DOCUMENT RESUME

ED 351 872 FL 020 748

AUTHOR Leung, Kei Wai; Maciejewski, Anthony A.
TITLE Technical Specifications of the Nihongo Tutorial System.
INSTITUTION Purdue Univ., West Lafayette, IN. School of Electrical Engineering.
SPONS AGENCY National Science Foundation, Washington, D.C.
REPORT NO TR-EE-90-27
PUB DATE Apr 90
CONTRACT NSF-INT-8818039
NOTE 69p.; For related document, see FL 020 747.
PUB TYPE Reports - Descriptive (141)

EDRS PRICE MF01/PC03 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Japanese; *Languages for Special Purposes; *Programing; Reading Comprehension; Second Language Instruction; Sentence Structure; Technical Writing; Uncommonly Taught Languages; *Vocabulary Development
IDENTIFIERS *Nihongo Tutorial System

ABSTRACT

The Nihongo tutorial system is an intelligent tutorial system designed to use a computer to assist scientists and engineers in developing reading competence in technical Japanese. It consists of three applications: the Nihongo Tutor, which provides useful information about an article (translation, syntax, pronunciation) to help understand the text of a personalized lesson, and records student performance; the Parse Tree Editor, used to prepare technical articles for the tutorial system; and the Administrator, which uses student data to assign an article that is technically and pedagogically appropriate. The program integrates the three functions. This report details the technical specifications for system implementation, including both the data structures and algorithms used. The section on the Administrator looks at the attributes of the system files, technical (discipline) fields included, user records, user article selection, and user database. Information on the Nihongo Tutor covers text data structures, student database management, and electronic dictionaries. An explanation of the Parse Tree Editor offers specifics on the four stages of parsing: segmentation, syntax, semantics, and mapping. Some screens are used for illustration. (MSE)

Technical Specifications of the Nihongo Tutorial System

Kei Wai Leung
Anthony A. Maciejewski

TR-EE 90-27
April 1990

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

TECHNICAL SPECIFICATIONS OF THE NIHONGO TUTORIAL SYSTEM

Kei Wai Leung

Anthony A. Maciejewski

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

TR-EE 90-27
April 1990

This material is based upon work supported by the National Science Foundation under Grant No. INT-8818039. The Government has certain rights in this material.

TABLE OF CONTENTS

CHAPTER I - INTRODUCTION

The Nihongo tutorial system presented here is an intelligent tutorial system designed to use a computer to assist scientists and engineers in developing reading competence in technical Japanese. It is comprised of three applications: the Nihongo Tutor, the Parse Tree Editor and the Administrator. The Nihongo Tutor provides useful information about an article such as an English translation, syntax, and pronunciation to help the student to comprehend the text of a personalized lesson. It also records the student's performance into his personal database which reflects his current Japanese language proficiency. The Administrator uses the personal data of a student to assign an article that is related to his technical interests and at his current level of Japanese competence. The Parse Tree Editor is used to prepare technical articles for the tutorial system by incorporating semantic, syntactic, phonetic, and morphological information into the parse tree of a sentence in each article. This report presents the technical details of the implementation of the Nihongo Tutorial System including both the data structures and algorithms employed.

CHAPTER II - THE ADMINISTRATOR

The function of the Administrator is to perform database management for the tutorial system. It receives information from the tutor about the performance of the student and selects an appropriate article for his next lesson based on his current Japanese reading competence. The student's personal database keeps a record of the Japanese words, readings, and grammatical structures that the student has currently mastered. By comparing this information with the content of a potential instructional article the Administrator can assign an article that has a desirable amount of new material. Not only does this preclude unnecessary repetition of already mastered materials but it also prohibits the introduction of a large amount of material that the student, at his current level, cannot handle at one time. The Administrator also attempts to provide articles which conform to the student's technical area of interest.

In addition to handling the student personal database, the Administrator also assists in system management and development. It groups all currently available articles into categories so that the student can be assigned course materials directly related to his profession and interest. It also saves important information about the student, such as his area of interest, his last access time of the Nihongo Tutor, etc, which aids the Administrator in selecting articles for him.

2.1 System Files

There are several important files that the Administrator maintains for system management. These files are called:

(a) Technical Fields
(b) User Record
(c) User Selection
(d) User Database

The first file records all the currently available articles categorized into different technical areas. The second one keeps personal information for each individual student, such as his area of interest, the article name of his last lesson, etc. The third one saves the article that should be assigned to a student as the next lesson. The last one stores information about the student's level of Japanese language proficiency. The attributes of the records stored in these files are presented in table 1.

Table 1. The Attributes of the Database System Files

Record type	Attributes				
Technical field	Technical area*	File title	Start/End index	Child node list	Parent node
User record	User name*	Technical area	Article area	Last access time	Last access file
User selection	User name*	File name			
User database	User name*	Well-known Area	Unfamiliar Area		

* the key of the record

2.2 Technical Fields

The Administrator attempts to assign articles that match the technical area of interest of the student. Since there is a great variety of different interests among the students, the list of articles must include topics like physical sciences, engineering, computer sciences, etc. These topics can further be divided into sub-topics if there is a demand for more specific technical areas. This suggests the usage of a generalized tree structure to maintain all the technical fields available. Figure 1 shows the tree configuration of the technical fields.

For each topic (or sub-topic), there will be a number of articles accessible to the student for instruction. Thus, each one of them is assigned a generic file name together with a unique file index. The first and last indices reflect how many articles are available in a specific area. New topics are added to the article list by inserting them into the tree. Any article can be appended to or removed from a technical area by updating the file indices of that area.

The records in the "Technical Fields" file are stored in a string list format (see figure 2). All the records are kept in sequence so that they can be identified by their position in the file. The child node list of a topic includes the sub-topics under it. The list is terminated by a sentinel value of "-1". If there is only the sentinel value, the child node list is empty and the topic is then one of the leaf nodes in the tree. The parent of a sub-topic represents a more general area that this sub-topic can be categorized under. If an area is in the top level of the tree with no parent topic, its parent node identification number will be set to zero. In figure 2, the first record physics is a top level area with two sub-topics: kinetics and kinematics which have two and three articles available respectively. If there is one more article to be introduced into the area physics, its ending file index has to be changed from 1 to 2. Initially, only those topics with no parents are displayed when a student starts searching through

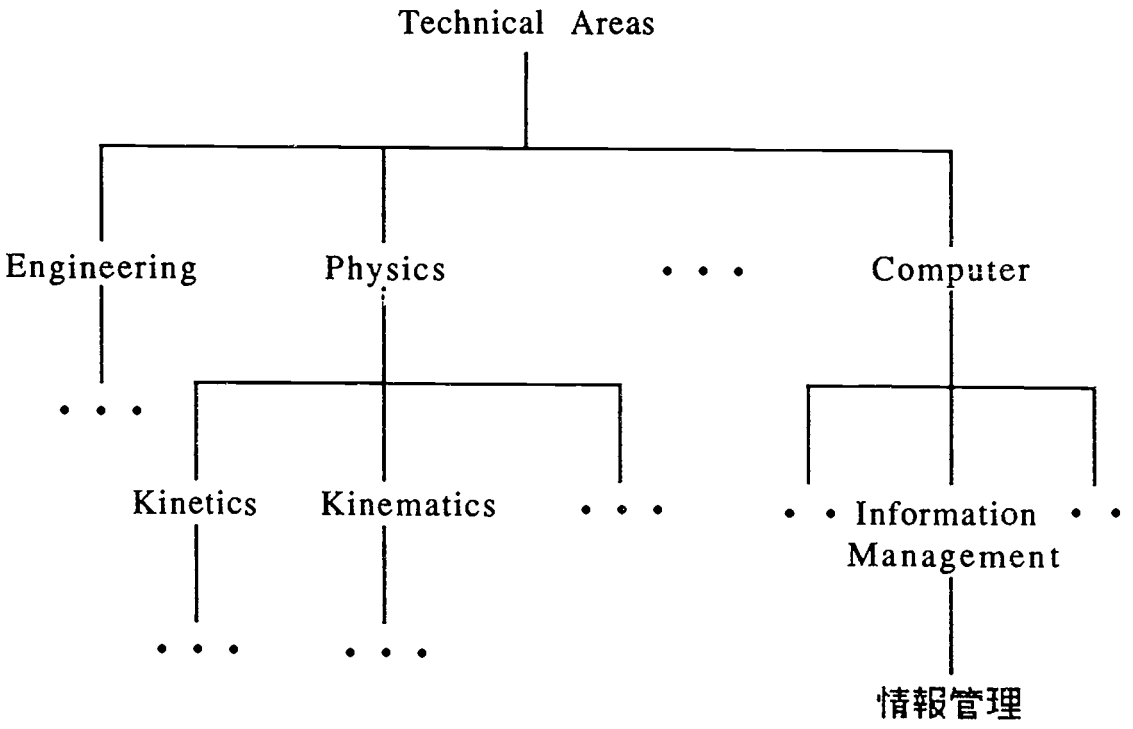


Figure 1. A Tree Representation of the Technical Fields

Physics; Physics; 1 1;2 3 -1; 0" 1
Kinematics; Kinematics; 1 2; -1; 1" 2
Kinetics; Kinetics; 1 3;-1;1" -1
① ② ③ ④ ⑤ ⑥

Legends:
① Technical Area
② File Title
③ Starting and Ending File Indices
④ Child Node List
⑤ Parent Node
⑥ String Number

Figure 2. The Format of the Technical Fields File

the tree. If the sub-topics of an area are requested, only those in its child node list are shown. On the other hand, when a parent topic is requested, only one area (the parent area) is displayed.

2.3 User Record

The Administrator must keep personal information of the student in the "User Record" file to select articles that are related to his profession and technical area of interest. If some articles are available in this area, an appropriate one will be assigned to him based on his level of Japanese proficiency level. Normally, the Administrator automatically searches through the database to find articles that match the student's technical area. If there are no such articles available, however, the student can manually choose an article area that is closest to his technical area.

The records of the "User Record" file are stored in a list format (see figure 3). Each record can be retrieved by using the Munger Function (available from the Macintosh operating system) to search for a particular key which is the name of the student. Once the name is found, the entire record is retrieved and update operations are performed with it. Any changes are then saved into the "User Record" file. For example, in figure 3, if the student finishes reading the article "Kinetics1" in the last lesson and proceeds to study the next article "Kinetics2", the last field of the record will be changed to "Kinetics2" and the last access time is modified.

Since the Administrator keeps the technical area and the article area as separated fields, the demand of the student for new technical areas can be reflected to the supervisor. This provides some feedback from the student as to what he wants to be included into the system. The last access time reveals to the student when he last used the tutor to study an article. The last file access shows the previous lesson that the student may want to review if he does not have access to the tutor for a long period of time.

Each time the Administrator attempts to assign a new article for a student the information in his user record is displayed. Then his personal database is retrieved and compared with the contents of the

Nelson Leung" Kinetics" Kinetics" -1579190216" Kinetics1"
① ② ③ ④ ⑤

Legends:
① User Name
② Technical Area
③ Article Area
④ Last Access Time
⑤ Last Access File

Figure 3. The Format of the User Record File

next lesson. In particular, the lexemes (i.e. all leaf nodes in the parse trees of the sentences) stored in the file containing the translation information of the article are compared with the student database in order to give a rough estimation of how much new material will be presented to him in this article (see figure 4).

The task of article selection for a student is carried out automatically by the Administrator based on his current Japanese competence. However, if the student is not satisfied with the article assigned to him, he can manually override the selection and invoke comparisons of his database with other articles in the area until he finds one that is most appropriate. Once he accepts an assignment, his user record will be updated with the new access file name and new access time. In case the student wants to study an article that is not in his technical area of interest, he will be assigned with one starting with the first file index.

