

# A table look-up parser in online ILTS applications

Liang Chen

Computer Science Department,  
University of Northern British Columbia,  
Prince George, BC, Canada V2N 4Z9  
Email: [lchen@ieee.org](mailto:lchen@ieee.org)  
Tel: (250)9605838  
Fax: (250)9605544  
URL <http://web.unbc.ca/~chenl>

Naoyuki Tokuda and Pingkui Hou  
SunFare Research & Development Center,  
Sunflare Company, Shinjuku Hirose BLDG,  
4-7 Yotsuya, Shinjuku-ku, Tokyo, Japan  
Email: [tokuda\\_n@sunflare.co.jp](mailto:tokuda_n@sunflare.co.jp)

## Abstract

A simple table look-up parser (TLUP) has been developed for parsing and consequently diagnosing syntactic errors in semi-free formatted learners' input sentences of an intelligent language tutoring system (ILTS). The TLUP finds a parse tree for a correct version of an input sentence, diagnoses syntactic errors of the learner by tracing and regarding the deviations of the input from the identified correct sentences as errors of the learner. The TLUP can now display the parse tree as well as diagnosed errors at leaves' level. This simple super-rule based TLUP turns out to be a powerful pedagogic tool coaching L2 (2nd language) learners on the grammar structures in L2 saving the computing time and running space requirements, improving the parsing accuracy and allowing the real time online running environments. The TLUP is effective as long as the input does not deviate too much from anticipated embedded patterns of model answers with committed errors remaining within the framework of anticipation for the level of the ITLS designed. We establish the validity of a TLUP by an  $(n, c)$  lot acceptance-sampling plan.

## 1. Introduction

This paper is motivated by a keen need for a compact but accurate parser in Azalea (Tokuda and Chen, 2001), an online ILTS (intelligent language tutoring system) developed for acquiring English writing skills, where L2 (2nd language) learners' responses (answers) to a sequence of translation exercises of L1 (1st language) sentences (questions) are diagnosed, corrected, and repaired in real time. In sharp contrast to the more restrictive multi-choice type or blank-filling type questions predominantly used in most of conventional tutoring systems, we

削除: |

allow the so-called “free format” input here. The “free format” does not mean a complete freedom in format, of course; it should be restricted to a pattern of format reasonably close to those used in the typical correct answers (model translations) expected at the level of ILTS systems intended. The good parser plays many important pedagogic roles. A good example for displaying the effectiveness of parsers can be found when we make full use of an error frequency table of Minimum Error Subtrees of grammar structures identifying the detailed structure of learners’ errors, whereby, we are often able to pinpoint learners’ inherent weak points peculiar to some groups of learners strongly depending on learners’ L1 language (Chen, Tokuda and Xiao, 2003). These inherent weak points include, for example, the misuse of articles common to Japanese or Chinese students whose language does not make any differentiation of singular or plural form of nouns and consequently any use of articles or definite articles in their native language. Naturally, the acquisition of the grammar structure of a new language plays a fundamental role in understanding the foreign (L2) language so that the importance of the parser is quite evident in the ILTS as long as users’ often error-ridden, free format input sentences can be parsed. A good parser is indispensable for diagnostic as well as for repairing strategies of L2 learning. The importance of a good parser is all the more evident because if we regard the tutoring process of translating a sentence in L1 to L2 in Azalea as a question (in L1) to an answer (in L2) process comprising a typical dialogue system, the need for a simple but accurate parser is quite high in all ILTS’s.

刪除；

刪除: displaying the effectiveness of parsers is that by

刪除: using

刪除，

刪除: definite

刪除:

刪除: Because the acquisition of the grammar structure of a new language plays a fundamental role in understanding the foreign (L2) language, the importance of the parser is quite evident in the ILTS where users’ free format input sentences must be parsed. Regarding

Developers of most of the state-of-the-art ILTS’s prepare a fixed number of exercises or questions in the question database, asking nonnative L2 learners to type in answers to these questions. Due to the flexible structural patterns of L2 sentences, the number of possible answers can easily explode. It is also a common practice to compel the learners to use specified key words in the answer (or translation) so that a huge number of combinations in possible answers to be processed can be reduced as much as possible. However, the situation is still tough because we have to deal with many unexpected L1-oriented errors. If we are able to implement a human teacher-like ILTS, we need to solve the following three critical problems:

1. Diagnosing ill-formed input sentences from L2 learners containing a variety of errors is a very challenging task. An expert human language teacher is able to track down the source of errors, because he/she has a firsthand understanding of the context and domain of the sentence, and is most likely to also have an extensive knowledge of the cultural and educational background of the L2 learners. This can be observed by noticing that an experienced language teacher is always able to provide a “best possible” correction under the given context-sensitive situation although ill formed sentences can be corrected in many plausible ways. Because of implicitness of the knowledge level acquired, the general buggy rule based grammar analyzer may not be applicable to ILTS’s.

