A New Diagnostic System for J-E Translation ILTS By Global Matching Algorithm and POST Parser

Liang Chen and Naoyuki Tokuda
Computer Science Department, Utsunomiya University, Utsunomiya, Japan 321-8505
lchen@alfin.mine.utsunomiya-u.ac.jp, tokuda@cc.utsunomiya-u.ac.jp

Abstract

A new diagnostic system has been developed for an interactive template-structured intelligent language tutoring system (ILTS) for Japanese-English translation where an efficient heaviest common sequence (HCS) matching algorithm and a ‘part-of-speech tagged (POST) parser’ play a key role. This is implemented by exploiting the system template which consists of a complex transition networks comprising both model (correct) translations and many typical erroneous translations characteristic of nonnative beginners all collected and extracted from translations of about 200 monitors.

By selecting, from among many candidates’ paths in the system template, a path having a HCS with the student’s input translation as a best matched sentence, the template structure of the diagnostic system allows the potentially complicated bug finding processes in natural language to be implemented by a much simpler and efficient HCS string matching algorithm [20]. To improve the precision of a parser, we have developed a ‘probabilistic POST parser’ where we have eliminated ambiguity in part-of-speeches by manually pre-assigning POS tags to all words in potentially correct paths of the template. Combining the template-based diagnostic system and the parser, we found that the ILTS is capable of providing most adequate diagnostic messages and a tutoring strategy with appropriate comments after analyzing the keyed-in translated sentences from students.

1 Introduction

An Intelligent Language Tutoring System is an outgrowth from extensive research results addressing the use of computers in language processing where extensive fruitful research results from various fields of natural language processing applications have converged. While it draws heavily upon the findings of computational linguistics and machine translation [11], we think that the ILTS is currently one of the most challenging subject of all applications in machine translation and natural language processing. While most of NLP and MT systems depend on semantic and syntactic analysis requiring processing well-formed sentences, the ILTS’ differ fundamentally from other NLP and MT applications in how and why they handle ill-formed inputs. This is essential in the ILTS applications because unlike most of NLP or MT systems, the ILTS need to provide error-contingent feedback to erroneous input sentences from students for implementing effective tutoring. We see that the additional capability required of processing ill-formed sentences in ILTS is a basic requirement for an ITLS. Obviously it is extremely difficult to develop a syntactical parser to process them.

Many ILTSs are implemented in declarative representation formalism, including notably government binding [8, 9], logical grammars [6], definite clause grammar [15], and head-driven phrase structure grammar [10] among others. Several ITLSs resort to Augmented Transition Networks [24, 22, 23, 5]. To parse ill-formed sentences, we use meta-rules by relaxing the constraints of strict grammars or buggy rules which are provided for processing every ill-formed constructions, They may also alter an unification algorithm that always performs a parse and keeps track of conflicting features whenever they find.

Just like nonlinear problems often encountered in many engineering and science disciples, researchers in natural language processing and machine translation areas are all grappling to find methods of resolving problems rooted in the context sensitiveness of natural language requiring an efficient disambiguation algorithm for POS-tagging [1, 2, 17], word sense disambiguation algorithm of [19, 25] and PP attachment disambiguation of [18]. In fact, understanding very little of human’s processes involved in language understanding and acquisition, the machine processing of natural languages and machine translation including intelligent language tutoring systems still poses a big challenge if intelligent processing at least at the level of human intelligence is desired [1, 2, 17] unless the application domains are restricted to some special domains.

We believe that a useful natural language processing system or an ITLS may still be developed by restricting its use to a specific domain under more restricted constraints. As a next best target to general purpose natural language processing and machine translation applications, we have chosen an intelligent language tutoring system to facilitate us to acquire the Japanese-English translation ability in a technical field with an assistance from the machine in this paper.

The present ILTS comprises the template-structured diagnostic system, a part-of-speech tagged parser, a spellchecker and a VTAT (visual template authoring tool) for automating the construction of the system templates. The spellchecker and VTAT are discussed elsewhere[21] and will not be discussed here. The FSA-structured system template forms the core of the system and is designed by an experienced native speaker by extracting a vast amount of important information collected from details of the translation processes analizing the responses of about 200 normative monitors including not only correct model translations but also all potential semantic, syntactic, or structural errors expected of nonnative speakers[21]. Measuring the similarity between paths embedded in the system template and the students' translated input sentence by the summed weights of a common sequence, we diagnose possible bugs of translation by identifying a valid path(s) having the heaviest common sequence in the template. We will show how the template structure introduced allows the potentially complicated bug finding processes in natural language to be implemented by a well established and quite efficient string matching algorithm of HCS. The simplicity of the present method should be compared with a rule-based bug finding algorithm of [13], for example.

A parser is needed for identifying syntactic errors. In spite of its importance in natural language processing, an accurate parser is hard to obtain due to the inherent context sensitive grammar of any natural language. A corpus-based approach has attracted keen attention recently in this regard because a huge statistical data can train the broader coverage grammar reflecting the context sensitive grammar . An attempt has been made in [16] but the reported accuracy of such a parser of around 72% is too low for practical applications. Unlike most of the parsers that have been used in NLP or ILTS so far, we have decided to devise a far simpler parser called 'POST parser', where using the template structure, we pre-assign manually all the POS tags to all words of selected sentences in the template as in Penn TreeBank of [4]. Eliminating ambiguity of POS does improve the accuracy of grammar analysis considerably. We also show that a compound word dictionary of phrases also makes a contribution to parser accuracy.

