An Online Tutoring System for Language Translation

Naoyuki Tokuda and Liang Chen
Sunflare Company, Japan

Our interactive intelligent language tutoring system acquires Japanese-English technical translations over the Internet. The system consists of an augmented transition network-based template, a global-matching algorithm embedded in the template structure, a parts-of-speech tagged parser, and a visual interface authoring tool. Acquiring key English patterns greatly simplifies the template structure, which effectively controls an exponential explosion in the number of possible combinations inherent in many natural-language applications.

Intelligent language tutoring systems (ILTSs) over the Internet have emerged as an important tool for distance education and computer-assisted language learning (CALL) systems, where students can learn foreign languages in multimodal form anytime and anywhere. However, most ILTSs permit only multiple-choice responses, which don't allow free-form input.

Researchers understand and accept these restrictions because we know that natural-language understanding and acquisition have remained one of the unresolved problems in artificial intelligence.[1] In fact, we still understand very little of a human's ability to process language. Intrinsic difficulties involved in human-language understanding include such features as context-sensitive grammar[1] and word-sense disambiguation.[2]

Strategies for creating intelligent systems largely depend on existing natural-language processing (NLP) techniques that researchers developed in overcoming the system restrictions.[3-5] Unfortunately, most of these systems can only deal with simple sentences, reflecting the state of art in the current NLP research level.

Here we introduce an ILTS over the Internet that could replace an experienced human-language teacher, processing complicated input sentences based on relatively simple algorithms.

To overcome some of the many difficulties expected in NLP, we adopted the following strategies:

1. The system emphasizes acquisition of basic English patterns[6] throughout the course.

2. The global-matching algorithm of longest or heaviest common sequences stabilizes the unstable augmented transition network (ATN) because of its local matching. The ATN provides a means of constructing well-formed sentences where arcs are traversed under certain conditions and the registers save the intermediate results.

Item 1 plays an important role in simplifying the template structure because it not only eliminates the use of unnecessarily complex sentence patterns throughout the tutoring system but controls and avoids a possible exponential explosion in the number of possible combinations inherent in any natural-language application.

Item 2 represents an important extension we made for providing a robust diagnosis of students' input. We embedded error categories compiled from many monitors' wide-ranging responses[6] into the template form so that the global-matching algorithm provides correct error-contingent feedback.

In this article, we refer to a template as part of an ATN[7] where every node consists of a word(s) or a phrase(s), a syntactically or semantically misused word(s) or phrase(s), and appropriate error messages. Hence, the template constitutes the so-called pedagogic knowledge base where we construct a syntactically valid sentence by linking these nodes from a sentence's starting node to a terminating final node. Because any sentence can usually take on several structures in network paths, each structure forms a different network, called a subtemplate. By assigning non-negative real numbers to each word of the template that represents the relative importance of the word within the sentence, we can design a global-matching algorithm into the ATN's data structure. This comprises our tutoring system's diagnosis engine.[8]

A parser is another important module in our tutoring system. It checks the syntax of input sentences, thus facilitating the subsequent diagnostic and remedial processing. We devised a parts-of-speech tagged (POST) parser, which uses the system's template structure. We manually preassign all the parts-of-speech (POS) tags to all words of selected sentences in the template as in the Penn TreeBank.[9] Eliminating ambiguities of POS improves the syntactic analysis' accuracy. In

1070-986X/01/$10.00 © 2001 IEEE

addition, a compound word dictionary of phrases also improves the parser's accuracy.

The visual interface module's visual template authoring tool (VTAT) is another important system component. We designed it to facilitate and simplify the language experts' task of template construction.

Our ILTS first presents students—via the Internet—English texts characterized by typical key English patterns and the pattern-based problems for translation. When ready, students send back translations of the assigned problems from their PCs, which the system accepts as keyed-in through the Common Gateway Interface (CGI). We can implement all this quite efficiently in HTML. Audio files of native speakers and point-and-click explanations of new vocabulary from our own dictionary accompany the texts and problems so that non-native students can learn correct pronunciations and new vocabulary words. The server then checks for possible spelling errors by initiating the spell-check module. The matching algorithm of heaviest common sequence (HCS) or longest common sequence (LCS) selects an optimal sentence having the highest similarity value (or HCS) with the input sentence, which in turn provides an error-contingent feedback consisting of error comments and an adaptive remediation to students.

Figure 1 shows our online tutoring system's main interface.

Acronyms

ATN	augmented transition network
CALL	computer-assisted language learning
CGI	Common Gateway Interface
HCS	heaviest common sequence
ITLS	intelligent language tutoring system
LCS	longest common sequence
NLP	natural-language processing
POS	parts of speech
POST	parts-of-speech tagged
VTAT	visual template authoring tool

