Multi-Structure Information Retrieval Method Based on Transformation Invariance

Fuminori ADACHI, Takashi WASHIO,
Atsushi FUJIMOTO and Hiroshi MOTODA

ISIR, Osaka University
8-1 Mihogaoka, Ibarakishi, Osaka 567-0047 Japan

{adachi, washio, fujimoto, motoda}@ar.sanken.osaka-u.ac.jp

Hidemitsu HANAFUSA

INSS, Inc.
64 Sata, Mihamacho, Mikatagun, Fukui 919-1205 Japan

hanafusa@inss.co.jp

Received 11 April 2003

Abstract The needs of efficient and flexible information retrieval on multi-structural data stored in database and network are significantly growing. Especially, its flexibility plays one of the key roles to acquire relevant information desired by users in retrieval process. However, most of the existing approaches are dedicated to a single content and data structure respectively, e.g., relational database and natural text. In this work, we propose "Multi-Structure Information Retrieval" (MSIR) approach applicable to various types of contents and data structures by adapting a small part of the approach to data structures. The power of this approach comes from the use of the invariant feature information obtained from byte patterns in the files through some mathematical transformation. The experimental evaluation of the proposed approach for both artificial and real data indicates its high feasibility.

Keywords Information retrieval, Flexibility, Mathematical transformation, Invariance, Similarity.

§1 Introduction

The recent progress of information technology increases the variety of the data structure in addition to their amount accumulated in the database and the network. The flexible environment of information retrieval on multi-structured data stored in the computers is crucial to acquire relevant information for users. However, the state of the art remains within the retrieval for each specific data structure, e.g., natural text, relational data and sequential data [1) 2) 3)]. Accordingly, the retrieval on mixed structured data such as multimedia data containing documents, pictures and sounds requires the combined use of the retrieval mechanisms where each is dedicated to a specific data type [4) 5)]. Because of this nature, the current approach increases the cost and the work of the development and the maintenance of the retrieval system. To alleviate this difficulty, we propose a novel retrieval approach called "multi-Structure Information Retrieval"(MSIR) to use the most basic nature of the data representation. All real world data are represented by the sequence or the array, i.e. the orders of bits or bytes. We assumed that the data having similar content are usually represented by simiar sequences within a specific coding method. Thus the retrieval among files having entirely different coding or format is out of scope of this study. Moreover, this assumption does not hold under some coding schemes such as encrypted codings and some compressed codings. Encrypted codings are also out of scope of this work because the function of the encryption is to block this type of data retrieval. However, the most of the compressed codings can be decomprssed to some format to meet with this assumption. According to this assumption, a flexible retrieval method is established if a set of data which is mutually similar on this basic representation can be appropriately searched.

The main issue on the development is the definition of the similarity in the low level representation which appropriately corresponds to the similarity on the content level. Though the perfect correspondence may be hardly obtained, the following points are considered to enhance the feasibility of our proposal.
(1) Commonly seen byte portions in approximately similar length are searched.
(2) The judgment of the similarity is not significantly affected by the location of the patterns in the byte portions.
(3) The judgment of the similarity is not significantly affected by the noise and the slight difference in the byte portions.
(4) The mutual similarity of the entire files is evaluated by the frequency of the similar byte portions shared among the files.

(5) The similar byte portions shared by most of the given files are removed to evaluate the similarity among the files as they do not characterize the specific similarity.

The last point addresses the matter that the excessively common patterns do not provide any key information to sufficiently reduce the scope of the retrieval. This has been also addressed in the idea of TFIDF (Term Frequency Inversed Document Frequency) in the information retrieval [6] and the idea of "Stop List" [7]. In this work, a flexible method to retrieve similar files in terms of the byte portions is studied. A certain mathematical transform on the byte portions is used by treating each byte as a numeral. This can extract invariant characters of the portions against many distortions which do not affect the similarity in contents level, and the relevant files can be retrieved under the aforementioned consideration. The basic performance of the proposed approach is evaluated through experiments under artificial data, realistic word processor data and real bitmap image data.

§2 Principle of Similarity Judgment and Its Preliminary Study

The aforementioned point (1) is easily achieved by the direct comparison among byte portions. However, the point (2) requires a type of comparison among portions that is invariant against the shit of the portions. If the direct pair wise comparison between all sub-portions contained in two files is applied, the computational complexity is $O(n_{12}n_{22})$ where n_1 and n_2 are the numbers of bytes in the two files. To avoid this high complexity in practical sense, our approach applies a mathematical transformation to the byte portion in each file. The transformation has the property of "shift invariance" where the value obtained through the transformation is hardly changed against the shift of the portion structure. To address the point (3), the result of the transformation should be quite robust against the noise and slight difference in the portion. Moreover, the transformation must be conducted within practically tractable time. One of the representative mathematical transformation to suffice these requirements is the Fast Fourier Transform (FFT) [8]. It requires only computation time of $O(n \log n)$ in theory where the length of the byte sequence is n, and a number of methods for practical implementation are available. In addition, the resultant coefficients can be compressed into 50% of the original if only their absolute values are retained. However, when the transformation is applied to very large

portions or sub-portions contained in a large file where each part of the file indicates a specific meaning, the characters of the local byte portion reflecting a meaning in the contents level will be mixed with the characters of the other local part. Accordingly, we partition the byte portion in a file into an appropriate length, and apply the FFT to each part to derive a feature vector consisting of the absolute values of the Fourier coefficients.

The feasibility and the characteristics of the proposed method have been assessed through numerical experiments on pieces of byte sequences in advance before the further study and implementation are proceeded. In the experiment, the length of each byte sequence is chosen to be 8 bytes because it is the length of byte sequences to represent a word in various languages in standard. A number 128 is subtracted from the value of each byte to eliminate the bias of the FFT coefficient of order 0, while each byte takes an integer value in the range of [0, 255]. First, we shift the byte sequences to the left randomly, and the bytes out of the left edge are relocated in the right in the same order. Thus, the byte sequences are shifted in circular manner. Because of the mathematical nature of FFT, i.e., shift invariance, we observed that this did not cause any change on the absolute value of the transformed coefficients. Next, the effect of the random replacement of some bytes is evaluated. Table 1 exemplifies the effects of the replacement in the basic sequence "26dy10mo" on the absolute coefficients. The distance in the table represents the Hamming distance, i.e., the number of the different bytes from the original. The absolute coefficients from f_5 to f_8 are omitted due to the symmetry of Fourier Transform. In general, only $n/2+1$ coefficients for an even number n and $(n+1)/2$ for an odd number n are retained. The numbers of the absolute coefficients are quite similar within the Hamming distance 2 in many cases. However, they can be different to some extent even in the case of distance 2 such as "(LF)5dy10mo" where the value of "(LF)" is quite different from that of "2". Accordingly, some counter measure to absorb this type of change or noise in the similarity judgment must be introduced.

