
A New Template-Template-Enhanced ICALL System for a Second Language Composition Course

LIANG CHEN

*Mathematics and Computer Science Program
University of Northern British Columbia*

NAOYUKI TOKUDA

*Research and Development Center
SunFlare Company LTD,
Tokyo, Japan*

ABSTRACT

To solve a bottleneck problem of authoring a finite state automaton (FSA)-based ICALL system capable of automatically correcting free format English composition sentences, we have developed a new template-template scheme to simplify and streamline the labor-intensive input processes of the network of possible answers. This is achieved by exploiting a simple rule-based approach to the template construction which allows us to integrate complex template patterns into a simpler single template and also to generate many patterns, particularly those of grammatical errors, automatically. This approach contributes to a reduction both in computational processing time and complexity of the system and opens a wide door for a new natural language processing (NLP)-based dialogue system (written and spoken) of future man-machine interactive systems.

KEYWORDS

Template-Template Structure, Dialogue System, Knowledge-Engineer (KE)-Free ICALL, Finite State Automaton

1. INTRODUCTION

In designing intelligent computer-assisted language learning (ICALL) systems, one of the most desirable goals, and perhaps the ultimate goal, we want to achieve is to implement a human-like written and/or spoken dialogue system capable of correctly analyzing and diagnosing errors of students' free-format sentences so that error-contingent feedback is returned to learners in accordance with the linguistic requirements of the grammar, word usage, and semantics of the target language. This concept is in sharp contrast to the currently predominant computer-based approach of gap-filling or multiple-choice type input in



which feedback can often only tell learners if the final answer is right or wrong. Processing free-format input requires natural language processing (NLP) if we want to be able to analyze and diagnose the thinking processes involved in learners making decisions about how to formulate sentences. Among the current NLP techniques, grammar analysis is one of the most important elements.¹ However, we know that, at least up to now, the level of NLP technology alone is not advanced enough to provide the tools for such an analysis, not to mention the analytical tools for the more difficult analysis of language use and semantics. Even a simple, purely syntactic parser with reasonably high precision, say of 90% or 95%, is hard to obtain for possible use by language teachers, due largely to the problem of context sensitive grammar (Allen, 1995).

To solve such problems, many researchers have developed a variety of parsers for special domains, some resorting to a corpus-based statistical approach (Sekine & Grishman, 1995; Henderson & Brill, 1999, 2000). This approach resulted in improved performance in special situations. In our tutoring application, taking advantage of a restricted number of model translations, we have developed a part-of-speech-tagged (POST) parser in which language teachers pre-assign tags to some of the most ambiguous words or phrases. The current version of the template-based online ICALL system, *Azalea*, has been developed for an English as a second language composition course (Tokuda & Chen, 2001). It consists of a template automaton structure (Hopcroft & Ullman, 1979) for knowledge representation, a diagnostic engine which is based on an heaviest common sequence (HCS) matching algorithm, a POST parser for syntactic analysis, a parser-based learner model, and a visual template authoring tool (VTAT). The template is best built by language teachers in order to take full advantage of their pedagogical expertise. It is also necessary to monitor students' translation samples that were gathered in advance. This knowledge-engineer (KE)-free ICALL construction (Tokuda & Chen, 2002) became possible partly by the visual authoring tool and the finite state automaton system adopted for this purpose. Although the automaton-like process is easy for language teachers to follow, the task of building a template corpus consisting of well formed model translations and erroneous ill formed sentences is quite labor intensive taking up considerable time in the entire project. It is to solve this bottleneck problem of the authoring task that we want to introduce the template-template scheme where we employ certain NLP techniques, such as rules and buggy rules (Huang & Tokuda, 1996), to facilitate the construction of larger templates with richer information.

Our paper is organized as follows: Section 2 will introduce the techniques used in constructing template-templates which are in turn expanded and integrated into the original form of templates. Section 3 gives a brief account of the error diagnosis engine based on the HCS matching algorithm and a unique learner model based on the minimum error subtree concept of the new POST parser. Together with the template-template construction, the error diagnosis engine and the learner model are the key components of our system. Section 4 gives an



evaluation of the system performance verifying the validity of the present template-automaton system.

2. TEMPLATE-TEMPLATE STRUCTURE

As mentioned in section 1, templates play a key role in our knowledge database: they are how we represent the domain knowledge. Once built as a corpus (of sentences), the knowledge database of templates can be easily retrieved and reused for facilitating the further construction of the template database. The template structure has been introduced to represent all of our pedagogical knowledge including not only well formed model translations but also ill formed or ill composed sentences containing typical errors made by students. Because the task of building a template amounts to collecting possible errors, it is understandably quite labor intensive.