Throughout the ILTS, we emphasize the importance of repeated learning of basic key English patterns. This conforms certainly with the well established traditional Friead's Michigan Method of language acquisition but most importantly from our point of view, this helps us to deal with simplified template network structures, avoiding and controlling unnecessary combinatorial explosions.

The flow chart of the main procedure of our language translation tutoring system can be shown as figure 1.

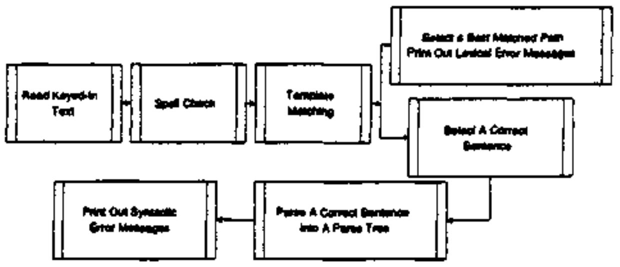

Read Keyed-in Text
Spell Check
Template Matching
Select a Best Matched Path
Print Out Lexical Error Messages
Select A Correct Sentence
Parse A Correct Sentence into A Parse Tree
Print Out Syntactic Error Messages

Figure 1: The Flow Diagram of ILTS

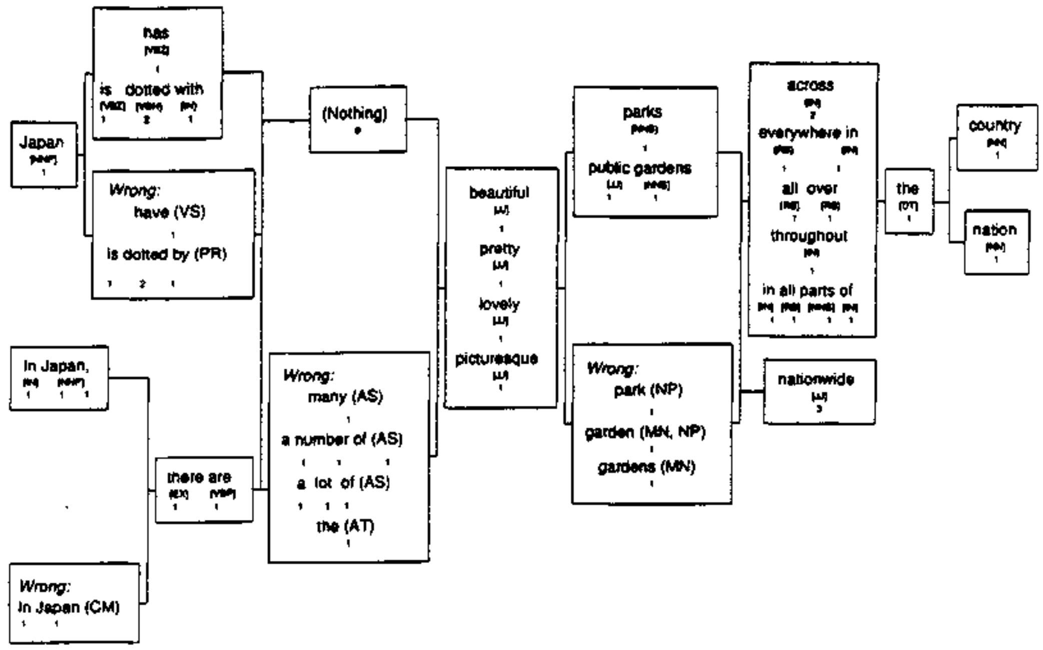
The purpose of the present paper is to introduce the concept of HCS matching algorithm and the POST parser both of which form the core of the diagnostic system of the present ILTS. The definition of the templates and their typical examples are given in section 2, while section 3 discusses the concept of HCS and ILTS matching algorithm we have developed. Section 4 will explain the POST parser. Concluding remarks and discussions will be given in section 5.

2 Template Structure

A template here refers to part of transition networks [12] with at least one starting node and one final node respectively, 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, with each word being assigned with a non-negative real number as its weight which is used to represent the relative importance of the word within the sentence. Here a larger weight emphasizes the importance of the words within the sentences. We pre-assign POS tags to the words of "syntactically valid" sentences but not to those of ill-formed sentences because we can trace syntactic errors by identifying differences between a parsed sentence and ill-formed sentence.

We may still have to allow several independent, disconnected templates for one sentence. We may assume that each sentence has only one template, however, since the template need not be fully connected (see explanations of figure 2 for example).

