AN ATTEMPT TO COMPUTERIZED DICTIONARY DATA BASES

M. Nagao, J. Tsujii, Y. Ueda, M. Takiyama
Department of Electrical Engineering
Kyoto University
Sakyo, Kyoto, 606, JAPAN

Summary

Two dictionary data base systems developed at Kyoto University are presented in this paper. One is the system for a Japanese dictionary (Shinmeikai Kokugojiten, published by Sansei-do) and the other is for an English-Japanese dictionary (New Concise English-Japanese Dictionary, also published by Sansei-do). Both are medium size dictionaries which contain about 60,000 lexical items. The topics discussed in this paper are divided into two sub-topics. The first topic is about data translation problem of large, unformatted linguistic data. Up to now, no serious attempts have been made to this problem, though several systems have been proposed to translate data in a certain format into another. A universal data translator/verifier, called DTV, has been developed and used for data translation of the two dictionaries. The detailed construction of DTV will be given. The other sub-topic is about the problem of data organization which is appropriate for dictionaries. It is emphasized that the distinction between 'external structures' and 'internal structures' is important in a dictionary system. Though the external structures can be easily managed by general DBMS's, the internal (or linguistic) structures cannot be well managed. Some additional, linguistic oriented operations should be incorporated in dictionary data base systems with universal DBMS operations. Some examples of applications of the dictionary systems will also be given.

2. Data Translation from Printed Image to Formatted Data

We decided to input the dictionary contents almost as they are printed, and to translate them into certain formatted structures by computer programs rather than by hand.

Ordinary dictionaries usually contain varieties of information. The description in English-Japanese dictionary, for example, consists of

1. parts of speech
2. inflection forms
3. pronunciations
4. derivatives
5. compounds
6. translation equivalents in Japanese
7. idioms and their translations
8. typical usages and their translations
9. antonyms and synonyms

An entry may have several different parts of speech (homograms) and to each part of speech, the other information 2-9 is described (even pronunciation may change depending on the parts of speech). 7, 8 and 9 are usually attached to one of the translation equivalents (see Fig. 1).

In such a way, the description for a dictionary entry has a certain structure, and the several parts of the dictionary descriptions are related to each other. In the printed dictionaries, these relationships are expressed implicitly in linearized forms. Various ingenious conventions
The printed version of a dictionary highly relies on human ability of intuitive understanding, and consists of many uncertain conventions. Gothic characters, for example, indicate that the phrases printed by them are idioms, and italic characters show that the entry words have foreign origins. In the input texts for computer, these different types of characters are indicated by a set of shift codes. Shift codes, together with various special symbols such as ( , ), [ , ] etc., give us useful clues for the data translation. However, these codes should be interpreted differently, when they are used in different parts of descriptions. "( " shows the beginning of the pronunciation when it appears just after a head word, and, on the other hand, when it is used in the midst of an idiomatic expression, it shows the beginning of an alternative expression of just the preceding one. Such conventions, however, have many exceptions. Moreover, the fact that there may be errors in the input texts makes the translation process more difficult.

If we use an ordinary programming language like PL/I, the program for this purpose becomes a collection of tricky mechanisms and hard to debug. Data translation of this kind is inevitable whenever we want to process unformatted linguistic data by computer. It would be very useful if we could develop a universal system for data translation (in fact, our system described below has been used not only for dictionary data translations but also for the translations of bibliographic data in ethology at the National Museum of Ethnology).

2-1. Data Translator/Verifier — DTV

The data translation can be seen as a translation from linearized character strings to certain organized structures. The relationships implicitly expressed in the linearized strings should be recovered and explicitly represented in the organized forms. It is basically a process of parsing. It has many similarities with parsing of sentences in artificial or natural languages. It has more similarities with natural language parsings in the sense that both are defined by many uncertain rules. Therefore, it is reasonable to expect that we can apply the same techniques to this problem that have been proven useful in natural language parsings. Several proposals have been made to define data syntax by using ordinary BNF notations (or CFG). However, we adopted here the framework of ATN (Augmented Transition Network) instead of CFG by the following reasons:

(1) CFG is essentially a model of a recognizer. Although it is possible to check the syntactic correctness of input texts by CFG rules, we need another component that transduces the parsed trees into formatted records we want. ATN gives us an adequate model of a data transducer. It has provisions for setting up intermediate translation results in registers (registers in ATN are called 'buffers' in our system) and building them up into a single structure (called BUILDQ operation in ATN).

(2) CFG provides an adequate framework for managing recursive structures such as embedded sentences in natural languages. Though recursive structures are also found in dictionary data, they are not so conspicuous. The structures in dictionaries are rather flat. In this sense, CFG is too powerful to define data syntax of dictionaries.

