uçak” (means “airplane”), and the second meaning is “düzlem” (means “flat surface”).

When the parsing algorithm runs on the lexical-level form of the input phrase, the parse trees in Fig. 2 are returned. When the translation is over, the results are presented to the user. Before doing this, the results are sorted in decreasing order of confidence values. As stated earlier, the confidence value of a translation result is determined as the product of the translation templates used in generating that translation result. The confidence value of the translation result in Fig. 2(a) is $0.9 \times 0.8 \times 1.0 = 0.72$, while the confidence value of the translation result in Fig. 2(b) is $0.9 \times 0.2 \times 1.0 = 0.18$. Therefore, the first translation result will be ranked above the second one. This complies with

our expectations, as semantically the first translation result is correct, while the second one is not.

3.4 Comparing translation templates with synchronous grammar rules

Synchronous context-free grammars [1, 23] define the correspondences between the grammatical structures of two languages. A synchronous CFG production rule has two right-hand sides. One of them belongs to a source language and the other belongs to a target language. The nonterminals appearing on both right-hand sides must have one-to-one correspondence. If a nonterminal symbol appears in the source language side, the same nonterminal must appear in the target language side too. A nonterminal symbol X in the source language side must be linked to the nonterminal symbol X in the target language side. For example, the following synchronous CFG rule

$$S \rightarrow \langle NP_1 V P_2, V P_2 N P_1 \rangle$$

indicates that S can be rewritten as $NP VP$ in the source language, and $VP NP$ in the target language. It also says that the nonterminals NP and VP in the source language side are linked to the corresponding nonterminals in the target language side, and the corresponding nonterminals are marked with identical subscripts.

Our basic translation templates without type constraints can be seen as synchronous CFG productions rules. For example, our translation templates in (3) can be rewritten as

the following synchronous CFG rules.

$$S \rightarrow \langle S_1 \text{drink+Verb +Past } S_2 \text{ +Sg,} \\ S_2 \text{ iç+Verb +Past } S_1 \text{ +A3sg +Pnon +Nom} \rangle$$

$$S \rightarrow \langle \text{tea+Noun, çay+Noun} \rangle$$

$$S \rightarrow \langle \text{coffee+Noun, kahve+Noun} \rangle$$

In our translation templates, both sides contain the same number of variables and there is one-to-one correspondence between the variables of both sides. Each variable in the translation templates is representable by a nonterminal symbol, and all variables of the translation templates are associated with the same nonterminal symbol in the synchronous CFG formalism. In fact, the corresponding synchronous CFG will have only one nonterminal symbol when the translation templates are converted into the synchronous CFG rules.

The variations of the probabilistic synchronous CFG are used in the statistical machine translation domain by many researchers [1, 7, 19, 24, 26]. Our translation templates with confidence factors can also be seen as probabilistic CFG production rules. When a translation template is represented as a synchronous CFG rule $S \rightarrow \langle \text{EnglishRule, TurkishRule} \rangle$, the confidence factor of the translation template for the translations from English to Turkish will be the conditional probability $P(\text{TurkishRule}/\text{EnglishRule})$. This conditional probability value is evaluated by dividing the number of the occurrences of *EnglishRule* and *TurkishRule* occurring together in the translation examples by the number of the occurrences of *EnglishRule* alone in the translation examples. Similarly, the confidence factor of the translation template for the translations from Turkish to English will be the conditional probability $P(\text{EnglishRule}/\text{TurkishRule})$.

Although our basic translation templates with confidence factors can be seen as probabilistic synchronous CFG rules, our translation templates have also type constraints and the type constraints can not be easily representable in the synchronous CFG formalism. Our type constraints can be seen as extra restrictions on the bindings of nonterminals to certain terminal strings. In other words, only certain forms of the translation templates discussed in our earlier works [2–4, 10, 21] can be directly representable by the probabilistic synchronous CFG formalism.

In both the earlier version of our EBMT system and the statistical machine translation systems based on the probabilistic synchronous CFG formalism, the learned translation rules are associated with some translation probabilities that are obtained from the translation examples using certain statistical methods. The probabilities of the translation rules are successfully used in the sorting of the translation results. Although the probabilities of the learned translation rules help

the correct translation results appear among the top results in many cases, the incorrect results or the results that would not be preferred by some users may also occur among the top results. There is no easy mechanism to order the translation results according to the user preferences, and our new user-feedback mechanism presented in this paper addresses this problem.

As we discussed earlier, the major contribution of this paper is the usage of a new user-feedback mechanism in order to learn the context-dependent co-occurrence rules. The context-dependent co-occurrence rules reorder the rule confidence factors in order to incorporate the user preferences. We are not aware of any similar mechanism to our context-dependent co-occurrence rule mechanism that is used in a statistical machine translation system based on the probabilistic synchronous CFG formalism.

4 Context-dependent co-occurrence rules

In the previous versions of our system, only the confidence factors associated with translation templates were used for sorting the translation results. This method was not flexible, since the confidence factors were calculated in the learning phase and were not updated throughout the system lifetime. Therefore, we propose the use of *context-dependent co-occurrence rules* in order to incorporate the user preferences into the result ordering mechanism. In our system, context-dependent co-occurrence rules are learned from the user feedback in the translation phase, and continually updated throughout the lifetime of the system.

A context-dependent co-occurrence rule specifies a tree arrangement of the translation templates and a list of contexts, each associated with a separate aggregate confidence factor. For example, the rule

$$1(2, 3(5, 6), 4(7)) - [8_{(2)}, 9_{(4)}](0.7) \quad (15)$$

specifies the template tree $1(2, 3(5, 6), 4(7))$ and it has a single context $[8_{(2)}, 9_{(4)}]$, which is associated with the aggregate confidence value of 0.7. A single template tree can be also associated with several contexts, all of which having a separate aggregate confidence factor. A sample context-dependent co-occurrence rule is

$$1(2, 3) - [4_{(1)}, 5_{(3)}](0.7) - [6_{(1)}, 7_{(4)}, 8_{(2)}](0.9) \quad (16)$$

in which a tree of translation templates, $1(2, 3)$, is followed by two contexts, $[4_{(1)}, 5_{(3)}]$ and $[6_{(1)}, 7_{(4)}, 8_{(2)}]$, associated with aggregate confidence factors 0.7 and 0.9, respectively. The rule (16) is depicted graphically in Fig. 3.

The numbers on the tree nodes denote the unique identifiers associated with each translation template. A context

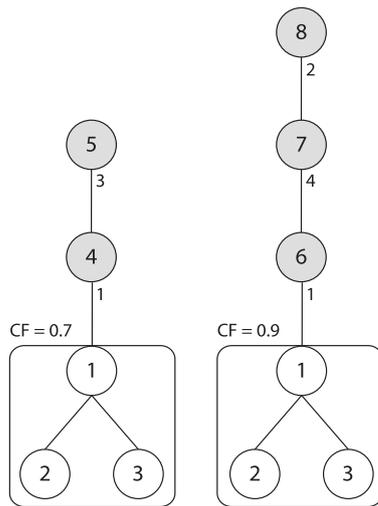


Fig. 3 The context-dependent co-occurrence rule (16)

such as, $[4_{(1)}, 5_{(3)}]$, specifies a sequence of translation templates, where each template is a child of the next template. In addition to that, each parent is marked with a subscript denoting the position of the child in the parent’s list of children. For example, for the context $[4_{(1)}, 5_{(3)}]$, the tree $1(2, 3)$ is the 1st child of template 4; and template 4 is the 3rd child of template 5. The order of the children of a given node is important, e.g., two trees, $1(2, 3)$ and $1(3, 2)$ are not equivalent.

As we aim bidirectional translation, two sets of co-occurrence rules are maintained in the system, one of which is used in English to Turkish translations and the other in the reverse direction. As the user runs the translation component and evaluates the generated translation results, the co-occurrence rules are continually updated, i.e., new rules are learned and context information of existing rules are updated.

4.1 Using the context-dependent co-occurrence rules

A co-occurrence rule specifies an aggregate confidence factor. If the parse tree built for a translation result has a subtree matching the rule, then this aggregate confidence factor overrides the individual confidence factors in that subtree. For example, assume that during the translation of the English phrase

“red haired man” (17)

whose lexical form is

red+Adj hair+Noun +Sg ^DB+Adj+Ed man+Noun +Sg (18)

the translation templates given below are used:

- 1: $X_{Adj\ Noun}^1 +Sg \wedge DB+Adj+Ed X_{Noun}^2 +Sg$
 $\leftrightarrow Y_{Adj\ Noun}^1 +A3sg +Pnon$
 $+Nom \wedge DB+Adj+With Y_{Noun}^2$
 $+A3sg +Pnon +Nom$ (19)
- 2: $X_{Adj}^1 X_{Noun}^2 \leftrightarrow Y_{Adj}^1 Y_{Noun}^2$
- 3: man+Noun \leftrightarrow adam+Noun
- 4: red+Adj \leftrightarrow kızıl+Adj
- 5: hair+Noun \leftrightarrow saç+Noun

where the English to Turkish confidence factors of individual templates are 0.8, 0.7, 1.0, 0.5, and 1.0 respectively. Suppose that the parse tree in Fig. 4 is built during the generation of the translation result:

“kızıl saçlı adam” (20)

whose lexical form is

kızıl+Adj saç+Noun +A3sg +Pnon
 $+Nom \wedge DB+Adj+With$
 adam+Noun +A3sg +Pnon +Nom (21)

The confidence value of this translation is

$$confidence = 0.8 \times 0.7 \times 1.0 \times 0.5 \times 1.0 = 0.28 \quad (22)$$