2. Developing an accurate general-purpose parser is always an extremely difficult and challenging task for any context-sensitive natural language applications. Natural language is so complicated that even a basic task of assigning part-of-speech tags is beyond the state-of-art in NLP (natural language processing) technology, if a high precision is required.

3. Parsing sentences requires a large search space. We expect a general language parser to require not only an extremely large working space in computer memory but also a lengthy running time<sup>1</sup>. For an online ILTS application, this may cause a big problem because many users may log on to the system simultaneously.

What would be a best way to resolve such problems for the grammar analysis in ILTS's?

The template structure and the so-called POST (part-of-speech tagged) parser described in (Tokuda and Chen, 2001; Chen and Tokuda, 2003; Chen, Tokuda and Xiao, 2002) have solved partially the problem posed in item 1. The template is used to store a finite set of correct answers (called as model translations in Azalea (Tokuda and Chen, 2001)) with part-of-speech tags pre-assigned to certain ambiguous words or phrases, as well as possible erroneous answers (erroneous translations in (Tokuda and Chen, 2001)), all of which are prepared or collected from the error database of many L2 learners by expert language teachers. Given a free format input sentence from students, the system first locates from among many possible answers a sentence (path) having the highest similarity to the keyed-in input sentence. The sentence (path) found may be an erroneous answer, but it is easy to locate a correct answer stored in the template closest to the input sentence by merely bypassing the error marked nodes along the path and tracing the validity-marked nodes. Regarding the differences between the two as errors committed by the students, we are able to diagnose and repair the errors identified in the input sentence. We have shown that as part-of-speech tags are pre-assigned to ambiguous words, a reliable parsing result for the syntactically correct sentence can be obtained by a POST parser, which is a modification of a general purpose PCFG (probabilistic context free grammar) parser by allowing certain words to be assigned with fixed tags. Matching the leaves of the grammar tree, which are words of the correct sentence, to the input sentence, we are able to provide grammatical or semantic error feedback.

The second problem remains unresolved, however, because there is no known parser providing a high precision comparable to a human language teacher, even if the correct part-of-speech tags for all the words in an entire sentence are provided beforehand. The third problem of large search space still remains, because a general-purpose parser of very large search space presents great

---

<sup>1</sup> For example, it is known that, to generate a grammar tree for a sentence with  $n$  terms, a general purpose probabilistic grammar parser requires  $O(n^3)$  time in the sense of complexity analysis.

difficulties for an online application to ILTS systems with many simultaneous users, and furthermore, a problem of high accuracy still remains.

The purpose of this paper is to provide a solution to the unsolved problems above in a surprisingly simple manner. While most applications resort to a general parser for grammar analysis, we address a naive-looking but a very basic question first -- is a fully-fledged statistical or a general purpose parser necessary for processing context-free or sensitive grammar of natural language in an ILTS where the number of the correct sentence patterns to be processed in the system remain large but finite?

We provide an answer "No" to this question by developing a simple table look-up parser (TLUP) for grammar analysis in the language tutoring system. We propose and provide a convincing statistical and physical argument to support the validity of the simplest possible TLUP for ILTS's where the number of sentences is restricted to a finite number.

削除: ILTS

In the rest of the paper, we will demonstrate that a simple TLUP meets all the requirements we set forth in ILTS applications. We will verify the validity of our assertion by an  $(n, c)$  lot acceptance-sampling plan (Ryan, 2000; Hoel, Port and Stone, 1971) ensuring the quality of the parsing performance for online purposes.

## 2. Basic Ideas and Analysis

### 2.1 Template structure

Before we go to detailed descriptions of our TLUP for the grammar analysis, we give a short introduction to templates, which are used to store many correct answers, as well as typical erroneous answers to questions posed. A template consists of separated segments of model sentences and / or learners' error-ridden sentences in L2 at the level of a word(s) or a phrase(s) so that the system can diagnose the sentences at the word or phrase level. A bilingual L2 expert teacher prepares the correct answers in conformity with the sentence patterns students have studied in the texts, collects possible errors that learners are most likely to make, and then integrates these correct answers as well as error data into templates. Matching the input with template paths, it is now easy to diagnose and correct various errors in input sentences.