Figure 2 shows a typical example of the template for English translations of a Japanese sentence mean-



Error Messages:

AS: an assumption has been made on the quantity of noun
AT: the article is not needed
CM: a comma is needed
CT: contraction is incorrect
MN: meaning is incorrect
NP: noun must be plural
PP: phrase must be plural
PR: preposition is incorrect
VS: verb must be singular, since subject is singular

Typical Part-of-Speech Tags:

DT: Determiner
EX: Existential *there*
IN: Preposition/Subord. conjunction
JJ: Adjective
NN: Noun, singular or mass
NNS: Noun, plural
NNP: Proper noun, singular
RB: Adverb
VBN: Verb, past particle
VBP: Verb, non-3rd ps. sing, present
VBZ: Verb, 3rd ps. sing. present

Notes:
The numbers under each of the words denote weights assigned to the word representing its relative importance
The node with "(Nothing)" on, is an empty node meaning that no word is needed

Figure 2: A typical template for an English translation of Japanese sentence meaning, 'Japan is dotted with beautiful gardens nationwide'

ing "Japan is dotted with beautiful gardens nationwide" (figure 2). There are three templates forming different networks starting respectively with "Japan / In Japan ...", "There are ..." and "Beautiful parks ...". Although the latter two templates disconnected are not shown here, they can be regarded as part of the template so that one template is adequate for one sentence. Information in this template is all extracted from among 200 Japanese students' wide-varying responses.

3 Heaviest Common Sequence (HCS) and HCS Matching Algorithm

The HCS matching algorithm plays a key role in our ILTS.

3.1 Heaviest Common Sequence (HCS)

A template can easily converted into ILTS dual diagram form of an acyclic weighted finite digraph where each of the nodes of the template are converted into one or several arcs in the graph with arcs labeled e with 0 weight being added for each empty node. The acyclic weighted finite digraph has the properties; each arc is labeled with a word or a null symbol with a non-negative real number W being assigned , and at least one of the vertexes is defined to be a starting vertex, and an accepting vertex respectively. Without loss of generality, we regard each of labels as a string of characters comprising a word(s) or phrase(s).

The corresponding acyclic weighted digraph of the part of template of figure 2 is shown in figure 3:

An accepted sequence, or a sentence, of a template (or the corresponding digraph) is the sequence of the arcs that form a path of the digraph starting from a starting vertex and ending at an accepting vertex. The weight of the accepted sequence is the sum of the weights of the corresponding arcs in the digraph.

The common sequence of a template and a sentence is defined as a sequence of characters which is a common sequence of the sentence and an accepted sentence of the template (digraphs). The weight of a common sequence is defined as the sum of the weights of the corresponding arcs in the template.

The HCS matching problem is defined as follows: Given an acyclic weighted digraph and a string sequence, it is required to find a common sequence as well as the corresponding sentence of the digraph, such that the common sequence has a heaviest weight.

The common sequence defines the similarity between the paths in the template and the keyed-in sentence. In another word, identification of an HCS path(s) in the template is equivalent to identifying or finding bugs of the input translation. The potentially complicated bug finding processes in natural language are now implemented by a well established and quite efficient string matching algorithm of a HCS algorithm, which is similar to a longest common sequence algorithm of ([7]). A so-called Michigan Method of acquiring key English sentence patterns of the present ILTS and the efficient HCS matching algorithm makes the present method a most attractive one from the point of view of online processing.

What are we going to do if a multiple number of paths share a same heaviest weight? The present HCS matching algorithm chooses the one having a minimum total weight of the entire corresponding sentence (path). This criteria gives more weight to the sentence having a higher relative weight to the entire weight of the target sentence. This is important in the tutoring in practice because error-mingled sentences tend to have a longer common sequence. For example, when a student keyed in the text "Japan has lovely parks across the country" (see figure 2), there are several paths that match the keyed in translation having the same heaviest common sequence; two of them are "Japan has lovely parks across the country" and "Japan has many lovely parks across the country". We choose the former sentence of weight 8 rather than the latter of weight 9 which has unnecessary 'many' added in the English translation. Now we elucidate the HCS matching algorithm.

3.2 HCS Matching Algorithm

We want to find a heaviest common sequence between the diagraph A and a sentence B. The sentence can also be considered as a diagraph. To simplify the notation, we denote the vertex set of A (B) as $V(A)$ $(V(B))$, the arc set of A (B) as $E(A)$ $(E(B))$, the character label of an arc $\overline{uv}$ by $c_{\overline{uv}}$, and the weight of an arc $\overline{uv}$ in digraph A by w_{uv}. For any vertices N_1 in A and N_2 in B, we define the heaviest common sequence of the sentences ending at N_1 and N_2 as one having a heaviest weight among the common sequences of pairs of sentences beginning at the starting vertex of A and B, ending at the vertex N_1 and N_2 respectively. We denote the weight of the heaviest common sequence of the sentences ending at N_1 and N_2 as $w(N_1,N_2)$.

Our algorithm for computing the heaviest common sequence is based on the following property:

Property 1

$$
\begin{aligned}
w(N_1,N_2) = \\
&\max\{\{w(a,N_2)|\overline{aN_1} \in E(A)\} \cup \{w(N_1,b)\} \\
&\cup\{w(a,b) + w_{\overline{aN_1}}|\overline{aN_1} \in E(A), c_{\overline{aN_1}} = c_{\overline{bN_2}}\}\},
\end{aligned}
$$

where $\overline{bN_2} \in E(B)$.