Now, suppose that a co-occurrence rule that specifies an aggregate confidence factor for the partial translation “red+Adj hair+Noun \rightarrow kızıl+Adj saç+Noun”, such as

$$2(4, 5) - [1_{(1)}](0.9) \quad (23)$$

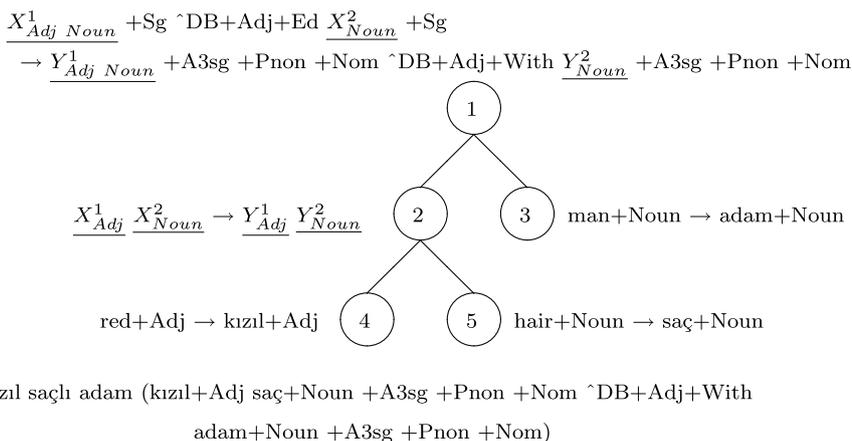
is learned beforehand. Since the template tree specified in the co-occurrence rule matches the subtree $2(4,5)$ in the parse tree of the result and the context of the matching subtree is $[1_{(1)}]$, the aggregate confidence factor specified in the rule overrides the original confidence factors of the nodes in the matching subtree; and the new confidence value of the translation result becomes

$$confidence = 0.8 \times 0.9 \times 1.0 = 0.72 \quad (24)$$

The confidence value calculation method exemplified above can be formalized by the algorithm in Fig. 5. Running this algorithm with the parameter *node* set to the root of the parse tree in Fig. 4 will return the confidence value 0.72 as the result.

CONFVAL-EXACT in Fig. 5 defines the confidence value of a parse tree recursively. If at any point of recursion, a rule matching the subtree rooted at the current parse tree node can be found, and a context matching the context of the current parse tree node is available in the rule, then the

Fig. 4 Parse tree built for the translation of the English phrase “red haired man” (red+Adj hair+Noun +Sg ^DB+Adj+Ed man+Noun +Sg) in (17)



```

CONFVAL-EXACT(node)
  tree ← the tree rooted at node
  context ← the context of node
  if (there exists a co-occurrence rule R that matches tree and
    there exists a context, R_context, in R, where R_context = context) then
    return the aggregate confidence factor associated with R_context
  else
    confidence ← confidence factor of the template represented by node
    children ← {child : child is a child of node}
    for all child ∈ children do
      confidence ← confidence × CONFVAL-EXACT(child)
    return confidence
  
```

Fig. 5 CONFVAL-EXACT. Returns the confidence value of a translation result

associated aggregate confidence value is returned. If these conditions are not satisfied, then the values returned by CONFVAL-EXACT(*child*), for all *child* in the children set of *node*, are multiplied with the confidence factor of the template represented by *node*; and the result of this multiplication is returned.

4.2 Partially matching contexts

CONFVAL-EXACT in Fig. 5 can use a co-occurrence rule in confidence value calculation of a translation result only, if the rule matches the current subtree and the rule contains a context that is identical to the context of the current subtree. If such a rule exists, then the aggregate confidence factor associated with the matching context of the rule is returned immediately; otherwise the confidence value calculation continues recursively. Requiring an exact match of a rule-context with the context of the current subtree can be too restrictive. In this section, we relax this constraint in such a way, that in the absence of an exactly matching context, one or more *partially matching* contexts are used for deriving an aggregate confidence factor.

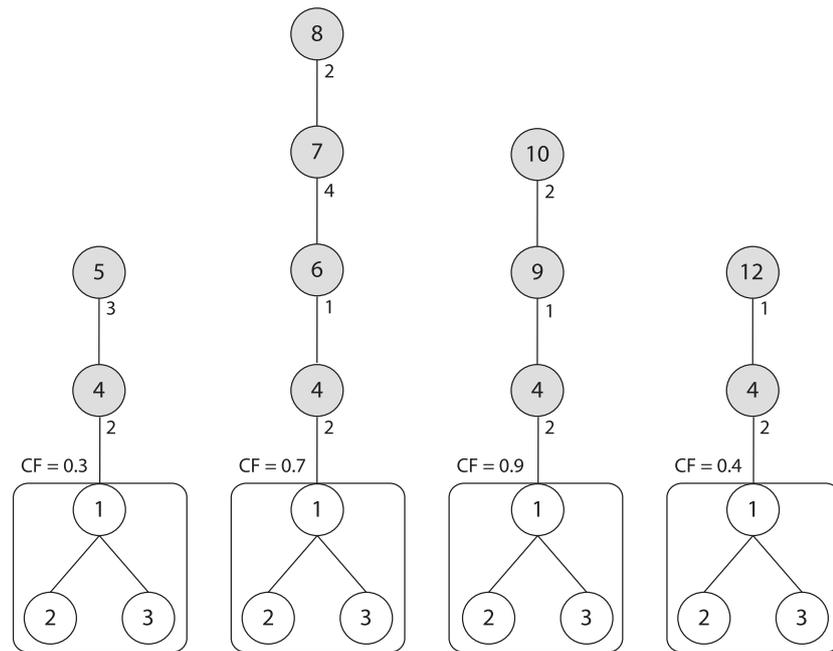
When we allow partial matching of contexts, we should first define a metric that reflects how close a given match is to a perfect one. Therefore, we define our metric, *match_ratio* as

$$\begin{aligned}
 & match_ratio(RC, TC) \\
 &= \begin{cases} 1 & \text{if } length(RC) = 0, \\ \frac{matched(RC, TC)}{length(RC)} & \text{otherwise.} \end{cases} \quad (25)
 \end{aligned}$$

where *RC* is the rule-context, *TC* is the context of the current subtree in the parse tree of the translation result, *length(RC)* is the total number of the elements in *RC*, and *matched(RC, TC)* is the number of matched elements between *RC* and *TC*. Note that the match-ratio calculated for an empty rule-context is always 1. Context matching is done simply by comparing the corresponding elements of a given pair of contexts from left to right, i.e., from child to parent. For example, comparing the contexts [4₍₂₎, 6₍₁₎, 7₍₄₎, 8₍₂₎] and [4₍₂₎, 6₍₁₎, 9₍₂₎] will yield two matching elements, 4₍₂₎ and 6₍₁₎.

The examples in this section use the context-dependent co-occurrence rule given below:

Fig. 6 The context-dependent co-occurrence rule (26)



$$\begin{aligned}
 &1(2, 3) - [4_{(2)}, 5_{(3)}](0.3) \\
 &\quad - [4_{(2)}, 6_{(1)}, 7_{(4)}, 8_{(2)}](0.7) \\
 &\quad - [4_{(2)}, 9_{(1)}, 10_{(2)}](0.9) \\
 &\quad - [4_{(2)}, 12_{(1)}](0.4)
 \end{aligned} \tag{26}$$

The rule above is depicted graphically in Fig. 6. This rule contains four contexts, namely $[4_{(2)}, 5_{(3)}]$, $[4_{(2)}, 6_{(1)}, 7_{(4)}, 8_{(2)}]$, $[4_{(2)}, 9_{(1)}, 10_{(2)}]$ and $[4_{(2)}, 12_{(1)}]$ which are associated with four different aggregate confidence factors, 0.3, 0.7, 0.9 and 0.4, respectively.

Given a current subtree and a rule that matches this subtree, we calculate an aggregate confidence value in three steps. In the first step we calculate match-ratios for all contexts available in the rule. Then we select a subset of the rule-contexts, elements of which are the best matching ones. Finally, we calculate an aggregate confidence factor using the selected subset. A subset of rule-contexts, elements of which match the context of the given subtree best, is selected as follows, and examples are given in Fig. 7:

- Case 1: If there is a unique rule-context with the highest non-zero match-ratio, then only that rule-context is selected.
- Case 2: If there are multiple rule-contexts with the highest non-zero match-ratio, then the longest one of those rule-contexts is selected.
- Case 3: If the longest rule-context is not unique, then all such rule-contexts are selected.

In the last step, the aggregate confidence factor for the current subtree T , and the matching rule R is calculated as

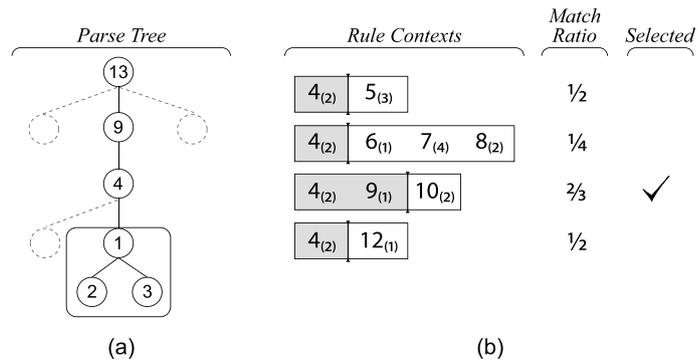
$$\begin{aligned}
 ACF(T, R) &= CV[T] + match_ratio[S] \\
 &\times \left(\frac{\sum_{RC \in S} (ACF[RC])}{|S|} - CV[T] \right)
 \end{aligned} \tag{27}$$

where $CV[T]$ is the original confidence value of T (calculated as the multiplication of the individual confidence factors of the templates in T), S is the selected subset of rule contexts, $match_ratio[S]$ is the match-ratio of the rule-contexts in S (which is shared by all), and $ACF[RC]$ is the aggregate confidence factor associated with the rule-context RC . The calculated aggregate confidence factor approaches to the original confidence value of the subtree, when match-ratio decreases. As the match-ratio increases, it approaches to the average of the aggregate confidence factors associated with the rule-contexts in S . For example, given that the original confidence value of the subtree $1(2, 3)$ in Case 3 of Fig. 7 is 0.6, the aggregate confidence factor calculated for this subtree is