To help language teachers develop pedagogical content of good quality, we will present a new scheme which facilitates the template construction—the template-template. Two different sets of rules for expanding a template-template into a number of templates are described below. The first set of rules can be used by language teachers directly to construct the template-template; the second set of rules helps language teachers to build non-preclassified bugs into the templates. In fact, the rules may be kept running in “background.”

2.1 Expansion Rules for Template-Template

If an adequate number of erroneous sentences are to be built into a corpus of templates, the possible number of L2 (English) translations of one single L1 (Japanese) sentence is huge unless some constraints are imposed. To restrict the number of translations and thereby reasonably constrain the templates, we ask the learner to use the keywords or idioms they have learned in a given lesson.

We will first define the term “template-template.” The template-template is defined as a template which can be expanded into many templates or into a large template. To activate the template-template structure, we mark the nodes of the template with rule-associated special symbols, introduced below, which allow the expansion of the template-template to one or more of the templates. We first present four simple rules to construct the template-template.

Rule A AP-NAP Rule (AP = appear, NAP = do(es) not appear)

In the new template-template representation, suppose that some nodes are marked with AP_i (i being an index number) and some other nodes marked with NAP_i . The rule imposes the condition that a new, expanded template can include either the nodes marked with AP_i or the nodes marked with NAP_i , but not both of them at the same time.



Rule B PPR-PPRP Form Rule (PPR = personal pronoun, PPRP = possessive pronoun)

Similar to rule A, the template-template imposes the condition that the nodes which are marked with PPR i (again, i being an index number) and the other nodes which are marked with PPRP i can generate a set of templates in each of which one personal pronoun appears in the nodes marked with PPR i and the corresponding possessive pronoun appears in the nodes marked with PPRP i .

Rule C AN Rule (AN = arbitrary number)

Rule C imposes the condition that the associated numbers of the nodes marked by AN can be chosen arbitrarily. Suppose that in translating a Japanese sentence of “I have 15 books on Zen,” a learner returns the translation “I have 5 books on Zen.” The AN rule can be used to denote an error node of 15 because any number other than 15 is an error, for example.

Rule D CHO Rule (CHO = choose one)

Rule D imposes the condition that one and only one node marked by CHO i can appear in any template extracted from the template-template.

We will illustrate how these rules are applied in section 2.4 below.

2.2 Buggy-Rule-Generated Templates

The rules of the preceding section are effective in introducing a simpler representation of otherwise complex templates. This process in turn leads to reduced storing space and processing time from a computational point of view. To further reduce the burden of language teachers, we introduce the buggy rules-based template construction. A buggy rule is used to represent a way of generating common erroneous expressions. Consider the following basic grammatical rules:

EX VBP → EX VBZ

PRP VBP → PRP VBZ

NNP VBZ → NNP VBP

EX = existential such as ‘there is’

PRP = personal pronoun

VBP = verb base form

VBZ = verb 3rd person singular present

NNP=proper noun singular.

Because the buggy rules used here are independent of the courses taught or of the number of learners involved, they can be generated automatically. While



they can be constructed manually, the buggy rules can also be obtained from a frequency table of the minimum error subtree used as part of the learner model for the POST parser described in section 3 below.

2.3 Error Classification and Error Messages

The error messages provided through template matching can be classified as (a) grammatical errors, (b) word and usage errors, and (c) non-preclassified errors. Grammatical errors are grouped into the following nine subgroups which all have a similar hierarchical structure.

1 Determiner	2 Noun	3 Punctuation
4 Subject-verb	5 Modifier	6 Tense
7 Preposition	8 Negation	9 Conjunction

Overall there is a total of 164 errors. Some errors will be explained in section 4.

Word and usage errors are also divided into the nine groups.

1 Wrong-word	2 Awkward-word	3 Misspelling
4 Wrong-meaning	5 Technically-wrong	6 Singular form
7 Plural form	8 Sexist	9 Colloquial

Obviously we cannot deal with all possible errors that could be made by students with our template mechanism. Non-preclassified errors can be grouped into three types. The first two we call ‘missing words’—a part of the template is missing from the input sentence—and ‘redundant words’—a part of the input sentence is not included in the template. The third group contains all other non-classifiable errors. Obviously, many of them could be regrouped and reclassified into new grammatical errors or word and usage errors, which would certainly enhance the tutoring. We will discuss how we can turn non-preclassified errors into classified errors in section 4.2.