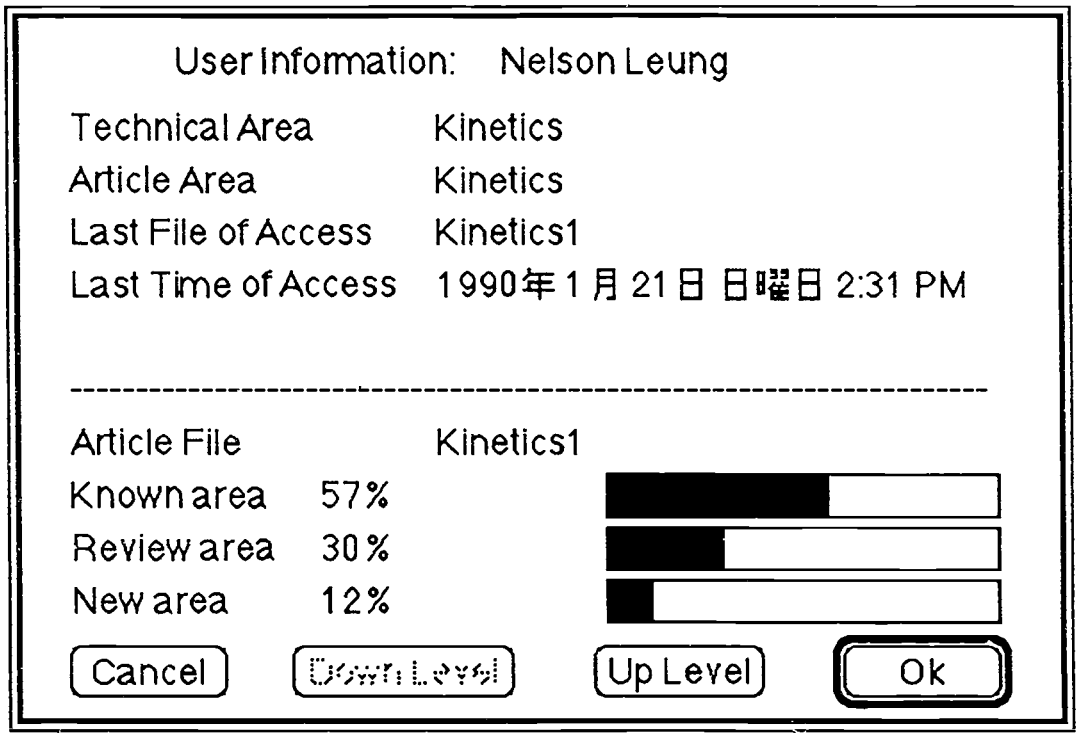


Figure 4. A Comparison of a Student's Database with the Lexemes of an Article

2.4 User Selection

After the Administrator selects an article for the student, it saves the assignment in the "User Selection" file under the student's name (see figure 5). The records, stored in a list format, are retrieved by using the Munger Function. The tutor knows about this assignment by searching through the selection file for the file name. Changes to the selection file can only be performed by the Administrator. This insures that if a student does not finish a lesson in a session, he can return to the same article in the next session. When the student finishes a lesson, he should use the Administrator to select the next article for him. What he learns from the last lesson is updated and saved in his personal database by the tutor as discussed in the next chapter. Thus the selection file serves as a communication link between the Administrator and the Nihongo Tutor.

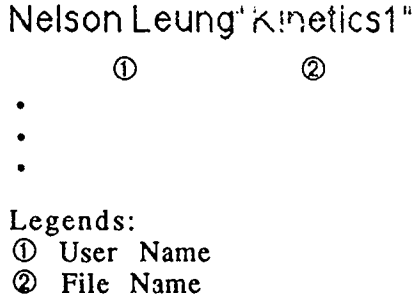
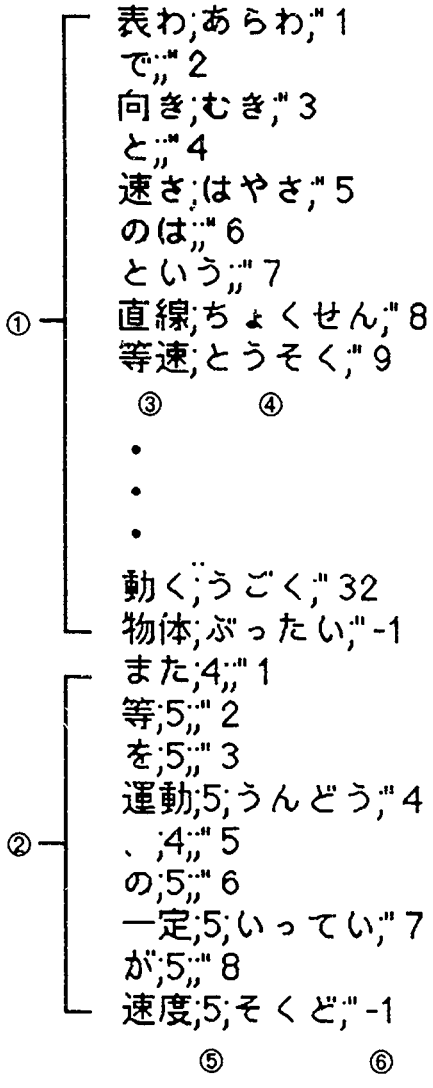


Figure 5. The Format of the User Selection File

2.5 User Database

Each student using the tutorial system has a personal database file that provides a measure of his Japanese language competence. The filename is defined as the student's login followed by a ".db" extension. This file consists of two areas: a well-known area and an unfamiliar area. The well-known area keeps all the material that the student has currently mastered whereas the unfamiliar area saves information about material that needs to be reviewed. Each time he studies a lesson, his database records are updated by the tutor to reflect his current level of competence. The Administrator's task of selecting an appropriate article for the student is closely related to the information on the student's proficiency of Japanese. The Administrator is designed to maximize comprehension through context and minimize rote memorization and dictionary use. Thus, it does not assign an article to the student that is too difficult for him to study. In other words, a certain percentage of the lexemes in the article should fall into the well-known area.

Figure 6 shows the contents of the two areas stored in a "User Database" file. The records are loaded into the system and stored in two hash tables, one for each area. When the Administrator is comparing the contents of an article with a student's personal database, the lexemes in the translation file will be hashed into the tables to check for a match with the student database. A match is said to have occurred if the two keys are identical. There is a chance that a lexeme in the article is not matched in neither the well-known area nor the unfamiliar area. This lexeme is then totally new to the student. The Administrator does not change any records of the student's database during the process of comparison.



Legends:
① Well-known Area
② Unfamiliar Area
③ Keyword
④ Reading
⑤ Frequency Count
⑥ String Number

Figure 6. The Format of a User Database File

CHAPTER III - THE NIHONGO TUTOR

The underlying task of the Nihongo Tutor is to improve a student's proficiency in comprehending technical Japanese by providing him with Japanese articles to study. The article assigned is carefully selected by the Administrator according to the student's current Japanese competence, the level of difficulty of the article and his technical area of interest. Presumably, if the student does not know the meaning or pronunciation of some portion of the text he will request semantic, syntactical, or phonetic information from the tutor. Consequently, there must be a fast and efficient interaction between the student and the internal data structures that store all of this information. In addition, the tutor keeps a record of what the student has learned from the current lesson and what is still unfamiliar. This information is stored into his database to be used by the Administrator in assigning an appropriate article as the next lesson. The tutor also furnishes several electronic on-line dictionaries for cross-referencing purposes.

3.1 Text Data Structures

Each sentence in an article is represented by a parse tree based on the syntax of the language. The semantics of the sentence is included in the parse tree as additional information stored in each node of the tree. Figure 7 shows how a Japanese sentence is represented as a parse tree. Note that each node in the parse tree may have an arbitrary number of child nodes.

A generalized tree is a suitable data structure to represent the parse tree. In the tutorial system, this generalized tree is implemented as a doubly-linked list. The linked list can effectively connect all nodes together for fast searching and retrieval. Moving upward or downward along different levels of the parse tree can be accomplished readily due to the double link between the nodes. On one hand, moving downward along a parse tree shows how a sentence or a phrase is broken down into its constituents. On the other hand, moving upward depicts how several clauses are combined together to form a sentence. The double link thus allows the student to easily investigate Japanese sentence structure. (Since there are no word delimiters in Japanese text, it is sometimes difficult to understand Japanese sentence structure since the first step of breaking a sentence down into phrases is not trivial.) This tree data structure is further augmented to include all of the translation information for the entire article. In particular, there is a sentence list consisting of nodes that keep the translation information for every sentence in the augmented parse tree.

Figure 8 illustrates how translation information is stored in the sentence list. With the help of this data structure, translation information can be retrieved by the scheme shown in figure 9. Whenever the student has difficulty in understanding some of the text in the article, he highlights this text to notify the tutor. Examining the internal edit record of the text window, the tutor identifies the selected text based on its relative position in the article. To retrieve the actual highlighted characters, however, the