As it was mentioned in Section 1, we usually reduce a huge number of possible correct answers by compelling the learners to use specified key words in an answer. However, the total number of correct answers is still very large, and we have to deal with many unexpected L1-oriented errors. Because each template path represents one answer, a template can include many possible combinations of words or phrases for constructing many varieties of sentences and thus makes it possible for a language teacher to construct a template that can extract all these answers but without listing all the answers by him/her-self. The reader is referred to articles (Chen and Tokuda, 2003; Tokuda and Chen, 2001) for further

discussions on template. We should emphasize again here that, although we use template to store the possible answers, our TLUP parser does not depend on the specific use of templates; the TLUP approach should work, as long as a system can store all the possible answers in one way or another.

A complete example of templates for syntactically equivalent sentences meaning “*Japan is dotted with beautiful parks nationwide.*” can be found in Chen and Tokuda (2003). Suppose a user keys in “Japan are dotted by beautiful park nationwide.” Given the template, the system returns the following error feedback:

Japan [are](1) dotted [by](2) beautiful [park](3) nationwide.

Comment(1): Error

- The subject of the sentence is singular. Therefore, the verb must be singular.

Change the verb to the singular.

- This part should be: is

Comment(2): Error

- This preposition is wrong.

- This part should be: with

Comment(3): Error

- This noun should be plural.

- This part should be: parks / gardens

## **2.2 The parsed trees having different structures**

As we discussed in the Introduction, we can avoid parsing erroneous or ill-formed sentences of nonnative learners directly by adopting the following strategy: First match and find a path of the template comprising a sentence having highest similarity<sup>2</sup> to the input sentence, then find the closest well-formed “model” sentence to the learner’s input sentence from among the well-formed model answers embedded in templates, and finally analyze the grammar tree of the well-formed matched sentence. Regarding any deviations of the input sentence from the matched model sentence as syntactic errors the learners have committed, it is easy to return relevant grammatical feedback to learners. Now how are we going to find the grammar trees of well-formed model sentences?

It is easy to understand that, if there are only a small number of correct answers (sentences), all we have to do is to look up the entries of a table, which could be pre-constructed comprising entries of all the correct sentences and their grammar structures. In practice, however, there are a large variety of correct answers to the same question, e.g., one L1 sentence can be translated into a variety of L2 sentences, so that the representation in the form of template automaton demands a huge number of different L2 sentences. This is so because the total number of possibly correct sentences is extremely large due to

---

<sup>2</sup> see (Tokuda and Chen, 2001) for the definition of similarity.

the flexibility of natural languages allowing many combinations of synonyms, or sentence patterns. Suppose we have a template for a correct response to a question with the total of 12 words and half of the words have 5 synonyms. This amounts to  $5^6=15625$  different sentences<sup>3</sup>. It would require a tremendously large table to include all the sentences and their grammar structure, if differences in word or phrase level are taken into account.

We can observe that, because a template for the answers in L2 in an ILTS, e.g., L2 translations of an L1 sentence, must necessarily include the sentences having the similar key frameworks of semantics and syntactic structure, we could expect the possible variations in grammar structure of the template paths embedded to be considerably restricted. By the same token, if we list sequences of the part-of-speech-tags for each possible sentence that critically depends on the selection of so-called production rules, we should not expect to have many variations in the sequence of part-of-speech tags for a template. Thus, we can expect a marked reduction of table size if we use only the sequences of part-of-speech tags of sentences as the keys instead of the whole sentences for searching a table listing all different grammar trees of all correct sentences in a template. Let us call the sequence of part-of-speech tags of a sentence a POST array of the sentence, hereinafter.

Now we want to establish a 1:1 relationship between the POST arrays and grammar trees.

First we want to show this by the parsing result of the PCFGs. Consider the same parsed trees of two well-formed sentences having the same POST array.

削除: PCFG's

The PCFGs generate a parse tree of a given sentence by selecting, among all the possible structures of the sentence, the one having the highest probability (Sekine and Grishman, 1995) of  $P_{tree} = \prod_{Struc_i \in tree} P_{Struc_i} \prod_{tag_j \in tree} P_{tag_j | word_j}$ <sup>4</sup>, where among all

削除: PCFG's

the possible structures starting with the same symbol  $Struc_i$ ,  $P_{Struc_i}$  denotes the probability of a structure to take on  $Struc_i$  while  $P_{tag_j | word_j}$  denotes the probability of a word  $word_j$  to take on the tag  $tag_j$ .