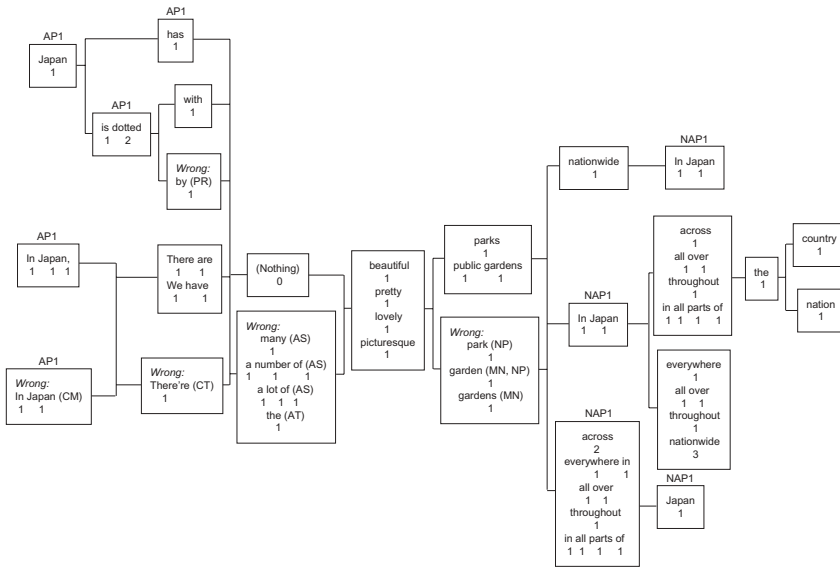
The system performance depends on how accurately errors can be diagnosed and on the fact that error-specific messages can be presented to learners. Compared with about 80 bugs of the Buggy Model (Huang & Tokuda, 1996), a total of 164 distinct errors are now used. As we will show on the evaluation of the system in section 4, the system worked well when tested with two groups with very different educational backgrounds (professional translators and first-year college students).

2.4 An Example of a Template-Template

First, we construct the template-template for an English translation of the Japanese sentence meaning “Japan is dotted with beautiful gardens nationwide” (see Figure 1).



Figure 1
Original Template-Template

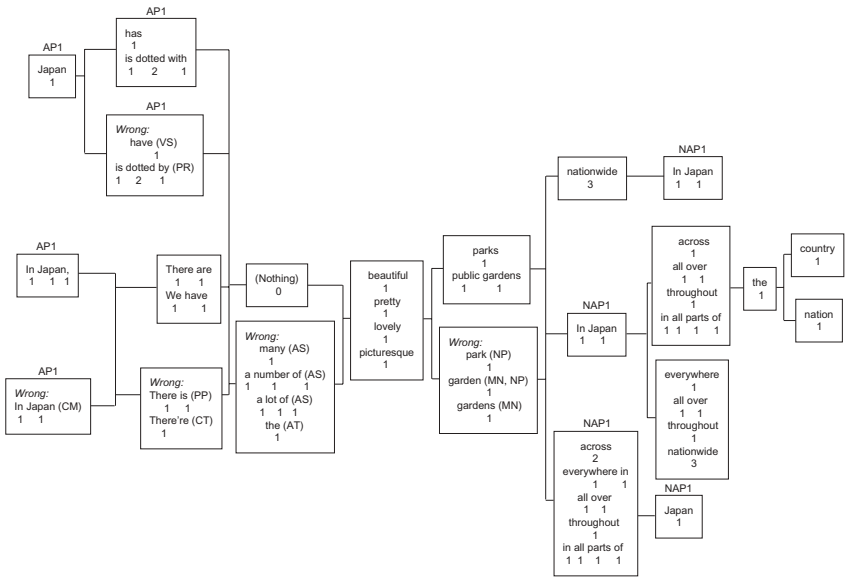


Notes: The following error messages are supplied: (AS) = an assumption has been made on the quantity of noun, (CM) = a comma is needed, (MN) = meaning is incorrect, (VS) = verb must be singular since subject is singular, (PP) = phrase must be plural, (AT) = the article is not needed, (CT) = contraction is incorrect, (NP) = noun must be plural, and (PR) = preposition is incorrect.

By applying the buggy rules listed in subsection 2.2, we can expand this template-template into the template-template in Figure 2. Actually the template-template of figure 1 has much more information than figure 2 has. This shows that a language teacher could ignore some very common errors when he/she is constructing the templates, since buggy rules will expand the template to include these erroneous expressions.