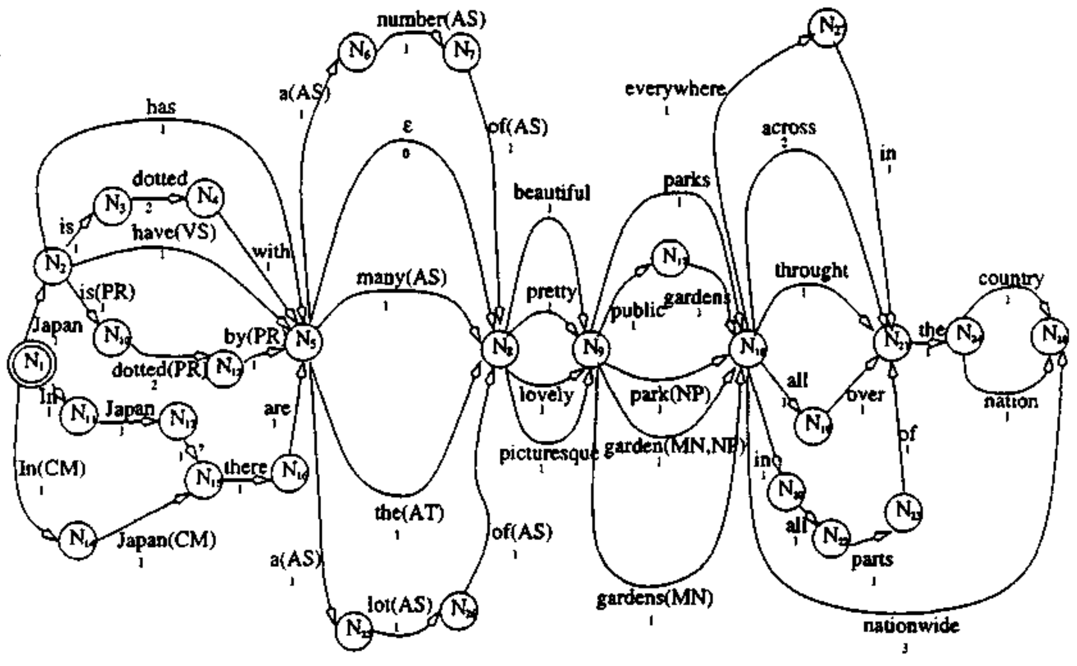
Once any digraph is topologically sorted, it is straight forward to design a dynamic programming method to find a Heaviest Common Sequence, with a computational cost of $O(|E(A)| \bullet |E(B)|)$. The algorithm could be found in [20]. A more detailed description of the matching algorithm can be found in [20].

A topological sort of an acyclic digraph G implies a linear ordering of all ILTS vertices such that for any edge $\overline{ab} \in E(G)$, a always appears before b in the ordering. A topological sorting algorithm can be found in [7] by a depth first search algorithm with a computational complexity of $O(|E(G)|)$.

Example: We demonstrate the advantage of global matching algorithm for the Japanese sentence "Japan is dotted with beautiful parks nationwide" by an example. Consider an input; *In Japan, is dotted with lovely park throughout nationwide.* A best matched path identified by the present HCS algorithm in the template is *Japan is dotted with lovely park nationwide* where we choose the weight of each word to be taken as unity throughout in this example. A greedy method on the other hand makes selection on local information, apparently choosing the path *In Japan, there are lovely park throughout the country* which is not obviously a best selection.

4 A Part-of-Speech-Tagged Parser

An efficient parser is basic to any natural language processing system [1, 2] including machine translation between languages. In our tutoring system, the parser



Notes:
The numbers denotes the weights
N_1 is the starting vertex, and N_{28} is the accepting vertex. Symbols within parentheses denote error messages which are stored in the database

Figure 3: An Example of Acyclic Weighted Digraph

is basic to diagnosing grammatical or syntactical errors of students' inputs needed for providing useful tutoring comments. Yet even just for well-formed sentences, it is difficult to construct a parser for processing a general class of sentences with a practical precision of, say, 90% or better. Since there is no such accurate parser announced even just for correct sentences in a natural language, we have enough reasons to believe that the meta- rule, buggy rule or unification algorithm based parser system could not have better performance in treating ill-formed sentences. The difficulty comes largely from the context sensitive grammar inherently associated with any natural language which in turn leads to well known ambiguities of natural language such as POS tags[2], semantic ambiguity [19, 25] and structural ambiguities[18].

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 Tree Bank corpus [3] to build up the statistical data. If tagging is done with accuracy over extensive domains of sentences of sufficiently large corpus size, we expect the probabilistic information accumulated to reflect the broad-coverage grammars including context sensitive grammars in particular. And the studies also show the benefit of using probabilistic information in parsing, and the large corpus allows us to train the probabilities of a grammar. An Apple Pie Parser [16] is a typical probabilistic parser of the kind developed at New York University based on the PennTree Bank's syntactically-bracketed corpus. But the reported precision of 72.61% is too low to use in a practical application.