The method taken to enhance the robustness against the replacement noises in this work is the discretization of the absolute value of the FFT coefficients. If the absolute coefficients are discretized in an appropriate manner, the slight differences of the coefficient values do not affect the similarity judgment of the byte sequence. An important issue is the criterion to define the threshold values for the discretization. A reasonable and efficient way to define the thresholds of the absolute coefficients for arbitrary sequences is that

Table 1 Effect of byte replacements on FFT coefficients.

Sequences	f_0	f_1	f_2	f_3	f_4	distance
26dy10mo	144	112.9	345.6	103.8	108	0
20dy10mo	150	112.4	350.7	103.9	102	1
19dy10mo	142	113.8	343.6	103.1	112	2
(LF)5dy10mo	174	89.9	361.2	136.2	156	2
(LF)5dy11mo	178	86.6	364.4	137.3	152	3
(LF)5dy09mo	180	88.6	365.8	136.8	152	4

the absolute coefficient obtained from a randomly chosen sequence falls into an interval under an identical probability. To define the thresholds of the absolute coefficient in every order for a certain length of byte sequences, i.e., the length n, we calculated the absolute coefficient value distribution for all 2^{8n} byte sequences for every order. This computation is not tractable in straight manner, however in practice, this is quite easily achieved by using the symmetric and invariant characteristics of the absolute values of the FFT coefficients on various sequence patterns. For example, the absolute coefficients are invariant against the aforementioned circular shift. They are also invariant against the reverse of the portion in the byte sequence and the reverse of the positive and negative signs of all byte numbers in the sequence. Furthermore, the absolute coefficients of the third order are invariant against the reordering of the units consisting of subsequent two bytes in the sequence. For example, their values do not change among "26dy10mo" and "dy26mo10". By combining these characteristics of the absolute FFT coefficients, the space of the sequences consisting of 8 bytes to be assessed for the derivation of the exact absolute coefficient value distributions is significantly reduced, and the distributions are obtained in a few hours computation. Upon the obtained absolute coefficient distribution for every order, $(m-1)$ threshold values for every order are defined where every interval covers the identical probability $1/m$ in the appearance of a coefficient. When the number of m is small, the character of each byte sequence does not become significant due to the rough discretization. We tested various number m, and chose the value $m = 16$ empirically which is sufficient to characterize the similarity of the byte sequence in generic means. Through this process, the information of a FFT coefficient for every order is compressed into 16 labels. In summary, a feature vector consisting of $n/2+1$ or $(n+1)/2$ elements for an even or odd

number n is derived where each element is one of the 16 labels.

Moreover, the moving window of a fixed byte portion is applied to generate a set of feature vectors for a file. Fig. 1 is for the case of a byte sequence in a file. First, a feature vector of the byte sequence of a length $n(= 8)$ at the beginning of the file is calculated. Then another feature vector of the sequence having the same length n but shifted with one byte toward the end of the file is calculated. This procedure is repeated until the feature vector of the last sequence at the end of the file is obtained. This approach also enhances the robustness of the similarity judgment among files. For example, the feature vectors of the first 8 bytes windows of "26dy10mo02yr" and "(LF)5dy10mo02yr" are quite different as shown in Table 1. However, the feature vectors for the 8 bytes windows shifted by one byte, i.e., "6dy10mn0" and "5dy10mn0", are very similar to each other. Furthermore, the vectors for the windows shifted by two bytes become identical because both byte sequences are "dy10mo02". This moving window approach enhances the performance of the frequency counting of the parts having similar patterns among files. Thus, the point (4) mentioned in the first section is addressed where the mutual similarity of the entire files is evaluated by the frequency of the similar byte sequences shared among the files. To address the point (5), the feature vectors which are obtained from a given set of files more than a certain high frequency threshold are registered as "Unusable Vectors", and such unusable vectors are not used in the stage of the file retrieval. In case of the other type of byte orders such as two dimensional array order, similar moving window approach ia taken as described later.

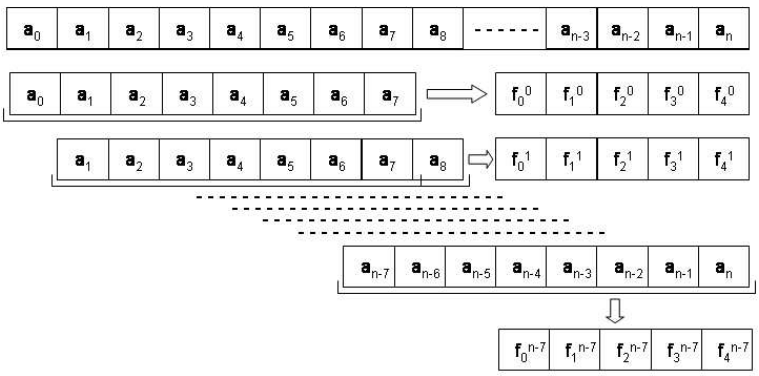

Fig. 1 FFT on moving windows

§3 Fast Retrieval Algorithm

The data structure to store the feature vectors for given vast number of files must be well organized to perform the efficient file retrieval based on the similarity of the byte portions. The approach taken in this work is the "inversed file indexing" method which is popular and known to be the most efficient in terms of retrieval time [3], [12]. Through the procedure described in the former section, the correspondence of each file to a set of feature vectors is derived. Based on this information, the inversed indexing from each feature vector to a set of files which produced the vector is derived. The data containing this inversed indexing information is called "inversed indexing data". By using the inversed correspondence in this data, all files containing patterns which are similar with a given feature vector are enumerated efficiently. Figure. 2 outlines our retrieval approach MSIR. The path represented by solid arrows is the aforementioned preprocessing. The "Data Extraction" part applies the moving window extraction of byte portions to each file in a given set of data files. The extracted byte portions are transformed by FFT in the "Mathematical Transformation" part. The "Vector Discretization" part discretizes the resulted coefficients by the given thresholds, and the feature vectors are generated. The "Vector Summarization" part produces the correspondence data from each file to feature vectors while removing the redundant feature vectors among the vectors derived from each file. Finally, the "Inversed Indexing" part derives the inverse correspondence data from each feature vector to files together with the "Unusable Vectors List".

