A New Buggy Rule And Template-Template Based Tutorial Dialogue System

(Extended Abstract)

Liang Chen
University of Northern British Columbia,
Prince George, B.C., Canada
Email: chenl@unbc.ca

Naoyuki Tokuda
Research & Development Center,
SunFlare Company, Tokyo, Japan
E-mail: tokuda_n@sunflare.co.jp

Abstract

To simplify the time consuming authoring task of template generation, we have developed a new buggy-rule and template-template based tutorial dialogue system which has not only simplified the authoring but has markedly improved the system performance. The template-template architecture comprises two schemes; extraction rules are used to expand the classical templates into a variety of template patterns by introducing rule-based symbols in some of transitional nodes of the template automata while the buggy rules introduced are capable of identifying and hence generating bugs from learners' erroneous responses automatically. Many different templates can now be integrated into, or equivalently extracted from, a single template-template so that a considerable reduction is achieved in both space and time complexity of the overall system performance.

The new template-template automaton is expected to play an important role in many other applications including intelligent tutoring systems of both text-based or speech-based dialogue systems, voice-enabling call center or voice portal systems or in fact any systems focused on enhanced as well as advanced human computer interfaces, implementing more natural human computer interactions between the system and humans.

I. Background

For a text-based tutorial dialogue system such as language tutoring system to be successful (Tokuda & Chen 2001), the template database containing a large collection of sentences are needed in representing many plausible combinations of both 'right' and 'wrong' responses from learners because they must be in turn used in selecting a variety of learner's (or user's) response from among the many semantically equivalent paths of the collected database. We see then that the system may now be able to make a right diagnosis of the learner's response by capturing the semantic meaning of the sentence, thus returning error contingent feedback (Chen & Tokuda 2003; Heift & Nicholson 2001; Tokuda & Chen 2001).

As many semantically equivalent sentences might share the same structure, perhaps differing only in one or several segments of the sentence, we can enumerate all possible combinations of segments listing all the equivalent sentences. On the other hand, if the input sentences are not 100% syntactically correct, we would have much more trouble in listing all the possible semantically equivalent sentences.

All this suggests to use the FSA (finite state automata) type templates to represent semantically equivalent sentences. By doing so, we are able to reduce the storage space, developing time, as well as the matching time complexity by exploiting a computationally efficient HCS matching algorithm (Aho & Ullman 1992). The template automaton architecture based on FSA and the corresponding robust matching algorithm have been discussed as applied to language tutoring system (Chen & Tokuda 2003; Chen, Tokuda, & Xiao 2002; Tokuda 2002; Tokuda & Chen 2001).

Having a distinct advantage of their pedagogic expertise in language teaching, the language teacher can best construct the templates with the help of monitoring students' sample responses. The task of building a template corpus comprising well-formed model expressions and erroneous ill-formed sentences is quite labour-intensive, however, taking up considerable time. The quality of the monitoring students' sample responses also affect the template authoring task because human errors are often erratic and are not easily amenable to taxonomy for clustering.

This paper will introduce the template-template architecture whereby the extracting rules embedded in certain restricted nodes of the transitional diagram allows a variety of templates to be integrated into a single template-template or equivalently allows the single template-template to be expanded into a variety of patterns of templates. Furthermore, to deal with cumbersome error taxonomy, we have introduced the concept of buggy rules which is capable of automatically generating erroneous nodes with the help of extracting rules of the template-template. In this way, the language teacher need not be concerned with details of error taxonomy when he/she constructs the template-template. Once completed by a human teacher in terms of buggy rules, the template-template is endowed with enriched information, providing many typical errors automatically.

II. Concept of Template-Template and The Buggy Rules

1. Template-Template Structure

We will first define the term "template-template" below. The template-template is defined as a special template where some of the nodes are marked with rule-associated symbols, which can be expanded into many templates or, into a large template if we regard non-connected templates representing the translation of one sentence as a single template.

Typically, a rule is related to a set of symbols, say $\{s_1, s_2, \cdots, s_n\}$, each of the symbols is assigned to one or many nodes in the templates. Several values may be assigned to these symbols which will represent the style of the nodes of the transitional diagram. The so-called label symbols related to a single rule are called related symbols. Related symbols should have certain restriction; when $s_i = 1$, s_k might have to be 2. When the value of one symbol s_i depends on the values assigned to a set of other symbols, the choice of the value of this symbol s_i is called a required choice of the other symbols.

Depending on the application, we may develop many rules. For the language translation system, we could exploit rules similar to the following one.