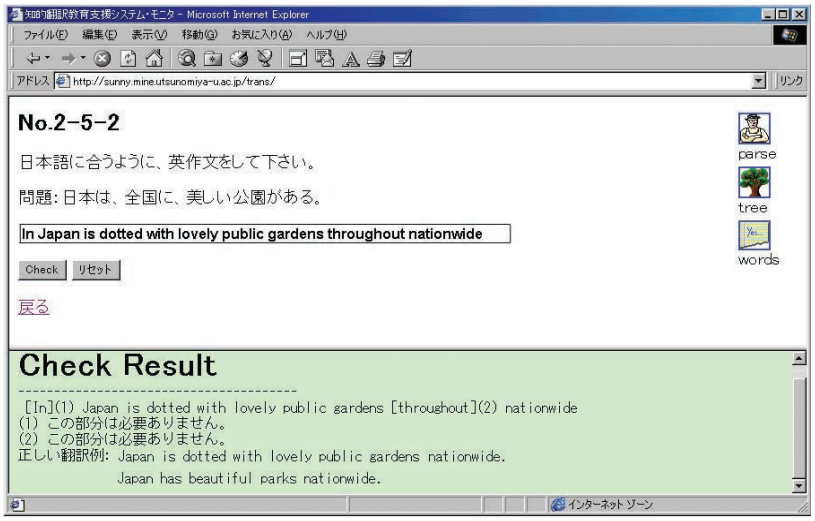


Figure 1. CGI form interface. The core interface of ILTS allowing free inputs from learners and returning error-contingent feedbacks to learners.

Template and error classification

Our tutoring system implements the efficiency and intelligence equivalent to an expert human tutor. We achieved this by using template structures that restrict sentence patterns.

Template structure

The template is our tutoring system's most important component. It consists of decomposed units of model sentences including learners' erroneous sentences consisting of either words or phrases so that the system analyzes the data in smaller units rather than the original sentences. A bilingual English native speaker collects the data into templates and

- presents model translations in conformity with the sentence patterns students have just studied in the texts, and
- diagnoses and corrects various errors in input sentences based on learners' responses.

Error classifications

The template structure we developed consists of mainly native speakers' model translations and detailed descriptions of ill-formed or ill-composed sentences from students. Typical errors are

1. spelling, including misused upper or lower cases such as "tokyo" or "two Days;"
2. syntactic, such as VS meaning "the verb must be singular, since the subject is singular;"
3. semantic, which implies the erroneous use of a word in a special situation: "MN: meaning is incorrect in this context;" and
4. unclassified, which can be grouped into the following three types of errors:

- *Wrong.* A part of the sentence is wrong because it differs from the assigned path node's template.

- *Missing*. A part of the template is missing from the input sentence.

- *Redundant*. A part of the input sentence isn't included in the template.

Obviously we can't deal with all the possible errors committed by students even with our template mechanisms. To reduce the possibility of our system missing an error, we first preprocess misspelled words not found in dictionaries. Our basic strategy for misspelled words is the following:

- When our system detects a word in the input sentences that's not in the dictionary, use the spell-check module based on the LCS algorithm.[10]

- Among all possible candidates in the template, find the words having the first three longest common sequences.

- Present the three most probable candidates of the misspelled word for students to choose from.

We present the top three candidates to students so that they can choose the best candidate by themselves. This simple method seems to work well in our matching module.

A detailed classification of errors and the contingent error-specific messages are vital for remedial purposes and hence for the completion of an effective ILTS. Unlike the roughly 80 bugs found in the Buggy Model,[11] we developed about 170 error classifications. As we show in the "System evaluation" section, the system works remarkably well when tested against the groups with an entirely different educational background from those groups that used the current template databases as long as input sentences were reasonably well formed. We may build in a machine-learning system for adding and expanding the template databases, which we hope will lead to a more practical tutoring system so that students will be able to better understand the error comments.

HCS matching algorithm

The template-matching algorithm plays a key role in guessing and selecting an optimal pattern from among the template's valid paths that have the largest similarity to the input sentence from the student. Here we measure similarity by a weight or a number of the resulting HCS or LCS[10] in matching the input sentence to the template paths. The latter (LCS) is a special case of the former (HCS) when the weights of words are uniform among all words. We'll now show how our global-matching algorithm based on HCS works using the input sentence and the ATN equivalent template structure.

The input to the system is a sentence consisting of a sequence of words. The system compares the input against a template consisting of a finite set of sentences specified by a regular expression. The expression provides a means of efficient string matching and is well understood in the field. A syntactically legal sequence of words in the regular expression is a sentence called a template path. We can associate the template's words with positive real numbers, which represent the word's weight and indicates its relative importance within the sentence.

We now search for a common sequence in words of the sentence from among all the possible valid template paths. An LCS has the largest number of common words between the input sentence and the candidate template path. A common sequence's weight is the sum of the weights of all the words in the sequence. The HCS algorithm finds the weight's HCS.

Here's how we describe the problem of finding an optimal path. Given the sentence and the template, find one path so that the weight of the path's HCS and the sentence isn't less than that of any common sequence consisting of any valid paths of the template and the sentence. To solve this problem, we use the HCS algorithm as follows:

1. Turn the template into an ATN so that the directed edges (transitions) are labeled by the corresponding words in the template, and the ATN accepts exactly all the template paths.

2. Topologically sort all the nodes (states) of the ATN into $N_1, N_2, \ldots, N_n$, such that for each pair of nodes N_i and N_j, there's no transition from N_j to N_i when $j > i$ (j and i are number indexes taking on integer values 1, 2, 3, …, n.