The file retrieval is conducted along the path represented by the dashed arrows. A key file for the retrieval is given to the "Data Extraction" part, and the identical information processing from "Data Extraction" to "Vector Summarization" with the former paragraph derives the set of the feature vectors of the key file. Subsequently, the unusable vectors are removed from the set in the "Unusable Vectors Removal" part. Finally, the files whose feature vector sets have large intersection with the feature vector set of the key file are enumerated based on the inverse correspondence data in the "Vector Matching" part. The size of the intersection is counted using the inverse corresponding data in this part. Then to focus the retrieval result on only files having strong relevance with the key file, a relevance measure of each file to the given keyfile is caluculated based on a function of the size of the intersection and the number of feature vectors generated from the key file and each retrieved file. This measure can be changed depending on the objectives and the conditions of the retrieval. If the

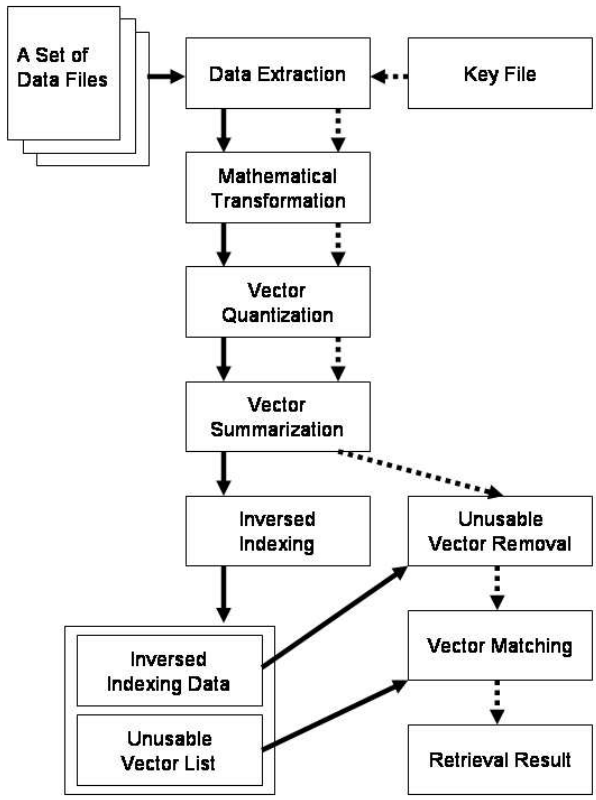

Fig. 2 Outline of retrieval system

evaluation value is less than a given threshold value, the file is not retained in the retrieval result. Moreover, the result is sorted in the order of the relevance measure.

§4 Basic Performance Evaluation

A C program based on the proposed method has been developed, and its basic performance was evaluated by using artificial data sets. The specification of the computer used in this experiment is CPU: AMD Athlon 1400MHz, RAM: PC2100 DDRSDRAM 384MB, HDD: Seagate ST340824A and OS: LASER5 Linux 7.1. 500 files having the normal distribution in their sizes were generated. Their average size is 30KB and the standard deviation 10KB. Once the size of a file is determined, the byte data in the file were generated by using the uniform random distribution. In the next stage, 5 specific sequences in the length of

16 bytes, which were labeled as No.1,$\cdots$, 5, were embedded in each file. They were embedded not to mutually overlap, and moreover the nonexistence of the sequences accidentally identical with these 5 sequences in the random generation of the byte data is verified upon this data preparation. The moving window size of 8 bytes, the 16 level of discretization of the FFT coefficients for each order and 70% for the threshold frequency to determine the unusable vector are set for the generation of the feature vectors along the solid line in Figure. 2.

Table 2 Retrieval by key file No.1

Sequence No.	Threshold	Retrieved Files	Correct Files	Precision	Recall	Comp. Time
1	1.0	250	250	1.00	1.00	0.6sec
	0.25	261	250	0.96	1.00	
	0.125	344	250	0.75	1.00	

Table 3 Retrieval by shifted key file No.1

(a) Result by proposed method using FFT

Sequence No.	Threshold	Retrieved Files	Correct Files	Precision	Recall	Comp. Time
1	0.66	2	2	1.00	0.01	0.7sec
	0.55	37	37	1.00	0.15	
	0.44	250	250	1.00	1.00	
	0.33	252	250	0.99	1.00	
	0.22	266	250	0.94	1.00	
	0.11	326	250	0.77	1.00	

(b) Result by conventional keyword matching

Sequence No.	Threshold	Retrieved Files	Correct Files	Precision	Recall	Comp. Time
1	-	0	0	0.00	0.00	0.5sec

The performance indices used in the experiment is the precision and the recall. In ideal situation, both values are close to 1. However, they have a trade off relation in general. Table 2 shows the performance of the retrieval by the key file consisting of the sequence No.1. The thresholds in the table are the

aforementioned threshold values on relevance measure to evaluate the similarity of the files in the "Vector Matching" part in Fig. 2. Under the condition of the high threshold values, only highly similar files are retained. The sequence No.1 is embedded in the 250 files among 500 test files. This is reflected in the result of the threshold equal to 1.0, i.e., the key file consisting of the sequence No.1 is certainly included in these files as a subsequence. In the lower value of the threshold, some files containing similar subsequence with the sequence No.1 are also retrieved. Thus, the precision decreases. In this regard, the proposed approach has a characteristic to retrieve a specified key pattern similarly to the conventional keyword retrieval when the threshold is high.

Table 3 (a) shows the result of the retrieval where the key sequence No.1 is shifted randomly in circular manner. Because the lengths of the embedded sequences and the key sequence are 16 bytes, but that of the moving window for FFT is only 8 bytes, the FFT coefficients do not remain identical even under its shift invariance characteristics. Accordingly, the feature vectors of the key sequence do not match with these of the embedded sequences completely. However, the coefficients of FFT reflects their partial similarity to some extent, and thus the excellent combination of the values of the precision and the recall is obtained under the frequency threshold values around [0.2, 0.4]. Similar results were obtained in case of the other key sequences. In contrast, when we applied the conventional retrieval approach based on the direct matching without using the FFT to derive the feature vectors, the values of the precision and the recall were zero as shown in Table 3 (b). Table 4 represents the results for noisy data. 2 bytes are randomly chosen in each original 16 bytes sequence, and they are replaced by random numbers. Similarly to the former experiment, the excellent combination of the precision and the recall was obtained for most of the key sequences under the threshold value of [0.3, 0.5]. If the distortion on the embedded sequences by the replacement becomes larger, i.e., the increase of the number of bytes to be replaced, the values of precision and the recall decreases. But, the sufficient robustness of the proposed retrieval approach has been confirmed under the random replacement of 3 or 4 bytes in the 16 bytes sequence through the experiments.