Original Japanese sentence:
速度というのは速さと向きで表わされるものである。

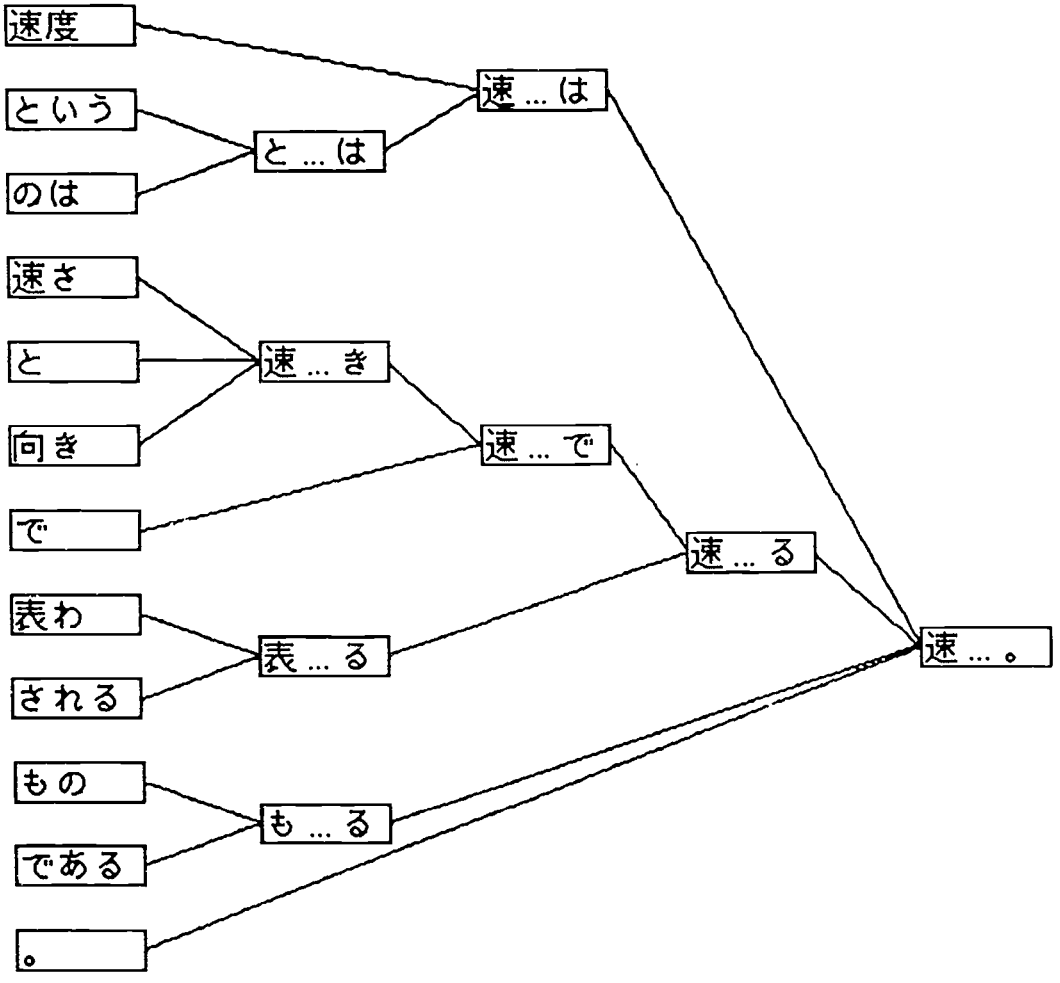


Figure 7. A Parse Tree Representation of a Japanese Sentence

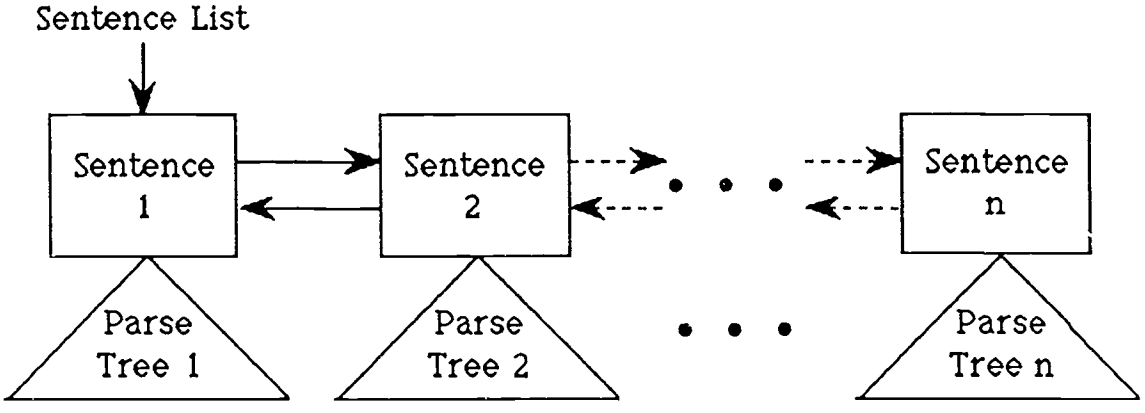


Figure 8. The Sentence List of a Japanese Article

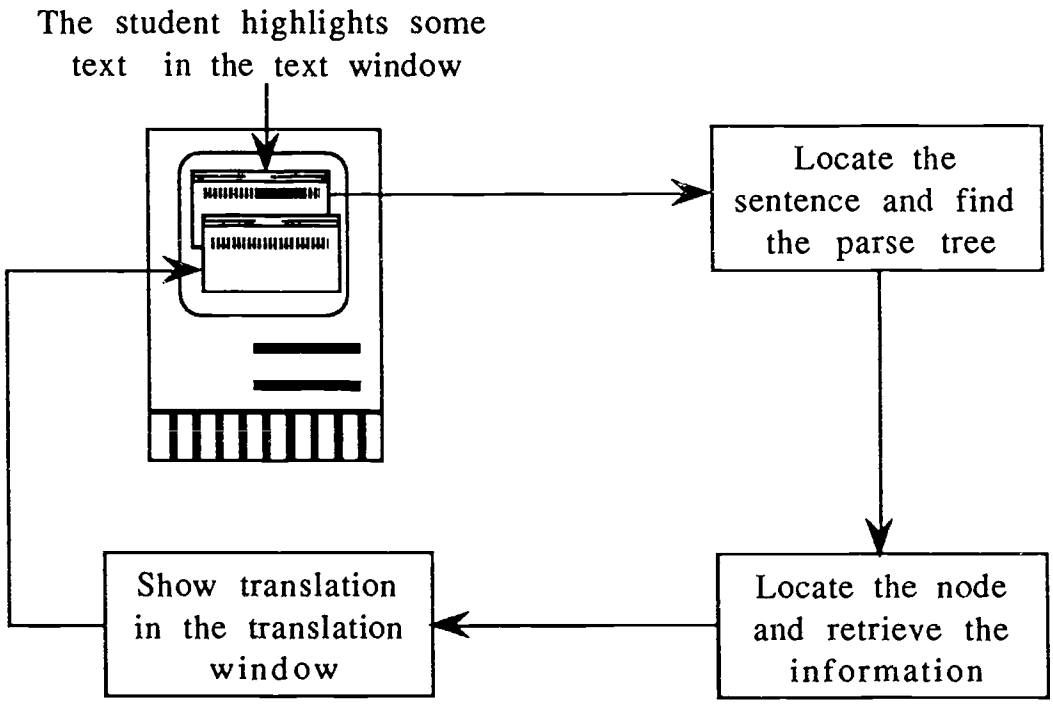


Figure 9. Interaction between the User and the Nihongo Tutor

Macintosh operating system (see Inside Macintosh, Vol I, p. 384) provides a routine called TEGetText that converts the entire text document into a character array. If this had to occur each time the student highlighted a region of text, it would be very inefficient. As a result, each node in the sentence list and each parse tree stores the location of the text that it encloses. To check whether the selected text is included in a node, only a comparison of its text location with the selection range is required. To get the translation information, the Nihongo Tutor locates the highlighted text in a sentence and returns the associated parse tree. It then finds a node in the lowest possible level of the parse tree enclosing all of the selected text. All the translation information based on this node is retrieved and then displayed. More about this searching algorithm will be described later.

Time Complexity Analysis

The retrieval time depends on two levels of searching: through the sentence list, and through the parse tree. The time complexity of the first one is O(k) for the worst case, where k is the number of sentences in the article. More often than not, the student will be reading the article sequentially from one sentence to the next one instead of randomly selecting a sentence to study. The tutor takes advantage of this locality of reference by putting a marker beside the sentence which has currently been comprehended. Assuming that most sentence selections do not change drastically from sentence to sentence, it is highly probable that the next sentence will be the target sentence to be searched. In this case, the search time will be O(1). Thus, searching from the marker can significantly reduce the amount of time required to look for the target sentence.

The retrieval time through the parse tree is more complicated since nodes in the parse tree have a variable number of child nodes. Suppose that m is the total number of nodes in a parse tree and each

node bears n children on average. The expected number of levels in this parse tree will be $\log_n m$, and in each selected node within a non-leaf level, half of its children are expected to be examined before the next searching path can be determined. Thus, the average case is $O([n\log_n m]/2)$. The overall expected time complexity is therefore $O(1+[n\log_n m]/2)$.

Attributes

The contents of a node must include all semantic, phonetic, and morphological information as well as any supplemental data for instruction but at the same time remove as much redundancy as possible. The attributes should include the lexical boundary of the stored text, its best English equivalent contextual meaning, Japanese readings, possible syntactical breakdown, and so on. Moreover, the node must be structured in such a way that it can be used to represent the sentence list. The attributes associated with a node are presented in figure 10 and are defined as follows:

(1) Identification number : identification within the parse tree
(2) Text location : starting and ending positions of the text in the article
(3) Child node list : a linked list of child nodes
(4) Parent link : a pointer to the parent node
(5) Actual text : strings of Japanese text (with readings) and English translation
(6) Character index : indices to the actual text strings for retrieving Japanese text and English equivalent

The actual text attribute pertains only to a sentence list node whereas the character index attribute applies only to a parse tree node. The purpose of this special implementation is to remove redundant storage of text including Japanese text, readings, and English translations, within a node. The root of the parse tree

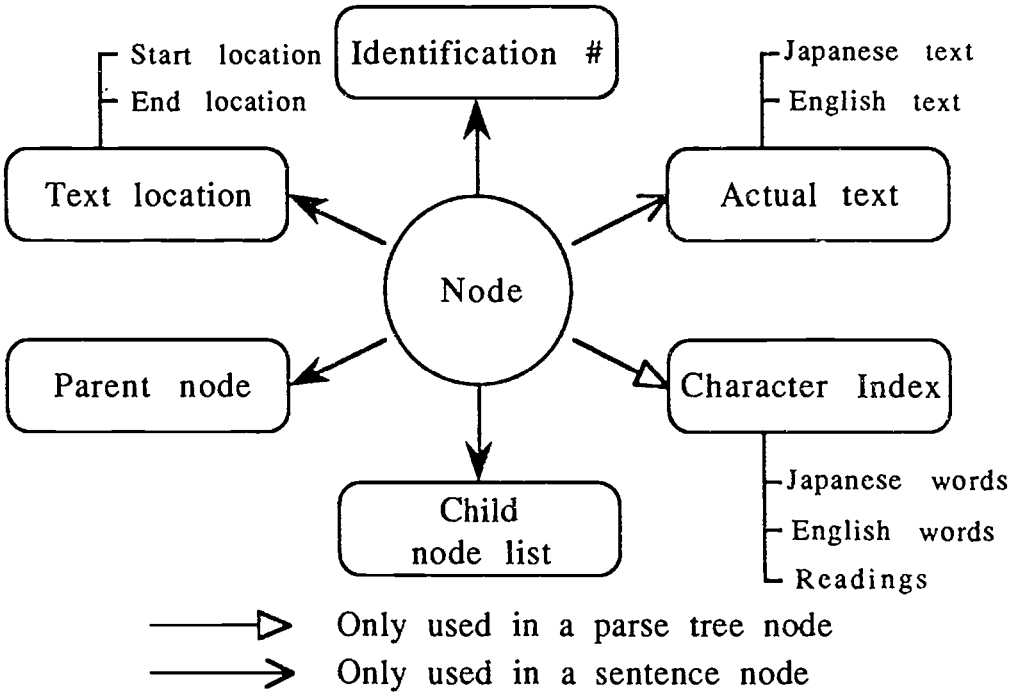


Figure 10. The Attributes Associated with a Node

normally contains most of the text contents that its descendants need. Consequently, instead of redundantly storing the text information in each node, it is kept only in the sentence node. Whenever translation information is requested, the tutor uses the character index attribute as an index into the actual text strings and retrieves the appropriate words. Table 2 shows the text contents stored in a sentence node and what text characters will be retrieved according to the word indices in the parse tree nodes.

The actual text string can be considered as a single string of infinite length used to store a large amount of text information. All attempts of text information retrieval should access this text string. The actual implementation, however, divides it into substrings of equal string length and connects them together in a linked list (see figure 11). This kind of data structure is frequently used in the tutorial system and will be referred to as a "string list".

The character index is a record consisting of two fields: the starting index (text begin) of the text and the length of the words (text length). The desired text is retrieved according to the following formulas:

Start index = (text begin) **mod** (string length) (1)
End index = Start Index + (text length) (2)
String number = (text begin) **div** (string length) (3)

String number indicates the location of the substring in the string list where the text can be found. To give an example of how to practically retrieve some Japanese text, its reading, and the associated translation, figure 12 shows the contents of the text strings and table 3 summarizes the results of the retrieval for some specific values of character indices.

To compare the storage savings, the three records of the character index account for six bytes of space in each node. If the text or readings to be stored in a node requires more than this

Table 2. Example of Text Information Stored in Nodes

In the sentence node:

Japanese text and readings:

速度というのは速さと向きで表わされるものである。

あらわむきはやさそくど

English translation:

velocity is a thing represented by speed and direction

the thing called velocitycan be representedperiod

Starting text location: 0

Ending text location: 48

In the parse tree:

Node	Text Location	Char. Index	Japanese text	Char. Index	English text	Char. Index	Reading
1	0	1	速度*	1	velocity	65	そくど*
	3	4		8		6	
2	4	5	という*	65	called	0	—
	9	6		6		0	
3	10	11	のは*	55	the	0	—
	13	4		9	thing	0	

* two bytes are required to represent a Japanese character

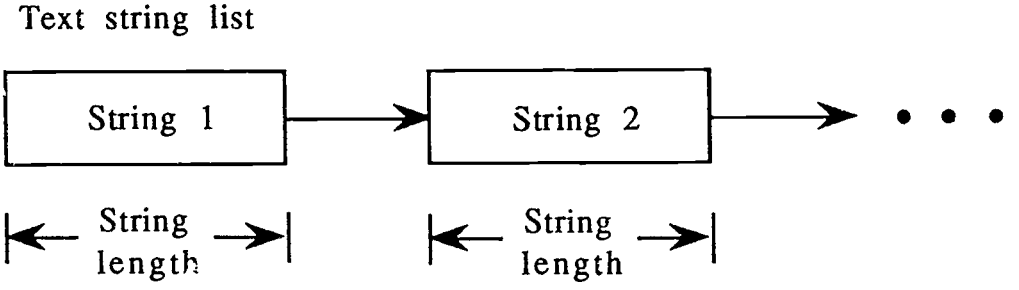


Figure 11. The String List Representation of the Text String

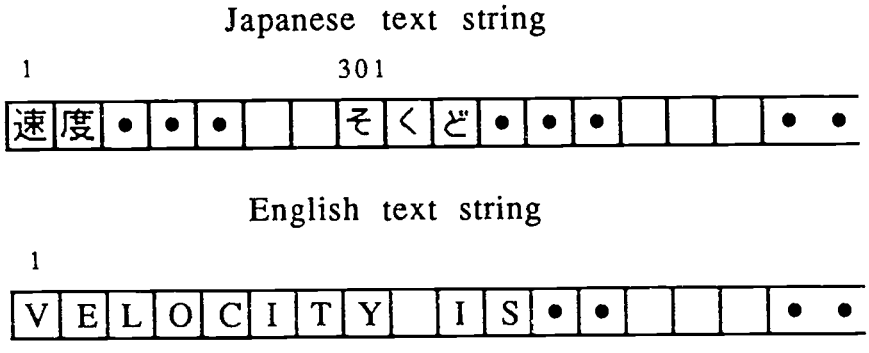


Figure 12. The Japanese and English Actual Text Strings

Table 3. Example of Text Information Retrieval

Character Index	Text Begin	Text Length	String Number	Text
Japanese text	1	4	1	速度
Pronunciation	301	6	2*	そくど
English Translation	1	8	1	velocity

*String length = 255

amount, the character index data structure will reduce the storage space requirement. For instance, the sentence in figure 7 has 20 nodes with 120 bytes used for the character indices. Assuming that actual text characters are stored instead, it would require 185 bytes for the Japanese text, 268 bytes for the English text, and 22 bytes for the readings, thus requiring a total of 475 bytes. The saving in this case is a ratio of 3.96 to 1. Thus the space requirements for the character index method are four times smaller than that required by keeping the actual text. This figure will, of course, vary based on the amount of redundancy in the node information which is in turn related to the depth of the parse tree. Counting all the sentences in an article, the reduction in space complexity is overwhelming. The only drawback of this scheme is the slight increase in time complexity i.e. the overhead of locating the correct substring in the string list.