(3) ATN provides a more procedural framework than CFG. Because a CFG based system assumes a general algorithm that applies the rules to the input text, the user who defines the rules cannot control the algorithm. This is a fatal disadvantage of CFG, when the input text contains many input errors. Whenever an input error is encountered during the translation process, a CFG system fails to produce a parsed tree. The system or the human user should trace back the whole process to find the input error which causes the failure. It would be a formidable task.

2-2. Definition of Rules, Codes, Buffers and Files

Based on ATN model, we modified it for data translation. In this section, we will explain the detailed syntax for the DTV (the formal definition of the DTV syntax is given in (8)).

(A) Definition of Codes

In the case of syntactic analyses of natural language sentences, the basic units are parts
of speech of individual words or individual words themselves. Special checking functions such as CAT and WORD are prepared in the original ATN model. On the other hand, in the case of data translation, the basic units are individual characters.

A restricted set of characters such as the character set defined by ISO or ASCII are used and sufficient for ordinary computer applications. However, when we want to process real documents or linguistic data like dictionaries, we need much richer set of characters. Though, in principle, a single kind of parenthesis is sufficient for expressing tree-like structures, several different sorts of parentheses such as (, [, , C, {, ~ etc. are used to identify different parts of descriptions in the published dictionaries. We also found out that a certain set of characters, for example, phonetic symbols, appear only in a certain specific position (the pronunciation in the case of phonetic symbols) in the dictionary descriptions. If we could recognize the scope of the pronunciation parts, we would not need to have extra sets of character codes for phonetic symbols. We could interpret ordinary ASCII codes in the pronunciation part as usual alpha-numeric characters but as phonetic symbols, according to certain pre-defined rules.

However, these redundancies of descriptions are especially useful for detecting input errors. Whenever we find out the codes for phonetic symbols in the positions other than the pronunciation fields, or inversely, when we encounter, in the pronunciation fields, the codes for the characters other than phonetic symbols, something wrong would be in the input texts.

Because we have about 10,000 or more different Kanji-(Chinese) characters in Japanese, a standard code system such as ISO, ASCII etc. is no more adequate, and therefore, a special code system has been determined as JIS (Japanese Industrial Standard). The code system assigns a 2 byte code to each character. We have 752 extra codes which are not pre-defined by JIS and to which the user can assign arbitrary characters. Various types of parentheses, shift code, phonetic symbols etc. have been defined by using these extra codes. Because each character, including alpha-numeric, Kanji, specifically designed symbols, shift codes etc., correspond to a 2 byte code, we can assign a decimal number to each character by interpreting the 2 byte code as an integer representation. By using this decimal number notation, we can define arbitrary subsets of characters as shown in Fig. 2. These subsets of characters play the same role for the data translation as the syntactic categories for sentence analysis. Notice that a character is allowed to belong to more than one character set.

(B) Definition of Rules
A rule of DTV is defined by a triplet as

(condition action next-state).

The condition-part is specified by using the code sets defined in (A). Two forms of specifications are possible.

1. < subset-1, ..., subset-n >
2. ( subset-1, ..., subset-n )

The first notation means that the characters in the specified subsets should appear in this order. The second is the notation for specifying OR-conditions, that is, a character in one of the specified subsets should appear. Arbitrary combinations of these two bracketing notations are allowed such as < ( ( < > ) ) >.

The action parts, which will be carried out when the condition parts are satisfied, are described by using a set of built-in functions. These are the functions for manipulating buffers and files. Some examples of such built-in functions are shown in Table 1.

| Function | Argument | Result |
|----------|----------|--------|
| WRITE    | '*[-number] BUF(Buf-name) | the currently scanned character(or the 'number' preceding character) is written in the buffer. |
| RECNO    | BUF(Buf-name) | the ID number of the current input record is written in the buffer. |
| PTR      | BUF(Buf-name) | the position of the scanned character in the input record is written in the buffer. |
| 'arbitrary character string' BUF(Buf-name) | the specified character string is written in the buffer. |
| BUF(Buf-name) FILE(File-name) | the content of the buffer is written out to the external file. |
| MERGE BUF(Buf-name1,.., Buf-name n) BUF(Buf-name) | the contents of the n buffers are merged into a single buffer specified by the second argument. |
| CLEAR CTR(Counter-name) or BUF(Buf-name) | the counter is cleared to 0 or the buffer is cleared by blank characters. |
| ADD CTR(Counter-name) Number | the counter is counted up by the number. |

Table 1 Built-in Functions in DTV
Several actions can be specified and they will be executed in sequence.