3. Suppose that all the m number of words of the sentence are $M_1, M_2, \ldots, M_m$ (all the words in a sentence in sequence from the beginning to the end). For all $1 \le i \le n$, $1 \le j \le m$, we associate the pair N_i, M_j with a variable $w_{i,j}$ as the temporary weight of a relatively good common sequence of a path in the ATN ending at N_i and the sentence $M_1M_2 \ldots M_j$. The pair N_i,M_j is associated with a link capable of remembering

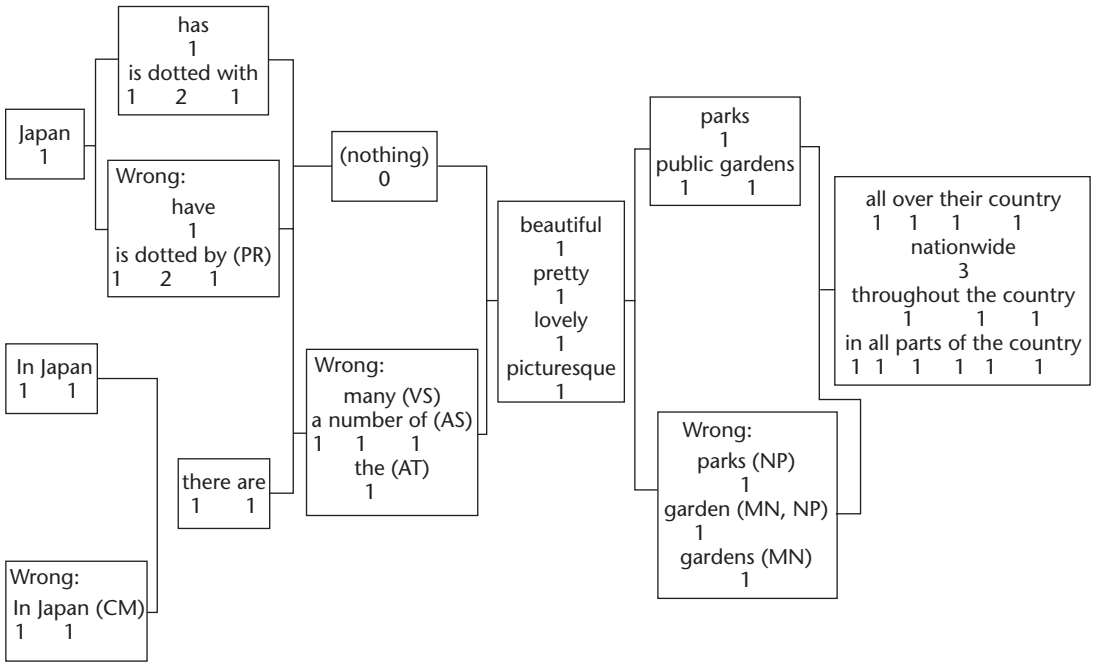


Figure 2. A typical template example of English translation for the Japanese sentence meaning "Japan is dotted with beautiful parks nationwide."

the information of the last transition of the path in the ATN spanning this temporary good common sequence with the sentence $M_1M_2 \ldots M_j$. For $i = 1$ to n, where n is the total number of nodes of the ATN, repeat the following procedures of steps 4, 5, and 6.

4. For each of the transitions starting from the node N_i, repeat the following procedures of steps 5 and 6.

5. For $j = 1$ to m, repeat step 6.

6. Suppose the transition considered is T, and it terminates at node N_p. Choose a maximum of real numbers from among the following real numbers: $w[i, j]$, $w[p, j - 1]$, and $w[i, j - 1]$ plus the weight of M_j if the label of T matches M_j. In case this real number chosen is larger than the $w[p, j]$, update $w[p, j]$ to the real number chosen, and reset the associated links that remember transitions.

7. Obtain the path according to the links that remember the transitions.

We can easily show that the time complexity of this matching algorithm is $O(MN)$, where M and N represent the number of the arcs of the template and the number of words in the sentence, respectively.

Suppose an input sentence from a student is "In Japan dotted with lovely public gardens through nationwide." The HCS algorithm will find the path "Japan is dotted with lovely public gardens nationwide" as an optimal path from the template of Figure 2. The HCS of the path and the sentence is "Japan … dotted … with … lovely … public … gardens … nationwide." By having the algorithm choose this sentence as the most probable target sentence that a student intended to type, we simplify all the subsequent diagnosis and remedial procedures. We think this HCS global-matching scheme is essential to our approach. The local-matching scheme, on the other hand, can't process the input sentence because the local algorithm fails to find a matching pattern with the input.

POST parser

An efficient parser is basic to any NLP system[12] including machine translation between lan-

Parser Notations

The notations we use follow those of the Penn TreeBank:

","	Comma
ADJP	Adjective phrase
ADVP	Adverb phrase
DT	Determiner
EX	Existential there
IN	Preposition or subordinating conjunction
JJ	Adjective
JJS	Adjective, superlative
MD	Auxiliary verbs including do, did, does, can, could, dare, may, might, must, ought, shall, should, will, would
NN	Noun, singular or mass
NNP	Proper noun, singular
NNS	Noun, plural
NP	Noun phrase
NPL	Low level noun phrase
POS	Possessive ending
PP	Prepositional phrase
PRP	Personal pronoun
RB	Adverb
S	Sentence or simple declarative clause
TO	to
VBN	Verb, past participle
VBP	Verb, non-third-person singular present
VBZ	Verb, third-person singular present
VP	Verb phrase

guages. In our tutoring system, the parser is the key to diagnosing students' grammatical or syntactical input errors for providing useful tutoring comments. Because context-sensitive grammar of natural languages results in well-known ambiguities of natural language such as POS tags,[12] semantic ambiguity,[2,13] and structural ambiguities, it's difficult to construct a parser capable of processing even a general class of well-formed sentences with a practical precision of say, 90 percent or better. In the absence of such a general parser, we have enough reason to believe that the metarule, buggy rule, or unification-algorithm-based parser system couldn't have higher performance in treating ill-formed sentences.