When all the part-of-speech tags of a sentence are specified beforehand, the above formula simplifies to  $P_{tree} = \prod_{Struc_i \in tree} P_{Struc_i}$  (Tokuda and Chen, 2001; Chen and

Tokuda, 2003; Chen, Tokuda and Xiao, 2002). When there are several possible parse trees for one sentence, the system chooses the one having the highest probability score. Then, it follows that, if the POST array is the same for two

<sup>3</sup> At <http://web.unbc.ca/~chen1/translation>, we can find that there are a total of 37944 different correct sentences (i.e., correct English translations) can be taken as the correct answer to the translation problem of a Japanese sentence meaning "Marine resources are available in abundance in Malaysia".

<sup>4</sup> In the formula of (Sekine and Grishman, 1995), a square of the term  $P_{tag_j | word_j}$  is used.

sentences, then the production of  $\prod_{tag_j \in tree} P_{tag_j | word_j}$  must be the same for  $P_{tree}$  of both of them. Since this implies that the production of  $\prod_{Struc_i \in tree} P_{Struc_i}$  of each of the sentences should be the same so as to maximize  $P_{tree}$  for each of them, we have now proved the following Observation.

Observation: *When parsed by any PCFGs, the parse trees of two well-formed, grammatically correct sentences having the same POST array should be exactly equivalent except for the leaves comprising distinct words.*

削除: PCFG's

This implies that any PCFG produces only one grammar structure if the POST arrays of two well-formed sentences are the same. It is possible that, even if two sentences have the same POST array, the grammar trees might be different as observed in the well-known PP (prepositional phrase) attachment problem. This is beyond the capability of PCFG parsers, of course. This problem should be able to be resolved by introducing the concept of the extended POST array, which comprises a POST array with additional disambiguation bits for establishing a 1:1 mapping from the extended POST arrays to the grammar trees as in the following conjecture.

**Conjecture:** *The parse trees of two well-formed, grammatically correct sentences having the same extended POST array should be exactly equivalent except for the leaves comprising distinct words.*

We believe that this conjecture should be true for any natural language because, technically, we can keep adding one disambiguation bit whenever we encounter an ambiguity. Among many disambiguation problems that require to be solved by adding disambiguation bits to POST arrays, we show here as an example the well-known PP attachment disambiguation problem. To resolve the ambiguity of PP attachments, Brill and Resnik (1994) suggest formally to introduce a quintuple pattern in the form of  $(v, n_1, p, n_2, a)$ , where  $v$  is a verb,  $n_1$  the head of its object noun phrase,  $p$  the preposition,  $n_2$  the head of the noun phrase governed by the preposition, and  $a$  the two-value variable indicating the PP being an attachment of  $v$  or  $n_1$ . Note, however, that to ensure a distinct 1:1 mapping onto parse trees the full 5-tuples information is not needed in addition to the POST array. For the parsing purpose, the ambiguity of PP attachment problems can be resolved by attaching the values of  $a$  to the preposition in POST arrays. For example, a POST array with a value of PP attachment “ $NN VB NNS PP(v) \dots$ ”, should be appropriate for a sentence “I buy a car with a Visa card”, while the extended POST array of “ $NN VB NNS PP(n) \dots$ ” is adequate for a sentence “He washes a cloth with holes”. Although there are other disambiguation problems, such as the so-called coordination problem in natural language analysis<sup>5</sup>, we can see that, by

<sup>5</sup> A simple example of the coordination problem: safety in (trucks and minivans); (safety in trucks) and minivans. Simply adding parentheses is enough to clarify the range of applicability of disambiguation.

simply adding some extra information, we should be able to separate different situations for disambiguation purpose. We will address the details on this topic in another paper. We see that an extended POST array comprising a POST array plus disambiguation bits is sufficient to disambiguate a large number of ambiguous representations.