The next-state specifies the state to which the control is to be transferred after the current rule is applied. A typical state-diagram is shown in Fig. 3.

![State-Diagram](image)

(C) Definition of Buffers and Files

We can define arbitrary numbers of buffers with various sizes as follows.

| BUF-NAME | SIZE(BYTE) | IF-OVERFLOW-STATE |
|----------|------------|--------------------|
| SPELLING | 40         | SPELL-ERROR       |
| IDiom    | 30         | IDIOM-EXPAND       |
|          |            |                    |

One of the typical input errors is the omissions of delimiters which cause serious problems in data translation. Various characters play the roles of delimiters. They are shift codes, several sorts of parentheses, etc. and they are used in pairs (right vs. left parentheses, shift-in vs. shift-out etc.) to delimit scopes of specific data fields. When one of the pair is missing, two situations would occur: the buffers corresponding to the fields may overflow or illegal characters for the fields may be scanned. The latter case can be easily detected because no transition rules are defined for that character. DTV put a message to the error message file which tells at which position the illegal character is found. The former case is rather troublesome. Checking overflow conditions by rules makes the whole definition very clumsy. We can specify in the definition of a buffer, to which state the control makes a transition if the buffer overflows. In that state, some error messages are printed out.

2-3. System Configuration

Fig. 4 shows the overall construction of DTV. By the compiling component, the definitions of codes, buffers, files, formats of input and output, and translation rules are compiled into several internal tables. Based on these tables, the executer scans the input characters one by one from the input file and applies the rules. During the execution, the system will report various error messages such as 'buffer overflow', 'illegal characters' etc. into the error message files. Because the detailed information, such as the position of the error in the input text, is associated with these messages, human proof-readers can easily recognize the input errors.

A flexible editor has been developed for correcting input errors. Because this editor has a special command to call DTV, the reviser can check the data syntax immediately after the correction (see Fig. 5).

Fig. 4 Overall Construction of DTV

Fig. 5 Data Editor Accompanied by DTV

2-4. Experience with DTV

We used DTV for data translation of the English-Japanese dictionary. About 500 rules and 150 states were necessary to manage exceptional description format of the dictionary. Because DTV should scan and check every input character...
and because the dictionary consists of 6,500,000 characters, the whole process was very time consuming (it took about 130 min. for translating the whole dictionary by FACOM M200 at Kyoto University Computing Center).

In order to show the effectiveness of DTV, Table 2 is prepared, which shows the input errors detected in the initial input. Some of them can be corrected automatically only by augmenting DTV rules. Moreover, the data editor accompanied by DTV was so effective that all of the detected errors were completely removed by 3 man-month efforts. However, DTV can check mainly the consistency of delimiting characters. There still remain a lot of input errors in the text such as errors in spellings of words. The detection of such input errors requires certain semantic knowledge and is hardly done by DTV rules. Human proofreader should do it. Human proofreaders can easily recognize these errors, but tend to overlook the errors such as omissions of delimiting characters. Certain effective co-operations between man and machine seem to be inevitable in correcting errors in a large amount of linguistic corpora like dictionaries.

Another point to discuss is the relations between DTV and data entry systems. Though our attempt here is highly batch-oriented, some considerations about interactive data entry systems will be necessary in future to augment the dictionary data in evolitional ways. An ordinary data entry system usually guides the user as to what he should input next, by printing prompting messages such as 'input the next word', 'input the part of speech of the word' etc. However, in the case that the input data have rich varieties in their description formats like the dictionary here, such a system becomes infeasible. Though some guidance by the entry system would be necessary, it is natural for the user to input data in arbitrary fashion. The data entry system should have the abilities of translating the texts into certain formatted structure, and of checking the existence of input errors. Our data editor accompanied by the DTV is the first step toward developing such data entry systems.

3. Data Base Systems for Dictionaries

A dictionary description has a certain hierarchical structure such as previously shown in Fig. 1. Such a structure can be well represented by a framework provided by ordinary DBMS's, because it is just a simple tree structure. However, the primitive data (or records) from which the whole structure is built have certain internal structures of their own. For example, idioms or typical usages in English-Japanese dictionary are the primitive records which are located at certain fixed positions in the whole structure and related to the other records such as translation equivalents, head words etc. They can be accessed as basic units through usual DBMS operations. At the same time, they are composite expressions which consist of several component words. These component words are related to each other inside the idioms. We call such structures inside the primitive records 'internal structures'. (See Fig. 6) In other words, the primitive records in a dictionary data base system are not primitive in a

* Note 1: * shows that the errors of that type can be automatically corrected only by augmenting DTV rules. ** shows that some of them can be corrected automatically by augmenting DTV rules. Note 2: The exceptional format errors are not input errors in a true sense.

