Development of Generic Search Method Based on Transformation Invariance

Fuminori Adachi[1], Takashi Washio[1], Atsushi Fujimoto[1], Hiroshi Motoda[1], Hidemitsu Hanafusa[2]

[1] I.S.I.R., Osaka University, 8-1 Mihogaoka, Ibarakishi,
Osaka 567-0047, Japan
{adachi,washio,fujimoto,motoda}@ar.sanken.osaka-u.ac.jp
[2] INSS Inc., 64 Sata, Mihamacho, Mikatagun,
Fukui 919-1205, Japan
hanafusa@inss.co.jp

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 key roles to acquire relevant information desired by users in active retrieval process. However, most of the existing approaches are dedicated to each content and data structure respectively, e.g., relational database and natural text. In this work, we propose a generic information retrieval method directly applicable to various types of contents and data structures. The power of this approach comes from the use of the generic and 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.

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 data type respectively [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 to use the most basic nature of the data representation. All real world data are represented by the sequence of bits or bytes. Accordingly, a generic 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 sequences in approximately similar order and length are searched.
(2) The judgment of the similarity is not significantly affected by the location of the patterns in the byte sequences.
(3) The judgment of the similarity is not significantly affected by the noise and the slight difference in the byte sequences.
(4) The mutual similarity of the entire files is evaluated by the frequency of the similar byte sequences shared among the files.
(5) The similar byte sequences shared by most of the 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 by 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 generic method to retrieve similar files in terms of the byte sequences is studied. A certain mathematical transform on the byte sequences is used by treating each byte as a numeral. This can extract invariant characters of the sequences, and the relevant files can be retrieved under the aforementioned consideration. The basic performance of the proposed approach is evaluated through numerical experiments and a realistic application to the retrieval of raw binary format data of a word processor.

2 Principle of Similarity Judgment and Its Preliminary Study

The aforementioned point (1) is easily achieved by the direct comparison among byte sequences. However, the point (2) requires a type of comparison among sequences that is invariant against the shit of the sequences. If the direct pair wise comparison between all subsequences selected from two sequences is applied, the computational complexity is $O(n_1^2 n_2^2)$ where n_1 and n_2 are the numbers of bytes in the two sequences. To avoid this high complexity in practical sense, our approach applies a mathematical transform to the byte sequence in each file. The transform has the property of "shift invariance" where the value obtained through the transform is hardly changed against the shift of the sequence. To address the point (3), the result of the transform should be quite robust against the noise and slight difference in the sequence. Moreover, the transform must be conducted within practically tractable time. One of the representative mathematical transform to suffice these requirements is the Fast Fourier Transform (FFT) [8]. It requires only computation time of $O(n \log n)$ in theory when 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 the amount of 50% of the original if only their absolute values are retained. However, when the transform is applied to very long sequences or subsequences contained in a large file where each part of the file indicates a specific

meaning, the characters of the local byte sequence reflecting a meaning in the contents level will be mixed with the characters of the other local part. Accordingly, we partition the byte sequence in a file into an appropriate length, and apply the FFT to each part to derive a feature vector consisting the absolute values of the Fourier coefficients.

The feasibility and the characteristics of the proposed method have been assessed though some numerical experiments on some 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 FTT 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 edge are located in the right in the same order. Thus, the byte sequences are shifted in circular manners. 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 a 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.

Table 1. Effect of byte replacements on FFT coefficients

Sequences	f0	f1	f2	f3	f4	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 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 vales 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 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 forward, 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 order 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 two byte units 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 length byte sequence is applied to generate a set of feature vectors for a file as depicted in Fig.1. 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 mutually very similar. 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.