The child node list is the linkage that connects all the siblings of a node together. Its linked list implementation can accommodate a variable number of nodes. Figure 13 shows how the different kinds of links join the nodes together within a parse tree. Each node in the child node list is a record consisting of the current child and a pointer to the next child. The sibling nodes are sorted according to their text location in the article, and each one bears its own child node list.

Using these links, the algorithm shown in figure 14 can be used to search the parse tree for the highlighted text. The search is difficult since there are no delimiters used to denote the boundary of a word, a phrase, or even a clause in Japanese. When some text is highlighted, it is important to know which node is being selected. Since each node encloses a certain amount of text, the relative starting and ending position of this text can be used to compare with the selection range. In this way, the tutor can identify the selected node by the unique text enclosure range stored in the nodes.

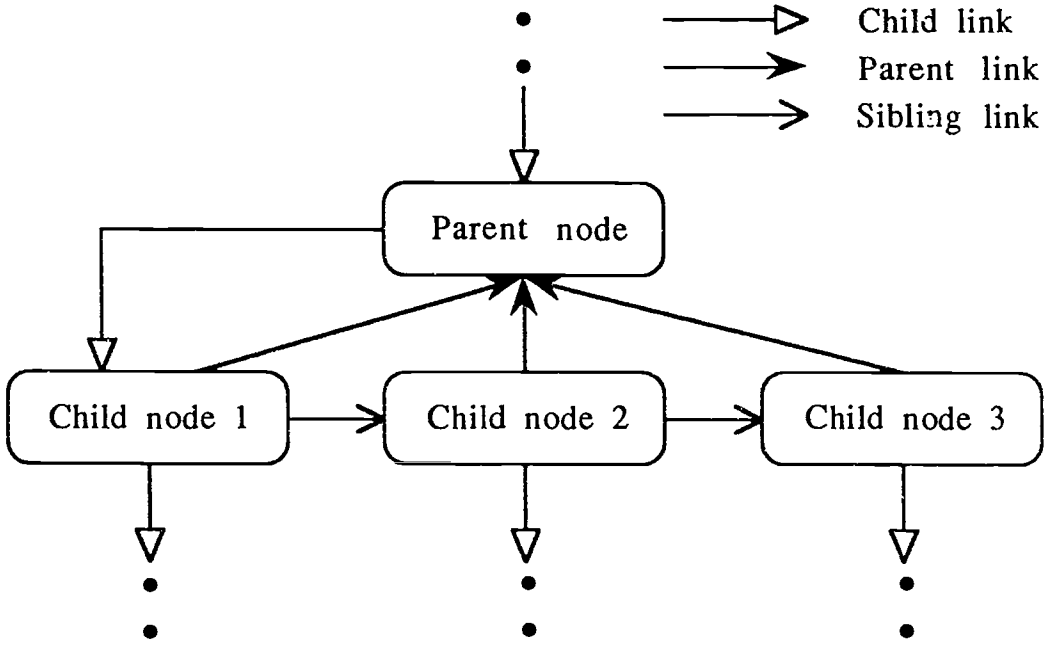


Figure 13. Connection of Nodes within a Parse Tree

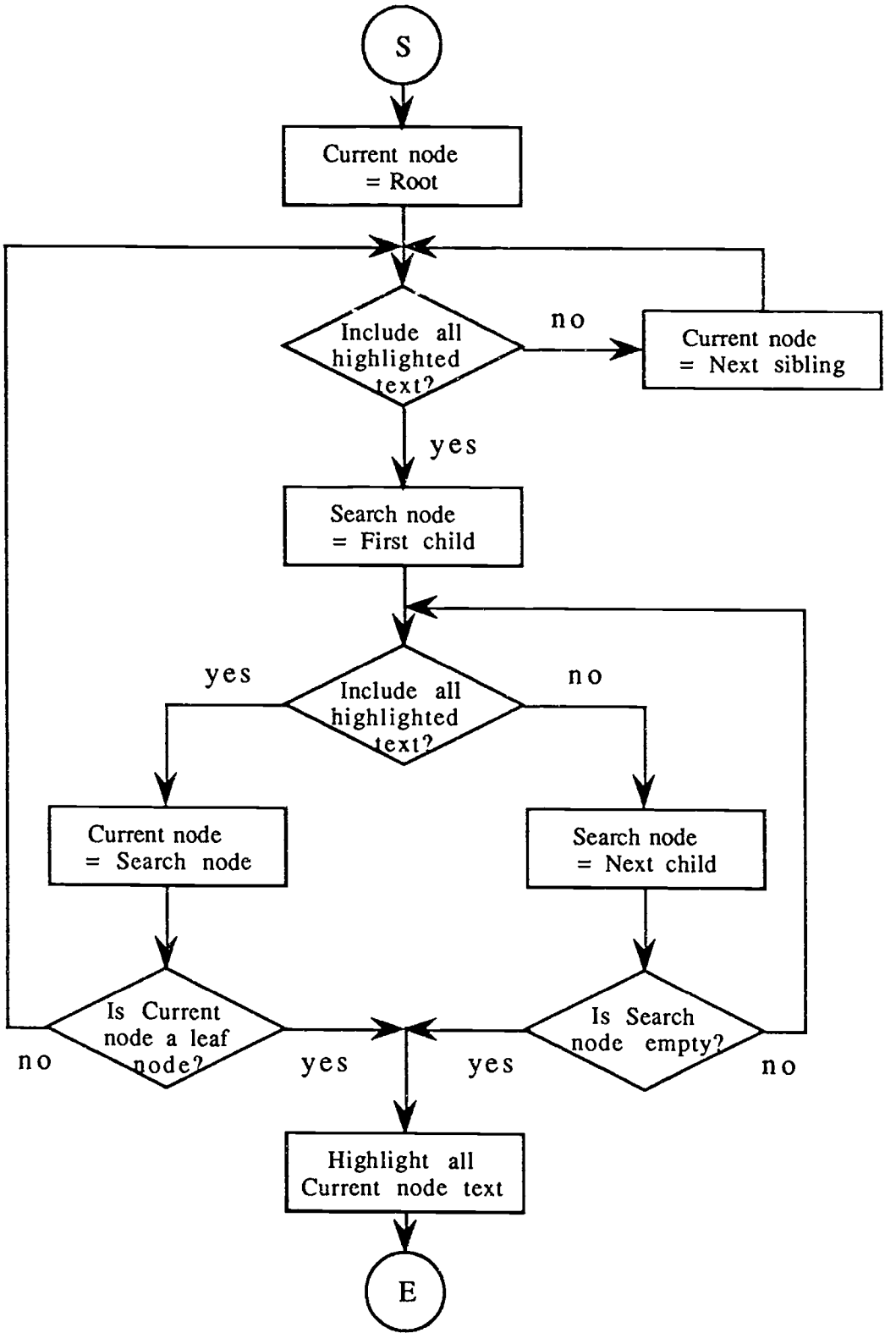


Figure 14. Algorithm for Searching through the Parse Tree

The algorithm starts searching from the root node of the parse tree. The "current node" is used to examine sibling nodes whereas the "search node" is for inspecting child nodes. For each level within a parse tree, the selection range is compared with the text location of the "current node." If the latter completely includes the first one, it proceeds to examine its child nodes. Otherwise, the "current node" will be set to be its next sibling for continual searching. For each "current node", however, it is necessary to check whether any one of its child nodes entirely encompasses the selection range. Hence, the "search node" begins with the first child node, and so on. If such a child node is found, it will be assigned to be the new "current node" and searching is started from it again. If this fails or the "current node" is already a leaf node, searching terminates with its text location highlighted. As a consequence, this is the node in the lowest level of the parse tree that includes the selected text and has correct lexical boundaries.

The notion of a child node list can be extended to construct the sentence list. Each node in the sentence list bears the parse tree associated with the sentence as its only child (see figure 15).

Syntactical Analysis

The tutor provides a utility, called syntactical analysis, used to assist the student in learning about the grammatical rules of Japanese. This is accomplished by showing how a clause or a sentence is split into its basic constituents. With the help of this kind of breakdown, the student can understand more about the syntactical structure of a Japanese phrase, in addition to its overall meaning.

Figure 16 presents an example of syntactical analysis. The task is achieved by moving upward along the parse tree through the parent link. Going to the parent of a node automatically includes all the sibling nodes. To show how several words or phrases are

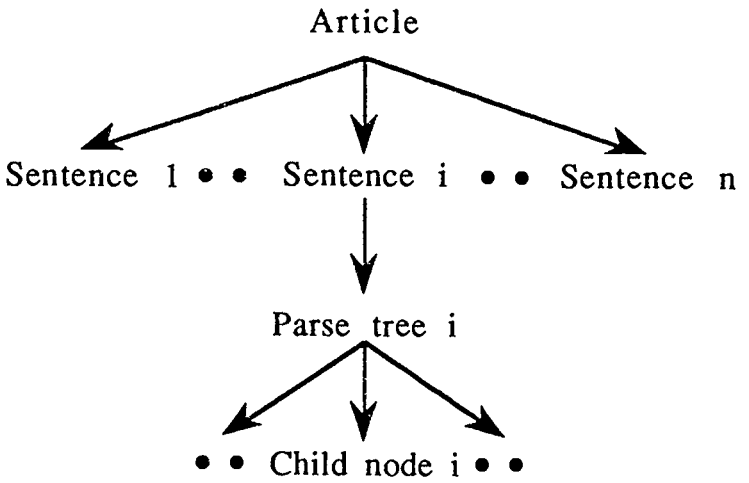


Figure 15. Connection of Nodes within the Sentence Tree

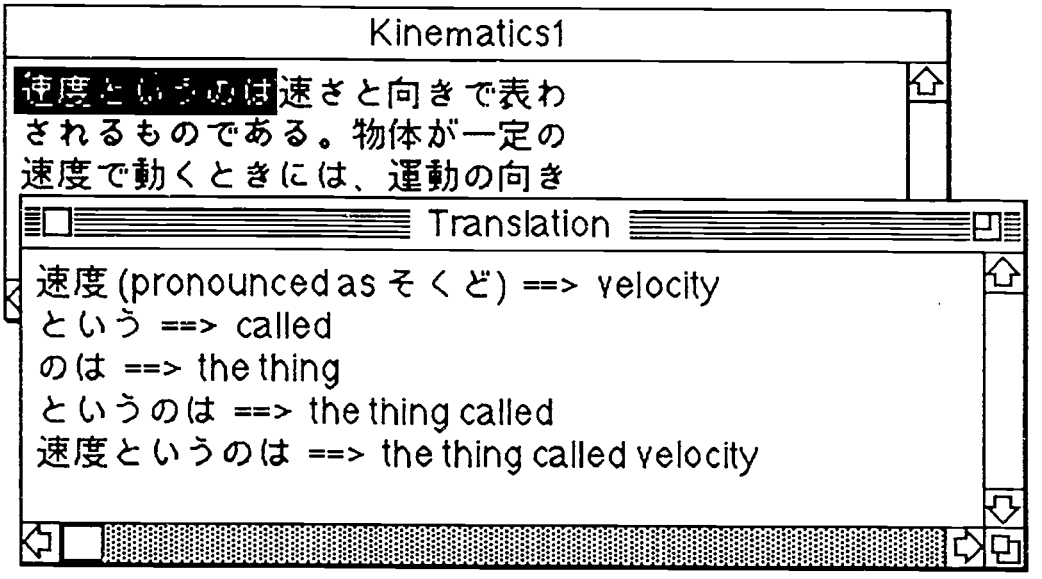


Figure 16. Example of a Syntactical Analysis

combined together to form a clause, the tutor uses a double-click facility together with the Macintosh mouse. If a double-click occurs when the mouse pointer is on some highlighted text, the selection range will be changed to that of the parent node. This happens because a node in the upper level of the parse tree is being selected. Similar double-clicks will move the selection upwards until the root node is reached. If syntactical analysis is requested, the translation information of all the descendant nodes will be displayed. This is designed to enhance the student's comprehension of word or clause boundaries in a Japanese sentence.