We can see that the number of possible parse trees in one template and thus in all the templates for the ILTS application is considerably constrained, although the number of possible sentences and the number of all possible correct sentences to be matched in one template might be very large. Consider a typical translation problem, "Marine resources are available in abundance in Malaysia" (In Japanese), there are around 40,000 correct responses (translations) embedded in the template<sup>6</sup>. Consider a typical, reasonable sized ILTS like Azalea (Tokuda and Chen, 2001) that contains 20 lessons each with 10 sentences for translation exercises. The number of correct responses (translations) alone could amount to around 8 million roughly in total. When the leaves of indivisible terminals comprising distinct words are ignored, we could expect the number of different parse trees will be reduced to less than 400,000. We will demonstrate how the reduction is implemented with a numerical example in Section 4.

Because we can extract an extended POST array from each of the parse trees, we are able to build up a table of parse trees for an ILTS, the entries of which are the extended POST arrays. When a sentence is keyed in by a user, we first match the input sentence to the template and find a best-matched correct sentence, and then we obtain the extended POST array of the sentence; the extended POST array is sufficient to find the parse tree by merely matching the extended array with an entry of the look-up table. We call the approach as a table-look up parser (TLUP) where we look up the table for the extended POST array finding a matched parse tree for the input sentence.

When expressed in a bracketed form, the parse tree actually expresses a production (rule). Thus the major difference between a TLUP and a general PCFG parser is that, while a general PCFG parser always selects the best possible list of rules from a multitude of "small sized" productions (rules) stored in the rule base, until the sentence is derived by composition of small productions, our simple TLUP captures the grammar structure of an entire sentence with a "large sized" production, or the so-called "super rule". We see that the TLUP plays the role of a super rule where the unique extended POST array constitutes a key entry to the rule.

---

<sup>6</sup> To be precise, the total number is 37944; see also the footnote [3]. It is important to note that, the developers actually did not collect/list all 37944 sentences for this question at the time of template construction. Rather, it is the template structure that allows us to collect the sentences from every possible combination of paths embedded in template. This means that, we are able to extract as many as 37944 sentences from the template, after the template is constructed by language experts using a much smaller number of samples. This is the advantage of using template. The reader is referred to articles (Chen and Tokuda, 2003; Tokuda and Chen, 2001) for examples of templates.

### **2.3 Assigning part-of-speech tags and disambiguation bits to error-free nodes in the templates**

Since the entries of the TLUP consist of an extended POST array, we require that the part-of-speech tags, as well as the disambiguation bits, if any, be assigned to each of words or phrases in the error-free nodes of the template. This can be easily done by parsing all the well-formed correct sentences into grammar trees as traced across the templates. To be consistent, we need the following assumption.

**Assumption** *The part-of-speech tag of a word, as well as the disambiguation bits, remains unchanged in all the sentences that can be derived from the same template.*<sup>7</sup>

To clarify the assumption, a simple example is given below. Consider the two sentences "Japan is dotted with beautiful parks nationwide." and "Japan has been dotted with pretty parks in the country." which constitute two paths (sentences) among many paths in a template and share the same words namely "Japan", "with" and "parks". The above assumption assumes that the part-of-speech tags of "Japan" in both of the two sentences are the same, and so are those of "with" and those of "parks".

We can assign the part-of-speech tags, as well as disambiguation bits, to each of the correct words in the template manually or automatically. Since the manual tagging task is too labor-intensive, we can use a general purpose parser and/or a tagger, perhaps a state-of-the-art parser, for tagging all the correct sentences of the templates, although we know their accuracy is not fully ensured. By assigning the part-of-speech tags to all the words on a path of correct sentence in the template, we keep parsing and/or tagging all other correct sentences in the template. These sentences (paths in the template) might have common words. The above assumption ensures that there is no contradiction in assigning the part-of-speech tags, as well as disambiguation bits, to the common words.<sup>8</sup>

We can easily find, from examples of templates, that most of the words in a template appear more than once in different paths of the template. This means that, if occasionally a parser/tagger assigns erroneous tags to some words when it parses a sentence (path) of a template, the errors are very likely to be caught when the parser/tagger parses other paths of the template<sup>9</sup>. Any inconsistency in

---

<sup>7</sup> If one word appears several times in one template, we regard them as separate words.

<sup>8</sup> Although it is an assumption and we should verify when our program is assigning part-of-speech tags to words in the templates, presently we find no exceptions. If there is such an exception, we should be able to divide the template into two or more templates so that the above assumption is satisfied.