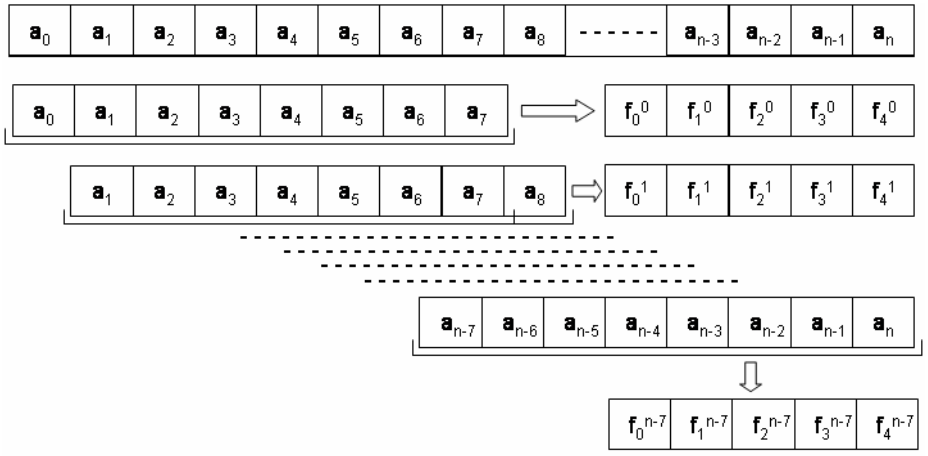

Figure1. FFT on moving windows

3 Fast Algorithm of Retrieval

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 sequences. 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],[9]. Through the procedure described in the former section, the correspondence from each file to a set of feature vectors derived from the file is obtained. 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. The path represented by solid arrows is the aforementioned preprocessing. The "Data Extraction" part applies the moving window extraction of byte sequences to each file in a given set of data files. The extracted byte sequences 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".