3.2 Student Database Management

To monitor the student's progress in terms of Japanese proficiency, the tutor records performance information which is used to aid the administration application in defining the student's competence. Such information includes material that the student knows well, as well as those items not currently mastered. This information is in terms of Japanese words, phrases, and general grammatical structures. Each time the student highlights some Japanese text and requests translation information, the tutor updates his personal database so that he will be presented with articles appropriate for his level of competence in the next lesson.

One of the difficulties in building a model of a student's language proficiency is evaluating why a student has asked for a particular piece of information. There must be some scheme used to evaluate the student's ability to comprehend an article. One direct way is to query the student to see if he has difficulty with a phrase whenever he requests translation information. However, this is too time-consuming and would interrupt the process of instruction in order to gather accurate data for assessment. The evaluation scheme used in the tutor is to classify text materials in an article into two areas: well-known and unfamiliar. In order to classify material into either the well-known or unfamiliar area, the tutor makes the following assumptions:

(1) The student requests translation information for some Japanese text if and only if he does not understand it. As a result, this material will fall into the unfamiliar category.

(2) The student knows some Japanese text well if he does not ask for a translation. This material should thus be grouped into the well-known area. This is a corollary of the first assumption.

Clearly, these assumptions can be violated in practice; however, they represent reasonable approximations and result in an efficient implementation.

Table 4 shows the contents of the records stored in the well-known and unfamiliar areas. A record includes at least a Japanese word or compound, and all of its associated readings. Since a Kanji may have various readings in different contexts, it is important that these readings are recorded to reflect what readings the student currently knows. Whenever he learns new readings of a Kanji, they are added to update his database. Each entry in the unfamiliar area includes a frequency counter which is a measure of the difficulty that the student is having with understanding that entry. A large value for the counter indicates a high level of difficulty while a small value suggests a low level. The frequency counter is initially set to an upper-bound value for each inserted entry.

Evaluation Scheme

In practice, there is not a clear-cut division between the two areas, and they are not independent of each other. For instance, when a student is introduced to new vocabulary items they will initially be entered in his unfamiliar area. For the next few lessons which include the same vocabulary, if the student does not ask for their translation, the tutor will transfer these items from the unfamiliar area to the well-known area. By the same token, any material in the well-known area is moved to the unfamiliar area if the student requests information about it. Hence the evaluation scheme must allow this kind of transferring activity to accommodate changes in the student's Japanese competence which result from both learning and forgetting.

In the tutorial system, there are two procedures to implement the transfer of data to and from the well-known and unfamiliar

Table 4. Attributes of the Record Stored in the Student Database

Area	Attributes		
Unfamiliar	Kanji character	Multiple Readings	Frequency Counter
	Compound	Reading	Frequency Counter
Well-known	Kanji character	Multiple Readings	
	Compound	Reading	

areas. Scheme A adds, updates, or transfers entries to the unfamiliar area whenever the student asks for a translation. Any text for which the student does not request information is added or transferred to the well-known area by Scheme B when he finishes reading the lesson. Thus Scheme A is performed while the student is reading the article whereas Scheme B is undertaken after the student finishes the lesson. As mentioned previously, there is no guarantee that text inserted by Scheme B is necessarily well-known by the student. However, putting text in the well-known area does not give rise to a problem. If the text really should fall into the other area, Scheme A can provide a remedy by moving it to the unfamiliar area. The two schemes work together to resolve conflicts so that the same Japanese text in different lessons will not be put into different areas.

Based on Scheme A (see figure 17), the following procedure will be performed when the student highlights some Japanese text in order to request translation information:

(1) If the text is in the unfamiliar area, its frequency counter is reset to the upper-bound value. The associated reading will be inserted if it is not included in the record.

(2) If the text is in the well-known area, it will be transferred from the well-known area to the unfamiliar area. The frequency counter is set to the upper-bound value. The associated reading will be inserted if it is not included in the record.

(3) If the text is not found in either one of the two areas, it will be inserted into the unfamiliar area as a new entry. The frequency counter is set to the upper-bound value.

Based on Scheme B (see figure 18), the following steps will be performed for any text for which the student has not requested translation information:

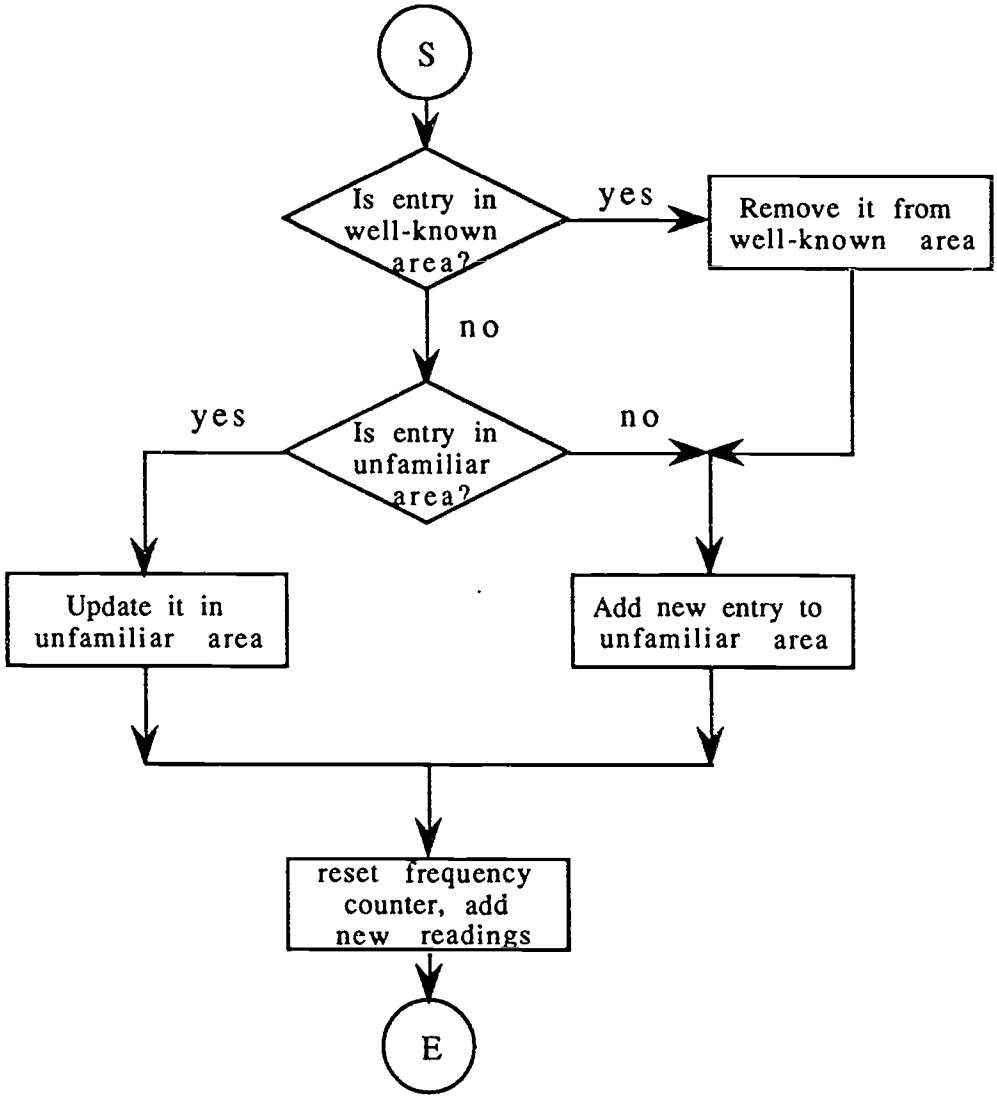


Figure 17. Evaluation Scheme A for Student Competence

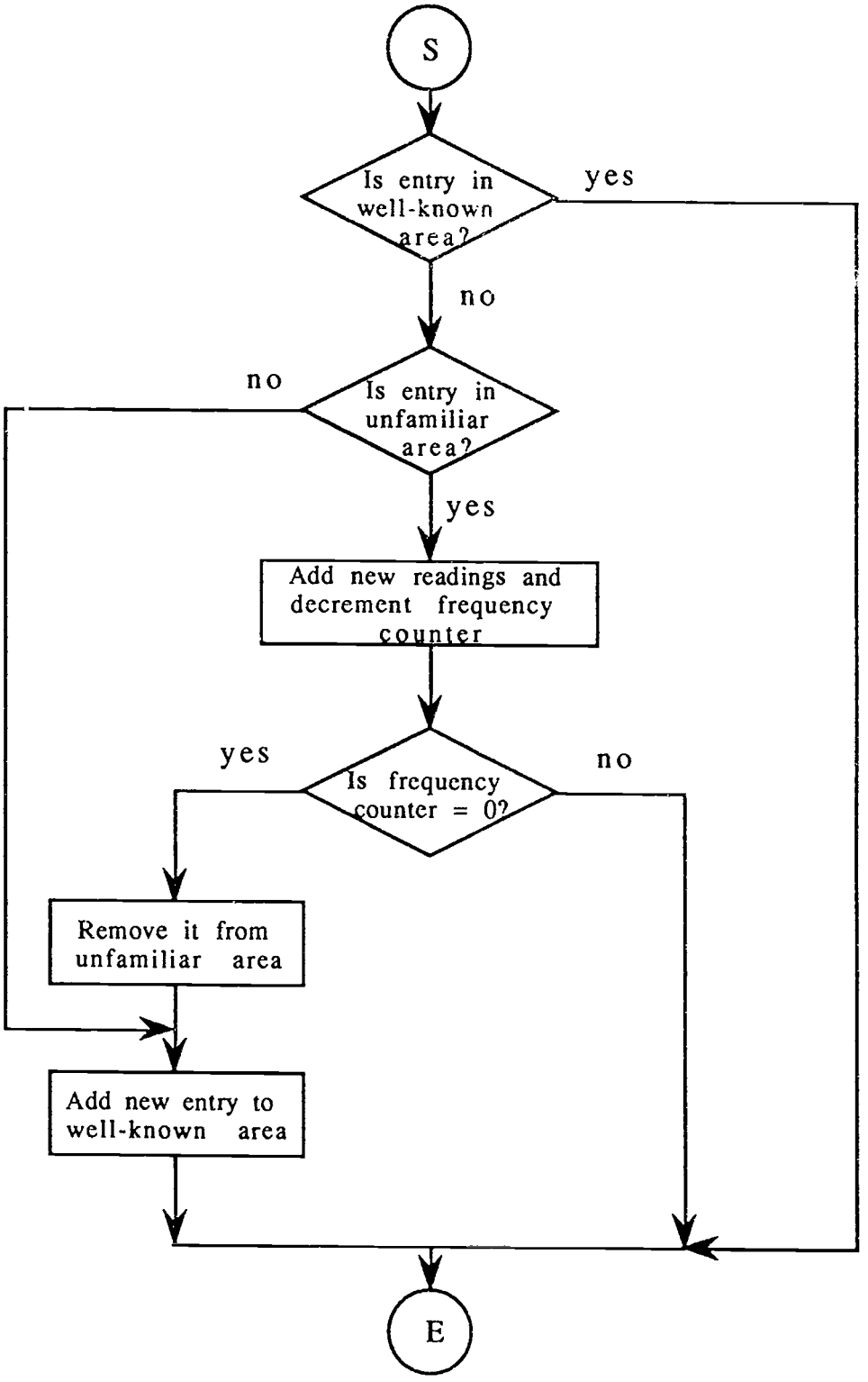


Figure 18. Evaluation Scheme B for Student Competence

(1) If the text is in the unfamiliar area, its frequency counter will be decremented. If the resulting value is equal to zero, it will be transferred from the unfamiliar area to the well-known area.

(2) If the text is not found in either of the two areas, it will be inserted into the well-known area as a new entry.

The two areas in the student's personal database are initially empty. Each time the student reads an article, his database is updated and new entries are inserted. The number of total entries increases with the two areas frequently interacting with each other in response to the student's performance.