Type A Rule: AP**(appear)-**NAP**(not appear) Rule**

In the template-template representation, suppose some nodes are marked with AP_i and some other nodes being marked with NAP_i (i being any integer, representing different Type A Rules); The rule imposes the condition that the template-template can be expanded to new templates including either the nodes marked with AP_i or the nodes marked with NAP_i separately, but not both of them simultaneously. If we assign $AP_i = 0$ thus implying that the nodes marked with AP_i do not appear in a template, we must assign $NAP_i = 1$ implying that the nodes marked with NAPi will appear in the template. Thus, we can say that $NAP_i = 1$ is the required choice of $AP_i = 0$. For the same reason, when $NAP_i = 0$, we must have $AP_i = 1$; and we can say that $NAP_i = 0$ is the required choice of $AP_i = 1$.

2. The Buggy Rules for Expanding Template-Template

A buggy rule here is defined as a generating scheme of common erroneous expressions which are characterized as deviations from a correct or well-formed expression. Consider the following form of a buggy rule:

$H_1 H_2 \cdots H_N \rightarrow R_1 R_2 \cdots R_M$

where $H_1 H_2 \cdots H_N$ represents a set of nodes tracking the correct path of any template-templates, or a set of grammatical part-of-speech tags representing basic components or segments of a correct expression. $R_1 R_2 \cdots R_M$ is the set of nodes which represents a typical erroneous expression whose correct form is $H_1 H_2 \cdots H_N$. We see immediately that errors are identified as deviations from the correct paths of the template-template.

Here is an example:

$EXVBP \rightarrow EXVBZ$

(Here EX-Existential, PRP-Personal pronoun, VBP–Verb for 1st and 2nd person present, VBZ-Verb, 3rd person singular present)

3. HCS (heaviest common sequence) Matching Algorithm

As we have discussed in some details in (Chen & Tokuda 2003), we are able to show that the DP (dynamic programming)-based HCS algorithm can now be applied directly to the template-template without expanding the template-template into the template form. This is so because the algorithm is now able to select a closest path to the student's input from among the many possible paths of the template-template databases. The algorithm now acts as an efficient diagnostic engine of the tutorial systems. Because of the simple template-template architecture and a DP-based high speed HCS-based computational algorithm as a diagnostic engine, the new system contributes to marked reduction in space and time in computational complexities in both authoring tasks and system performance.

III. Example of Template-Template

We demonstrate below our template-template architecture. We first construct the template-template for an English translation of a Japanese sentence meaning "Japan is dotted with beautiful gardens nationwide". The numbers in the picture are the weights of each word emphasizing its relative importance in the sentence. The weights of the words in the template are set to 1 by default, and they must be assigned in accordance with the importance of the words as judged by experts in the field. The symbols within "[" and "]" are the part-of-speech tags. The grey nodes of the transitional diagram denote starting nodes.

Now, simply applying the buggy rules listed in subsection II.2, we will be able to expand it into the template-template of figure 2. This shows that, as a language teacher, he/she could ignore some very common errors when he/she is constructing the templates, since buggy rules will then be able to expand the template to include these erroneous expressions.

By applying the Type A rule listed in subsection II.1, it is easy to see that, we are able to extract two templates from the template-template of figure 2: One template is obtained by simply allowing the nodes marked with AP_1 appear and accordingly deleting the nodes marked by NAP_1; another template is obtained by deleting the nodes marked by AP_1 of figure 2 and accordingly let the nodes marked with NAP_1 appear.

We now see that a language teacher is able to construct the template-template integrating a large combination of templates by simply using some label symbols.

IV. Concluding Remarks

The template-template architecture we have developed in this paper plays an important role in many text-based and hopefully in speech-based dialogue systems where a free format input in the form of text or spoken sentences are analysed by NLP (natural language processing) technology syntactically as well as semantically. This is done firstly by syntactical parsers and then by matching the respective response

to the semantically equivalent paths of template databases prepared. A distinct advantage of the new dialogue system is a reduction of computational complexities in both authoring tasks and computational processing of the system which owes much to improved space and time complexities of the new system. We hope the new dialogue system opens a way to an enhanced human-computer interface facilitating more natural human computer interactions.