Figure 2
Expanded Template-Template

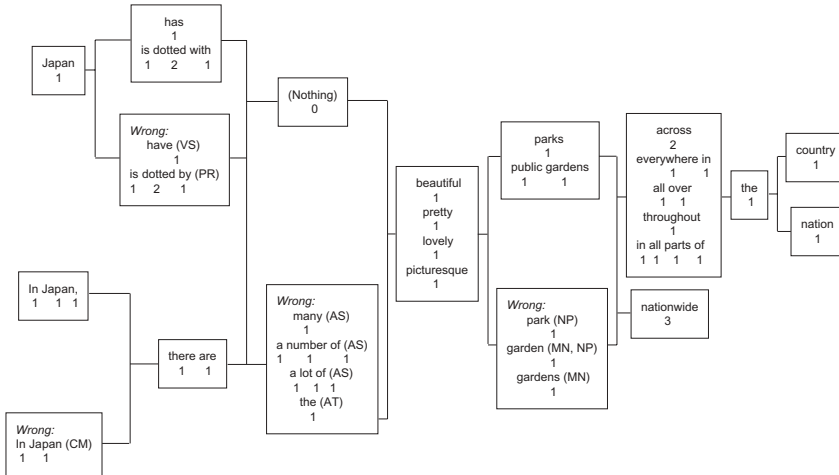


By applying rule A in section 2.1, we can extract from the template-template in Figure 2 template 1—in Figure 3 by deleting the nodes—marked with NAP_i in Figure 2. Similarly, template—2 in Figure 3 has been created by deleting the nodes marked with AP_i in Figure 2. We assume that a language teacher is able to construct the template-template which integrates a large number of templates through simple expressions in the template-template.

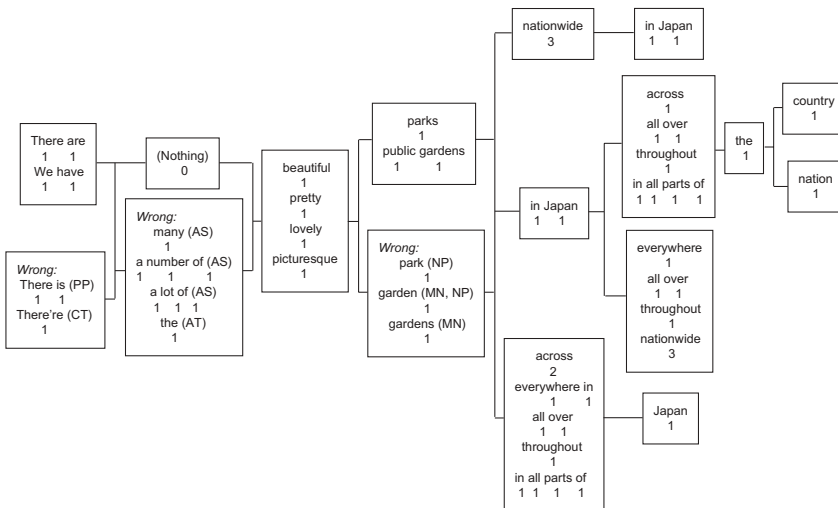


Figure 3
Two Sample Templates

Template 1



Template 2



3. TEMPLATE MATCHING, PARSER, AND LEARNER MODEL

We now apply the heaviest common sequence (HCS) matching algorithm to the template-template. So far, we have applied this algorithm to each of the extracted templates independently (Tokuda & Chen, 2001). We expect this process to be computationally less expensive.



3.1 HCS Matching Algorithm

The knowledge acquisition process of this system exploits a template matching algorithm in which an optimal path is sought from among all the paths of the template. The optimal path is defined

here to have a highest similarity with the sentence provided by the student. Here similarity is determined by the heaviest common sequence (Cormen, Leiserson, & Rivest, 1990) between a student's input and the template path in question. We will show how our heaviest common sequence template matching algorithm works on input sentences and the finite state automaton (FSA) template structures.

The problem of finding an optimal path can be described as follows. Given the input sentence and the template, find one path such that the weight of the heaviest common sequence of the path and the sentence is not less than that of any other common sequence of a path and a sentence. Should two or more paths have the same weight in their HCS with a given sentence, we choose the one which has the lowest error weight. In the case of identical error weight, we choose the one which has the shortest path length. If even this is identical, we choose the one with the fewest erroneous blanks. Further details of the algorithm can be found in Chen and Tokuda (2002).