Our aim here is to build a parser best suited for our purpose with the precision of 90% or better when grammatically correct sentences are parsed. The parser developed here is used to parse syntactically correct sentences which are embedded in templates so that all we want to do here is for us to check students' syntactically incorrect sentences against the correctly parsed sentences pointing out possible errors and for providing appropriate coaching strategies. Manual labor involved in POS tagging of paths in templates is reasonable with respect to other human interactions needed. Even if we eliminate ambiguity of POS tags, we have to deal with versatile English grammars to construct correct parse trees but this is certainly far easier.

Following Apple Pie Parser, we write the grammars using the two non-terminals 5 and *NP.* All units starting with 5 and *NP* are called structures which 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 Pen TreeBank:
(S (PP (BR) (PP (IN) (NP (NP (NNP)) (POS) (JJ)

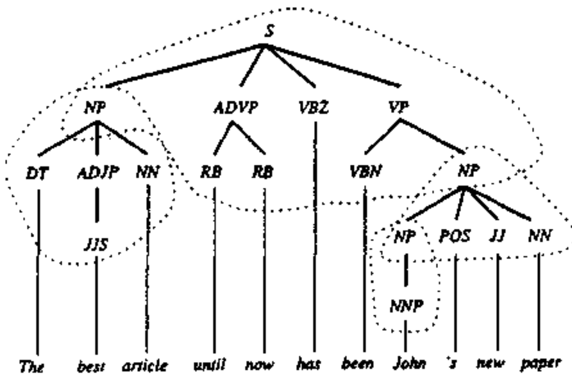


Figure 4: A Parse Tree for The Sentence *The best ...*

(NN)))) (, ,) (S (NP (DT) (ADJP (JJS)) (NN)) (ADVP (RB) (RB)) (VBZ) (VP (VBN) (NP (DT) (NNP) (NNP) (NN)))))

As we see in the following, the sentence is split into 6 "small" structures each of which starts with S or *NP:*

1. *(S(PP(BR)(PP(IN) NP))(,)S)*
2. *(NP NP (POS) (JJ) (NN))*
3. *(NP (NNP))*
4. *(S NP (ADVP(RB)(RB)) (VBZ) (VP (VBN) NP))*
5. *(NP (DT) (ADJP (JJS)) (NN))*
6. *(NP (DT) (NNP) (NNP) (NN))*

It is possible to split each of the syntactically-bracketed structures into several smaller structures until all comprise the constituents or leaves of the tree. In the language tutoring environment where all the POS tags are known, our main 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 sentence having 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)),* a parse tree of figure 4 can be obtained.

Given a sentence having all part-of-speech tags manually specified, we expect most probably that many different subsets of structures can be selected for constructing parse trees for the sentence. To deal with the situation, a so-called probabilistic chart parser is called in. In our language translation tutoring system, we compute the probabilities of grammars by the following formula giving scores for the tree structures chosen:

$$P_{tree} = \prod_{Struc_i \in tree} P_{Struc_i}$$

where

$$P_{Struc_i} = \frac{F_i}{T_i},$$

F_i denotes the frequency of the structure $Struc_i$ in the corpus, T_i the Total frequency of the structures starting with the same symbol of $Struc_i$.

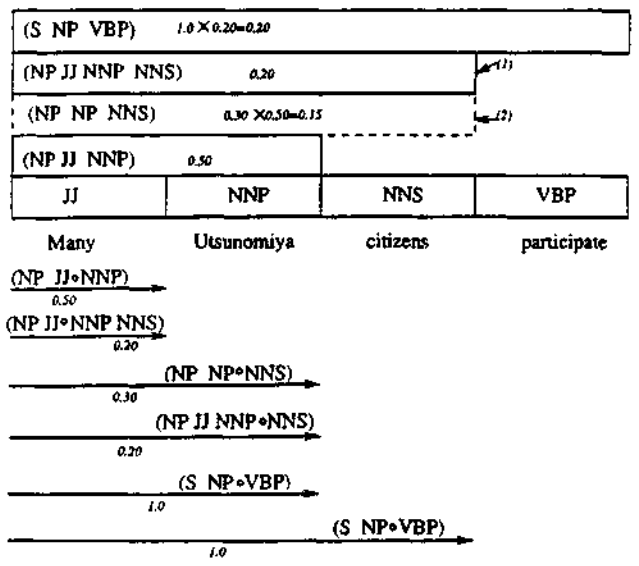
When there are several possible parse trees for one sentence, we choose the one with a highest score. To construct the probabilistic parser, a standard bottom-up chart parsing algorithm of [2, pp.53-60] can best be used with the POS tags fixed. In the approach we add a step that computes a score of each entry to the chart so that we may select a best constituent from many candidates having the same type of the input strings. Consider parsing a sentence *"Many Utsunomiya citizens participate"* which has POS tags of "*JJ, NNP, NNS, VBP*" (Figure 5(1)).

Here we allow the grammar to have the following four structures: Suppose that according to the corpus statistics, *(NP JJ NNP)* has a score of 0.50, *(NP NP NNS)* a score of 0.30, *(NP JJ NNP NNS)* a score of 0.20, and *(S NP VBP)* a score of 1.0. It follows readily that among (1) and (2), the constituent of the structure of (1) is chosen because of ILTS higher score of 0.20. Parse tree of figure 5(2) is easily generated.