The computation time to finish a retrieval for a given key file is around 1 second due to the efficient inverse indexing approach. Thus, the proposed method shows practical efficiency for this size of problems. In short summary, the basic function of our approach subsumes the function of the conventional

Table 4 Retrieval on Noisy Data

Sequence No.	Threshold	Retrieved Files	Correct Files	Precision	Recall	Comp. Time
1	0.33	3	2	0.67	0.01	0.9sec
	0.22	27	18	0.67	0.07	
	0.11	159	92	0.59	0.37	
2	0.77	1	1	1.00	0.01	1.2sec
	0.66	15	15	1.00	0.04	
	0.55	125	125	1.00	1.00	
	0.22	140	125	0.89	1.00	
	0.11	205	125	0.62	1.00	
3	0.625	1	1	1.00	0.01	1.0sec
	0.5	3	3	1.00	0.03	
	0.375	31	28	0.90	0.28	
	0.25	120	100	0.83	1.00	
	0.125	229	100	0.44	1.00	
4	0.375	1	1	1.00	0.02	1.1sec
	0.25	38	14	0.37	0.28	
	0.125	178	50	0.28	1.00	
5	0.66	3	3	1.00	0.12	1.2sec
	0.55	25	25	1.00	1.00	
	0.22	34	25	0.74	1.00	
	0.11	127	25	0.20	1.00	

retrieval approach, because the retrieval equivalent to the conventional one is performed by setting the frequency threshold value of the feature vector matching at a high value as shown in Table 2. Our approach can retrieve the files having a certain generic similarity.

§5 Evaluation on Real-World Text Documents

For the evaluation of the practicality of the proposed approach, we measured the performance of the conventional text retrieval by using a dataset consisting of plain text documents. The dataset includes 2253 files of text documents written in Japanese, and each file contains sentences on the topics of atomic power generation. Every file in the dataset has several keywords labeled

by human experts to represent the contents. Some files do not contain keywords which is labeled to the files. After applying the preprocessing to generate the inverse indexing data, 3 keywords are selected from the set of labeled keywords randomly for the evaluation as shown in the second column of Table 5. Each keyword is given to the retrieval system and processed along the dashed line in Fig. 1. Because of the nature of the mathematical transformation, i.e., FFT, of the original byte sequence in the files under the moving window approach, our retrieval system enumerates the files containing not only the original keyword given to the system but also similar string of the keyword. For example, the file containing "fuel", "nuclear", "nuclear fuel" and "fuel cycle" can be retrieved under the keyword of "nuclear fuel cycle". Accordingly, the performance of the retrieval system has been evaluated in terms of not only the number of the given keyword contained in the retrieved file but also the number of the partial strings of the keyword. The files containing the partial strings of the original keyword is considered to have some relevance. The partial strings shown in the third column in Table 5 are used in the evaluation in addition to the original keyword for the 3 retrieval cases from No.1 to No.3.

Table 5 Keywords and its partial strings

No.	Keywords	Partial strings
1	nuclear fuel cycle	nuclear fuel, cycle, fuel
2	fault-tree analysis	fault-tree, analysis
3	dynamical characteristic of nuclear reactor	dynamical characteristic nuclear reactor

Table 6 (a), (b) and (c) show the retrieval results for the several frequency threshold values for the feature vector matching in the retrieval. In these tables, the first column from left shows the frequency threshold value, and the second column shows the retrieved file ID. Because the retrieval result of the lower threshold value subsumes that of the higher threshold value, only the additionally retrieved file IDs are indicated in the lower threshold case. The rest of the columns except for last one show whether the strings written in the top row are contained in each retrieved file. If it is contained, the value is "1" otherwise "0". The last column is the total number of "1". If the value is larger, the file is considered to be more related to the keyword.

Table 7 shows the retrieval result on the same keyword by well-known

text retrieval system of NAMAZU[12] where the keyword matching is used. The results of NAMAZU include some files which contain no keyword. This is because NAMAZU divides keyword into morphologies by famous morphological analysis system, ChaSen[13], then calculates frequency in which each morphology appears in every file. So the results of NAMAZU include files in which morphologies appear many times but no keywords do. Our approach also retrieve files that contain strings that are partial and/or similar to the original keyword. For example, in the result of "fault-tree analysis", the file 1636 appears in the retrieval result even when the frequency threshold of relevance measure is set to 1.0 while the file does not contain keyword. We checked the file 1636, and found a misspelled word "falt-tree analysis" in the file. In this manner, our approach can find files that include similar keyword including misspelling in addition to files including right keywords without having any dictionary. Moreover in the case of the keyword "nuclear fuel cycle", the file numbered 675 which does not include any keywords or partial string is retrieved when the threshold of retrieval is set to 0.7. We found a phrase "fuel cycling" in this file. Since the phrase "fuel cycling" approximately means "nuclear fuel cycle", the retrieval result supposed to include files containing similar meaning words. In case of NAMAZU, the files numbered 224 and 183 are retrieved on the keyword "nuclear fuel cycle" which do not appear in our result of the Table 6(a). However our system could retrieve them when the threshold of retrieval is set to 0.7. In addition, on the "fault-tree analysis", no file is retrieved by NAMAZU since "fault-tree" is not registered in word dictionary of ChaSen. So we registered "fault-tree" for the dictionary and experimented again. The second column in Table7 is the result of experiment after registration of "fault-tree". Our approach, in contrast, could retrieve files containing keyword without knowing the word since it does not use the concept of words. It only compares the invariant freature vector generated from both keyword/keyfile and stocked files. The performance of our approach is approximately equivalent or superior in a certain condition to the performance of NAMAZU which is a retrieval system specialized to text file format.