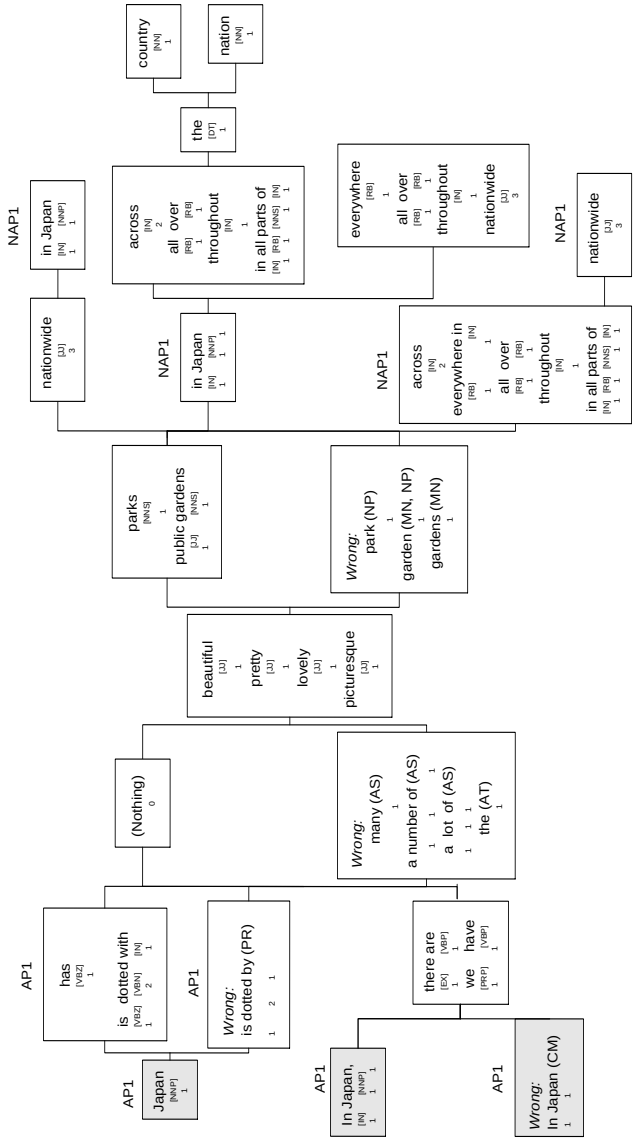

Figure 1: Original Template-Template

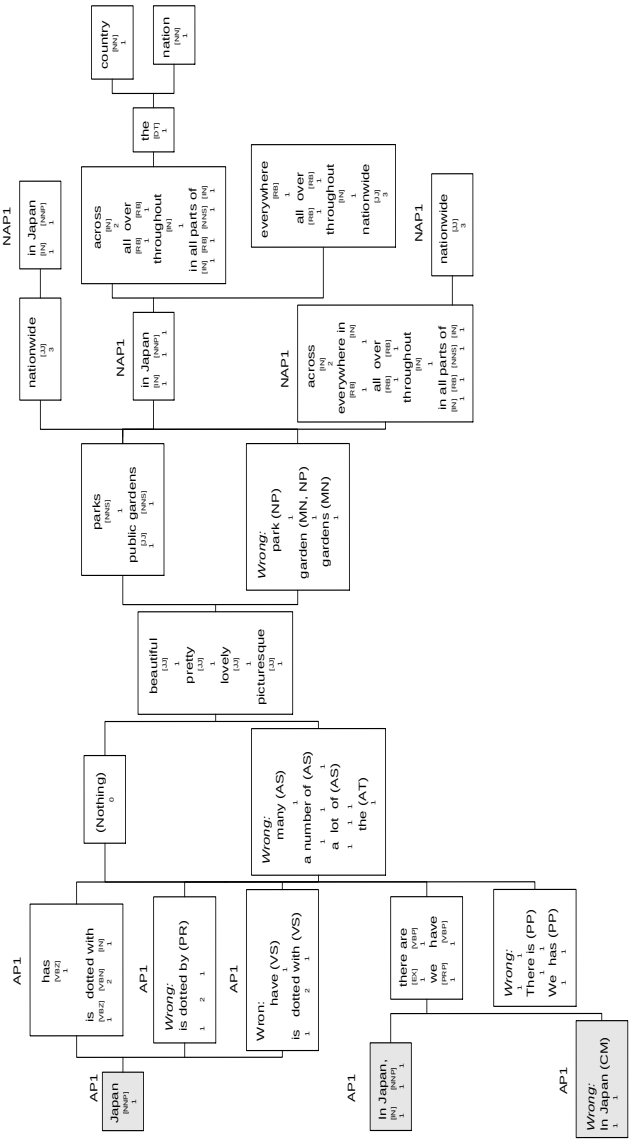

Figure 2: Expanded Template-Template

References

Aho, A. V., and Ullman, J. D. 1992. *Foundations of Computer Science*. New York: Computer Science Press.

Chen, L., and Tokuda, N. 2003. Bug diagnosis by string matching: Application to ILTS for translation. *J. CALICO*. In Press.

Chen, L.; Tokuda, N.; and Xiao, D. 2002. A post parser-based learner model for template-based ICALL for Japanese-English writing skills. *Journal of CALL* 15(4):357–372.

Heift, T., and Nicholson, D. 2001. Web delivery of adaptive and interactive language tutoring. *International Journal of Artificial Intelligence in Education* 14:310–325.

Tokuda, N., and Chen, L. 2001. An online tutoring system for language translation. *IEEE Multimedia* 8(3):46–55.

Tokuda, N. 2002. Guest editor's introduaction: New developments in intelligent CALL systems in a rapidly internationalized information age. *Journal of CALL, Special Issue on Recent Development in ICALL* 15(4):319–327.