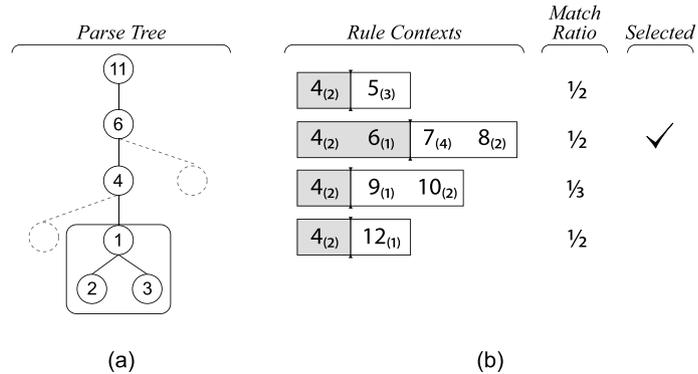
$$0.6 + 0.5 \times \left(\frac{0.3 + 0.4}{2} - 0.6 \right) = 0.475 \tag{28}$$

Up to now, we have studied the cases for which at least one rule-context has a non-zero match ratio. Another case is the one where a context-dependent co-occurrence rule matching the current subtree exists, but all of the contexts have a match ratio of zero. The naive solution is simply calculating the confidence value recursively without using the matching rule, if a rule-context with a non-zero match

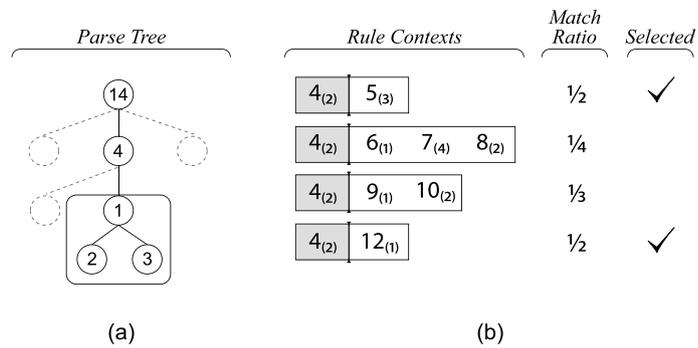
Fig. 7 Partial matching of contexts: (a) An example parse tree where a confidence value will be calculated for the subtree surrounded with the square. Nodes that are not important are drawn in dashed line pattern. (b) The rule contexts, their match ratios, and the selected rule contexts



Case 1. Third context has the highest match-ratio, therefore it is selected for confidence value calculation.



Case 2. As there are multiple rule-contexts with the highest match-ratio, the longest one of those, which is the second rule-context, is selected.



Case 3. As there are two rule-contexts with the highest match-ratio, and the lengths of them are equal, both of the contexts are selected.

ratio is not available. This approach may not satisfy the user expectations. Consider a situation where the user dislikes a combination of templates. He evaluates the combination as incorrect, but the combination appears over and over in completely different contexts. We cannot expect a user to evaluate that combination for all possible contexts. Therefore—even if a non-zero rule-context does not exist—the previous evaluations should influence the confidence value calculated for the current subtree. We achieve this effect by taking the average of the aggregate confidence factors of all rule-contexts, and the confidence value of the subtree is calculated recursively, as given in the equation be-

low:

$$ACF(T, R) = \left(CV_recursive[T] + \sum_{RC \in A} ACF[RC] \right) / (|A| + 1) \tag{29}$$

In this equation $CV_recursive[T]$ is the confidence value calculated for the current subtree T recursively, and A is the set that contains all rule-contexts in the rule. The whole partial context matching process is formalized in Fig. 8, which provides the procedure CONFVAL-PARTIAL.

```

CONFVAL-PARTIAL(node)
  tree ← the tree rooted at node
  context ← the context of node
  rule_found ← false
  if (there exists a co-occurrence rule R that matches tree) then
    Calculate match-ratios for all contexts in R.
    Select the subset S, from the contexts in R, that best match context.
    if (S ≠ ∅) then % use formula (27) in calculation
      return  $CV[T] + match\_ratio[S] \times \left( \frac{\sum_{RC \in S} (ACF[RC])}{|S|} - CV[T] \right)$ 
    else
      rule_found ← true
  % Calculate the confidence value recursively.
  confidence ← confidence factor of the template represented by node
  children ← {child : child is a child of node}
  for each (child ∈ children) do
    confidence ← confidence × CONFVAL-PARTIAL(child)
  if (rule_found = true) then % use formula (29) in calculation
    A ← {RC : RC is a rule context in R}
    return  $\left( confidence + \sum_{RC \in A} ACF[RC] \right) / (|A| + 1)$ 
  else
    return confidence

```

Fig. 8 CONFVAL-PARTIAL. Returns the confidence value of a translation result

5 Learning context-dependent co-occurrence rules

In our EBMT system, the context-dependent co-occurrence rules are learned from the user feedback. After retrieving the translation results, the user has the option of evaluating them. The system provides two different evaluation interfaces, the *Shallow Evaluation*, which provides minimum detail for inexperienced users, and the *Deep Evaluation*, which is targeted for advanced users. First, we discuss the deep evaluation mechanism to clarify the learning mechanism for context dependent co-occurrence rules. Then, the shallow mechanism which imitates the deep evaluation is discussed.

5.1 Deep evaluation of translation results

The deep evaluation is targeted for advanced users and can be used to learn more fine-tuned co-occurrence rules compared to shallow evaluation. In the deep evaluation, the user can evaluate individual nodes of the parse tree associated with each translation result.

The user interface provides two check boxes for each node of a parse tree in order to input the correctness judgment from the user. Check Box 1 can be set to 3 different values, which are *correct* (☑), *incorrect* (☒) and *indeterminate* (☐). The indeterminate state can be chosen for a node when the user does not want to evaluate the subtree rooted at

that one. Check Box 1 is always shown to the user, whereas Check Box 2, is only shown when Check Box 1 is set to incorrect and the node has a child evaluated as incorrect. Check Box 2 has two states, namely *correct* (☑) and *incorrect* (☒). The different configurations of the two check boxes constitute a total of 5 states for the nodes, the meanings of which are explained in detail in Table 1.

For a given node, the user determines the state of Check Box 1 by answering the question: “*Is the translation implied by the subtree rooted at this node correct?*”. Therefore, if Check Box 1 is set to (☑) for a node, then the partial translation implied by the subtree rooted at that node must be correct. Likewise, if it is set to (☒) then the partial translation implied by the subtree rooted at that node is incorrect.

Similarly, for a given node, the user determines the state of Check Box 2 by answering the question: “*Can the translation error be isolated to some erroneous child(ren) of this node?*”. If the partial translation implied by the subtree rooted at a node is incorrect, that node may not be the actual source of the translation error. In other words, the error can be isolated at one or more children nodes. If this is the case, the Check Box 2 is set to (☑) denoting that the node is not a cause for the erroneous translation. If the error cannot be isolated to a child node, then Check Box 2 is set to (☒).

As an example, suppose that the translation system knows only the following translation templates:

Table 1 States used in deep evaluation

| State | Symbol | Explanation |
|-------|--------|---|
| 1 | □ | This is the initial state assigned to every node at the beginning of the evaluation. It simply denotes that there does not exist any node in the subtree rooted at this node that is evaluated by the user. |
| 2 | ☑ | This state denotes that the user evaluated the partial translation, which is implied by the nodes in the subtree rooted at this node, as <i>correct</i> . It also indicates, that all children of this node are also in state 2. |
| 3 | ☒ | This state denotes that the user evaluated the partial translation, which is implied by the subtree rooted at this node, as <i>incorrect</i> . It also indicates that the user has not evaluated any of the children nodes as <i>incorrect</i> , or the node is a leaf. |
| 4 | ☒☒ | This state denotes that the user evaluated the partial translation, which is implied by the subtree rooted at this node, as <i>incorrect</i> . In order for a node to be in this state, the node has to have a child that is evaluated as <i>incorrect</i> . |
| 5 | ☒☑ | This state has all properties of state 4. In addition to that, this state denotes that, although the translation is erroneous, the use of this translation template in the current context is not the cause of the error. That is, the translation error is isolated in some children of this node. |

$$\begin{aligned}
1: & X_{\text{Adj Noun Sg}}^1 \hat{\text{DB+Adj+Ed}} X_{\text{Noun}}^2 + \text{Sg} \\
& \leftrightarrow Y_{\text{Adj Noun A3sg Pnon Nom}}^1 \hat{\text{DB+Adj+With}} Y_{\text{Noun}}^2 \\
& \quad + \text{A3sg} + \text{Pnon} + \text{Nom} \\
2: & X_{\text{Adj}}^1 X_{\text{Noun Sg}}^2 \hat{\text{DB+Adj+Ed}} \\
& \leftrightarrow Y_{\text{Adj}}^1 Y_{\text{Noun A3sg Pnon Nom}}^2 \hat{\text{DB+Adj+With}} \\
3: & \text{blonde+Adj } X_{\text{Noun}}^1 + \text{Sg} \\
& \leftrightarrow \text{sarı+Adj saç+Noun +A3sg +Pnon} \\
& \quad + \text{Nom} \hat{\text{DB+Adj+With}} \\
& \quad Y_{\text{Noun}}^1 + \text{A3sg} + \text{Pnon} + \text{Nom} \\
4: & \text{hair+Noun} + \text{Sg} \leftrightarrow \text{saç+Noun} + \text{A3sg} + \text{Pnon} + \text{Nom} \\
5: & \text{woman+Noun} \leftrightarrow \text{kadın+Noun} \\
6: & \text{yellow+Adj} \leftrightarrow \text{sarı+Adj}
\end{aligned} \tag{30}$$

where the Turkish to English confidence factors are 0.9, 0.8, 0.5, 1.0, 1.0 and 1.0, respectively. Let us assume that the user has translated the Turkish phrase

$$\text{“sarı saçlı kadın”} \tag{31}$$

whose lexical representation is

$$\begin{aligned}
& \text{sarı+Adj saç+Noun +A3sg +Pnon} \\
& \quad + \text{Nom} \hat{\text{DB+Adj+With}}
\end{aligned}$$

$$\text{kadın+Noun +A3sg +Pnon +Nom,}$$

and the translation system returned two different results, as shown in Fig. 9. The first result, “*yellow haired woman*” is a literal translation and it is less appropriate when compared to the second one, “*blonde woman*”. The confidence value of the first translation, $0.9 \times 0.8 \times 1.0 \times 1.0 \times 1.0 = 0.72$, is greater than the confidence value, $0.5 \times 1.0 = 0.5$, of the second one; therefore, the first translation is listed over the

second one. However, suppose that the user prefers the second translation, “*blonde woman*”, over the first one. In that case, the user may enter the Deep Evaluation screen to teach his preference to the system.