Moreover, our approach was compared with the conventional signature files approach[10]. Signature files approach is as follows. First, sequences having fixed a length which is set by user are extracted from sequence in a file by moving window. Then a set of values which are calculated from each extracted sequence by a function is generated. For exmple, let an original file F be a byte sequence $\{10, 14, 25, 36, 7, 1\}$, the extraction length 2 bytes and the function

Table 6 Results of Retrieval

(a) Result on "nuclear fuel cycle"

Threshold	Retrieved file ID	nuclear fuel	cycle	fuel cycle	nuclear fuel cycle	Total
1.0	945	1	1	1	1	4
	913	1	1	1	1	4
	885	1	1	1	1	4
	819	1	1	1	1	4
	805	1	1	1	1	4
	6	1	1	1	1	4
	682	1	1	1	1	4
	530	1	1	1	1	4
	385	1	1	1	1	4
	372	1	1	1	1	4
	29	1	1	1	1	4
	257	1	1	1	1	4
	1732	1	1	1	1	4
	1541	1	1	1	1	4
	1488	1	1	1	1	4
	1407	1	1	1	1	4
	1348	1	1	1	1	4
	1312	1	1	1	1	4
	1194	1	1	1	1	4
	1189	1	1	1	1	4
	1148	1	1	1	1	4
0.8	922	1	1	1	0	3
	122	0	1	1	0	2
0.7	675	0	0	0	0	0
	⋮					

Only a protion of thr retrieved files are indicated for the threshold 0.7 due to space limiation.

(b) Result on "fault-tree analysis"

Threshold	Retrieved file ID	fault-tree	analysis	fault-tree analysis	Total
1.0	72	1	1	1	3
	268	1	1	1	3
	1636	0	1	0	1
0.7	1640	0	1	0	1
	1594	0	1	0	1
	1392	0	0	0	0

(c) Result on "dynamical characteristic of nuclear reactor"

Threshold	Retrieved file ID	nuclear reactor	dynamical characteristic	dynamical characteristic of nuclear reactor	Total
1.0	894	1	1	1	3
	1587	1	1	1	3
	1051	1	1	1	3
0.8	1216	1	0	0	1

Table 7 Retrieval result of NAMAZU

nuclear fuel cycle	fault tree analysis	dynamical characteristic of nuclear reactor
885	72	1051
1407	268	1587
183*		894
805		
372		
6		
945		
257		
224*		
922*		
29		
1541		
1194		
682		
1312		
1189		
1148		
1488		
1348		
913		
530		
385		
1732		

* Contain no keyword

"$(a_1 + a_2)$ *mod* 8". By applying this function to the 2 bytes moving window, the set $\{0, 7, 5, 3, 0\}$ is obtained. Subsequently, the redundant elements are gathered into one element in the set. Then new set becomes $\{0, 7, 5, 3\}$. Let this set be A. All elements in A are less than 8 since each element is modulation of 8. A bit vector called "signature" whose length is 8 bits is generated from A in the following manner. The x-th bit in the signature is set to "1" if x is in A, othewise "0". In case of this A, the signature becomes "10010101". This signature represents some features of the original file F, and the signature is labeled to the file F. On retrieval, the signature is calculated by the same function from the keyword or key file, and the signature is compared with the signatures of files accumurated in the computer breforehand. Files are listed in retrieval when all non-zero digits in key file's signature are also non-zero in the files' signatures. In short, when the logical product between the signature of the keyword or key file and the signature of the file is identical to the original key word's or key file's signature, the file is retrieved. The signature files approach used on evaluation uses the 1536 bits signature whose values are caluculated by

the hashjpw function[11]. Table 8 shows that signature files approach retrieves only the files which contain the keyword. Because small change in the sequence causes large change on the hash value. The signature becomes significantly is different from that of the original sequence. Accordingly, the signature files method is very sensitive to the miss-spell or synonyms.

Table 8 Retrieval result of signature files approach

nuclear fuel cycle	fault tree analysis	dynamical characteristic of nuclear reactor
1148	72	1051
1189	268	1587
1194		894
1312		
1348		
1407		
1488		
1541		
1732		
257		
29		
372		
385		
530		
6		
682		
805		
819		
885		
913		
945		

In short summary, the performance of our approach is approximately equivalent or superior under a certain conditions to the performance of NAMAZU which is a retrieval system specialized to text file format and the performance of the signature files approach which is another representative retrieval approach.

§6 Evaluation on Semi-Real-World Word Processor Files

The practical performance of our proposed method is also evaluated by using real world data. The data is a set of 2253 word processor files having Microsoft Word doc format. The average size of a file is around 20KB. Each contains a document consisting of almost 600 characters coded in a specific binary format. Accordingly, the conventional text keyword retrieval is not applicable

to this retrieval problem. To evaluate the ability to retrieve similar content files within the proposed approach, the raw Microsoft Word data are numbered from No.1 to No. 2253, and they are processed to have stronger similarity in terms of contents when the number labels of the files are closer. Initially, a seed file is selected from the original set of 2253 word processor files and numbered as No.$1'$. Then a sequence consisting of 16 charactors is extracted from the content of a file which is randomly selected from the left 2252 files. The sequence overwrites the sequence having length of 16 charactors selected from the content of file No.$1'$, and the overwritten sequence is stored as file No.$2'$. Subsequently, a sequence consisting of 16 charactors is extracted from the content of a file selected from the left 2251 files. A sequence consisting of 16 charactors selected from file No.$2'$ is overwritten by the sequence, and so on. This process is repeated until the set of files becomes empty to gradually and randomly change the original seed file, and newly generate similar files. As a result of this process, 2253 files in total are generated where the files having close number labels have similarity.

Table 9 Retrieval on semi-real world data

KeyFile No.100	KeyFile No.500	KeyFile No.1000	KeyFile No.1500	KeyFile No.2000
100	500	1000	1500	2000
102	676	789	1499	2001
99	664	979	1494	1999
104	508	648	1498	1995
96	554	999	1502	2158
97	503	967	1497	2223
105	579	997	1496	1868
106	561	856	1503	2019
103	543	852	1504	1989
98	485	543	1506	1877
Std. 17.0	Std. 142.4	Std. 176.4	Std. 190.0	Std. 108.2
$\chi^2 =$ 1352	$\chi^2 =$ 316	$\chi^2 =$ 256	$\chi^2 =$ 1765	$\chi^2 =$ 385
0.642sec	0.466sec	0.422sec	0.844sec	0.370sec