Data Structures

The well-known and unfamiliar areas are implemented as "string list" structures in the tutorial system. Each entry in an area contains Japanese text, with all the associated readings, and the frequency counter if it belongs to the unfamiliar area. Thus the student's database consists of records storing Japanese text entries. A hash table is used to store the records when they are loaded into the system. The hash table is a sequential array of buckets where the records are mapped via a hash function. The hash function f(X), shown in equation 4, is obtained by using the modulo (mod) operator.

$$f(X) = X \bmod b \qquad (4)$$

$$X = \Sigma \text{ JIS of each Japanese character} \qquad (5)$$

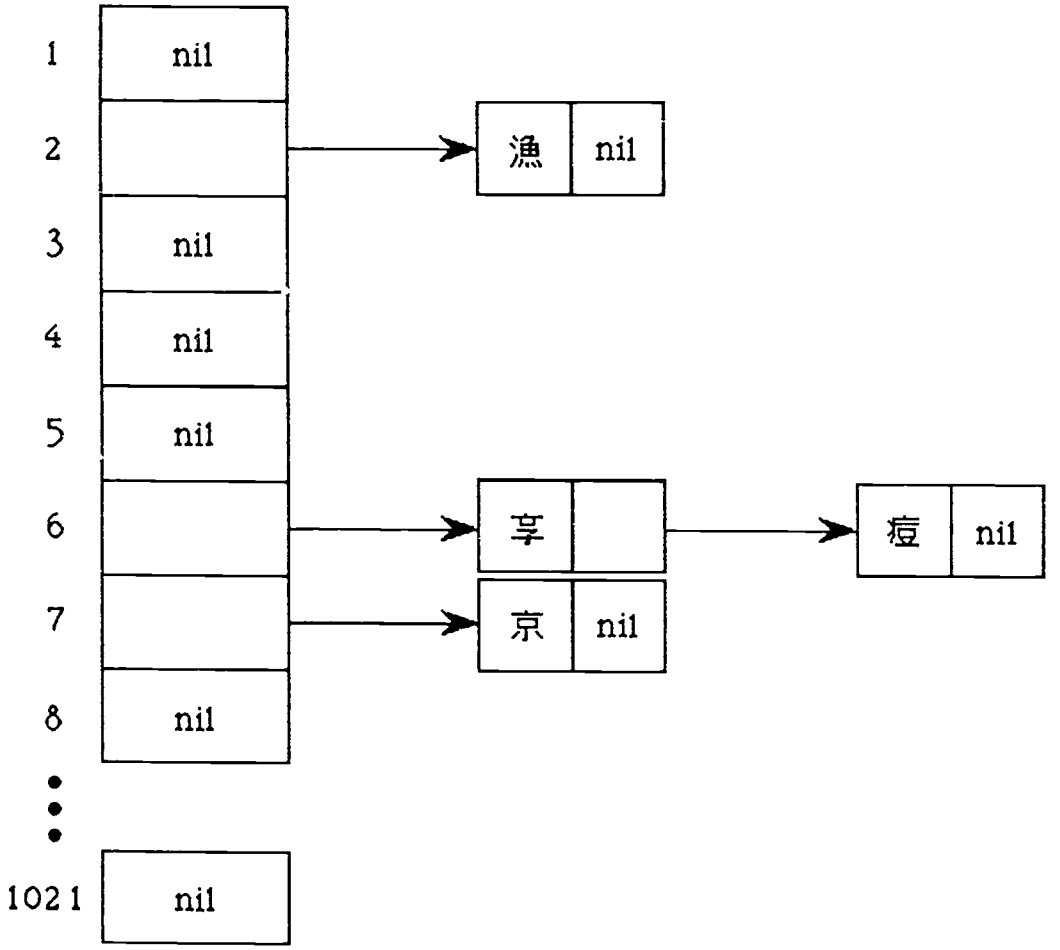
This function gives bucket addresses in the range of 0 to (b-1) and so the hash table is at least of size b. In the tutorial system, b is selected to be 1021, a prime number that makes the hash function uniform. This prevents a biased use of the hash table for random inputs, i.e. if X is the key identifier of a record chosen at random and the probability that f(X) = i is 1/b for all buckets i, then f is a

uniform hash function. Thus a random record X has an equal chance of being hashed into any of the b buckets.

Suppose X is the key identifier of the record. If the key is a compound of several characters, the JIS code for each character is summed together as the value for X (see equation 5). There is a possibility that the hash function may map several different key identifiers into the same bucket. In this case, a collision is said to have occurred and these identifiers are called synonyms. In order to be able to detect and handle collisions, the buckets are initialized to be the head pointer of an empty linked list. This linked list holds all the synonyms for that bucket in the form of a chain. The hash function computes the value of X for a record and returns the bucket address of a head pointer. Updating is done with this headed chain and new records are added at the front. Figure 19 illustrates the contents of a hash table when several Kanji are hashed into it. Using chaining to resolve collisions makes the hash table capable of holding a large number of records while reducing the retrieval time.

Time Complexity Analysis

The expected number of key identifier comparisons can be shown to be $1 + \alpha/2$, where α is the loading density n/b and n is the number of records to be hashed. The time to compute the bucket address is constant O(1) and, assuming uniform distribution, each chain holds on average n/b records. Half of these records will be examined while searching; thus, the total expected search time is $O(1 + \alpha/2)$. With one thousand records, this figure is 18. If there are ten thousand records, the figure becomes 5.9. Assuming that a student's database includes several thousand records, the hash table data structure wili always reduce the search time to less than ten comparisons.



Hash table with 1021 buckets; each bucket can hold a chain.
Hash Function: f(X) = X mod 1021 where X = Σ JIS(keyword)

Hash sequence	Keyword	X = JIS of Keyword	f(X) = X mod 1021
1	漁	35737	2
2	痘	37783	6
3	享	35741	6
4	京	35742	7

Figure 19. A Hash Table Using Chaining to Resolve Collisions

3.3 Electronic Dictionaries

The goals of the electronic dictionary can be summarized as follows:

(1) To provide the student with an understanding of the meanings, readings and characteristics (stroke count, radical number, Nelson's dictionary reference number) of Kanji and Kanji compounds.

(2) To enable the student to look up Kanji characters and compounds with high speed and different search schemes.

(3) To facilitate a cross-reference between the translation in the article and the definitions in the dictionary.

The Macintosh User Interface Toolbox offers a powerful function called Munger which can efficiently search through a large amount of data for a specific target string. This utility is described in the chapter on Toolbox Utilities of Inside Macintosh, Volume I, from page 468 to page 470. Munger allows different types of byte manipulations such as finding a string in some text data, or replacing a target string with a replacement string in the destination string. With the help of this Munger function, each dictionary search based on a particular key value can be accomplished in a short period of time. All meanings and readings related to the search key are then displayed.

Character Dictionary

Each entry in the character dictionary is sorted according to its key, the Kanji character, based on the JIS code value. This allows indexed-sequential searching for the retrieval of definitions of a specific Kanji requested by the student. The information stored in each entry includes a Nelson's dictionary reference number, a radical number, a stroke count, some common meanings and the associated

readings (see figure 20). The search methods currently available for the character dictionary require one of the following seven keys: Kanji, Hiragana (on readings), Katakana (kun readings), Nelson's dictionary reference number, radical number, stroke count, or English translation.

There are three separate index files, called CDNelson, CDRadical and CDStroke, which are used to facilitate the searching of Kanji with respect to the Nelson's dictionary reference number, the radical number and the stroke count respectively. In the CDNelson file, each Kanji is only associated with its Nelson's dictionary reference number. In the CDRadical file, the Kanji are grouped under their radical numbers, and similarly under their stroke counts in the CDStroke file (see figure 21). For example, if the student wants to search for the Kanji with radical number 140, the tutor, instead of examining the entire character dictionary, will display all those Kanji in the CDRadical file under the radical number 140.

Kanji Compound Dictionary

The general dictionary consists of common Kanji compounds, including their meanings and the associated pronunciations. The search key is either the entire Kanji compound or one of its individual Kanji. If the latter is provided for searching, all relevant compounds are retrieved so that a cross-reference can be obtained. The search methods currently available for the compound dictionary require one of the following keys: Kanji, Hiragana (on readings), Katakana (kun readings), or English translation.

Specialty Dictionary

The specialty dictionary is a group of special lexicons containing terms devoted to a specific technical discipline. In the current implementation, there is a communications dictionary with

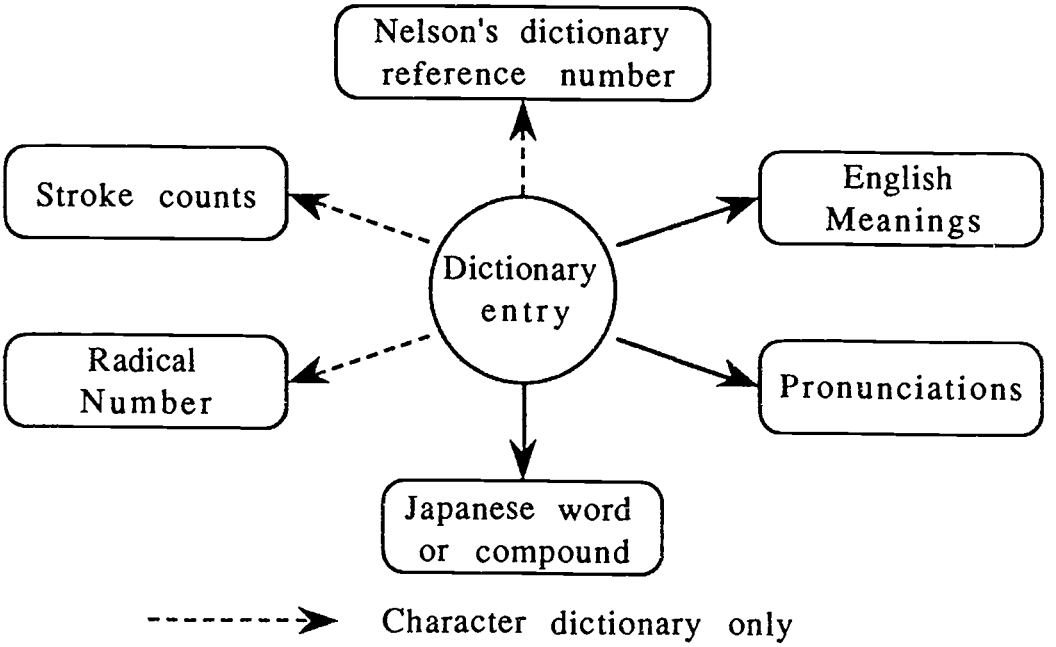
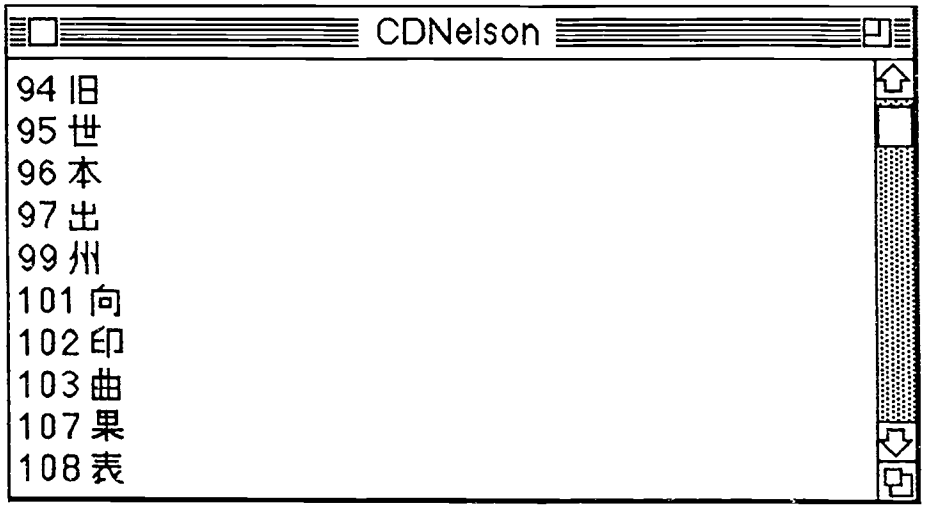


Figure 20. The Attributes of a Dictionary Entry



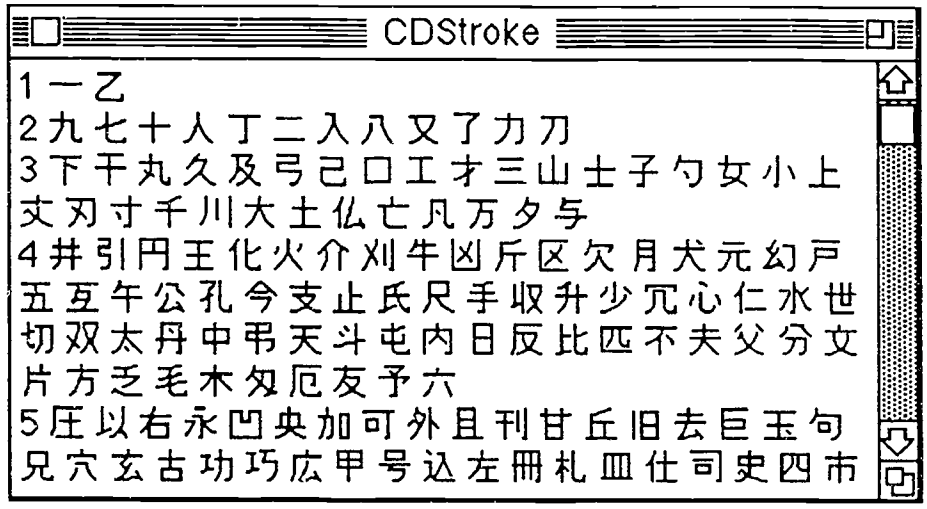


Figure 21. The Contents of the Index Files

8,889 entries and their associated meanings. The search key is either the entire term or one of its characters. If the latter is provided for searching, all relevant entries are retrieved so that a cross-reference can be obtained. The search methods currently available for the specialty dictionary require one of the following keys: Kanji, Hiragana (on readings), Katakana (kun readings), or English translation.