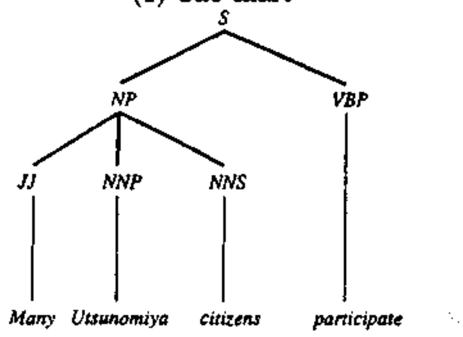
We will next demonstrate the so-called POST parser we have developed showing its obvious advantage over the other statistical parsers like APPLE-Pie. Two examples are given in figure 6 and figure 7. Note that for POST parser, each of words must be assigned with the POS before we “parse” the sentence.

To further improve the accuracy of the parser, we plan to develop a compound word dictionary of phrases. For example, 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 part-of-speech tagged sentence *"There/EX are/VBP a/DT lot/NN of/I pens/NNS on/IN the/DT table/NN"* will be parsed into the tree: (S (NP (EX There))(VP (VBP are)(NP (NP (JJ (DT a) (NN lot) (In of)) (NNS pens))(PP (IN on) (NP (DT the) (NN table)))))) (figure 8). Note that, without compound word phrase dictionary the result will 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)))))))).

When a keyed-in sentence is matched to a correct sentence in the template, a correct parse tree will be most probably generated for the correct sentence by the probabilistic POST parser. Then by collating the resulting parse tree and the keyed-in sentence, it should be easy to provide appropriate grammatical comments to students. It is expected to be only effective when the keyed-in sentence *DOES NOT* have



(1) The chart



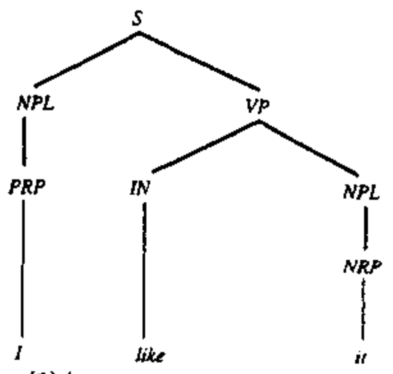
(2) The corresponding parse tree

Notes:
Constituents (1) and (2) cover same inputs and the structure having the constituent of (1) is selected because it has a higher score of 0.20 > 0.15

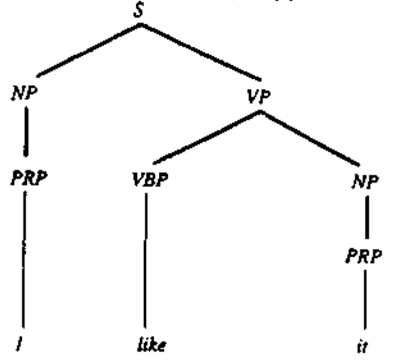
Figure 5: A Modified Bottom-up Chart Parsing Algorithm

too many errors. Figure 9 gives an example where an input sentence *There aren't disadvantage to the model* is matched to a correct sentence *there is no disadvantage to this model.* When the parse tree of the correct sentence is obtained, the system will return the following grammatical comments. One example is given below.

There [aren't 1] [2] disadvantage to [the models 3].
1. aren't—should be *is*, which is the 3rd ps. sing. present in agreement with feature unification of the following Noun phrase no *disadvantage.*
2. *"no"* is needed; together with the Noun *disadvantage,* this forms the Noun phrase of a Verb phrase is above.
3. *"the models"* should be replaced by a noun phrase *this model,* where *this* is a Determiner and *"model"* is a singular noun.



(1)A parse tree by Apple Pie



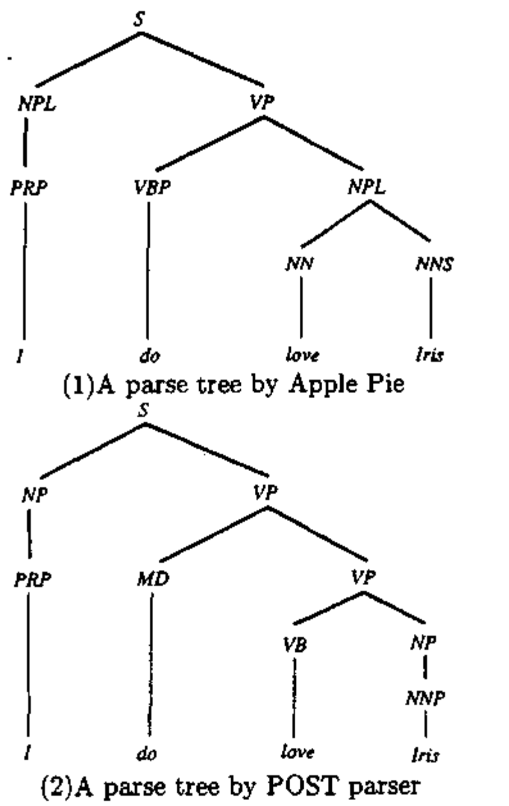
(2)A Parse Tree by POST parser

Meaning of the Grammatical Symbols:
NP: Noun Phrase VP: Verb phrase
NPL-Lowest level of Noun Phrase. This symbol is used only by Apple Pie. We prefer to use NN in our parse tree.