Based on this semi-real world data, the inversed indexing data and un-

usable vector list are generated in the preprocessing stage of our approach. Subsequently, 5 key files arbitrary chosen from the semi-real world files are used to retrieve their similar files. Each key file is given to the retrieval system and processed along the dashed line in Fig. 2. Table 9 show the result of the top 10 retrieved files in the order of the similarity judged in the feature vector matching the 5 key files. The result clearly shows that the files having close number with the key file are retrieved. Some files are not retrieved even when their numbers are closer to the number of the given key file. This is because the character sequence for the replacement can be quite different from the original overwritten sequence as numerical series data, and this replacement significantly affects the coefficients of FFT in the feature vectors. This effect has been already discussed in the example of the feature vectors of "26dy10mo02yr" and "(LF)5dy10mo02yr" in Table 1. Though the moving window approach alleviates this type of distortion in the judgment of similarity, the judgment is infected to some extent even by this approach when the character sequence for the replacement is largely different from the overwritten sequence. The third row from the bottom in the table indicates the standard deviation of the label numbers of the top 50 retrieved files, and the second row from the bottom shows the chi-squared value on the difference of the standard deviation from that of the label numbers randomly sampled. In this case, the chi-squared value follows the distribution whose degree of freedom is 49. If the chi-squared value is more than 94.6, the probability that the files retrieved are randomly sampled is less than 0.0001. In this regard, the distributions of the retrieval results are sufficiently skewed around the key files in the sense of the similarity. The bottom row represents the computation time to retrieve the 50 files for each key file. The 50 similar files are retrieved within a second among the 2253 doc files for each key file. The difference of the time for retrieval is due to the difference of the number of the feature vectors which are not unusable for each key file. For example, the number of the usable feature vector of the key file No.1500 is 1474 while it is only 593 for the key file No. 2000. The retrieval time is almost linear with the number of the effective feature vectors of each key file.

§7 Evaluation on Real-World MS Word Documents

In addition to the ordinary ascii text data and semi-real MS Word data, we evaluated the performance of our approach by using a dataset consisting of

MS Word documents which is coded into binary data. The dataset includes 2253 files of MS Word documents that are converted from text files used in the previous section 5. The inversed indexing data and unusable vector list are generated from this dataset. Then 3 files are chosen as a key file from the dataset, and these key files are given to the retrieval system and processed along the dashed line in Fig. 2. To evaluate the performance of the retrieval, some representative keywords characterizing the contents are selected from each key file, and the frequency of appearance of keywords are counted in each retrieved file. Table 10 shows the average frequency of the keyword occurrence per file and the average ratio of the keyword occurrence normalized by the number of letters included in each retrieved file. Table 11 shows the average frequency and the average ratio with the same definition for randomly chosen 20 files from the dataset without using our retrieval approach. From these tables, the performance of our approach is clearly higher than that of random file sampling. Accordingly, our approach is also capable of retrieving similar files on binary MS word files.

Table 10 Retrieval performance of our approach

File No.	200	1000	2000
Ave. frequency of keywords	3.00	2.04	2.55
Ave. ratio of letters	0.013	0.022	0.024

Table 11 Result of random sampling

File No.	200	1000	2000
Ave. frequency of keywords	0.40	1.15	0.5
Ave. ratio of letters	0.002	0.008	0.005

An important point through section 4 to 7 is that our identical retrieval approach is used for each of these tasks. Nevertheless, our approach shows good performance in each evaluation. This shows that proposed approach has flexibility to retrieve data which constraints the contents represented by byte sequences.

§8 Evaluation on Complex Structued Data

In this section, we propose a slight extention of our approach to complex structured data format. Our previous approach considers data which have

similar orders to be relevant each other. Then by taking into account invariant feature based on mathematical transformation from order, files in dataset which is similar to keyfile is retrieved. For complex structured data, the "Data Extraction" part in Fig. 2 is needed to be modified for each data type. For example, we explain the case of bitmap image files. An image file has grid structure where each point of the grid is related to information of colors. Humans recognize the similarity among images arranging similar colors in similar order. Moreover, humans recognize the similarity among images including many similar partial blocks. Because an image has two dementional structure, "Data Extraction" and "Mathematical Transformation" in Fig.2 must transform two dimensional byte data, and convert the two dimensional coefficients into one dimensional feature vectors. To generate feature vectors, all color information is divided into two components, i.e., intensity and color difference, and two dimensional moving window which has a fixed size area sucn as 8×8 or 16×16 is applied to intensity component of the image in "Data Extraction" part in Fig.2. The color difference component is not used bucause of the charasteristic of human's vision that is more sensitive to intensity than color difference. Two dimensional Fourier transformation is applied to every area and the two dimensional Fourier coefficients are obtained for every area. High vertical and/or horizontal frequency components are removed from coefficients since human's vision is insensitive to fine change in the image. Then a zigzag-scan on a coefficients matrix shown in Fig.3 to retain low order coefficients. The feature vectors are generated by quantization of coefficient vectors. After generating feature vector, the procedure in Fig.2 identical with the case of one dimensional order is applied. This approach needs minimum change to deal with complex structured data.

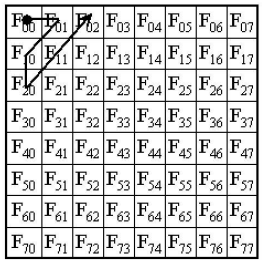

Fig. 3 Zigzag-scan

This extended approach is applied to the dataset consisting of 101 bitmap

files whose sizes lie between 5 KB and 8MB. The moving window size is set as 8×8 pixels. The FFT coefficients are discretized into 12 levels for each order. The threshold frequency to determine the unusable vector are set to 70% for the generation of the feature vectors. The specification of the computer used in this experiment is CPU: AMD AthlonXP 1900+, RAM: PC2100 DDRSDRAM 2GB, HDD: Quantum Atlas10KIII and OS: RedHat Linux 7.2.

The evaluation function used on judgement of similarlity is:

$$F(i) = \frac{f(i)}{\sqrt{x \cdot y(i)}} \tag{1}$$

where $f(i)$ is the number of matched feature vector of the i-th file in the data with those obtained from the key file. x is the number of feature vector generated from the key file, and $y(i)$ is the number of feature vector generated from the i-th file. This evaluation function is chosen after a number of empirical analysis. The details of the analysis can be seen in the literature[9].

The evaluation was repeated several times by changing the key file. We show an example of the result. The key file given to the system is shown in Fig.4. The top 5 files retrieved by our MSIR approach are shown in Fig.5. Futhermore the 50th through 54th files of the retrieval result are shown in Fig.6

According to this result, all files in the top 5 are a kind of diagram which is same with the key file. Their appearance is similar to the key file. In contrast, files shown in Fig.6 are photographs or illustrations whose appearances are not as similar as those of the top 5 files. Our approach can obtain good result on complex structured data such as image file by changing a small part of the MSIR into suitable one.