One way to cope with context-sensitive grammar is to use large corpora with syntactically bracketed tags such as the University of Pennsylvania's Penn TreeBank corpus[9] to build up statistical data. An Apple Pie Parser[14] is a typical probabilistic parser of the kind developed at New York University based on the Penn TreeBank's syntactically bracketed corpus. But the reported precision of 72.61 percent is too low to use in a practical application.

Our aim here is to build a parser best suited for our purposes with a precision of 90 percent or higher when the system parses grammatically correct sentences. We use the parser we developed to parse syntactically correct sentences embedded in templates. That way, all we need to do is to check students' syntactically incorrect sentences against the correctly parsed sentences for pointing out possible errors and for providing appropriate coaching strategies. Manual labor required for POS tagging of paths in templates is reasonable with respect to other human interactions. Even if we eliminate the ambiguity of POS tags we must deal with various English grammars to construct correct parse trees; our parser is certainly easier to use.

Following the Apple Pie Parser, we write the grammars using the two nonterminals S and NP (see the sidebar "Parser Notations" for definitions). All units starting with S and NP are called structures that can be split into smaller structures until all terms on the right-hand side consist of constituents or leaves. For example, the sentence, "Aside from Nomura's injured pride, the biggest victim so far has been the New Zealand government" has the following syntactically bracketed structure in the Penn TreeBank:

(S (PP (RB) (PP (IN) (NP (NP (NNP)) (POS) (JJ) (NN)))) (, ,) (S (NP (DT) (ADJP (JJS)) (NN)) (ADVP (RB) (RB)) (VBZ) (VP (VBN) (NP (DT) (NNP) (NNP) (NN)))))

We can split the sentence into six small structures, each of which starts with S or NP:

- (S(PP(RB)(PP(IN) NP))(,)S)
- (NP NP (POS) (JJ) (NN))
- (NP (NNP))
- (S NP (ADVP(RB)(RB)) (VBZ) (VP (VBN) NP))
- (NP (DT) (ADJP (JJS)) (NN))
- (NP (DT) (NNP) (NNP) (NN))

It's possible to split each of the syntactically bracketed structures into several smaller structures until all contain the parse tree's constituents or leaves. From the pedagogical purpose, particularly in the present language-tutoring environment, all the POS tags of well-formed sentences should be known beforehand whether by system or human experts. To implement this by system, our

strategy is to construct parsed trees from among a list of structures in the corpus having the parsed tree with the specified POS tags. For example, consider parsing the following POS tagged sentence: "The/DT best/JJS article/NN until/RB now/RB has/VBZ been/VBN John/NNP 's/POS new/JJ paper/NN." Splitting the sentence into the following structures (S NP (ADVP (RB) (RB)) (VBZ) (VP (VBN) NP)), (NP (DT) (ADJP (JJS)) (NN)), (NP NP (POS) (JJ) (NN)), and (NP (NNP)), we can obtain the parse tree shown in Figure 3.

Given a sentence having all POS tags manually specified, we expect that the system can select many different subsets of structures for constructing parse trees for the sentence. To deal with the situation, we need a so-called probabilistic chart parser. In our language-translation tutoring system, we compute the probabilities of grammars by the following formula, which gives scores for the chosen tree structures:

$$P_{\text{tree}} = \prod_{\text{Struc}_i \in \text{tree}} P_{\text{Struc}_i}$$

where $P_{\text{Struc}_i}=F_i/T_i$, F_i denotes the frequency of the structure Struc_i in the corpus and T_i the total frequency of the structures starting with the same symbol of Struc_i.

When several possible parse trees exist for one sentence, we choose the one with the highest score. To construct the probabilistic parser, we use a standard bottom-up chart parsing algorithm with the POS tags fixed. In our approach we add a step that computes a score for each entry to the chart so that we can select the best constituent from many candidates having the same type of input strings.

The POST parser we developed has advantages over the other statistical parsers such as Apple Pie. Figure 4 shows parsing for the sentence "I do love Iris" with Apple Pie and our POST parser. Obviously the tree obtained by Apple Pie isn't correct. Note that for the POST parser, we must assign each word to the POS before we parse the sentence.