Grammar Dictionary

The grammar dictionary has an English translation, a description of the grammatical structure and one or more examples for each entry. Each example contains a Japanese sentence depicting the usage of the entry, followed by a sentence showing the English translation for that Japanese sentence. The search key of the dictionary is the entire grammatical structure. The search methods currently available for the grammar dictionary require one of the following keys: Kanji, Hiragana (on readings), Katakana (kun readings), or English translation.

CHAPTER IV - THE PARSE TREE EDITOR

The Parse Tree Editor is used to prepare texts for use by the Nihongo Tutor. The tutor requires that each article have a translation file which keeps all of the syntactic, semantic, phonetic, and lexical information for each sentence. Manually configuring this kind of translation file is very inefficient and subject to typographical errors. It is even more difficult to edit a translation file if the parse tree of a sentence has to be adjusted. Thus, the Parse Tree Editor is used to automate the creation and modification of the translation file.

There are four stages in the processing of the parse tree — the segmentation stage, the syntactic stage, the semantic stage, and the mapping stage. These stages define the boundary of a sentence, configure the parse tree structure, incorporate the syntactic, semantic, phonetic, and possibly morphological information into the parse tree, and finally output all the data into an internal format recognizable by the tutor.

4.1 Segmentation Stage

The first stage of forming a parse tree is to define the boundary of a sentence in the article. This is accomplished by highlighting the sentence under consideration in the text window. The relative starting and ending positions of the sentence are revealed by accessing the selected text in the internal edit record. The ending position is usually signified by the presence of a maru (the Japanese period). However, there are cases when text is represented as individual sentences without having a sentence marker (i.e. maru), such as the title of the article, the list of keywords, etc. As a result, the parse tree editor requires that the user provide the correct sentence boundary instead of automating the sentence segmentation process.

The highlighted text of the sentence must be contiguous i.e. all the words and punctuation marks have to be included. The sentences must also be processed sequentially — the first sentence being processed first, followed by the second sentence, and so on. Since the Nihongo Tutor is not matching actual words but is instead identifying each character by its relative position in the article, special characters such as blanks or carriage returns cannot be totally ignored. Thus, it is essential to include these special characters into the boundary of a sentence if they exist at either end of the sentence. In summary, there cannot be any undefined characters between or within sentences.

During the segmentation stage, there is a new node created in the sentence list for the current sentence that is being processed. This node, with its parse tree initially empty, is appended to the end of the sentence list based on its relative boundary in the article (the internal data structure used by the Nihongo Tutor is repeated in figure 22). The user is requested to furnish the translation of the entire sentence so as to give the partial content of the English actual text attribute of the node. The Japanese text of the sentence is

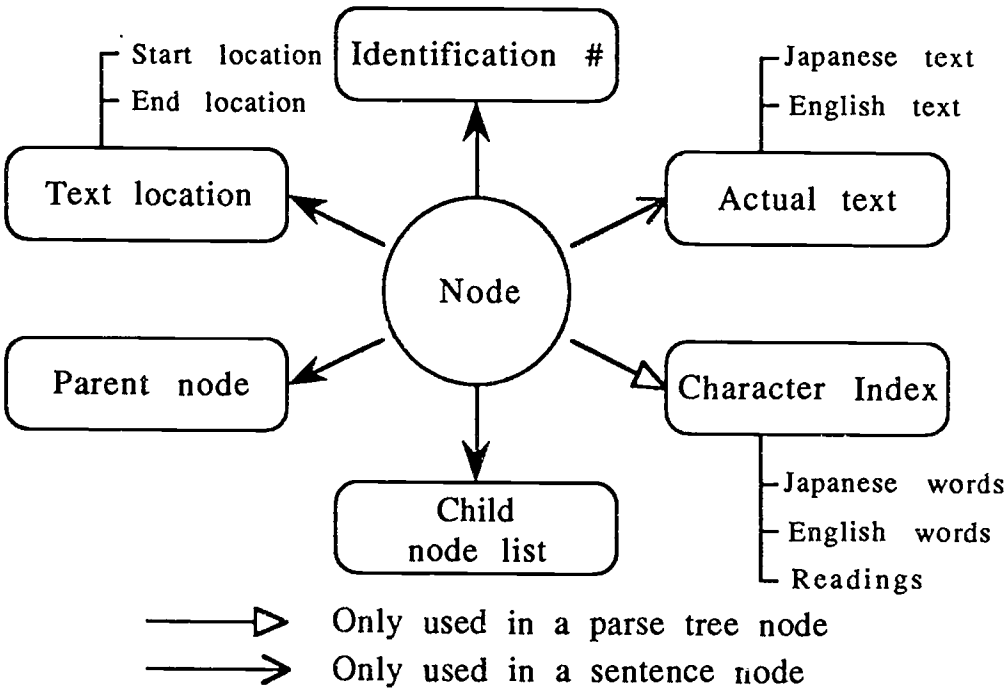


Figure 22. The Attributes Associated with a Node

extracted from the text currently highlighted. As a result, at the end of this stage, the sentence node acquires the content of its sentence identification number (sequentially assigned to each sentence), its text location and its actual text attributes.

4.2 Syntactic Stage

The second stage of building a parse tree employs a bottom-up strategy to construct the parse tree. This stage can be divided into two steps — defining the leaf nodes and configuring the parse tree from the leaf nodes. The leaf nodes are defined by highlighting the text of each node, one by one, within the sentence boundary as defined by the segmentation stage. All characters must be selected. The leaf nodes form a basis for the construction of the parse tree.

The data structures used in the Parse Tree Editor conform with those in the Nihongo Tutor. However, there is an additional attribute called node location in the data structure of a parse tree node. In the graphic window, each parse tree node is displayed as a rectangular box with its Japanese text enclosed and lines drawn to show the links between nodes. This node location field, specifying the node location in the graphic window, helps the user to visualize the configuration of the parse tree. This attribute, however, is not saved in the translation file since it is not required by the Nihongo tutor. Each node location is composed of three fields: LocationR, Highlighted, and Connected. The first field specifies the location of the node's rectangular box relative to the origin of the graphic window (which is initially set to be the upper left hand corner) while the other two fields are Boolean flags which will be set if the node is selected or connected to its parent node respectively.

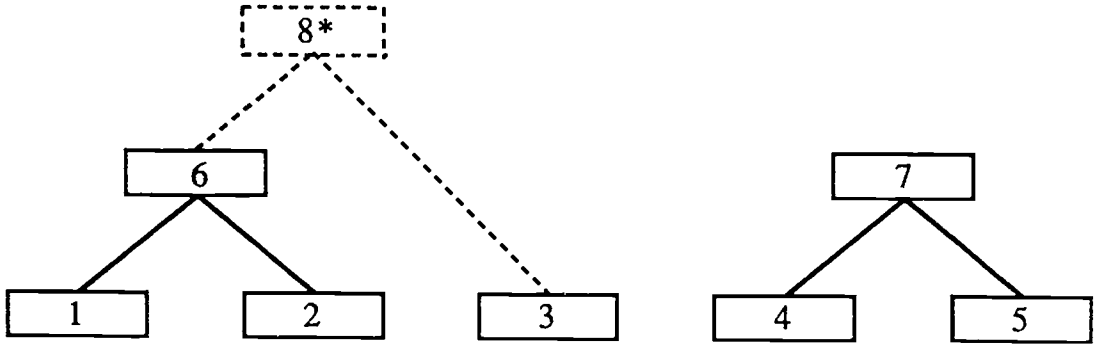
Each leaf node created in the first step is assigned a node identification number, a text location index, and a Japanese character index. The English translation text and Japanese readings for the node are filled in during the semantic stage. All of the leaf nodes are aligned on the left hand side of the graphic window in order of their node identification number. The highlighted and connected fields are initially set to false. Whenever a click occurs within the rectangular box of a node and if this node is not yet connected, it becomes selected (the highlighted flag is set) and its Japanese text is displayed

in inverse-video. Another click on a selected node will undo the selection.

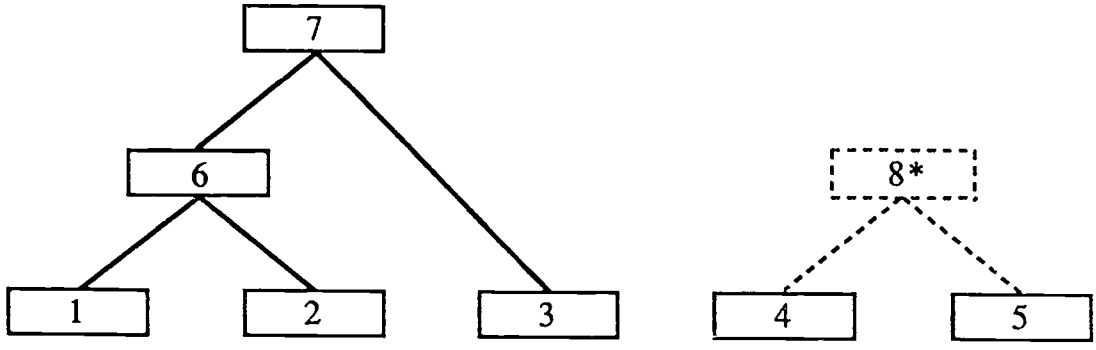
The second step of configuring the parse tree is to create parent nodes by joining the leaf nodes together. New nodes are created until the entire tree is completed. For instance, when the user wants to create a new parent node, the child nodes are selected and then, performing the "Join Nodes" operation, the system draws lines between these nodes to show the new relationship. All the selected nodes are included into the child node list of their parent node, and they are sorted according to their text location. The parent link of each child node is connected to this parent node and its connected field is thus set. The rectangular box of the parent node is placed to the right of its rightmost child in the graphic window, and to the middle of its topmost and lowest children. Any node already connected cannot be re-selected to be a possible child for a new parent node. Not only does this ensure a unique parent for each node but it also prevents the possible occurrence of a cycle.

Two strategies can be used in joining nodes to build the parse tree — level-by-level and subtree-by-subtree (see figure 23). The level-by-level method is to build all new nodes in one level at a time, working up to the root. The subtree-by-subtree method is to create all nodes in a branch at one time and then constructing the next branch until the root can combine all its branches. As shown in figure 23, this corresponds to either a breadth-first or depth-first sequence. Ordering of the parse tree nodes is not important since they rely on pointers for maintaining connection information. Thus, the two methods work equally well to configure a parse tree. When the parse tree configuration is completed, all nodes must be connected — there cannot be any disconnected components.

The configuration of the parse tree built in the syntactic stage can be changed in the "Manual" mode. For example, the links between nodes can be adjusted so that a node will have a new parent node or one of its child nodes removed. Figure 24 shows how



(a) Level-by-Level



(b) Subtree-by-Subtree

* Dashed line shows the next node to be created.
The number shown on each box is the node id.