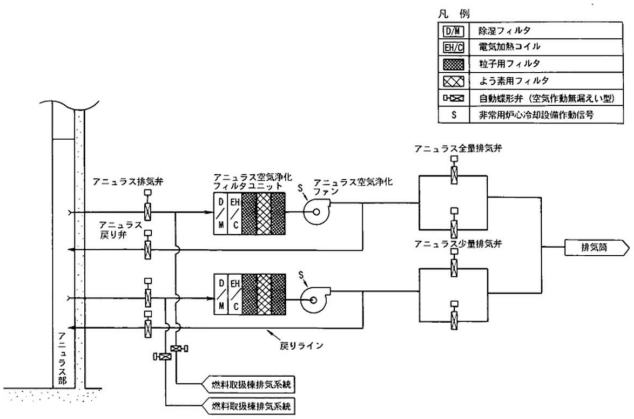

図5 アニュラス空気浄化設備系統図

〔出典〕原子力安全研究協会（編）：軽水炉発電所のあらまし（改訂版）、原子力安全研究協会（平成4年10月）

4-2-3-10-6

Fig. 4 Image of key file

However, the problem of the computation time remains. It took 25 seconds to retrieve in the previous example. More efficient retrieval is neccessary.

§9 Discussion and Related Work

The signature files method to use moving windows of byte sequences having a fixed length in the files has been proposed for file retrieval [2]. This method compresses each byte sequence in an incomplete and irreversible fashion by introducing hash functions, and efficiently focuses on similar key sequence patterns on the reduced size of binary signature data. However, the direct matching of key sequence is required at the final stage of the retrieval to achieve the complete retrieval because of the incompleteness of the signature matching. On the other hand, the inversed indexing approach where the files containing each key are listed in advance are often used for the fast and practical retrieval [3]. One of the representative systems is Namazu for Japanese documents [12]. Though this approach needs a considerably large space for the indexing data storage, recent increase of the capacity of the storage devices is alleviating this difficulty. However, this approach is for the complete keyword matching in files such as documents.

In contrast, the proposed method applies a mathematical transformation having some invariance and compression properties to retain the information of certain similarities among files rather than the ordinary hash compression function. Because of the nature of the mathematical transformation, the complete matching is easily achieved in our framework if the threshold value for feature vector matching is taken at a high frequency. Moreover, the incomplete matching to retrieve files containing similar patterns in terms of the invariance and robustness of the transformation is also achieved by applying a lower threshold value. The efficiency of the retrieval is comparable with the ordinary inversed indexing approach because our approach also uses the inversed indexing on the representation of feature vectors.

§10 Conclusion

In this work, "Multi-Structure Information Retrieval"(MSIR) approach is proposed. Our approach has high flexibility since it is applicable to various data format by slightly changing the generation process of feature vectors. The proposed approach covers the most advantage of the conventional approaches. In addition, our approach provides less cost to adapt to a certain data format than to develop a new system specialized to the format. However there are some exceptions, e.g., enciphered data, compressed data and more complex structured data. The measure to handle these formats is left for future work. Also the problem on the computation time of retrieval on complex structured data must be addresed in future work.

References

1) Baeza-Yates, R.A, "String Searching Algorithms" *Information Retrieval, Data Structures & Algorithms, Chapter 10*(ed. Baeza-Yates, R.A.), pp. 219-240, 1992.

2) Faloutsos, C. , "Signature Files" *Information Retrieval, Data Structures & Algorithms, Chapter 4*(ed. Baeza-Yates, R.A.), pp. 44-65, 1992.

3) Harman, D., Fox, E. and Baeza-Yates, R.A., "Inverted Files" *Information Retrieval, Data Structures & Algorithms, Chapter 3*(ed. Baeza-Yates, R.A.), pp. 28-43, 1992.

4) Ogle, V.E., Stonebraker,M., "Chabot: Retrieval from a Relational Database of Images", *IEEE Computer, Vol. 28, No. 9* pp. 1-18, 1995.

5) Faloutsos, C., Equitz, W., Flickner, M., Niblack, W., Petkovic, D. and Barber, R., "Efficient and Effective Querying by Image Content", *Journal of Intelligence Information Systems, 3, 3/4*, pp. 231-262, 1994.

6) Salton, G. and McGill, M.J., *Introduction to Modern Information Retrieval*, McGraw-Hill Book Company, 1983.

7) Fox, C., "Lexical Analysis and Stoplists", *Information Retrieval, Data Structures & Algorithms, Chapter 7*(ed. Baeza-Yates, R.A.), pp. 102-130, 1992.

8) The Institute of Electronics, Information and Communication Engineers (IEICE), *Digital Signal Processing 10th Ed.*, Gihoudou, pp. 49-61, 1983. (in Japanese)

9) Fujimoto, A., Adachi, F., Washio, T., Motoda, H., Niwa, Y. and Hanafusa, H., "Expansion of Generic Search Method for Two-dimensional Data", *Proc. of the 17th Annual Conference of Japanese Society for Artificial Intelligence(JSAI)*, 2C3-01, 2003. (in Japanese)

10) Masui, T., "An Retrieval System Based on Signature Files Approach", *Monthly UNIX magazine 1999.11*, ASCII, pp.170-176, 1999. (in Japanese)

11) Aho,V.R., Ethi, R. and Ullman, D.J., "Compilers: Principles, Techniques, and Tools", Addison-Wesley, 1986