<sup>9</sup> It is understandable that, even if a parser has only an accuracy of 75%, which is not a high standard for most of the parsers available, when it is used to parse three sentences, the possibility of getting wrong parsing with all of the three sentences will be an extremely low 2.5%, which should be acceptable for

tag assignment for the same word or phrase in different paths of a template immediately suggests that the parser assigns an erroneous tag, giving off a warning of errors in parsing. Of course, when an error is caught, we can correct it manually. Or, when a word appears in more than 3 sentences, a simple voting could be employed to determine which one is correct when the results conflict<sup>10</sup>. In fact, the assumption ensures that our tagging by the template structure provides an effective means of reducing the risk of providing wrong tags.

## **2.4 Statistical Assurance by Neyman-Pearson Lemma**

The accuracy of parsing that we expect in an online tutoring system is quite high. How reliable is our parsing results of all the correct sentences in the templates as described in Section 2.3? Here we assume that all the possible corrections are made either by voting or by changing manually all the errors caught by machine. The well-known Neyman-Pearson Lemma (Ryan, 2000; Hoel, Port, and Stone, 1971) will be fully exploited to ensure this.

From the production view point, we can set two numbers  $p_0 = 0.01$  and  $p_1 = 0.05$  in equations (1) and (2) below<sup>11</sup>, such that we will be satisfied with the parsing results if the error rate  $p \leq p_0$  and we will never accept the results if  $p \geq p_1$ .

This implies that we shall have to check all the parsing results of correct answers (sentences) in a template. if  $p \geq p_1$ , then we are not satisfied with the parsing result. But how can we get the value of  $p$ ? In view of the vast number of sentences involved in a template, it is not practical to check the grammar trees one by one. The Neyman-Pearson Lemma allows us to do this by a sampling test. Choose two numbers  $\alpha = 0.05$  (Producer's Risk) and  $\beta = 0.10$  (Consumer's Risk); then for a given  $(n, c)$ , which we obtain by solving the MP testing formula of equations (1) and (2) below, we have a chance less than  $\alpha$  of no satisfaction when  $p \leq p_0$  and we have a chance less than  $\beta$  of not insisting the rejection when  $p \geq p_1$ .

Now suppose we choose  $n$  samples for careful testing, and no more than  $c$  samples of these  $n$  samples are not correctly parsed (i.e., the parsing results fail to satisfy us). The so-called  $(n, c)$  lot acceptance sampling plan (LASP) (Ryan,

---

teaching purposes. We can see that, when a parser gets a wrong tagging to a sentence, it is very likely to set wrong POST tags (extended POST tags) to many words in a sentence; this indicates that when several sentences share the same word in a template, the chance that a parser makes wrong parsing without being caught is very low.

<sup>10</sup> When the POST array of one word in a sentence is corrected manually or by "voting", we should parse the whole sentence again to ensure the correctness of the tags of other words in the sentence.

<sup>11</sup> If we regard the parser as a producer and the user as a consumer of the parser,  $p_0$  actually denotes an Acceptable Quality Level, representing the base line requirement for the quality of the producer's product, which is a parse tree of a set of correlated sentences, while  $p_1$  is the Lot Tolerance Percent Defective, a designated high defect level that would be unacceptable to the consumer.

2000, Hoel, Port, and Stone, 1971) will be found as follows. The  $(n, c)$  plan should satisfy:

$$\sum_{k=0}^c \binom{n}{k} p_0^k (1-p_0)^{n-k} \geq 1-\alpha, \quad (1)$$

and

$$\sum_{k=0}^c \binom{n}{k} p_1^k (1-p_1)^{n-k} \leq \beta. \quad (2)$$

Noting that  $T(p) = (c - np) / \sqrt{np(1-p)}$  approximately for a standard normal distribution  $N(0, 1)$  for a sufficiently large  $n$ ,  $T(p_0) = 1.65$  and  $T(p_{-1}) = -1.30$  satisfy the above two equations (1) and (2) giving  $n = 125$  and  $c = 3$ .

This means that if we randomly choose 125 parsed sentences for manual checking and if there are more than three errors, we should check carefully to see if we should provide certain help.<sup>12</sup>

### 3. Technical Methods

#### 3.1 Setting Up Table Look-Up Parser (TLUP)

The look-up table for the TLUP has two columns: one for the extended POST arrays, and another for the corresponding parse trees. By the construction scheme of the table, it is evident that every entry of all the correct sentences extracted from template has a match in the array of the TLUP table.

In the process of setting up the look-up table for TLUP, we shall also place the extended part-of-speech tags to the correct words in the templates<sup>13</sup>.