To further improve the parser's accuracy, we had to develop a compound word dictionary of idiomatic phrases. Consider parsing POS tagged sentences having idiomatic phrases where any parser has difficulty in getting correct parse trees. For example, in our compound dictionary, we always regard "a lot of" as an adjective (JJ). Thus, the POS tagged sentence "There/EX are/VBP a/DT lot/NN of/IN pens/NNS on/IN the/DT table/NN" will be parsed into the tree: (S (NP (EX There)) (VP

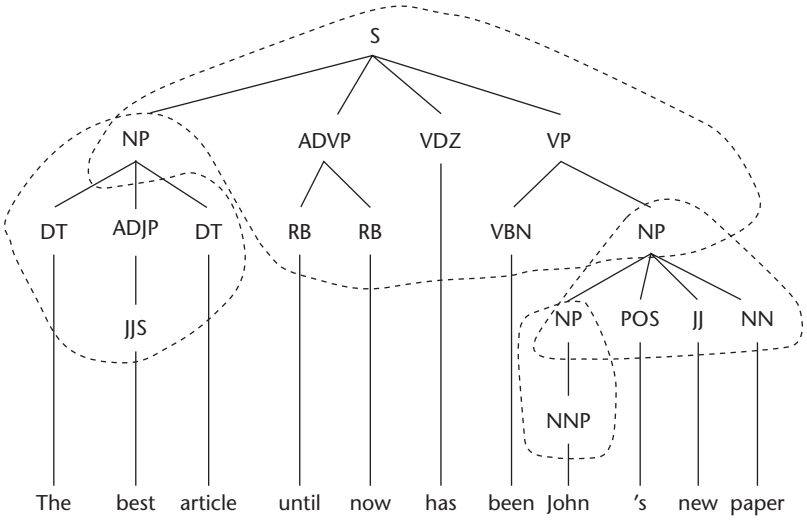
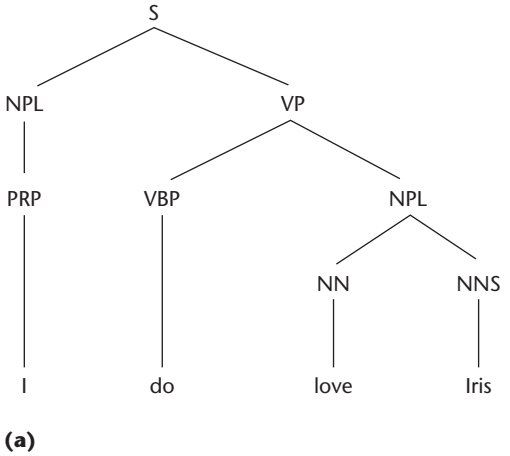
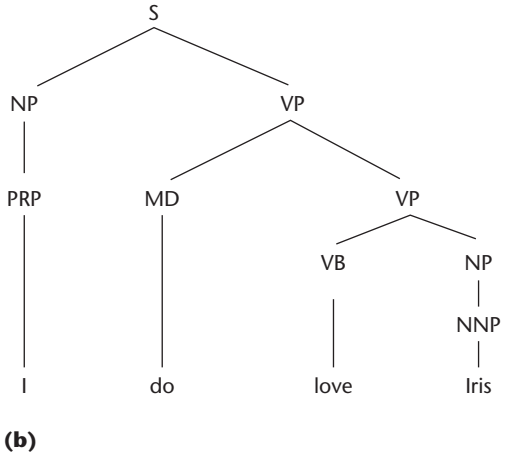


Figure 3. A parse tree for the sentence "The best article until now has been John's new paper."



(a)



(b)

Figure 4. Parse trees of the sentence "I do love Iris." (a) A parse tree by Apple Pie. (b) A parse tree by our POST parser. (See the sidebar "Parser Notations" for explanations of the grammatical symbols.)

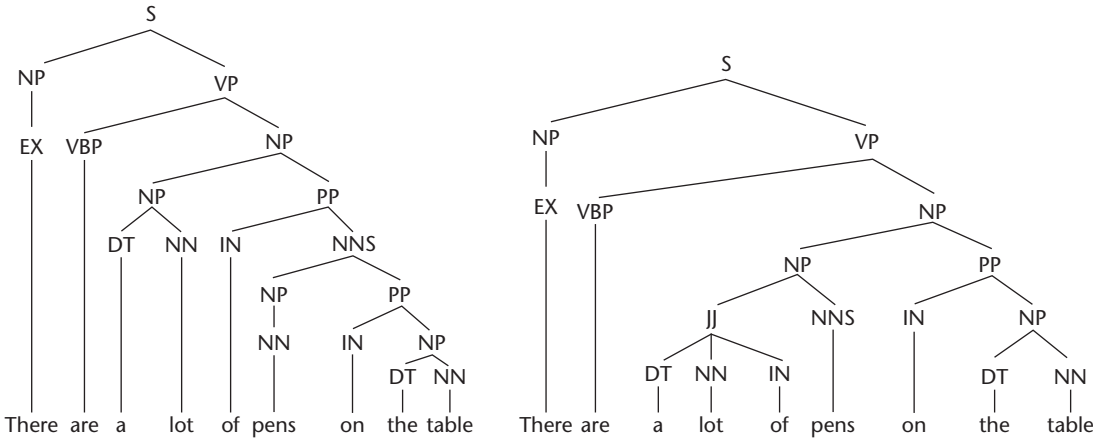


Figure 5. Parse trees without (a) and with (b) a compound word dictionary of phrases.