12) http://www.namazu.org/

13) http://chasen.aist-nara.ac.jp/

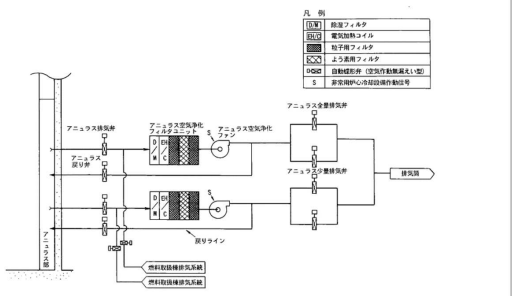

図5　アニュラス空気浄化設備系統図

〔出典〕原子力安全研究協会（編）：軽水炉発電所のあらまし（改訂版）、原子力安全研究協会（平成4年10月）

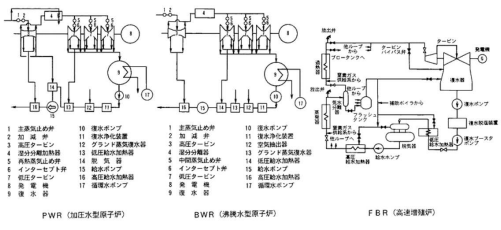

図1　タービンサイクル系統概略図

〔出典〕旭火力原子力発電技術協会：原子力発電講座、昭和57年5月；
動力炉・核燃料開発事業団：高速増殖炉原型炉－設計・建設・試運転の軌跡－、1994年7月

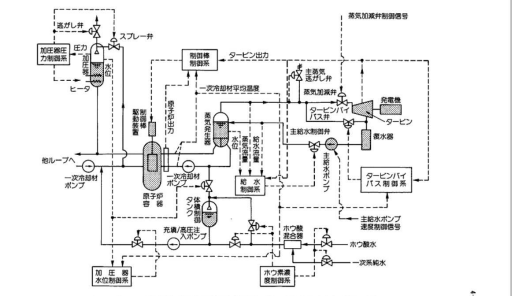

図1　PWR原子炉制御系系統概略

〔出典〕原子力安全研究協会(編)：軽水炉発電所のあらまし（改訂版）、原子力安全研究協会（平成4年10月）

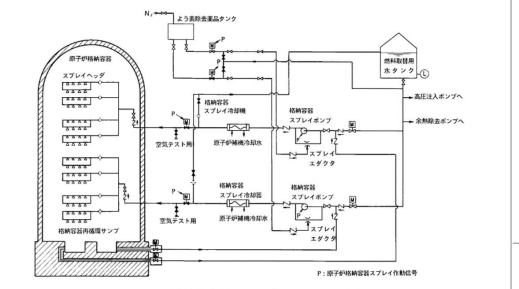

図4　原子炉格納容器スプレイ設備系統図

〔出典〕原子力安全研究協会（編）：軽水炉発電所のあらまし（改訂版）、原子力安全研究協会（平成4年10月）

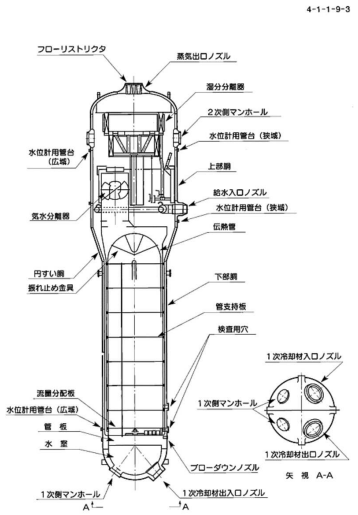

図3　PWRの蒸気発生器構造図

〔出典〕関西電力：大飯発電所原子炉設置変更許可申請書、
昭和61年12月

Fig. 5　Top5 files retrieced by keyfile

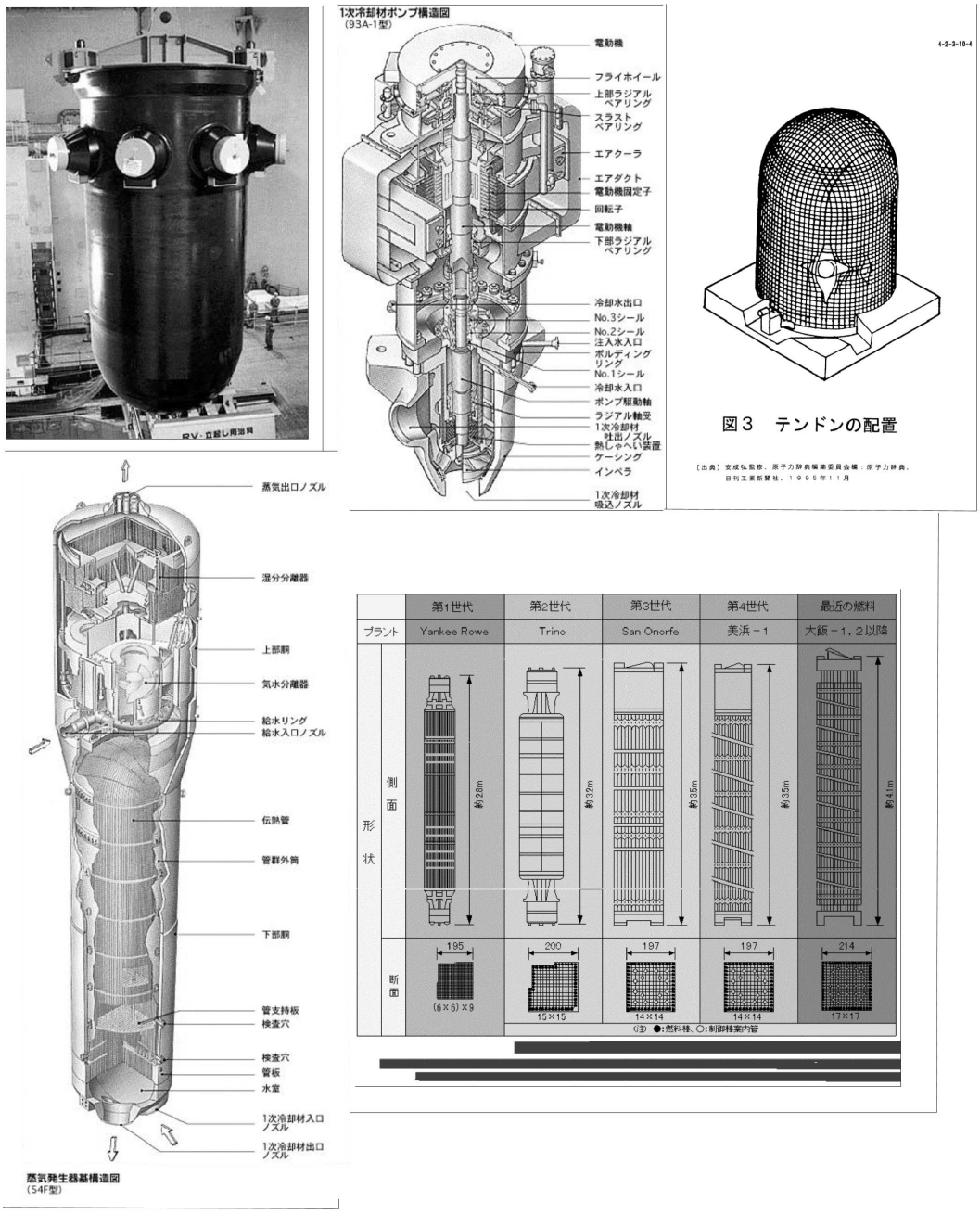

Fig. 6 The 50th through the 54th file of retrieval result