For any template, do steps 1-5:

1. Extract a new correct answer (sentence) from the template;
2. Use a general-purpose parser to parse the sentence;
3. Add the tags of each of the words of the sentence to the template;
  - 3.1 If a word has already been assigned with a different tag from the new tag being assigned, then manually check it: Find out which one is in error, re-parse the sentence that modifies the erroneous tagging by assigning the correct tag to the ambiguous word, and go to step 3 again;
4. Add the pair of an extended tag array and a parser tree as an item to the look-up table, if it does not already exist;<sup>14</sup>
5. Extract a new correct answer from the template, if any, and go to step 2; otherwise terminate the parsing process.

<sup>12</sup> Possible "help" includes the manual assignment of part-of-speech tags to certain words or phrases, re-parsing of all the correct sentences, and manual adjustment of the tree structures.

<sup>13</sup> We can use a general-purpose parser to do both of the jobs or use a tagger for tagging purposes and a parser for the rest.

<sup>14</sup> The assumption in Section 2.3 guarantees that no conflict will occur at this stage. If it does occur, we should come back to use more disambiguation bits in the parser.

As it was described in Section 2.4, Neyman-Pearson Lemma can be applied to predicate the accuracy of the TLUP. We still use  $p_0=0.01$  and  $p_1=0.05$ , and  $\alpha=0.05$  (Producer's Risk) and  $\beta=0.10$ . According to Section 2.4, we should choose 125 sentences from the templates and see if the number of errors is smaller than 3. If so, it means that the look-up table parser will have less than 10% chance of having an accuracy less than 95%.<sup>15</sup> We tested several groups of 125 sentences, and the number of errors has never exceeded 2. Because most of the possible errors in tagging can be caught while the sentences in a template are being parsed, it remains within our expectations (see Section 2.3).

### 3.2. Grammar analysis by table look-up parsing

A simple scheme of syntactic analysis of input sentences will be demonstrated below:

1. Given an input sentence from a student, select the most closely matched correct sentence from the templates prior to the syntactic analysis.
2. Extract the part-of-speech tag as well as the disambiguation bits of each word in the matched sentence to obtain an extended POST array.
3. Obtain the grammar tree for the most closely matched correct sentence by simply matching its extended POST array with those registered in the look-up table.
4. By collating the resulting parse tree and the keyed-in sentence, provide appropriate grammatical feedback to students.

The most closely matched sentence could be found by the HCS (heaviest common sequence) matching method described in (Tokuda and Chen, 2001; Chen and Tokuda, 2003) or by editing distance (Baeza-Yates, 1992).

We can see that we actually do save all the possible super rules for the correct sentences in the ILTS. For any input sentence, as we only have to look-up a table to find a grammar tree, we can know that it costs very small amount of computing time, in comparing with that of any general PCFG.<sup>16</sup>

---

<sup>15</sup> For the templates we used in Azalea, all of which have a complexity much higher than that of figure 1, we use Brill's tagger (Brill, 1994) to generate POST tag arrays, and we use ApplePie (Sekine and Grishman, 1995) to get the parse trees in the above look-up table construction process.

<sup>16</sup> Readers interested in theoretical analysis will be able to prove the following theorem:

**Theorem** If the length of a matched correct sentence is  $n$ , and the size of part-of-speech tag set is  $m$ , we are able to find the corresponding grammar tree within  $O(n \log_2 m)$  time.

As a matter of fact, the part-of-speech tag set is always quite small, and thus the time complexity for finding a grammar tree can be regarded as  $O(n)$ . Since the lengths of a matched correct sentence and the input sentence are of the same order, the complexity of generating the grammar tree is of  $O(n)$ . This shows a marked improvement over the complexity of a general parser, which is  $O(n^3)$ .

## 4. Numerical Examples

We shall now examine the size of look-up table for an ILTS system by first examining the template for a typical ILTS question: Translate the Japanese sentence meaning "Japan is dotted with beautiful parks nationwide" into English. A reduced version of the template was given in (Chen and Tokuda, 2003). A total of 1032 different correct sentences can be extracted from the templates. But among them, we can see that there are only 136 different grammar trees. Notice that 136 of 1032 is about 1 out of 8.

Now consider a more realistic application. For a full-sized template that we use for the English translations of Japanese sentence, meaning "Fisheries resources are available in large numbers around Malaysia.", we have built in a total of around 40,000 different answers (translations). We use Brill's tagger<sup>17</sup> in tagging the sentences, and a slightly modified ApplePie parser, which we called as a POST parser in (Tokuda and Chen, 2001) in obtaining grammar trees. It is found that we have only 1614 different part-of-speech arrays<sup>18</sup>, implying that the table for the TLUP only requires 1614 entries for this template<sup>19</sup>.