Table 2 Error Frequencies in the Initial Data

| Error Type                        | Explanations                                      | Frequencies |
|----------------------------------|---------------------------------------------------|-------------|
| Settings of shift codes          | Shift-out codes (the code for normal characters) are often missing. | 792         |
| Confusions of similar characters | The phonetic symbol "y" is, for example, often confused with the number character 3. | 5434        |
| Fluctuations in character sequences | Certain functional character sequences can express a same thing. It is impossible to standardize them in the case that several key punchers work in parallel. | 1166        |
| Exceptional formats which were not expected beforehand | The description formats for acronyms, for example, are quite different from those of ordinary words. | 550         |
| Misunderstandings of key punchers | Though the key punchers consented to several standardization rules for input, some of them misunderstood them. | 1298        |
| Miscellaneous errors             |                                                   | 1276        |
| **Total**                        |                                                   | **10516**   |

Note 1: * shows that the errors of that type can be automatically corrected only by augmenting DTV rules. ** shows that some of them can be corrected automatically by augmenting DTV rules. Note 2: The exceptional format errors are not input errors in a true sense.

Table 2 Error Frequencies in the Initial Data

Fig. 6 External Structure and Internal Structure in a Dictionary Data
usual sense of DBMS. Though the external structures among primitive records can be managed by an ordinary DBMS, the internal linguistic structure, in some sense, cannot be well manipulated. Moreover, what we want to do on the dictionary data base systems is not only concerned with external structures, but also, in many cases, concerned with their internal, linguistic structures. Some additional operations should be incorporated with the usual DBMS operations for treating such intermixed structures.

3-1. Japanese Dictionary Data Base

The first thing we have to do is to incorporate morpho/graphemic level of processings. Because Japanese has a very peculiar writing method, special techniques are required to utilize the dictionary. The main difficulty we encountered in developing the dictionary consultation system is from the fact that dictionary entries of Japanese usually have more than one spelling. They have basically two different forms of spellings, Kana-spellings (spellings by Japanese phonetic symbols) and Kanji-spellings (spellings by ideographs — Chinese characters). Corresponding to these two spellings, we have two types of printed dictionaries, one for Kana and the other for Kanji spellings. However, in actual sentences, there often appear mixed forms of these two spellings. (See Fig. 7) Though these mixed spellings are not entered in the ordinary, printed dictionaries of both types, human readers are intelligent enough for converting them into one of the two basic spellings. We can obtain Kana-spellings from mixed ones, by systematically changing the Kanji-characters in the mixed spellings into corresponding Kana-strings. However, in actual sentences, there often appear mixed forms of these two spellings. (See Fig. 7)

![Fig. 7 Various Spellings of a Single Word](image)

In our system, the intermediate indexing structures are provided for both Kana and Kanji-spellings (Fig. 8). The dotted line shows the access path for Kana-spellings and the bold line is for Kanji-spellings. The relationships among FCT, FFCT, SCT and IT are illustrated in Fig. 9, and the required memory spaces for these structures are given in Table 3.

![Fig. 9 Relationships among FCT, FFCT, SCT and IT](image)

| Index Table          | Storage Requirement |
|----------------------|---------------------|
| FCT                  | 24 KB               |
| FFCT                 | 18.6 KB             |
| SCT                  | 700 KB              |
| IT                   | 4.3 MB              |

Table 3 Required Memory Space

Mixed spellings are normalized into one of these basic spellings. We can obtain Kana-spellings from mixed ones, by systematically changing the Kanji-characters in the mixed spellings into corresponding Kana-strings. However, because each Kanji-character corresponds to three or four (or more) different Kana-strings (each Kanji-character has several pronunciations), the resultant Kana-strings are to be matched against the Kana-spellings in the dictionary. Some examples of retrieval results are shown in Fig. 10.

Another problem is the incorporation of the morphological analysis component. Because the word inflection system for Japanese is much richer than English, the morphological analysis component is indispensable for the Japanese dictionary system. The morphological analysis program developed for our another project, i.e., Machine Translation Project (7), has been incorporated into the system. The retrieval program has Japanese inflection rules in it and can convert inflectional variants to their...
1) Input spelling by KANA-character

(1) Input spelling by KANA-character

KANA ニャーイ 表現→Retrieved entry

2) Input spelling by a Mixed-spelling

KANJ KANJ→input

The Kanji-character is replaced systematically by its corresponding KANA-strings.

Note: Two entries are retrieved because they have the same spelling.