Let us assume that the student provided the sentence "In Japan dotted with lovely public gardens through nationwide." The heaviest common sequence algorithm will find the path "Japan is dotted with lovely public gardens nationwide" as an optimal path from the templates in Figure 3. The heaviest common sequence of the path and the sentence is: "Japan .. dotted .. with .. lovely .. public .. gardens .. nationwide." By choosing this sentence as the most probable target sentence which the student had intended to type in, all the subsequent diagnosis and remedial procedures are considerably simplified because all errors are now classified and comments and repair schemes are specified.

We think this global matching of a heaviest common sequence is essential to our approach. Unlike unstable local matching which fails if a specific label is not provided in the transitional node, global matching is stable, robust and indeed accurate, as we show in section 4 on system evaluation.

3.2 POST Parser

The part-of-speech-tagged (POST) parser involves a simple modification of so-called corpus-based probabilistic parsers, whereby we pre-assign tags to certain ambiguous words or phrases which would have a large number of candidate tags. Not tagging such words or phrases results in greater processing time often without contributing to improved accuracy. Accordingly, we add constraints on the unnecessarily huge number of potential combinations of tag assignments in the part-of-speech tags of certain words or phrases. We use an Apple-Pie parser (Sekine & Grishman, 1995) which chooses a combination that maximizes the probability of the final grammar tree.



$$P_{tree}(T) = \prod_{rule_i \text{ in } T} P_{rule_i} \cdot \prod_{tag_j \text{ of word}_j \text{ in } T} (P(tag_j | word_j))^2$$

Here P_{rule_i} denotes the probability of a rule ($rule_i$), while $P(tag_j | word_j)$ is the probability of the word $word_j$ being assigned a certain part-of-speech tag (tag_j). Two modifications are needed in the algorithm of the Apple-Pie parser to accommodate pre-assigned part-of-speech tags: (a) Any phrase pre-assigned to a POS is regarded as *one* word and (b) The probability of a pre-tagged word is always set to 1.

The parsing activity completes the analysis. Below we provide the sequence of steps carried out after a student has entered a sentence.

1. Read a keyed-in sentence.
2. Check the sentence with a standard spell-check model and correct the spelling errors.²
3. With the template matching algorithm (Tokuda & Chen, 2001), find the path which has the highest HCS value, provide the lexical error information, feedback information as well as the score for the input sentence.
4. According to the error feedback information, find the nearest correct path in the template.
5. Apply the POST parser to obtain a syntactically bracketed grammar structure for the relevant correct path.
6. Draw the parse tree of the correct path and mark the errors at the leaves of the relevant tree.

3. 3 The Learner Model

We now discuss the learner model which evaluates the learners’ writing competence from two different perspectives—one is based on the HCS matching algorithm, and the other relies on the notion of a minimum syntactic subtree.

The HCS matching algorithm (Chen & Tokuda, 2002) allows us to evaluate the basic writing level of learners by matching each input sentence with the template paths with which the score of the input sentence is calculated:

$$Score = \frac{\text{Weight of HCS}}{\text{Weight of The Matched Patch}} \times \frac{\text{Number of Matched Correct Words in Input}}{\text{Length of Input}}$$

The level of learners is determined on the basis of the scores of the most recent inputs. For example, we can evaluate and re-group learners into three groups, high level learners with mean scores exceeding 0.9, low level learners with mean scores less than 0.7, and middle level learners with scores between 0.7 and 0.9.

Let us start by explaining the notion of a minimum syntactic subtree. Suppose a is the nearest ancestors of leaf b , which has at least two direct descen-



dents. This set of trees is called a minimum syntactic subtree of leaf b , and it includes all direct descendants of a and the ancestors of b up to a . A syntactic error is thus defined as the minimum syntactic subtree of a leaf of the grammar tree that is matched with a word marked as an error.

The procedure of recording the detected syntactic errors is as follows:

1. Obtain the grammar tree of the matched correct sentence in the template by using the POST parser.
2. Match the input sentence to the leaves of the grammar tree.
3. For each leaf l of the grammar tree which is matched with a word marked as an error, find the minimum syntactic subtree of leaf l and associate l with the subtree.
4. For all the subtrees found, combine the leaves that are associated with the same subtree, that is, allow subtrees associated with more than one leaf.
5. For each subtree with the associated leaves, search the user's syntactic error table.
 - 5.1 If there is an identical subtree in the table, add 1 into the frequency field of this row and add all the leaves of the subtree into the associated leaf field of the row, if any of the leaves are not registered as yet.
 - 5.2 Otherwise, add the subtree as well as the associated leaves into the user's syntactic error table and assign 1 to the frequency field for initializing the table.