Figure 6: A Parse Tree of Sentence "I like it"

5 Concluding Remarks and Discussions

This paper presents an efficient template-structured intelligent tutoring system for language translation where the HCS matching algorithm and a POST parser play a key role in diagnosis of students' translations. We have shown that by introducing the template structure into the language tutoring system, an apparently difficult bug diagnosis can be implemented by a remarkably efficient and simple string matching procedure. The system can always find a best matched path of the template within reasonable short time presenting semantic or structural error messages as the result. The part-of-speech tagged parser on the other hands can be used to present syntactic error messages as well. A bi-language and preferably multi-language translation tutoring system is the most desirable but the prospect for such a perfect system endowed with human intelligence is still far and dim. We have developed a language tutoring system of this paper as a next best target where we have made a best use of a classical string matching algorithm and a POST parser. In the present system, we have made full use



Meaning of the Grammatical Symbols:
MD: Auxiliary verbs including *do, did, does, can, could, dare, may, might, must, ought, shall, should, will, would.* Note that MD here is defined to include a wider class of auxiliary verbs than PennTree Bank does such as *do, did, does.*

Figure 7: Parse Trees of Sentence "I do love Iris"

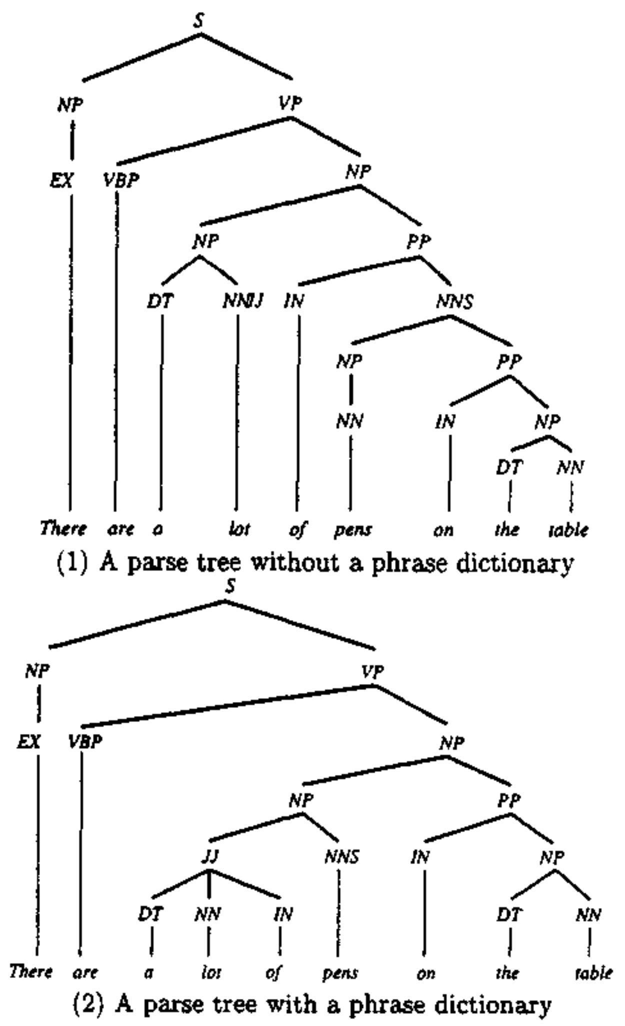

S
NP VP
EX VBP NP
NP PP
DT NNIJ IN NNS
NP PP
NN IN NP
DT NN
There are a lot of pens on the table
(1) A parse tree without a phrase dictionary
S
NP VP
EX VBP NP
NP PP
JJ NNS IN NP
DT NN IN DT NN
There are a lot of pens on the table
(2) A parse tree with a phrase dictionary

Figure 8: Parse trees with and without a compound word dictionary of phrases

of human interactions firstly in building up template databases and then parts-of speech tagging of some model sentences. We provide all helpful comments to students because it is the human who acquires the final language capability. It seems that the template and template matching algorithm are also useful in other applications. In the areas of information retrieval and language tutoring, the template may comprise the type of the information which we would like to retrieve or to be retrieved. The HCS algorithm can best be used giving the relative weights with which particular symbols or words are to be assigned. We believe that the heaviest common sequence matching algorithm may be used to improve the performance of a search engine in the information retrieval systems. Also we think an evaluation system of an individual student can best be achieved by the use of the system.

The work reported here is undertaken as a joint research with SunFlair Company of Tokyo. We thank the chairman of the company, Mr.Y.Sasai, for encouragement and financial support.

References

[1] Aho, A.V. and Ullman, J. D.(1992), "Foundations of Computer Science", Computer Science Press.

[2] Allen, J.(1995), "Natural Language Understanding", The Benjamin/Cummings Publishing Company Inc., pp.53-60, pp. 83-123 and pp. 227-295.

[3] Bertino, E. and Martino, L. (1994), "Object-Oriented Database Systems Concepts and Architectures", Addison-Weseley, pp. 232-242.

[4] Bies, A.and Ferguson, M. (1990), "Bracketing Guidelines for Treebank Style Penn Treebank Project", Linguistic Data Consortium, University of Pennsylvania.