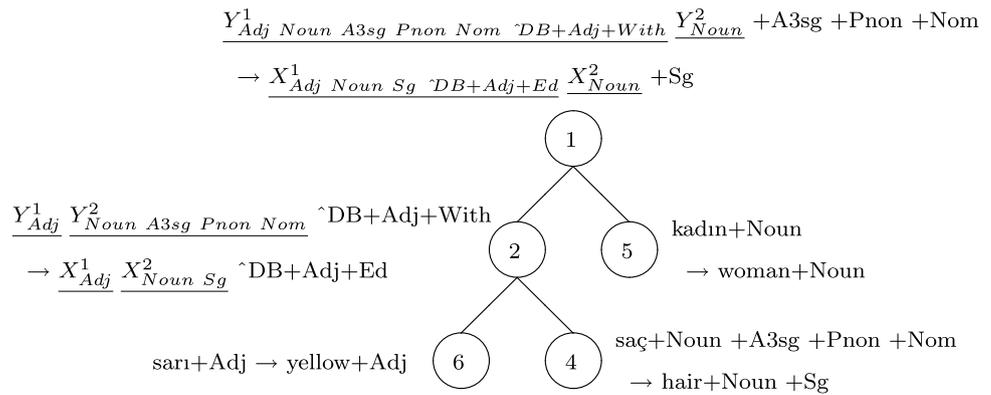
To simplify the evaluation process, rather than showing the contents of the non-atomic templates as node labels, the Deep Evaluation screen shows the partial translations implied by those nodes. The partial translation implied by a node is defined recursively, and found by replacing each variable in the template by the partial translation implied by the corresponding child node. Since the leaf nodes always represent an atomic template in the parse tree of a result, the partial translation implied by a node can always be found. Also the partial translation of the root node is equal to the lexical form of the translation result associated with that tree. For example, for the template tree of the first result in our example, rather than showing the contents of the 2nd template as the label of the node 2, the Deep Evaluation screen shows the partial translation implied by node 2, which is

$$\begin{aligned}
& \text{sarı+Adj saç+Noun +A3sg +Pnon} \\
& \quad + \text{Nom} \hat{\text{DB+Adj+With}} \\
& \rightarrow \text{yellow+Adj hair+Noun +Sg} \hat{\text{DB+Adj+Ed}} \tag{32}
\end{aligned}$$

At the beginning of the evaluation, in order to simplify the user interface, the roots of the translation trees are collapsed, i.e., the children of the root nodes are hidden from the user. The children of a node are only expanded (shown) when the partial translation implied by that node is evaluated as incorrect by the user. By using this method, the user marks paths from the root to the subtrees that are the sources of the erroneous translation.

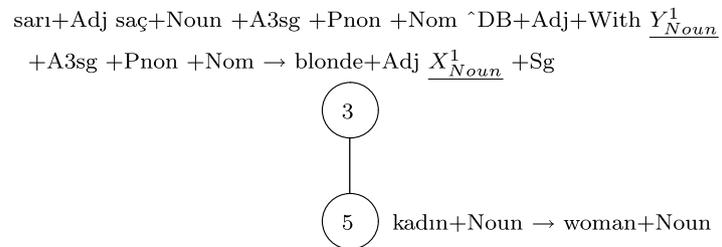
The evaluation is simple for the translation results that are perceived as correct by the user. When the user marks the root node of the parse tree of such a translation result as

Fig. 9 Translation results for the Turkish phrase “sarı saçlı kadın” (sarı+Adj saç+Noun +A3sg +Pnon +Nom ^DB+Adj+With kadın+Noun +A3sg +Pnon +Nom) in (31)



(a) yellow haired woman

(yellow+Adj hair+Noun +Sg ^DB+Adj+Ed woman+Noun +Sg)



(b) blonde woman (blonde+Adj woman+Noun +Sg)



Fig. 10 Evaluation of the translation result given in Fig. 9(b)

correct, all other nodes in the parse tree are considered to be correct too. This is intuitive, as we expect a correct translation to be made up of partial translations, that are correct in the context of the translated phrase. In our example, the user perceives the second translation result, “blonde woman”, as correct. So, the node 5 in the parse tree of that result is marked as correct along with the root node. The Deep Evaluation process for this result is depicted in Fig. 10(a–b).

For the translation results that are perceived as incorrect, or *inappropriate*, by the user, the evaluation requires more attention. The user starts by setting the root node to state \square , and walks through the tree by expanding the nodes on the paths to erroneous subtrees. For our translation result “yellow haired woman”, the process of Deep Evaluation is depicted in Fig. 11(a–e). One should note that, although this

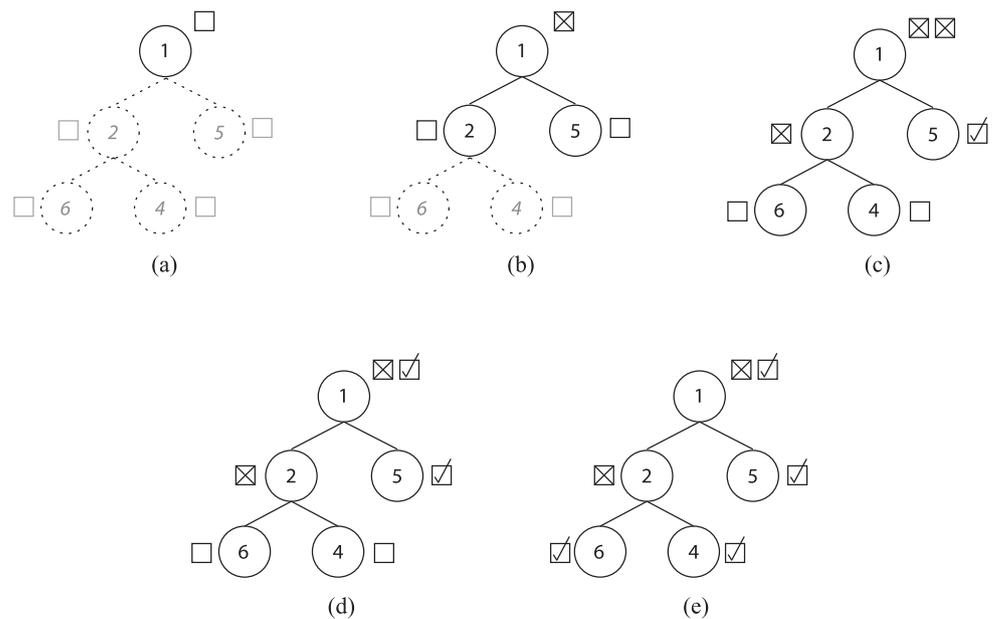
translation result is not a completely wrong one, it is less desired compared to the other result. The user can treat this result as if it was incorrect to teach his preference to the system. In deep evaluation, treating a not-that-appropriate result as if it was incorrect is safe, since the system will never assign a 0 confidence factor to such a translation result. The learned co-occurrence rules will be fine-tuned to place this kind of results just below the more desired ones.

Initially, only the root node is shown to the user (Fig. 11(a)), along with the translation result in its lexical form. When the user sets the state of the root node to \square , the root node is expanded and its children are shown (Fig. 11(b)).

As the partial translation implied by node 5, “kadın +Noun → woman+Noun”, is correct, the user sets the state of node 5 as \checkmark . Since the partial translation implied by node 2, as given in (32), is perceived as incorrect, the user sets the state of node 2 to \square (Fig. 11(c)). Also, since the error can be isolated in node 2, the user changes the state of the root node to $\square\checkmark$ (Fig. 11(d)).

Lastly, the user evaluates the nodes 6 and 4. Node 6 implies the partial translation “sarı+Adj → yellow+Adj”. It is not wrong to use this node in the context of [2(1), 1(1)]. Similarly, node 4 could well be used in the same context

Fig. 11 Evaluation of the translation result given in Fig. 9(a)



correctly if node 6 was not there. In other words, the reason for the error is using nodes 6 and 4 together. When considered separately, using these nodes in the context they appear is not wrong. So, in the Deep Evaluation, the states of both of the nodes are set to ✓ by the user (Fig. 11(e)).

5.2 Determining the desired confidence values

Each translation result is either marked as correct or incorrect regardless of the evaluation method used. The user also has the option of leaving a translation result unevaluated. In that case, no co-occurrence rule is learned from that particular translation result.

The co-occurrence rules learned from the user evaluation guarantee that during the next translation of the same input phrase, results marked as correct will be placed above the results marked as incorrect, i.e., learned rules will adjust the confidence value of correct and incorrect translations in such a way that confidence values of correct translations will be higher than that of incorrect translations.

Suppose that the translation of an input phrase returned five different results, A, B, C, D and E; and the user evaluated the results as shown in Table 2. We can see that all translation results except B are evaluated. While A is the result with the highest confidence value, it is marked as incorrect. Although, C and D are marked as correct, they are assigned lower confidence values compared to A, thus ranking below A. Therefore, the co-occurrence rules, that will be learned from the evaluation should change the order of A, C and D in such a way, that A comes below C and D. Even though E is marked as incorrect, we do not have to change its position in the ordering, since there are no correct results

Table 2 Sample translation result evaluation

| Translation result | Original confidence value | Evaluation assessment |
|--------------------|---------------------------|-----------------------|
| A | 0.9 | ⊗ |
| B | 0.8 | □ |
| C | 0.6 | ✓ |
| D | 0.4 | ✓ |
| E | 0.3 | ⊗ |

with confidence values lower than that of E. So, we will not learn any rules from E.