It is worthwhile to note that the syntactic error tree obtained in this manner can also be used to generate buggy rules which can be used to expand template-templates automatically.

4. SYSTEM EVALUATION

To create the initial set of template model sentences as well as template bugs, we formed a monitoring group, Group 1, of about 200 professional translators, which also included some semi-professional translators, from whom we sampled and collected about 160 responses to assigned translations. The core of the template automaton system is thus based on the data of Group 1 and the expertise of language teachers who participated in our project. To test how the system performs for a different group, we formed another group, Group 2, of 100 freshmen at a typical Japanese university of whom 52 belonged to an Engineering School and 48 to a School of International Studies. The distinct difference between the two groups was their educational background with respect to English composition. Those in Group 2 rarely practiced English composition in the Japanese educational system. The results of the evaluation are given in Table 1 for problems of Templates 02-2-3, 02-3-3, 02-4-3.



Table 1
Test Results of Selected Problems

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

Diagnosis A in Table 1 refers to the group of responses for which we had an exact match with the finite states of the template. A1 refers to a group of excellent students whose translations perfectly matched those of the model translations provided, while A2 refers to those whose errors in translations had exact matches with those of the finite states of the templates, that is, those with preclassified bugs. B1 refers to correctly diagnosed errors which did not exactly match the finite states of the templates. This happened if diagnosed errors did not have a match with preclassified bugs in the templates. B2 refers to those which could not be diagnosed with the information available at the time. Translations were classified as B2 if we judged the original intention of the student input to be model sentence other than the one found. A classification of “Too many errors” was given if the student input matched less than half of the template path or if more than two thirds of the input sentence was not spelled correctly. This message was often returned when a student gave up before completing the translation.

As one would expect, the difference in A1 very clearly shows the superiority of Group 1 in the translation of Japanese sentences into English compared to Group 2. Also as expected, the data for A2 and B1 show that the translations of Group 1 had a better chance of correct diagnosis by the FSA of the templates currently available. On the other hand, the number of B2 for Group 2 indicates that the error recognition ability of the system was remarkably robust. We believe this was due to the fact that the HCS matching algorithm correctly identified the most likely paths for the student input from among the templates. This enabled the system to provide an error-contingent and (we believe) effective tutoring comment. Finally, we would like to emphasize that the data analysis of this section was done by us one by one manually.

We demonstrate below the cases of A1, A2, B1, B2, and “Two many errors” from the evaluation tests.



Figure 4
Template 02-3-3 Case of A1 and A2

Model translation	<i>This process takes advantage of the reaction of oxygen and hydrogen.</i>
Input sentence	<i>This process takes advantage of a reaction of oxygen and hydrogen.</i>
Diagnosis given	Your English composition has an error This process takes [advantage](1) of [a](2) reaction of oxygen and hydrogen.

Comment (1):

Case of A1:

Gives detailed accounts on usage of “take advantage of,” “use,” “utilize,” “exploit,” ...

- 1 “Use,” “utilize,” “employ,” and “exploit” are all suitable translations of “**利用する**.”
- 2 “To use” means to employ something for a purpose or as a means to gain something.
- 3 “To utilize” is a more formal alternative to “to use.” It implies that the use of something is achieved in an efficient manner.
- 3 “To employ” means to make use of something toward a particular goal. (It also means “to hire someone and pay him/her to do a job.”)
- 4 “To exploit” means to make use of something to obtain the greatest possible gain. (It is often used to refer to cases in which workers are treated unfairly to make greater profits.)
- 5 A number of English phrases are also good translations of “**利用する**.” These include “make the most of,” “take advantage of,” “put to use,” “make full use of,” and “capitalize on.”
- 6 “To take advantage of something” is a stock phrase meaning “**利用する**.” It means to employ something as a means for your own gain. It is often used in relation to opportunities. When used in relation to people, it has a negative connotation of “exploit.” (Note that no article is needed before “advantage.”)
- 7 “Making the most of something” is a stock phrase meaning “**十分に活用する**.” It always uses the article “the.”

Comment (2):

Case of A2 Error in the use of articles

- 1 Use “the” before nouns that refer to specific things.
- 2 A chemical reaction between specific substances is always described as “the reaction between A and B.” “The” is used because the reaction is viewed as always being the same process, regardless of when or under what circumstances it takes place.
- 3 This part should be: the



The input uses the sentence pattern of “take advantage of.” Comment (1) gives a detailed account of how the keyword “take advantage of” can be used and of how translations would differ if other verbs such as “make the most of,” “use,” “utilize,” or “exploit” were used.