Figure 23. The Strategies of the "Join Nodes" Operation

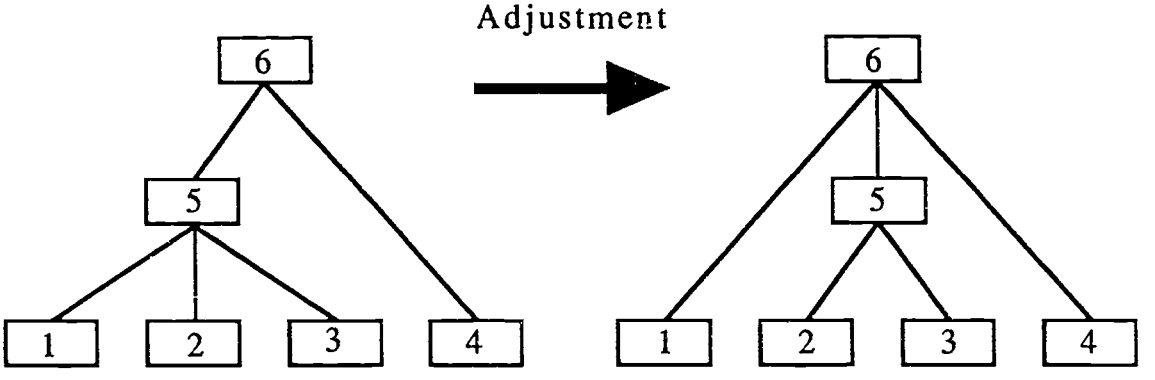


Figure 24. An Example of Adjustments to a Parse Tree Configuration

adjustments can be made to change the configuration of a parse tree. The change in the figure is accomplished in the following steps:

(1) change the parent node of node 1 from node 5 to node 6 (the root)
(2) remove node 1 from the child node list of node 5
(3) add node 1 to the child node list of node 6

As the links between nodes are specified by node numbers, the "Manual" mode allows easy editing of the parse tree configuration. Should there be any changes to a parse tree, the user must make sure that it is properly re-configured. The graphic window reflects the new parse tree configuration after any adjustment.

When the Parse Tree Editor loads in an article with an existing translation file, the node location field is automatically appended to the data structure for further editing of the parse tree. If a double click occurs within the rectangular box of a node in the graphic window, a dialog box is displayed to show all the information stored in this node. Changes can then be made if the user realizes that the contents of some attributes are not correct.

4.3 Semantic Stage

While the syntactic stage defines the sentence structure, the semantic stage specifies the meanings and readings associated with each of the parse tree nodes. The readings are required only if the Japanese text of a leaf node has at least one Kanji character. The semantic stage uses a top-down approach to process the nodes in the parse tree. It starts with the root, followed by its first child, second child, and so on until all nodes in one level are processed. Then the nodes in the next level down the parse tree are prepared and so on to the leaf nodes. This approach removes redundancy since it is probable that the translations of a node are included in those of one of its ancestor nodes. Therefore, extra storage space can be eliminated if nodes in the top level of the parse tree are processed first and then referenced by those in the lower levels.

The user is prompted to provide the semantic information of each node which includes the English translation and pronunciation of its Japanese text. Under the "Auto Setup" mode, the pronunciation is not required unless the node is a leaf node and its Japanese text contains one Kanji character. The text characters are stored into the actual text attribute only if they are not already included in it. New characters are appended to the end of the text string as the attribute is updated. Hence, each parse tree node contains a character index attribute with the actual text attribute of the sentence node containing all of the actual text characters. Adjustments to the semantic contents of a node can be performed under the "Manual" mode. When this stage is completed, all of the sentence information is incorporated in the augmented parse tree.

4.4 Mapping Stage

The last stage of the process is to transform the sentence information into an internal representation to be stored in a translation file. Figure 25 shows the format of a translation file. The translation file of an article is in a list format. It starts with a list of sentence information followed by a lexeme list. The lexemes are the Japanese text of the leaf nodes in each sentence. The sentence information includes data about the sentence and its parse tree. The sentence data is composed of a sentence identification number, its text location, string lists of Japanese characters, English translation text and readings, and finally a list of parse tree nodes. The data for a parse tree node contains a node identification number, its text location, three character indices (one each for the Japanese text, English translation, and readings), a child node list, and then the parent node.

Figure 26 summarizes the contents of a translation file. The end of a list is signified by a sentinel value implemented as "-1" in the system. This value is used in the sentence list, the string list, and the child node list. For example, the text information of a sentence is stored in the form of a string list. Each member of the string list consists of some text followed by a text delimiter which is a double quote (") and then the string number ("-1" for the last string). The child node list is a list of node identification numbers followed by "-1". If there is only the sentinel value, the child node list is empty and the node is thus a leaf node in the parse tree.

The lexemes are also contained in a string list similar to that of the text information. However, each member of the lexeme list is headed by the number of lexemes in the string, and the lexemes (in the form of Japanese text) are separated by string delimiters which are semi-colons (;). The entire list of lexemes is separated from the sentence information by a list delimiter which is a double occurrence of double quote (""). The number of strings in the lexeme list is equal

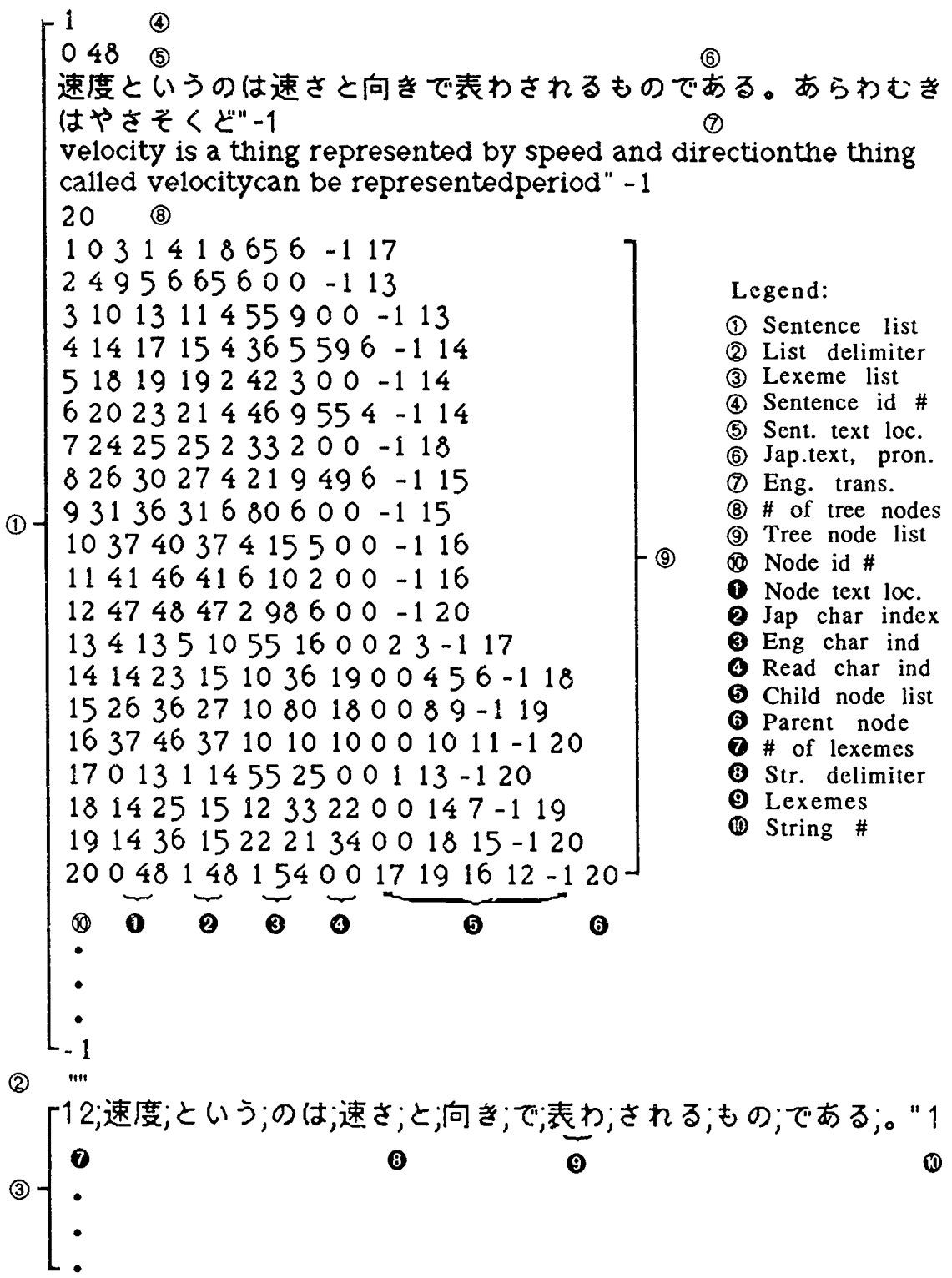


Figure 25. The Contents of a Translation File

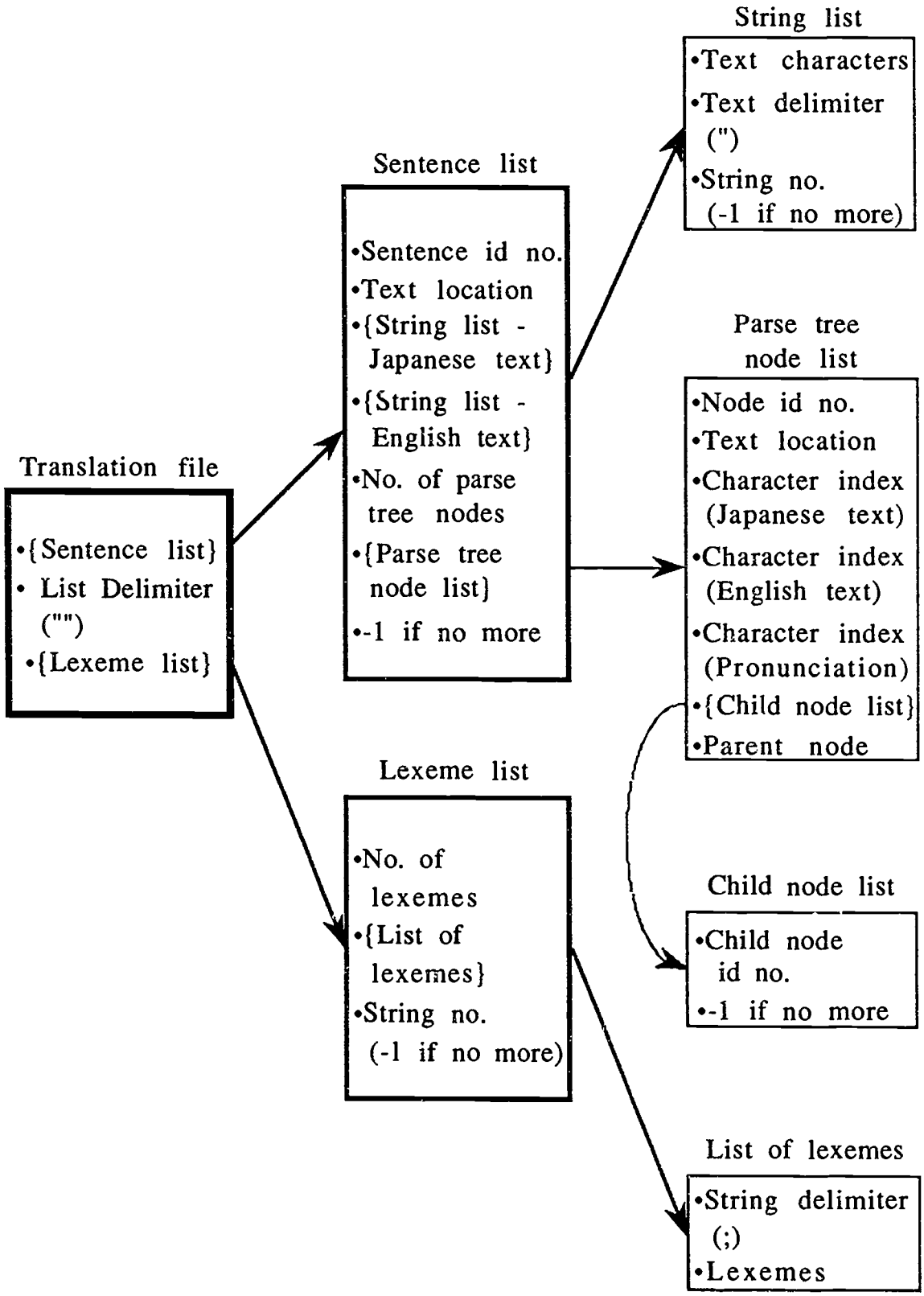


Figure 26. The Format of a Translation File

to the number of sentences in the article. All of this translation data is saved in a translation file with the same file name as the article along with the special extension ".tr".


Purdue University