The next step for learning co-occurrence rules, is to determine the desired confidence values for the translation results. In order to do that, we have to calculate six values, namely *lower_hinge*, *upper_hinge*, *length₁*, *length₂*, *gap_{avg}* and *scale_factor*. The first four of these values for the example in Table 2 are shown in Fig. 12.

Let the incorrect translation result with the highest confidence value be R_{inc_high} and the correct result with the lowest confidence value be R_{cor_low} . *Upper_hinge* is the confidence value of the correct result that is ranked just above R_{inc_high} . If such a correct result does not exist, then $upper_hinge = 1$. Symmetrically, *lower_hinge* is the confidence value of the incorrect result that is ranked just below R_{cor_low} . If such an incorrect result does not exist, then $lower_hinge = 0$. Also, $length_1$ and $length_2$ are defined as

$$length_1 = |upper_hinge - confidenceOf(R_{cor_low})| \quad (33)$$

$$length_2 = |lower_hinge - confidenceOf(R_{inc_high})| \quad (34)$$

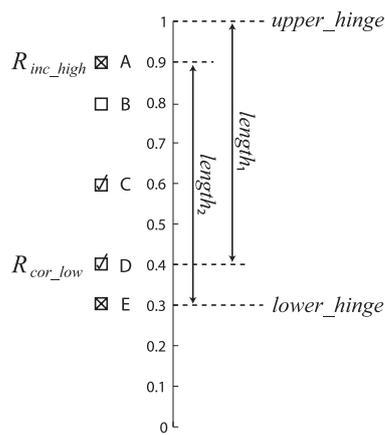


Fig. 12 *lower_hinge*, *upper_hinge*, *length₁* and *length₂* for the Example in Table 2

The average gap, *gap_{avg}*, between the original confidence values of the subsequent evaluated translation results in range [*lower_hinge*, *upper_hinge*] for Table 2 is

$$gap_{avg} = \frac{(0.9 - 0.6) + (0.6 - 0.4) + (0.4 - 0.3)}{3} = 0.2 \tag{35}$$

Lastly, the *scale_factor* is calculated as

$$scale_factor = \frac{upper_hinge - lower_hinge}{length_1 + gap_{avg} + length_2} \tag{36}$$

which is $(1 - 0.3)/(0.6 + 0.2 + 0.6) = 0.7/1.4 = 0.5$ for our evaluation. After calculating the *scale_factor*, the desired confidence value of a translation result *R*, that is in range (*lower_hinge*, *upper_hinge*) is assigned by formula (37).

$$desired\ confidence\ of\ R = \begin{cases} confOf(R) & \text{if } R \text{ is not evaluated,} \\ upper_hinge - (upper_hinge - confOf(R)) \times scale_factor & \text{if } R \text{ is correct,} \\ lower_hinge + (confOf(R) - lower_hinge) \times scale_factor & \text{if } R \text{ is incorrect.} \end{cases} \tag{37}$$

For our evaluation, the correct results have been ranked above the incorrect ones after assigning the desired confidence values, as shown in Table 3. The process is depicted graphically in Fig. 13. One should note that, our formula in (37) preserves the order among correct results, which is also true for incorrect results.

Table 3 The new ranking of the results in Table 2

| Translation result | Desired confidence value | Evaluation assessment |
|--------------------|--------------------------|-----------------------|
| B | 0.8 | □ |
| C | 0.8 | ☑ |
| D | 0.7 | ☑ |
| A | 0.6 | ☒ |
| E | 0.3 | ☒ |

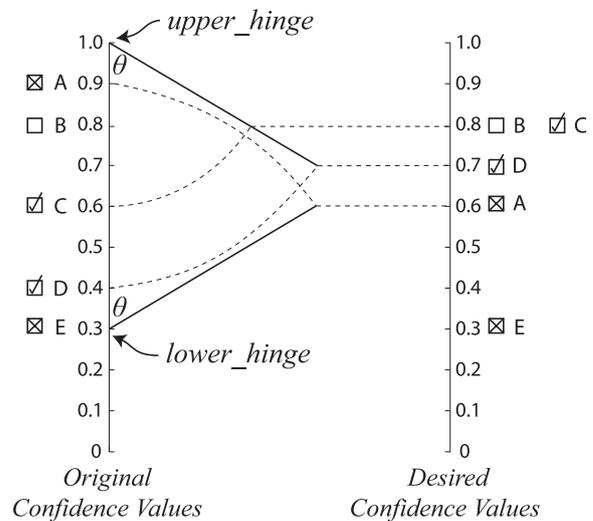


Fig. 13 Assigning the desired confidence values. ($\theta = \arccos(scale_factor)$)

Now, let us return to our example translation of the Turkish phrase (31) (“sarı saçlı kadın”). In our scenario, the translation system returns two different translation results for this input phrase, which are shown below with the corresponding confidence values:

“yellow haired woman” : 0.72 “blonde woman” : 0.5

In this example, the first result will be evaluated as an incorrect result, and the second one will be evaluated as a correct result. In this case, when we apply the methods described in this section we will obtain the following parameters:

$$\begin{aligned} lower_hinge &= 0.0 & length_1 &= 0.5 \\ upper_hinge &= 1.0 & length_2 &= 0.72 \\ & & gap_{avg} &= 0.22 \end{aligned}$$

Therefore,

$$\begin{aligned} scale_factor &= \frac{upper_hinge - lower_hinge}{length_1 + gap_{avg} + length_2} \\ &= \frac{1}{0.5 + 0.22 + 0.72} = 0.694 \end{aligned}$$

Using formula (37), the desired confidence values for the translation results become:

$$\begin{aligned} \text{"yellow haired woman"} &: 0.499 \\ \text{"blonde woman"} &: 0.653 \end{aligned} \quad (38)$$

One should note, that the desired confidence values comply with the expectations of the user. The more proper result, "blonde woman", has a higher desired confidence value than that of the first result, "yellow haired woman".

5.3 Extracting context-dependent co-occurrence rules

The last step in learning co-occurrence rules, is to extract them from the parse trees of the evaluated translation results with the desired confidence values. After finding the desired confidence values for each translation result in the range (*lower_hinge*, *upper_hinge*), the system extracts context-dependent co-occurrence rules, using the EXRULES procedure given in Fig. 14. The first parameter to this procedure is an array of the translation results, while the second parameter is an array of the desired confidence values. The desired confidence value for each translation result is calculated as described in Sect. 5.2. EXRULES uses EXRULES-INCORR and EXRULES-CORR procedures (given in Fig. 15 and Fig. 16, respectively) in order to learn co-occurrence rules for the given parse trees and their subtrees.

EXRULES-INCORR procedure is used to extract co-occurrence rules from the parse trees of the translation results that are marked as incorrect by the user. The parse trees of the incorrect translations are marked as \boxtimes , $\boxtimes\boxtimes$ or $\boxtimes\boxtimes\boxtimes$, and EXRULES-INCORR procedure is only invoked for those incorrect parse trees. EXRULES-INCORR performs a depth-first traversal on the parse tree of a given incorrect result. During the traversal, since a translation rule at the root position of the nodes marked as \boxtimes or $\boxtimes\boxtimes$ is a reason for an incorrect translation, a co-occurrence rule is extracted for that node. The incorrect children of the current node are also explored in order to extract more co-occurrence rules. Although a node marked as \boxtimes represents an incorrect translation, its children will never be explored during the depth-first traversal, since such a node cannot have a child marked as incorrect. On the other hand, the children of nodes marked

as $\boxtimes\boxtimes$ or $\boxtimes\boxtimes\boxtimes$ will be explored, as this kind of a node must have at least one incorrect child.

EXRULES-INCORR procedure in Fig. 15 takes 3 arguments. The first one is a node in the parse tree of a translation result, the second one is the context in which the node exists, and the last argument is the desired confidence value for the subtree rooted at the given node. When EXRULES-INCORR is called for the node p , with the desired confidence value *desired-confidence* _{p} , first a context-dependent co-occurrence rule is learned by LEARNRULE if the node p is marked as \boxtimes , or $\boxtimes\boxtimes$. The learned rule will have an aggregate confidence factor that is lower than the original confidence value of the subtree rooted at p , penalizing the subtree.

Now, let us assume that a node p has children c_1, c_2, \dots, c_n , where c_1, c_2, \dots, c_k are marked as incorrect (\boxtimes , $\boxtimes\boxtimes$ or $\boxtimes\boxtimes\boxtimes$) and c_{k+1}, \dots, c_n are either marked as correct (\boxtimes) or left unevaluated (\square). Then, an incorrect-child-multiplier value β is calculated as

$$\beta = \sqrt[k]{\frac{\text{desired-confidence}_p}{\text{original-confidence}_p}} \quad (39)$$

where *original-confidence* _{p} is the original confidence value of the tree rooted at node p . This multiplier is used to distribute the penalty evenly to each of the incorrect children of p . One should note that, the inequality

$$\text{desired-confidence}_p / \text{original-confidence}_p < 1$$

will always hold, as p is incorrect, therefore $\beta < 1$ is also true. Next, for each child c_i , $1 \leq i \leq k$, EXRULES-INCORR is called recursively with the desired confidence parameter

$$\text{desired-confidence}_{c_i} = \beta \times \text{original-confidence}_{c_i} \quad (40)$$

Thus, for $1 \leq i \leq k$,

$$\text{desired-confidence}_{c_i} < \text{original-confidence}_{c_i} \quad (41)$$

EXRULES-CORR is very similar to EXRULES-INCORR, except it is used to learn rules from correct translations. As all nodes in the parse tree of a correct translation result would be marked as \boxtimes , the depth-first traversal performed by recursive calls of EXRULES-CORR will effectively explore all the nodes in such a tree. When EXRULES-CORR is called for the node p , with the desired confidence value *desired-confidence* _{p} , first a context-dependent co-occurrence rule that rewards the subtree rooted at p is learned.