Comment (2) gives error-contingent feedback on a grammatical error. The use of the definite article is explained for the sentence in question.

Figure 5
Template 02-4-3 Case of B2

Model Translation 1	<i>Unlike large mainframes, notebook PCs are portable.</i>
Model Translation 2	<i>Unlike huge mainframe computers, notebook PC's can be carried.</i>
Model Translation 3	<i>Unlike a huge mainframe computer, a notebook PC can be taken with you.</i>
Input sentence	<i>Unlike big computers, we carry about personal computer which size is the same as note.</i>
Diagnosis given	Unlike big [(1) computers, [we carry about](2) personal computer [which size](3) is [same as note](4).

Comment (1):
Something is missing: “mainframe ”

Comment (2):
This part should be: “a notebook ”

Comment (3):
This part is not needed

Comment (4):
This part should be: “portable ”

Model translations 1, 2, and 3 are typical model translations the language teacher has prepared within the templates. The particular student input is a pattern we had not expected in preparing the original templates. Although our HCS algorithm diagnosed the input closest to model sentence 1 and corrections were suggested in accordance with the sentence, we judged that the student input was intended to be model sentence 2 rather than 1. Therefore, the errors in comments (1) to (4) were classified as B2. In view of the remarkably small number of B2 cases compared to B1 cases, we can conclude that the current system is robust for many types of sentences. Many B1 errors can be converted to more desirable A2 errors by building the necessary bugs into the buggy model.



Figure 6
Template 02-4-3 Case of Too Many Errors

Model Translation	<i>Unlike large mainframes, notebook PCs are portable.</i>
Model Translation	<i>Unlike huge mainframe computers, notebook PC's can be carried.</i>
Model Translation	<i>Unlike a huge mainframe computer, a notebook PC can be taken with you.</i>
Input sentence	<i>Unlike disctop P.C., P.C that like note book can mobile.</i>
Diagnosis given	Unlike [disctop](1) P.C., P.C that like note book can mobile.

Comment (1):

Spelling wrong, maybe: “desktop”. Too many errors, please try again.

In this example, words such as “disctop,” “P.C.,” and “mobile” were far away from the template paths prepared, and the diagnosis was aborted. The case of “Too many errors” frequently occurs when students give up answering altogether.

5. CONCLUDING REMARKS

Based on the template-template structure and the application of the rules for expanding a template-template into templates, we are able to record not only correct translations, but also the erroneous ones in a compact template-template with rich information embedded. We have also verified that the simple HCS matching algorithm is remarkably accurate in its diagnostic capability and is capable of providing a most effective repair or remediation scheme to learners. The POST parser and the related learner model we developed can be used to extract valuable information from the learner’s common errors. For this purpose, we use the compiled “minimum error subtree” in the user’s table. This information allows us to provide personalized remediation to address the learners’ weak points in their L2 which are often influenced by their L1. By making full use of the two-stage repairing process, our system is able to provide personalized remediation not only for syntactic errors, but also for semantic or word usage errors.

The advantages of the system partly come from the template (template-template) structure and our adoption of an advanced NPL-based human-computer interface which allows language teachers to integrate their valuable pedagogical expertise into the system. The advantages also stem from the POST parser which improves parsing precision to a practically useful level. This way we are able to construct personalized syntactical error tables.

Problems still remain, for example, in dealing with the structural ambiguity of the parsing tree. Consider the sentence, “We see a girl with glasses.” Even a



human needs further information from the context to decide if “with glasses” modifies “the girl” or the verb “see.” This should be done by adding certain “tags” to correct sentences in the templates (template-templates). We expect to add certain structural tags to such ambiguous sentences.

We believe that the present approach opens a new door to NLP-based dialogue systems which, we believe, will have many applications in many different areas.

NOTES

¹ A most illuminating finding in this regard has recently been given by Sakai and his colleague Hashimoto (Hashimo'o & Sakai, 2002). They show that a distinct feature of humans in comparison to other animals is their powerful capability to acquire the grammatical structure of a language. To our knowledge, Sakai and his colleague have been the first to prove that the language ability of humans is special and distinct from other general cognitive abilities such as learning and memory. Their most recent f-MRI-based tests provide neuroscientific evidence for hypotheses by Chomsky to the effect that the human brain is indeed specialized in sentence comprehension, distinct from that of other animals.