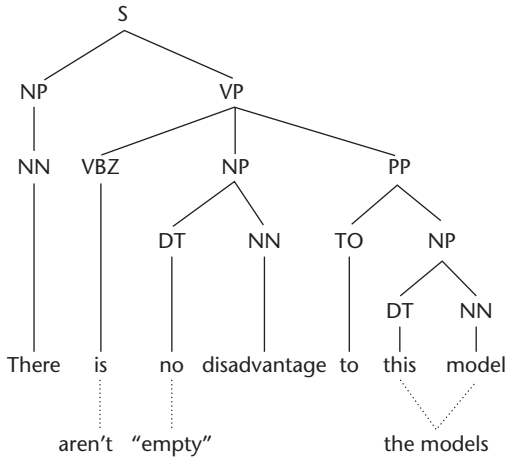


Figure 6. A parse tree of a correct sentence by the POST parser.

(VBP are) (NP (NP (JJ (DT a) (NN lot) (In of)) (NNS pens)) (PP (IN on) (NP (DT the) (NN table)))))) (see Figure 5). Note that, without a compound word phrase dictionary, the result would be: (S (NP (EX There)) (VP (VBP are) (NP (NP (DT a) (NN lot)) (PP (IN of) (NP (NP (NNS pens)) (PP (IN on) (NPL (DT the) (NN table)))))))).

Given the additional POS tags, the POST parser should perform much better than the parser without POST. Our experiments with the POST parser including the compound words found no obvious errors if well-formed sentences are input for parsing. Under general conditions, ill-formed sentences may not be correctly parsed, but fortunately we don't need to parse such sentences. To diagnose syntactic bugs and give error-contingent feedback, we must tell the difference between well-formed and ill-formed sentences.

When the system matches a keyed-in sentence to a correct sentence in the template, the probabilistic POST parser will generate a correct parse tree for the correct sentence. Then by collating the resulting parse tree and the keyed-in sentence, it should be easy to provide appropriate grammatical comments to students. We expect the parser to be effective only when the keyed-in sentence doesn't contain too many errors. Figure 6 gives an example where the system matches the input sentence "There aren't disadvantage to the models" to the correct sentence "There is no disadvantage to this model." When the system obtains the correct sentence's parse tree, it will return the following grammatical comments (noted in parentheses):

There aren't (1) (2) disadvantage to the models (3).

1. "Aren't" should be "is," which is the present tense of the third-person singular for the noun phrase "no disadvantage" (see comment 2).

2. The word "no" must be added together with the noun "disadvantage," which forms the noun phrase of the verb phrase "is."

3. "The models" should be replaced by the noun phrase "this model," where "this" is a determiner and "model" is a singular noun.

Visual template authoring tool

To help a native speaker of the target language (an English expert in our case) input and edit the templates and comments or error messages associated with the templates, we developed a visual authoring tool called VTAT (see Tokuda et al.[6] for the operation manual). Using this tool, the native speaker can not only draw the template-automaton on screen by simple operations such as drag and drop but also construct the links of connections between words, manage the comments, and link them to the proper position of the templates.

The tool has three parts: a template canvas, node editor, and comments editor. The template canvas draws and constructs the template structure. The node editor defines words and phrases and ATN's properties. The comments editor orga-

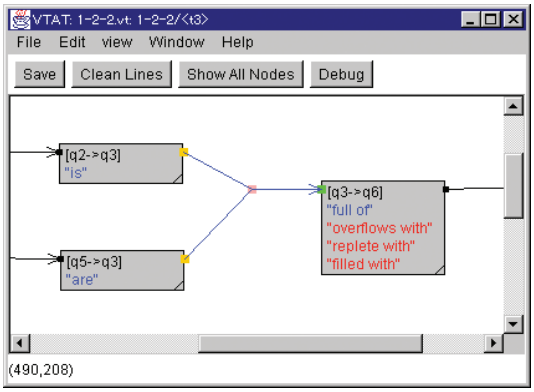


Figure 7. Template canvas.

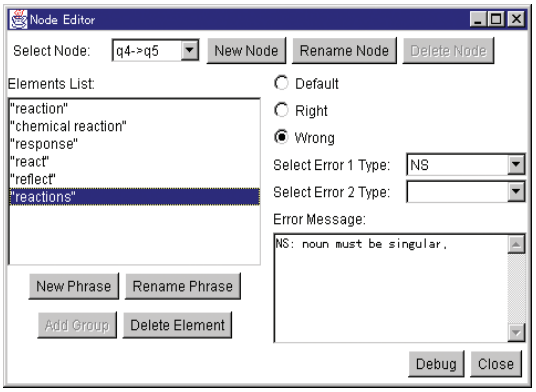


Figure 8. Node editor.

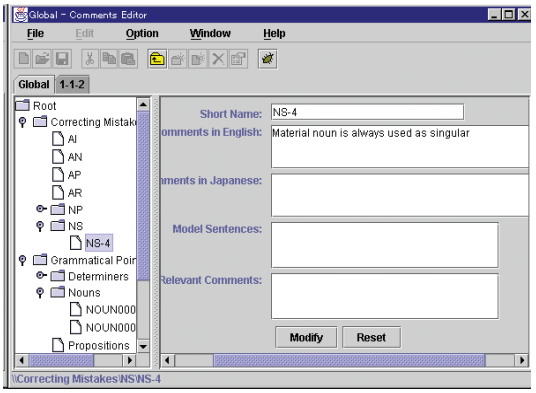


Figure 9. Comments editor.

nizes the comment databases. The comments can be organized in a tree structure defining several comment categories or groups such as error-correcting messages, grammatical errors, and/or word usages, and so on. To facilitate easy management of hundreds of comments under each category, we defined several subcategories. Figures 7, 8, and 9 show the different interfaces of VTAT to implement easy template editing and subsequent visualization of templates.