Assume that a node p has children c_1, c_2, \dots, c_m , where all the children are marked as correct. Then a correct-child-multiplier value δ is calculated as

$$\delta = \sqrt[m]{\frac{\text{desired-confidence}_p}{\text{original-confidence}_p}} \quad (42)$$

```

EXRULES(results, new_confidences)
  for i = 1 to length[results] do
    root ← root node of results[i]
    context ← [] % an empty context
    if (root is in state  $\boxtimes$ ) then
      EXRULES-CORR(root, context, new_confidences[i])
    else if (root is in state  $\boxtimes$ ,  $\boxtimes\boxtimes$  or  $\boxtimes\boxtimes\boxtimes$ ) then
      EXRULES-INCORR(root, context, new_confidences[i])

```

Fig. 14 EXRULES. Extracts context-dependent co-occurrence rules from evaluated translation results

```

EXRULES-INCORR(node, context, desired_confidence)
  tree ← the tree rooted at node
  if (state of node is ☒ or ☒☒) then
    LEARNRULE(tree, context, desired_confidence)
    old_confidence ← the confidence value of tree
    incorrect_children ← { c : c is a child of node in ☒, ☒☒ or ☒☒ state }
    if (incorrect_children ≠ ∅) then
       $\beta \leftarrow (\text{desired\_confidence} / \text{old\_confidence})^{1/|\text{incorrect\_children}|}$ 
      for each (child ∈ incorrect_children) do
        index ← getChildIndex(node, child)
        child_context ← add(copy(context), ⟨node, index⟩)
        child_confidence ← confidence value of the subtree rooted at child
        desired_child_confidence ←  $\beta \times \text{child\_confidence}$ 
        EXRULES-INCORR(child, child_context, desired_child_confidence)

```

Fig. 15 EXRULES-INCORR. Extracts context-dependent co-occurrence rules from incorrect translation results

```

EXRULES-CORR(node, context, desired_confidence)
  tree ← the tree rooted at node
  if (desired_confidence ≤ 1) then
    LEARNRULE(tree, context, desired_confidence)
  else
    LEARNRULE(tree, context, 1)
    old_confidence ← the confidence value of tree
    correct_children ← { c : c is a child of node in ☑ state }
    if (correct_children ≠ ∅) then
       $\delta \leftarrow (\text{desired\_confidence} / \text{old\_confidence})^{1/|\text{correct\_children}|}$ 
      for each (each child ∈ correct_children) do
        index ← getChildIndex(node, child)
        child_context ← add(copy(context), ⟨node, index⟩)
        child_confidence ← confidence value of the subtree rooted at child
        desired_child_confidence ←  $\delta \times \text{child\_confidence}$ 
        EXRULES-CORR(child, child_context, desired_child_confidence)

```

Fig. 16 EXRULES-CORR. Extracts context-dependent co-occurrence rules from correct translation results

where *original-confidence*_{*p*} is the original confidence value of the tree rooted at node *p*. This multiplier is used to distribute the reward evenly to each of the correct children of *p*. One should note that, the inequality

$$\text{desired-confidence}_p / \text{original-confidence}_p > 1 \quad (43)$$

will always hold, as *p* is correct, therefore $\delta > 1$ is also true. Next, for each child *c_i*, $1 \leq i \leq m$, EXRULES-CORR is called recursively with the desired confidence parameter

$$\text{desired-confidence}_{c_i} = \delta \times \text{original-confidence}_{c_i} \quad (44)$$

Thus, for $1 \leq i \leq m$,

$$\text{desired-confidence}_{c_i} > \text{original-confidence}_{c_i} \quad (45)$$

Note that, for some child *c_i*, $1 \leq i \leq m$, the inequality *desired-confidence*_{*c_i*} > 1 can be true, since $\delta > 1$. This is

not allowable,⁴ since we do not want to learn a context-dependent co-occurrence rule with an aggregate confidence factor > 1. EXRULES-CORR prevents this kind of situations by simply setting the rule confidence to 1 for subtrees rooted at such *c_i*.

Now, let us return to our deep evaluation scenario for the translation results of the Turkish phrase “sarı saçlı kadın” in (31). The parse trees of the translation results were evaluated as shown in Fig. 10(b) and Fig. 11(e), and the desired confidence values were determined as given in (38). The translation result in Fig. 11(e) was an incorrect one. Therefore, EXRULES will call EXRULES-INCORR for the root node of the parse tree of this result, with the desired confidence value of 0.499. As the root node is marked as ☒☒, no rules will be extracted at that node. Then the β value will

⁴A confidence factor represents a probability value, therefore cannot be > 1.

be calculated as

$$\beta = \sqrt{\frac{0.499}{0.72}} = 0.693 \quad (46)$$

Next, EXRULES-INCORR will be called for the incorrect child of the root, which is node 2, recursively, with the desired confidence value parameter of $\beta \times 0.8 = 0.693 \times 0.8 = 0.554$, where 0.8 is the original confidence value of the subtree rooted at node 2. Since node 2 is marked as \boxtimes the context dependent co-occurrence rule

$$2(6, 4) - [1_{(1)}](0.554) \quad (47)$$

will be extracted. Since there are no other erroneous nodes in the tree, this rule will be the only rule that is learned for this translation result.

The translation result in Fig. 10(b) is correct. Therefore, EXRULES will call EXRULES-CORR for the root node of the parse tree of this result, with the desired confidence value of 0.653. As this node is marked as \boxplus , the context dependent co-occurrence rule

$$3(5) - [](0.653) \quad (48)$$

will be extracted. Note that the only context associated with this rule is an empty one, as the rule was extracted from the root node. Then, the δ value will be calculated as

$$\delta = \sqrt{\frac{0.653}{0.5}} = 1.306 \quad (49)$$

Next, EXRULES-CORR will be called for the correct child of the root, which is node 5, with the desired confidence value parameter of $\delta \times 1.0 = 1.306 \times 1.0 = 1.306$, where 1.0 is the original confidence value of the subtree rooted at node 5. Since the desired confidence value is greater than 1, the extracted rule will be assigned the maximum possible aggregate confidence factor, which is 1. Therefore the second extracted rule will be

$$5 - [3](1.0) \quad (50)$$

As no other nodes remain in the parse tree, the execution will be over.

5.4 Shallow evaluation of translation results

Shallow evaluation is the second evaluation interface of our translation system, which is targeted for inexperienced users, as it provides much simpler means of user interaction, compared to deep evaluation. In shallow evaluation, the translation results are shown in their surface forms, instead of their lexical forms. This makes it much easier to interpret the results during the evaluation. While in deep evaluation,

the nodes in the parse trees of translation results can be evaluated individually by the user, in shallow evaluation the user makes only a single correctness judgment for each result. Thus, a translation result is either marked as correct (\boxplus) or incorrect (\boxtimes), or left unevaluated (\square).

In fact, shallow evaluation is a front-end to deep evaluation with a simplified interface. The input for the shallow evaluation taken from the user is automatically converted to an instance of the deep evaluation input, on which the co-occurrence rule learning methods described in the previous subsections are applied. IMITATE-DEEPANALYSIS procedure, given in Fig. 17 performs this input conversion.

In IMITATE-DEEPANALYSIS, each incorrect result is compared with the correct results. COMPARE-TREES procedure is used as a subprocedure for the comparison. The nodes that might have caused the incorrect translation are tried to be identified during successive comparisons. Note that COMPARE-TREES ensures that the comparison order does not change the final configuration of the incorrect results.

When an incorrect result is compared with the correct results, in some rare occasions, all nodes in the parse tree of that incorrect result may be set to \boxplus . This is an undesired effect, since it prevents learning any co-occurrence rules from that particular incorrect result. This happens if successive comparisons validate all of the nodes in the parse tree of an incorrect result. IMITATE-DEEPANALYSIS handles this situation and sets the root node to an incorrect state. Therefore, it is guaranteed that at least one co-occurrence rule is extracted from each incorrect result. An example run of the algorithm in Fig. 17 is given in Fig. 18.

6 Test results and evaluation

In this section, we present the results of the translation tests on the user evaluation mechanisms. We used two different metrics for the performance evaluation. Our first metric is the position of the first correct translation result among the translations results. For this purpose, we examined whether the first correct result appears in the first position, in the positions 2–3 and in the positions 4–5. The second metric, *Bilingual Evaluation Understudy* (BLEU) [22], measures the closeness of a translation result generated by a machine translation system to a correct translation reference by using n -gram based method. To judge the quality of a machine translation result, BLEU calculates its closeness to one or more reference human translations using n -grams. A BLEU score varies between 0 and 1, where a score of 1 denotes that the result is an exact translation. In our experiments we take the candidate as the translation result with the highest confidence value. In cases where multiple results with the highest confidence value exist, the candidate is taken as the first result generated. We use a single reference translation for each

```

IMITATE-DEEPANALYSIS(correct_results_list, incorrect_results_list)
  for each (correct_result ∈ correct_results_list) do
    set all nodes in the parse tree of correct_result to  $\checkmark$ 
  for each (incorrect_result ∈ incorrect_results_list) do
    set all nodes in the parse tree of incorrect_result to  $\square$ 
    incorrect_root ← the root node in the parse tree of incorrect_result
    for each (correct_result ∈ correct_results_list) do
      correct_root ← the root node in the parse tree of correct_result
      COMPARE-TREES(incorrect_root, correct_root)
    if (state of incorrect_root is  $\checkmark$ ) then
      set all nodes in the parse tree of incorrect_result to  $\square$ 
      set state of incorrect_root to  $\boxtimes$ 

COMPARE-TREES(incorrect_root, correct_root)
  if (state of incorrect_root is  $\checkmark$ ) then
    return true
  else if (templateno of incorrect_root = templateno of correct_root) then
    flag ← true
    for  $i = 1$  to  $n$ , where  $n$  is the number of children of incorrect_root do
      incorrect_child ←  $i^{th}$  child of incorrect_root
      correct_child ←  $i^{th}$  child of correct_root
      if (COMPARE-TREES(incorrect_child, correct_child) = false) then
        flag ← false
    if (flag = true) then
      set state of incorrect_root to  $\checkmark$ 
    else
      set state of incorrect_root to  $\boxtimes\boxtimes$ 
    return flag
  else
    if (state of incorrect_root is  $\square$ ) then
      set state of incorrect_root to  $\boxtimes$ 
    return false

```

Fig. 17 IMITATE-DEEPANALYSIS. Converts a shallow evaluation input to a deep evaluation input automatically

element in the testing subset. We also used the parameter value $N = 4$, as recommended in [22] in our experiments.