² To further reduce the possibility of unproductive confusion, we first pre-process misspelled words, which, of course are not available in the dictionary of the system. Our basic strategy for misspelled words is the following:

1. When a word is encountered in a student's input sentence which is not found in the dictionary, a spellcheck module is employed. This module was developed using a longest-common-sequence algorithm by Cormen, Leiserson, and Rivest (1990).
2. The spellcheck module finds three plausible candidate words from among all words of the templates which have the longest sequence in common with the missing word. These words are then presented to the student. We have chosen to present a top candidate to students, and this simple method seems to work very well.

REFERENCES

- Allen, J. (1995). *Natural language understanding*. Menlo Park, CA: The Benjamin/Cummings Publishing Company Inc.
- Chen, L., & Tokuda, N. (2002). Bug diagnosis by string matching: Application to ILTS for translation. *CALICO Journal*, 20 (2), 227-244.
- Chen, L., Tokuda, N., & Xiao, D. (2002). A POST parser-based learner model for template-based ICALL for Japanese-English writing skills. *CALL*, 15 (4).
- Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.
- Hashimoto, R., & Sakai, K. L. (2002). Specialization in the left prefrontal cortex for sentence comprehension. *Neuron*, 35, 589-597.



- Heift, T., & Nicholson, D. (2001). Web delivery of adaptive and interactive language tutoring. *International Journal of Artificial Intelligence in Education*, 12 (4), 310-325.
- Henderson, J. C., & Brill, E. (1999). Exploiting diversity in natural language processing: Combining parsers. In *Proceedings of the 4th conference on empirical methods in natural language processing* (pp. 187-194). College Park, MD: .
- Henderson, J. C., & Brill, E. (2000). Bagging and boosting a treebank parser. In *Proceedings of the 1st meeting of North American Chapter of the Association of Computational Linguistics* (pp. 34-41), Seattle, WA:.
- Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction to automata theory, language, and computation*. New York: Addison-Wesley Publishing Co.
- Huang, Z., & Tokuda, N. (1996). A syntactical approach to diagnosing multiple bugs in an intelligent tutoring system. *IEEE Transactions on Systems, Man, and Cybernetics*, 26 (2), 280-285.
- Murray, T. (1999). Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education*, 10, 98-129.
- Sekine, S., & Grishman, R. (1995). A corpus-based probabilistic grammar with only two non-terminals. In *Proceedings of the fourth international workshop on parsing technology* (pp. 216-223). Prague and Karlovy Vary: Institute of Formal and Applied Linguistics, Charles University.
- Tokuda, N., & Chen, L. (2002, August). *A new development in online ICALL systems: KE-free English composition course with automatic corrections*. Paper presented at CALL 2002, Antwerp, Belgium.
- Tokuda, N. (2002). Introduction: New developments in intelligent CALL systems in a rapidly internationalized information age. *CALL*, 15 (4), .
- Tokuda, N., & Chen, L. (2001). An online tutoring system for language translation. *IEEE Multimedia*, 8 (3), 46-55.
- Tokuda, N., Chen, L., & Sasai, Y. (2002). *System and method for accurate grammar analysis using a learner's model and part-of-speech tagged (POST) parser*. US Patent Application No. 10/072,954.
- Toole, J. , & Heift, T. (2002). The tutor assistant: An authoring tool for an intelligent language tutoring system. *CALL*, 15 (4), .



AUTHORS' BIODATA

Liang Chen is currently an associate professor of computer science at *the University of Northern British Columbia*, Prince George, BC, Canada. He has published in the areas of CALL, fuzzy systems, image and pattern recognition, computer graphics, theoretical computer science, game, and discrete mathematics.

Naoyuki Tokuda, an Emeritus Professor of Computer Science, Utsunomiya University, Japan, is now Director of research and development in SunFlare Co., Tokyo, Japan. He has published extensively in the areas of intelligent tutoring systems including CALL, image and pattern recognition, and natural language processing.

AUTHORS' ADDRESSES

Liang Chen
Math & Computer Science Program,
University of Northern British Columbia,
Prince George, BC,
Canada V2N 4Z9
Phone: 250/960-5838
Fax: 250/960-5544
Email: chenl@unbc.ca

Naoyuki Tokuda
R & D Center, SunFlare Co. LTD., Tokyo Japan
Shinjuku Hirose Bldg, Yotsuya 4-7, Shinjuku-ku
Tokyo 160-0004
Phone: +81-3-3355-1383
Fax: +81-3-3355-1204
Email: tokuda_n@sunflare.co.jp