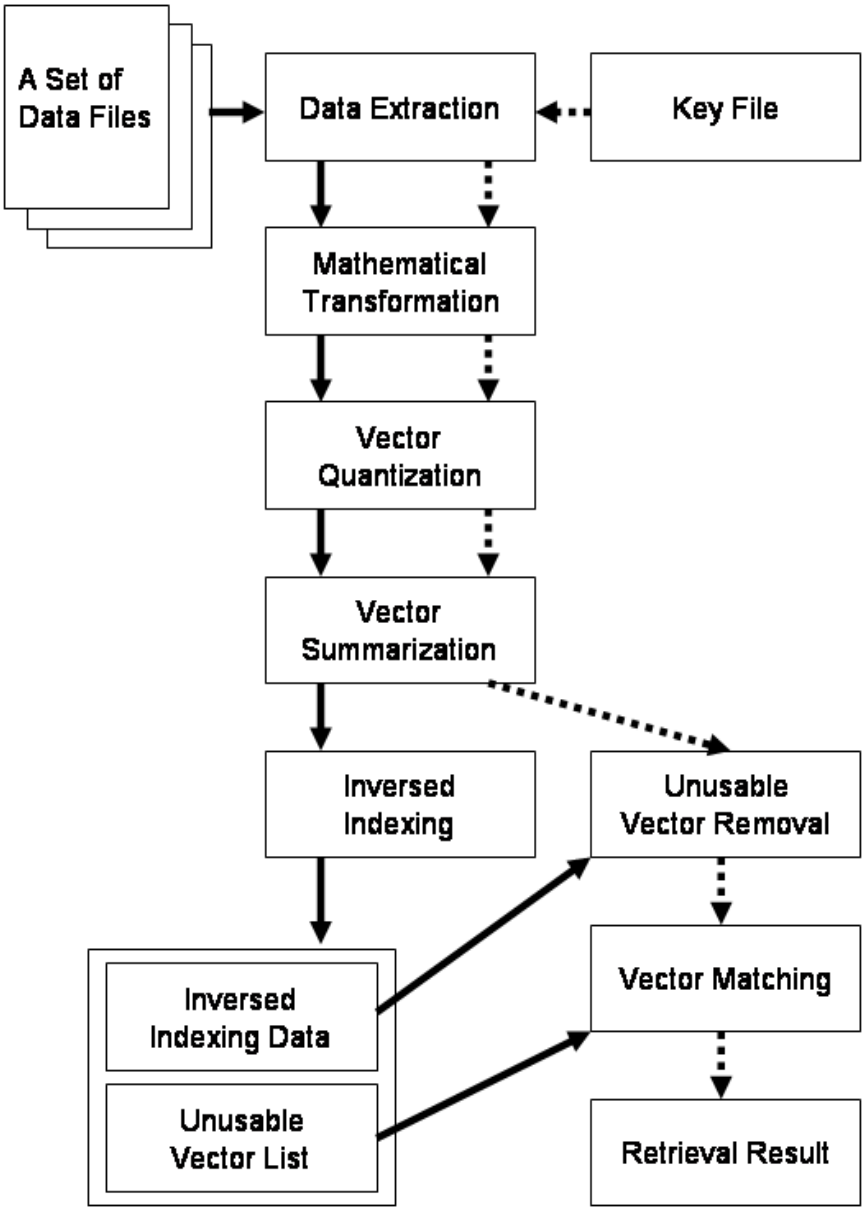

Figure2. FFT on moving windows

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 corresponding to the feature vectors in the set are enumerated based on the inverse correspondence data in the "Vector Matching" part. First, the "frequency" of the complete match of every feature vector in the set to the identical vector in the inverse corresponding data is counted in this part. Then to focus the retrieval result to only files having strong relevance with the key file, the total sum of the frequencies of all feature vectors in the set are calculated. If the total frequency of the vector matching is less than a given "frequency threshold value", the file is not retained in the retrieval result. Moreover, the result is sorted in the order of the matching frequency.

4 Basic Performance Evaluation

A 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 348MB, 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, …, 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. 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.

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	1.6
	0.25	261	250	0.96	1.00	
	0.125	344	250	0.73	1.00	

Table 3. Retrieval by shifted key file No.1

(a) Result by proposed method using FFT

Sequence	Threshold	Retrieved	Correct	Precision	Recall	Comp.

No.		Files	Files			Time
1	0.66	2	2	1.00	0.01	0.7
	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	0.5

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 frequency threshold values 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, our proposing 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 in complete fashion. 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, very low values of the precision and the recall were obtained 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 the 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 under the random replacement of 3 or 4 bytes in the 16 bytes sequence has been confirmed through the experiments.

Table4. Retrieval on Noisy Data

Sequence No.	Threshold	Retrieved Files	Correct Files	Precision	Recall	Comp Time
1	0.33	3	2	0.67	0.01	0.9
	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	1.2
	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	203	125	0.62	1.00	
3	0.625	1	1	1.00	0.01	1.0
	0.500	3	3	1.00	0.03	
	0.375	31	28	0.90	0.28	
	0.250	120	100	0.83	1.00	
	0.125	229	100	0.44	1.00	
4	0.375	1	1	1.00	0.02	1.1
	0.250	38	14	0.37	0.28	
	0.125	178	50	0.28	1.00	
5	0.66	3	3	1.00	0.12	1.2
	0.55	25	25	1.00	1.00	
	0.22	34	25	0.74	1.00	
	0.11	127	25	0.20	1.00	

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 scale of problems. In short summary, the basic function of our approach subsumes the function of the conventional retrieval approach, because the retrieval equivalent to the conventional retrieval is performed by setting the frequency threshold value of the feature vector matching at a high value as shown in Table 2. Moreover, this approach can retrieve the files having some generic similarity.

References

[1] Baeza-Yates, R.A.: String Searching Algorithms, Information Retrieval, Data Structures & Algorithms, Chapter 10, ed. Baeza-Yates, R.A., New Jersey: Prentice Hall, pp. 219-240 (1992).

[2] Faloutsos, C: Signature Files, Information Retrieval, Data Structures & Algorithms, Chapter 4, ed. Baeza-Yates, R.A., New Jersey: Prentice Hall, 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., New Jersey: Prentice Hall, 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., 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., New Jersey: Prentice Hall, pp. 102-130 (1992).

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

[9] http://www.namazu.org/