A data collection of 8012 translation examples has been created for the experimental evaluation. The translation examples in the data collection consist of simple translation examples that are manually created, and the translation examples that are created from the bilingual manuals of household electronic equipment and bilingual traveling brochures. This data collection is an extended version of the data set that is used in the evaluations of the earlier versions of our EBMT system. The longest English sentence contained 31 words, and the longest Turkish sentence contained 26 words. On the average, the number of words in a typical English sentence is approximately 11 words, and the number of words in a typical Turkish sentence is approximately 10 words. All of English and Turkish surface-level sentences in the data collection are manually converted into the sentences in lexical form by using a simple tagging tool. Thus, we obtained our bilingual corpus in lexical form, and it is used as our data collection. The number of morphemes in the longest English sentence in lexical form is 67, and the

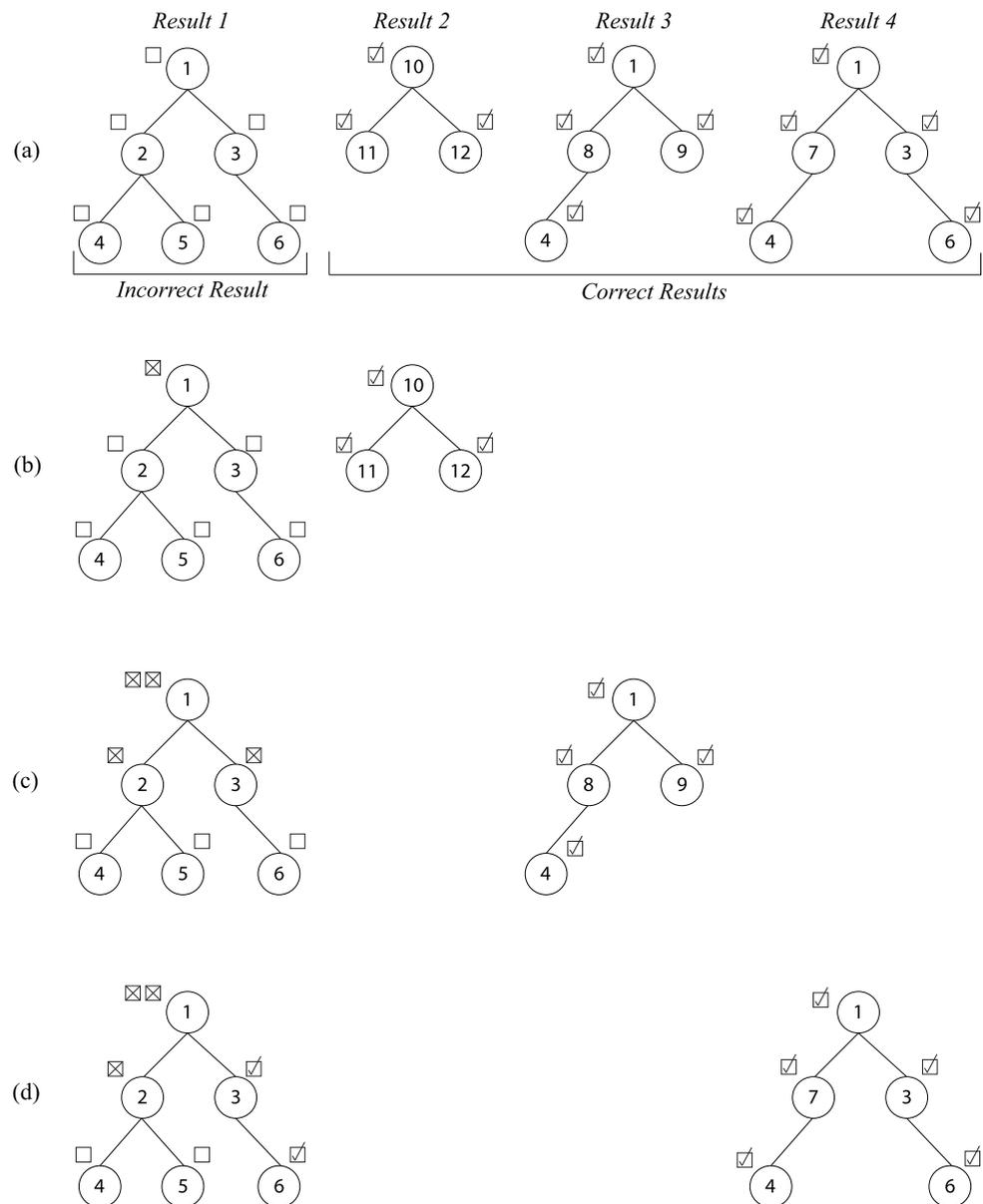
number of morphemes in the longest Turkish sentence in lexical form is 106.

The data collection in lexical form is pseudo randomly divided into 3 subsets as follows in order to evaluate the performance of our EBMT system:

- **Training Subset** is the 75% of the data collection (6012 examples), and it is used for the extraction of the translation templates.
- **User Evaluation Subset** is the 5% of the data collection (400 examples), and it is used to train the system during deep and shallow evaluations.
- **Testing Subset** is the 20% of the data collection (1600 examples), and it contains the translation examples which are used during the performance evaluation of the system.

All three subsets contain unique elements, i.e., there is no translation example that is shared by any two subsets; however, it is allowed for the translation examples in the subsets to contain common substrings. In this way, we guarantee that the system is not directly trained for the elements of the testing subset, and thus avoid a flawed experiment. The 20%

Fig. 18 An example to automatic conversion of *Shallow Evaluation* input into *Deep Evaluation* input. (a) The initial situation of the parse trees associated with 4 translation results. *Result 1* is the only incorrect result, while *Result 2–4* are evaluated as correct by the user. At this point, all nodes of the incorrect result are initialized to \square , while all nodes of the correct results are initialized to \checkmark . (b)–(d) The situation after each successive comparison of the incorrect result with one of the correct translation results. Note that changing the comparison order does not effect the final configurations of the parse trees



of the data collection (1600 examples) is selected randomly, and it is reserved as the *Testing Subset*. From the remaining 80% of the data collection (6412 examples), the sentences of the *User Evaluation Subset* (the 5% of the data collection = 400 examples) are pseudo randomly selected, and the remaining 75% of the data collection (6012 examples) is used as the *Training Subset*.

During the selection of the sentences into the *User Evaluation Subset*, we did not make our selection completely randomly because we tried to make sure that some of the sentences in the *User Evaluation Subset* have common substrings with the sentences in the *Testing Subset*. Thus, we can observe the effects of the learned context-dependent co-occurrence rules during the user evaluation phase in the translation of the sentences in the *Testing Subset*. The

learned context-dependent co-occurrence rules play a role in the order of the translation results if they are only used in the translation process of the sentences in the *Testing Subset*. If the sentences of the *User Evaluation Subset* and the *Testing Subset* do not have any common substrings, the learned context-dependent co-occurrence rules will not be used in the translation of the sentences of the *Testing Subset*. In that case, we can say that the user-evaluation does not have any effect on the performance of our EBMT system, and its performance will be same as the performance of the earlier version of our EBMT system. When the overlap between the substrings of the sentences of the *User Evaluation Subset* and the *Testing Subset* increases, the effect of the user-evaluation phase on the performance of the system increases too. Of course, the main point of using the user-

Table 4 Experimental results for English to Turkish translation

| | Avg. BLEU score | Pos. of first corr. result | | |
|---------------|-----------------|----------------------------|-----|-----|
| | | 1 | 2–3 | 4–5 |
| Initial | 0.815 | 65% | 21% | 3% |
| Shallow eval. | 0.842 | 74% | 13% | 2% |
| Deep eval. | 0.856 | 76% | 11% | 2% |

Table 5 Experimental results for Turkish to English translation

| | Avg. BLEU score | Pos. of first corr. result | | |
|---------------|-----------------|----------------------------|-----|-----|
| | | 1 | 2–3 | 4–5 |
| Initial | 0.778 | 59% | 19% | 10% |
| Shallow eval. | 0.802 | 68% | 13% | 7% |
| Deep eval. | 0.817 | 70% | 12% | 6% |

evaluation mechanism is that the user prefers certain translations for some of the strings during the user-evaluation and he/she wants that his/her preferences are used in the subsequent translations.

After a set of translation templates were learned from the translation examples in the *Training Subset*, we measured the performance of the system using the translation examples in the *Testing Subset* without doing any user-evaluation. These measurements reflect the initial performance of our system without using any co-occurrence rules. Of course, the system is tested in both directions, namely English to Turkish and Turkish to English. In fact, these measurements reflect the performance of the earlier version of our EBMT system [4] on the extended data set described in this paper. In other words, the first rows (marked as Initial) in Tables 4 and 5 indicate the performance of the earlier version of our EBMT system on the new data set described here using two different measurement metrics. The data set described in this paper is an extended version of the smaller data set described in [4]. According to the performance results given in [4], the BLEU scores of the earlier version of the system for the smaller data collection are 0.82 for English to Turkish direction, and 0.78 for Turkish to English direction. For English to Turkish direction, 66% of the correct results appear in the first position, 23% of the correct results appear in the second or third position, and 1% of the correct results appear in the fourth or fifth position. For Turkish to English direction, 59% of the correct results appear in the first position, 19% of the correct results appear in the second or third position, and 11% of the correct results appear in the fourth or fifth position. These performance results of the earlier version of our EBMT system for the smaller data collection are very similar to the performance results in the first rows (marked as Initial) in Tables 4 and 5 for the extended data set described here.