[5] Catt, M. and Hirst, G. (1990), "An Intelligent CALI System for Grammatical Error Diagnosis", Computer Assisted Language Learning, 3:3-27.

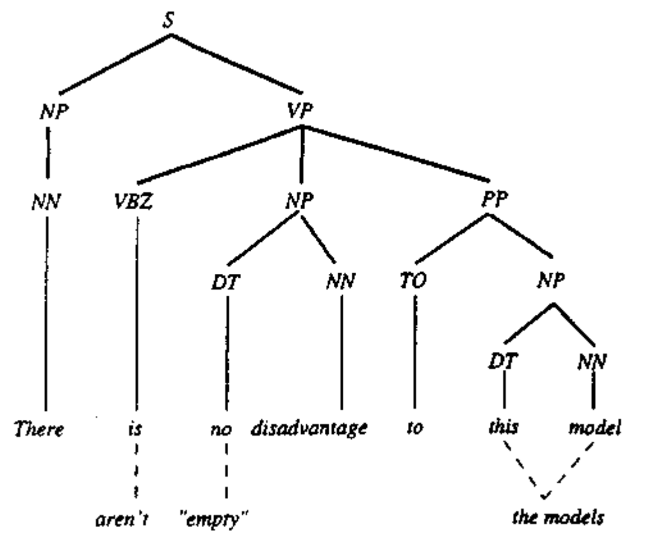


Figure 9: A parse tree of a correct sentence by POST parser

[6] Colmerauer, A. (1978), "Metamorphosis Grammers", Natural Language Communication with Computers, ed. Bolc, L., Springer Verlag.

[7] Cormen, T. H., Leiserson, C. E. and Rivest, R. L.(1990), "Introduction to Algorithms", MIT Press, Cambridge, Mass.

[8] Chomsky, N.(1981), "Lectures on Government and Binding", Dordrecht: Foris.

[9] Chomsky, N.(1986), "Barriers", MIT Press, Cambridge, MA

[10] Hagen, L. K. (1994), "Unification-Based Parsing Applications for Intelligent Foreign Language Tutoring Systems", CALICO Journal, 12(2):5-31.

[11] Heift, G. (1998), "Designed Intelligence: A Language Teacher Model", Doctor Thesis, Department of Linguistics, Simon Fraser University, July, 1998.

[12] Hopcroft, J. E. and Ullman, J. D. (1979), "Introduction to Automata Theory, Language, and Computation", Addison-Wesely Publishing Co.

[13] Huang, Z. and Tokuda, N. (1996), "A Syntactical Approach to Diagnosing Multiple Bugs in an Intelligent Tutoring System", IEEE Transactions on Systems, Man, and Cybernetics, Vol.26, No.2, pp. 280-285.

[14] Miller, D. A. (1990) "Introduction to WordNet: An On-line Lexical Database", International Journal of Lexicography, vol.3, pp. 1-86.

[15] Pereira, F. C. N. and Warren, D. H. D. (1980), "Definite Clause Grammars for Language Analysis - a Survey of the Formalism and A Comparison with Augmented Transition Networks", Artificial Intelligence, 13: 231-278.

[16] Sekine, S. and Grishman, R. (1996), "A Corpus-based Probabilistic Grammar with Only Two Non-terminals", 4-th International Workshop on Parsing Technology, pp. 216-223.

[17] Shieber, S. M .(1986), "An Introduction To Unification-based Approach To Grammar", CSLI Lecture Notes 4,: Chicago U. Press, Chicago

[18] Stetina, J. and Nagao, M. (1997), "Corpus-Based PP Attachment Ambiguity Resolution with a Semantic Dictionary", Proc. Fifth Workshop on Large Corpora of ACL, pp.66-79

[19] Tokuda, N., Gu, Z. and Ye, Y (1998), "An Unsupervised Word-Sense Disambiguation by Auto-clustering", To be submitted to IEICE Transactions on Information and Systems.

[20] Tokuda, N. and Chen, L. (1999) "A New Efficient Diagnostic System for Language Translation ILTS by Global Template Matching", 1999 International Conference on Artificial Intelligence (IC-AI'99), Las Vegas, USA.

[21] Tokuda, N., Chen, L., Huang, L. and Nagai, A. (1999) "An Interactive, Multimedia-Based Intelligent Tutoring System For Language Translation" , Accepted for presentation at IFIP Sixth International Conference on Distributed Multimedia Systems (DMS'99), University of Aizu, July 26-30, 1999

[22] Wang, Y. and Garigliano, R. (1992), "An Intelligent Language Tutoring System for Handling Errors Caused by Transfer", Intelligent Tutoring Systems, C. Frasson, G. Gauthier and G.I.McCalla eds. Lecture Notes in Computer Science, Springer Verlag:395-404.

[23] Wang, Y. and Garigliano, R. (1995), "Empirical Studies and Intelligent Language Tutoring", Instructional Science, 23: 47-64.

[24] Woods, W. (1970), "Cascaded ATN Grammars", American Journal of Computational Linguistics, 6(1):1-12.

[25] Yarowsky, D. (1995) "Unsupervised Word Sense Disambiguation Rivaling Supervised Methods" *Proceedings of ACL-95,* pp.189-196.