System evaluation

The current ILTS performance depends on the error diagnosis based on the global HCS algorithm. Table 1 summarizes the results of the two groups we tested. *N* represents the number of responses.

In Table 1, group 1 denotes the group of learners comprising about 200 (strictly 160 or 163) professional or semiprofessional translators or those taking such courses at the Sunflare Translation Academy used in Tokyo, Japan, while group 2 denotes the group of learners consisting of 100 freshmen at a typical Japanese university used (52 belonged to the Engineering School and 48 to the School of International Studies). The distinct difference between the two groups can be traced to their educational background of English writing. Group 2 seldom practices speaking or writing English in the current Japanese educational system. Table 1 shows the evaluation results for templates 02-2-3, 02-3-3, and 02-4-3 and the two groups for comparison (see Tokuda et al.[15] for details of the problems associated with these templates).

The diagnosis classified by group A (A1, A2) in Table 1 refers to the group of responses that exactly match the template classification nodes. A1 refers to the student group whose translations perfectly matched the model translations while A2 refers to those whose translation errors exactly

Table 1. ILTS test results.

Problem Number	02-2-3 Group 2 (N = 100)	02-2-3 Group 1 (N = 163)	02-3-3 Group 2 (N =100)	02-3-3 Group 1 (N = 163)	02-4-3 Group 2 (N = 100)	02-4-3 Group 1 (N = 160)
A1	3 (3%)	97 (60%)	8 (8%)	59 (36%)	0 (0%)	22 (14%)
A2	7 (7%)	23 (14%)	5 (5%)	51 (31%)	2 (2%)	41 (26%)
B1	77 (77%)	39 (24%)	82 (82%)	53 (33%)	73 (73%)	91 (57%)
B2	1 (1%)	2 (1%)	1 (1%)	0	5 (5%)	5 (3%)
TooManyErrors	12 (12%)	2 (1.2%)	4 (4%)	0	20 (20%)	1 (1%)

match those of the template nodes. On the other hand, B1 refers to the group of students who correctly diagnosed the errors although the errors don't match the template nodes. This is possible for the unclassified errors we noted in the "Template and error classification" section.

By adopting the paths having the HCS algorithm's highest similarity score, the system either supplies missing word(s) or phrase(s) to students' input or deletes redundant phrase(s) of word(s) from the students' input. This is classified as B. When we judge that the students' input is corrected to a class of model translations, it's classified as B1. If we judge the original intention of students' input to be the other model sentence, we classify this as B2. The system gives a classification of TooManyErrors if the students' input doesn't match more than half of any template paths or if more than two-thirds of the input sentences have misspellings. More often than not, the system displays this message when students give up the solution because of their inability to translate.

A distinct difference in the statistics of A1 shows the superiority of group 1 in English writing ability over group 2. Because the responses obtained from group 1 are used as a basic input to the template database of the current system, the statistics for A2 and B1 show that translations by group 1 have a better chance of being correctly diagnosed by the current templates' ATN. On the other hand, the dominating number of B2 for group 2 implies that the tutoring ability of the current system is remarkably robust and excellent because the HCS algorithm correctly identifies the most likely paths of the student input from among the template and provides a valid tutoring comment. We should emphasize that we analyzed the test results manually. The system performance in error diagnosis achieved about a 96 percent level even for group 2 as compared to 97 percent for group 1. For further details of the error analysis, see Tokuda et al.[15]

Concluding remarks

Heift and Nicholson[3] recently published an article about a German tutoring system, which operates in a Web environment for introductory German courses. In this system, a learner's model and a parser operate on the Prolog system server while the client side uses Java. Compared to our system, which works even in technical translation applications as long as the language ambiguity problems are minimal, their system is intended for basic beginning classes dealing only with the basic patterns of the language. For example, a Prolog parser may be useful for their purposes but it may not be able to process context-sensitive grammar. In addition, the genetic algorithm used for error diagnosis may not be useful for analyzing the compound errors we want to diagnose because of their computational complexities.

As we confirmed with the group 2, the present ILTS is efficient and robust. Above all, template construction seems quite easy for many language teachers.

Our ILTS has the following characteristics:

- Enhanced pedagogical effects and simple template construction. To ensure this, we adopted Fries' teaching method,[16] which advocates repeatedly learning key sentence patterns.

- Efficient error diagnosis and ease of exploiting the expertise of experienced language teachers. We implemented this into the current ILTS by adopting the string-matching algorithm.

These two characteristics are basic to implementing our online interactive tutoring system over the Internet. Because any natural language can be decomposed into an ATN-based template system, we believe the system can be applied to a language learning system of any language. In addition, our system applies to ILTSs for any combination of natural languages.