During the process of parsing the sentences extracted from the template and assigning extended part-of-speech tags to words in the template, some errors happened occasionally (it is understandable that a general parser normally does not have a very high accuracy), but as it is explained in Section 2.3 they were all detected. For example, for the sentence, "*Malaysia overflows with marine living resources.*", Brill's parser provides a POST array as "*JJ NNS IN NN VBG NNS*", which is obviously not correct. But, this error can easily be caught because in most of other sentences such as "Malaysia has a great many living aquatic resources.", the word "Malaysia" is tagged with "NN". After parsing the sentences in the templates one by one, manually making modifications when errors are detected (automatically), and making the look-up table, we can come back to check if the table-look up parser has an acceptable accuracy. We use the method described in Section 2.4 to verify the accuracy of the table-look up parser, using  $p_0=0.01$ ,  $p_1=0.05$ ,  $\alpha=0.05$ , and  $\beta=0.10$ . So we need to check only the parse trees of 125 randomly chosen sentences from among the 40,000 sentences. In our testing, we did not even encounter any error. This means that we can expect that the accuracy is most likely much higher than 95%.

As it was explained in Section 2.3, this result is well expected; the probability that an error made by a parser is not caught is extremely small as we have many

---

<sup>17</sup> The tagger can be downloaded from <http://www.cs.jhu.edu/~brill/>.

<sup>18</sup> At <http://web.unbc.ca/~chen/translation>, the file all-translations.txt contains all 37944 different correct sentences (i.e., correct English translations) as correct answers for the translation problem of a Japanese sentence meaning "Marine resources are available in abundance in Malaysia". The file all-different.txt contains all 1614 sentences that have different part-of-speech arrays. The part-of-speech array of any of other 36330 sentences must reduce to that of one of these 1614 sentences.

<sup>19</sup> We can reduce the number of the sentences even further if we take advantage of the common structures of sentences prevailing among different templates.

sentences sharing some common words in a template. We are quite confident now that our online TLUP approach is practical in real situations. Our approach saves not only search time but also the dynamic search space that any general purpose or statistical parsers demand for successful parsing.

## 5. Concluding Remarks

This paper has provided a simple but powerful scheme for constructing a table look-up parser (TLUP) for an ILTS application. The success of a TLUP comes from the fact that most of the ILTS applications allowing free input analysis process a fixed number of questions and a fixed but very large number of answers, which may or may not be stored in template format, although the examples in this paper used template structure for storing the answers. The TLUP has the following advantages over the general purpose parsers when used in ILTS applications: (1) It requires much less time and space in parsing semi-free format input sentences, enabling many learners to use the system in an online real time environments; (2) It obtains a high accuracy that may be comparable with expert human language teachers.

Certain further work can be performed to save the storage of the look-up table for the TLUP. It is understandable to consider using another template structure to store entries of the table, but how to store the corresponding grammar tree structure in such a template remains to be solved.

## Reference:

- Baeza-Yates, R. A. (1992). Introduction to data structures and algorithms related to information retrieval. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval*. Prentice Hall, New Jersey, pages 13--27.
- Brill, E. (1994). Some advances in rule-based part of speech tagging. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA.
- Brill, E. and P. Resnik. (1994). A rule-based approach to prepositional phrase attachment disambiguation. In *Proc. of the Fifteenth International Conference on Computational Linguistics*.
- Chen, L. and N. Tokuda. (2003). Bug diagnosis by string matching: Application to ILTS for translation. *CALICO Journal*, 20(2):227--244.
- Chen, L., N. Tokuda, and D. Xiao. (2002). A post parser-based learner model for template-based ICALL for Japanese-English writing skills. *CALL*, 15(4):357--372.
- Hoel, P. G., S. C. Port, and C. J. Stone, (1971). *Testing Hypotheses*, Chapter 3, pages 56--67. Houghton Mifflin, New York.

Ryan, T. P. (2000). *Statistical Methods for Quality Improvement*. Wiley, New York, NY, 2nd ed.

Sekine, S. and R. Grishman. (1995). A corpus-based probabilistic grammar with only two non-terminals. In *Proc. 4th Int'l Workshop Parsing Technology*, pages 216--223, Prague.

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