2) An Example of Irregularly Inflected Words

Input word:

FORE

Japanese translation:

fore-go

2. Literal string matching is performed. In the matching process, extra words in the input and retrieved idioms or usages are ignored. Only the order of words is taken into consideration.

3. Similarity value is computed for each idioms and usages. The expressions with the highest value are printed out. In the current system, the similarity value is determined by a simple formula as

\[ \frac{\text{the number of matched words}}{\text{the number of words in idioms or typical usages}} \]

Some results are shown in Fig. 12. We should develop more sophisticated method of computing the similarity value. Especially, information
Fig. 12 Retrieval Results of Similar Expressions about semantic relationships among words should be taken into consideration. Computerized thesauri will be useful. Certain words in idioms and usages play the role of variables. 'Oneself' in such a idiom as 'be a law unto oneself' is looked as a variable and should be able to be matched with 'myself', 'himself' etc. 'Person' in 'take a person about a town' should be matched with any person such as John, he, and so on. In the latter case, a 'town' can also be replaced by many other words that have certain semantic features in common, for example, 'placehold'. We are now designing such semantically guided pattern matchings.

3-3. Data Structure for English-Japanese Dictionary Data Base

The formats of records which are obtained as the result of data translation are shown in Fig. 13. The records in these formats contain a large amount of extra spaces, because they are fixed in length and, on the other hand, the length of the descriptions in the dictionary varies very much, depending on individual words. The necessary memory size for these records amounts to 150 Mbyte. We have reorganized them for the purpose of reducing the memory size. The actual data organization is shown in Fig. 14, in which the required memory size is 35 Mbyte. All kinds of text data of variable length are maintained in the same place (the Text Data File) in this organization.

Fig. 13 Examples of Formatted Records

![Diagram of data structure](image)

NOTE: i) Numbers in boxes are numbers of CHARACTERs. (1 CHARACTER = 2 bytes)

2) For example, 'pointer to JAPANESE TRANSLATION' stands for the pair illustrated below.

Fig. 14 Data Structure of English-Japanese Dictionary

The information which is necessary for managing the records in the Text Data File is contained in a TCR. A TCR consists of a pointer to the corresponding text record, the number of occupied text data records, a tag field etc. The tag field indicates what kind of text data are stored in the record. Arbitrary numbers of text data can be linked together. The memory efficiency of this data structure is obvious. And, the average time for retrieving and displaying the result on the screen is 320 msec. About half of the time is spent on the display control (about 120 msec ~ 160 msec, depending on the data size). To access a head word record from its spelling requires only 3 msec. The remainder is spent on retrieving the other records such as pronunciation, P.O.S. idioms etc.
4. Concluding Remarks

All the systems described in this paper have been implemented on FACOM M-200 (Kyoto University Computing Center) mostly by PL/I. Because the computing center has introduced an MSS (Mass Storage System) and will begin the service this summer, these systems are to be maintained on it. Several other groups in our university, especially a research group of the Faculty of Literature, are very interested in utilizing the dictionary systems for their own researches. Up to now, a set of utility programs have been developed. We hope that such a joint effort between computer scientists and linguists will lead us to new, fruitful research areas.

Acknowledgement

We would like to thank the other members of Prof. Nagao's laboratory, and in particular, Mr. Yukinori YAMAMOTO for his efforts of implementing the first version of DTV and for his valuable suggestions for the data organization of Japanese dictionary data base.

References

(1) Fry, J.P., Frank, R.L. et.al. : A Developmental Model for Data Translation, ACM SIGFIDET Workshop on Data Description and Access, 1972
(2) Fry, J.P., Smith, D.P. et.al. : An Approach to Stored Data Definition and Translation, ACM SIGFIDET Workshop on Data Description and Access, 1972
(3) Michiels, A., Moulin, A., Mullenders, J., Noel, J. : Exploiting the Longman Computer Files for MT Lexicography and other Purposes, Technical Report, University of Liege, Belgium
(4) Michiels, A., Moulin, A., Noel, J. : Working with LDOCE, Technical Report, University of Liege, Belgium
(5) Liu, S., Heller, J. : A Record Oriented, Grammar Driven Data Translation Model, ACM SIGFIDET Workshop on Data Description, Access and Control, 1974
(6) Nagao, M., Tsujii, J.: Data Structure of a Large Japanese Dictionary and Morphological Analysis by Using It, Journal of Information Processing Society of Japan, Vol. 19, No. 6, in Japanese
(7) Nagao, M., Tsujii, J., et.al. : A Machine Translation System from Japanese into English, to be included in Proc. of this conference, 1980
(8) Ueda, Y. : A Study for English-Japanese Dictionary Data Base, BS Thesis, Kyoto University, 1980, in Japanese