In order to measure the performance of the deep evaluation method, we trained the system by doing the user-evaluation using the translation examples in the *User Evaluation Subset*. For English to Turkish direction, the English sentence of each example in the *User Evaluation Subset* was translated using the translation templates extracted earlier from the training subset. Then, a deep evaluation was performed for the translation result. The evaluator marked correct results and incorrect results using the evaluator user interface [5], and evaluated the subtrees of the incorrect results in detail. As a result, the co-occurrence rules were learned from the deep evaluation. Then, in order to test the effect of the deep evaluation, the examples in the testing subset were translated using the initial templates together with the co-occurrence rules that were learned during the deep evaluation. The same steps were repeated for Turkish to English direction.

A similar evaluation process was applied in order to test the performance of the shallow evaluation process. In this case, the evaluator only marked the translations results as correct or incorrect without doing anything else. Then, the examples in the testing subset were translated using the initial translation templates together with the co-occurrence rules that were learned during the shallow evaluation process.

Because the deep evaluation is a much more detailed process compared to the shallow evaluation, the former takes approximately twice as much time as the latter. Furthermore, in the deep evaluation, the user has greater control on the process. As a result, the number of context-dependent co-occurrence rules learned in the deep evaluation is less than the rules learned in the shallow evaluation.

Table 4 presents the results of the tests done in English to Turkish direction of the translation. In this direction, initially the average BLEU score was 0.815. When the context-dependent co-occurrence rules learned from the deep evaluation were used in the ranking of the translation results, the BLEU score increased to 0.856. The BLEU score increased less when the rules learned from the shallow evaluation were used. The number of the correct translations appearing in the first position of the translation results increased 11% for the deep evaluation and 9% for the shallow evaluation. Both the deep evaluation and the shallow evaluation moved up the position of the first correct translation result. The last three columns of Table 4 show the distribution of the position of the first correct result among the generated results.

In Turkish to English direction as given in Table 5, the average BLEU score was 0.778 initially. When the context-dependent co-occurrence rules learned from the shallow evaluation were used in the ranking of the translation results, the BLEU score increased to 0.802. For the shallow evaluation, the number of the correct results appearing in the first position increased from 59% to 68%. The results for the

deep evaluation were better. When the context-dependent co-occurrence rules learned from the deep evaluation were used in the ranking of the translation results, the BLEU score increased to 0.817. Similarly, the number of the correct results appearing in the first position increased from 59% to 70%. The last three columns of Table 5 show the distribution of the position of the first correct result among the generated results.

The results for the deep evaluation are better than that of the shallow evaluation. This is because of the fact that, in the deep evaluation the user can fine-tune the templates that will be learned from the evaluation, while this is not possible in the shallow evaluation. Therefore in general, we expect the number of incorrect context-dependent co-occurrence rules learned by the shallow evaluation to be higher. Also, since the number of rules learned by the deep evaluation is usually less than that for the shallow evaluation, the time consumption of the ranking process will also be lower if the former approach is followed. However, we expect that the users will prefer the shallow evaluation, due to its simplicity.

7 Conclusion

In this paper, we described the extension of an existing example-based machine translation system with a user-evaluation module. The major contribution of this work is an improved ranking mechanism for the translation results that learns gradually from the user feedback. After a translation, the user always has the option of evaluating the generated results. From the evaluation, the system learns context-dependent co-occurrence rules which may be consulted in the result ordering phases of the upcoming translations.

In order to sort the translation results, the earlier versions of the system solely used the confidence factors associated with each template. Confidence factors were calculated in the learning phase once, and never updated thereafter. In our current approach, confidence factor scheme is improved by the inclusion of context-dependent co-occurrence rules. The system continues to learn context-dependent co-occurrence rules with each user evaluation in the translation phase.

Certain translation templates may be assigned low confidence factors when considered individually, but their co-existence in a translation result may deserve a higher confidence. The reverse can also be true. The original confidence factor assignment scheme did not handle template combinations, but considered each template individually. In our approach, the user has the chance to influence the confidence values of translation template combinations, without affecting the original confidence factors that will be used when the templates are utilized individually.

The system provides two different interfaces for getting the user feedback. In the shallow evaluation interface, the

user simply marks correct and incorrect translations. On the other hand, in the deep evaluation, as the name implies, the user can evaluate individual nodes of the parse trees associated with each translation result, where each node represents a separate translation template. Therefore, the deep evaluation takes more time, as it requires more attention and expertise. However, the deep evaluation provides fine-tuning capabilities which are not offered by the shallow evaluation.

In our tests, we observed significant performance improvements in the average BLEU scores and precision values at the top results. The improvements for the deep evaluation were better than those of the shallow evaluation, as expected.

In Sect. 4, the context of a subtree in the parse tree of a translation result (where the subtree corresponds to a phrase in the translation) was defined as a chain of nodes. This abstract definition allowed us to develop a context matching algorithm (see Sect. 4.2) in a simple manner. However, we expect a linguistically influenced definition to be superior, as a more natural definition of the context that a phrase occurs in would be based on the words surrounding that phrase.

References

1. Chiang D (2005) A hierarchical phrase-based model for statistical machine translation. In: Proceedings of the 43rd annual meeting of the ACL (ACL 2005), Ann Arbor, Michigan, pp 263–270
2. Cicekli I, Güvenir HA (2001) Learning translation templates from bilingual translation examples. *Appl Intell* 15(1):57–76
3. Cicekli I, Güvenir HA (2003) Learning translation templates from bilingual translation examples. In: Carl M, Way A (eds) Recent advances in example-based machine translation. Kluwer Academic, Boston, pp 247–278
4. Cicekli I (2005) Inducing translation templates with type constraints. *J Mach Transl* 19:283–299
5. Daybelge T (2007) Improving the precision of example-based machine translation by learning from user feedback. Master's thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey
6. Daybelge T, Cicekli I (2007) A rule-based morphological disambiguator for Turkish. In: Proceedings of recent advances in natural language processing (RANLP 2007), Borovets, Bulgaria, pp 145–149
7. Ding Y, Palmer M (2005) Machine translation using probabilistic synchronous dependency insertion grammars. In: Proceedings of the 43rd annual meeting of the ACL (ACL 2005), Ann Arbor, Michigan, USA, pp 541–548
8. Doğan H (2007) Example based machine translation with type associated translation templates. Master's thesis, Department of Computer Engineering, Bilkent University, Ankara, Turkey
9. Gough N, Way A (2004) Robust large-scale EBMT with marker-based segmentation. In: Proceedings of the 10th conference on theoretical and methodological issues in machine translation (TMI-2004), Baltimore, MD, pp 95–104
10. Güvenir HA, Cicekli I (1998) Learning translation templates from examples. *Inf Syst* 23(6):353–363
11. Imamura K (2002) Application of translation knowledge acquired by hierarchical phrase alignment for pattern-based MT. In: Proceedings of the 9th conference on theoretical and methodological issues in machine translation (TMI-2002), Valetta, Malta, pp 74–84

12. Imamura K, Sumita E, Matsumoto Y (2003) Feedback cleaning of machine translation rules using automatic evaluation. In: Proceedings of the 41st annual meeting of the association for computational linguistics, pp 347–350
13. Istek O, Cicekli I (2007) A link grammar for an agglutinative language. In: Proceedings of recent advances in natural language processing (RANLP 2007), Borovets, Bulgaria, pp 285–290
14. Font Llitjos A, Carbonell JG (2004) The translation correction tool: English-Spanish user studies. In: Proceedings of LREC-2004, pp 447–454
15. Font Llitjos A, Carbonell JG, Levie A (2005) A framework for interactive and automatic refinement of transfer-based machine translation. In: Proceedings of EAMT-2005, pp 87–96
16. Menezes A, Richardson SD (2001) A best first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora In: Proceedings of the workshop on example-based machine translation' im MT summit VIII, pp 35–42
17. Meyers A, Kosaka M, Grishman R (2000) Chart-based transfer rule application in machine translation. In: Proceedings of COLING-2000, pp 537–543
18. Nagao M (1984) A framework of a mechanical translation between Japanese and English by analogy principle. In: Proceedings of the international NATO symposium on artificial and human intelligence, New York, pp 173–180
19. Nesson R, Shieber SM, Rush A (2006) Induction of probabilistic synchronous tree insertion grammars for machine translation. In: Proceedings of the 7th conference of the association for machine translation in the Americas (AMTA 2006), Boston, Massachusetts
20. Oflazer K (1994) Two-level description of Turkish morphology. *Lit Linguist Comput* 9(2):137–148
21. Oz Z, Cicekli I (1998) Ordering translation templates by assigning confidence factors. In: AMTA '98: Proceedings of the third conference of the association for machine translation in the Americas on machine translation and the information soup. Lecture notes in computer science, vol 1529. Springer, London, pp 51–61
22. Papineni K, Roukos S, Ward T, Zhu W-J (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the association for computational linguistics, Philadelphia, pp 311–318
23. Shieber SM, Schabes Y (1990) Synchronous tree-adjointing grammars. In: Proceedings of the 13th COLING, pp 253–258
24. Venugopal A, Zollmann A, Vogel S (2007) An efficient two-pass approach to synchronous-CFG driven statistical MT. In: Proceedings of NAACL HLT 2007, Rochester, NY, pp 500–507
25. Xerox (2010) Xerox English morphological analyzer. Accessible at <http://legacy.xrce.xerox.com/competencies/content-analysis/demos/english.en.html>, accessed 7 March 2010
26. Yamada K, Knight K (2001) A syntax-based statistical translation model. In: Proceedings of the 39th annual meeting of the ACL (ACL 2001), pp 523–530



Turhan Daybelge graduated in 2005 from the Department of Information Technologies at Isik University. He received his M.S. degree in Computer Engineering in 2007 at Bilkent University in Ankara. His main research interests are natural language processing, example-based machine translation and information retrieval.



Ilyas Cicekli received his Ph.D. degree in computer science from Syracuse University. He is currently a faculty member of the Department of Computer Engineering at Bilkent University. From 2001 till 2003, he was a visiting faculty member at University of Central Florida. His current research interests include natural language processing, example-based machine translation, question-answering, and information extraction.