Our current diagnostic system deals with grammatical errors as well as word and usage errors. We can expand our system by building in as many versatile responses from different backgrounds as possible. For example, we can adopt the machine-learning algorithm for this purpose. **MM**

Acknowledgments

We acknowledge the help of Diana Cripps, Misako Tokuda, and Hiroyuki Sasai of Sunflare Company, Tokyo, Japan, for providing valuable data for this project including the original English texts and detailed data analysis of about 200 learners used in the template databases. We also thank Zenglei Gu, Zhomping Huang, and Liang Huang—graduate students at Utsunomiya University—for an initial investigation of parsers and ATN and VTAT, respectively.

We presented part of this article at the International Symposium on Intelligent Multimedia and Distance Education (ISIMADE 99) conference in Germany, 2 through 7 August 1999.

References

1. S.M. Shieber, *An Introduction To Unification-Based Approaches to Grammar* (Center for the Study of Language and Information Lecture Notes), vol. 4, Stanford Univ. Press, Stanford, Calif., 1986.
2. D. Yarowsky, "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods," *Proc. Assoc. for Computational Linguistics 95*, Cambridge, Mass., 1995, pp. 189-196.
3. T. Heift and D. Nicholson, "Theoretical and Practical Considerations for Web-Based Intelligent Language Systems," G. Gauthier, C. Frasson, and K. VanLehn, eds., *Proc. Intelligent Tutoring Systems. 5th Int'l Conf.* (ITS 2000), Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 1839, 2000, pp. 354-362.
4. H. Yamamoto, M. Sakayauchi, and R. Yoshioka, "Development System of Japanese Teaching Material Using C{A}{S}{T}{E}{L}," *Proc. 20th Ann. Conf. Japan Society for Science Education*, Japan Soc. for Education, Tokyo, Japan, 1996, pp. 105-106 (in Japanese).
5. J.C. Yang and K. Akahori, "Error Analysis in Japanese Writing and Its Implementation in a Computer Assisted Language Learning System on the World Wide Web," *CALICO J.*, vol. 15, no. 1-3, 1996, pp. 46-66.
6. N. Tokuda et al., *Progress Report for Tokyo Government on Online Interactive Intelligent Tutoring Systems for English-Japanese Translation via Internet*, tech. report, Computer Science Dept., Utsunomiya Univ., Utsunomiya, Japan, 1995.
7. W.A. Woods, "Transition Network Grammars for Natural Language Analysis," *Comm. ACM*, vol. 13, no. 10, 1970, pp. 591-606.
8. N. Tokuda and L. Chen, "A New Efficient Diagnostic System for Language Translation ITS by Global Template Matching," *Proc. Int'l Conf. Artificial Intelligence*, Computer Science Research Educational and Application, Las Vegas, 1999, pp. 129-135.
9. A. Bies and M. Ferguson, *Bracketing Guidelines for Treebank Style Penn TreeBank Project*, tech. report, Linguistic Data Consortium, Univ. of Pennsylvania, 1990.
10. T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Mass., 1990.
11. Z. Huang and N. Tokuda, "A Syntactical Approach to Diagnosing Multiple Bugs in an Intelligent Tutoring System," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 26, no. 2, 1996, pp. 280-285.
12. J. Allen, *Natural Language Understanding*, Benjamin/Cummings Publishing, San Francisco, 1995.
13. Z. Gu, N. Tokuda, and Y. Ye, "An Improved Yarowsky's Unsupervised Word-Sense Disambiguation Algorithm by Stairway-Like Threshold Step," submitted to *Trans. Information Processing Soc. of Japan*, 2001.
14. S. Sekine and R. Grishman, "A Corpus-Based Probabilistic Grammar with Only Two Non-terminals," *Proc. 4th Int'l Workshop Parsing Technology*, Prague, 1995, pp. 216-223.
15. N. Tokuda et al., "Template-Automaton-Based ILTS for Japanese-English Composition," *The Institute of Electronics, Information, and Communication Engineers Trans. Information and Systems*, vol. J.84-D-1, no. 7, 2001, pp. 1089-1101 (in Japanese).
16. C.C. Fries, *Teaching and Learning English as a Foreign Language*, Univ. of Michigan Press, Ann Arbor, Mich., 1945.

Naoyuki Tokuda is the director of the research center of SunFlare. in Tokyo, Japan. His present interest includes intelligent tutoring systems in artificial intelligence, sorting analysis, computer graphics, image analysis and recognition, and informatics education. He has received his BS, MS, and PhD degrees from Yokohama National University in 1959, Stanford University in 1962, and University of Michigan in 1966—all in mechanical engineering, respectively. He also received his DSc from the University of Tokyo in 1975. He is a member of the IEEE Computer Graphics Society, Information Processing Society of Japan, Society for Science on Form of Japan, Physical Society of Japan, and Japan Society of Fluid Mechanics.

Liang Chen is a senior research scientist at Sunflare, in Tokyo, Japan. His current research interests include pattern recognition, computational geometry, intelligent tutoring systems, and the computational intelligence fields, including fuzzy system, neural network, and fast approximate practical algorithms for solving some NP hard problems. He received his BS in computer software at Huazhong University of Science and Technology, China, in 1988, and his PhD degree in computer science from the Software Institute, Academia Sinica in Nanking, China, in 1994.

Readers may contact the authors at Sunflare Company, Shinjuku Hirose Bldg, Yotsuya 4-7, Shinjuku-ku, Tokyo, Japan 160-0004, email tokuda@cc.utsunomiya-u.ac.jp or lchen_97@yahoo.com.