2003－ＩＣＳ－133　(1)
2003／9／14

# A Fast Algorithm of Generic Pattern Retrieval and Its Application

Fuminori ADACHI[†], ○ Takashi WASHIO[†], Atsushi FUJIMOTO[†], and Hiroshi MOTODA[†]

† I.S.I.R., Osaka University　Mihogaoka 8–1, Ibaraki-shi, Osaka, 567–0047 Japan
E-mail: †{adachi,washio,fujimoto,motoda}@ar.sanken.osaka-u.ac.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 real data indicates its high feasibility.

**Key words** active data mining, information retreival, mathematical transformation, transformation invariance

# 高速汎用パターン検索手法とその実適用

足立史宣[†]　○鷲尾　隆[†]　藤本　敦[†]　元田　浩[†]

† 大阪大学産業科学研究所　〒567–0047 大阪府茨木市美穂ケ丘 8–1
E-mail: †{adachi,washio,fujimoto,motoda}@ar.sanken.osaka-u.ac.jp

**あらまし** データベースやネットワーク上には，マルチメディアデータの蓄積が進行しており，それらの効率的で柔軟な情報検索が求められている．特にアクティブな検索過程では，関連性の高い情報を柔軟に収集することがユーザーにとって鍵となる．しかしながら，現状の大半の手法は，リレーショナルデータベースや自然文のような各形式や構造に特化した検索を行うものである．本報では，種々のタイプのコンテンツやデータ構造で使用可能な一般的な情報検索手法を示す．これはファイル中のバイトパターンに数学的変換を施し，一般的で不変性を有する特徴情報を抽出して使用する方法である．本手法を実データに適用し，高い性能を確認した．

**キーワード** アクティブデータマイニング，情報検索，数学的変換，変換不変性

## 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]～[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 a realistic application to the retrieval of raw binary format data of a word processor.

## 2. Principle of Similarity Judgment

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 f5 to f8 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.

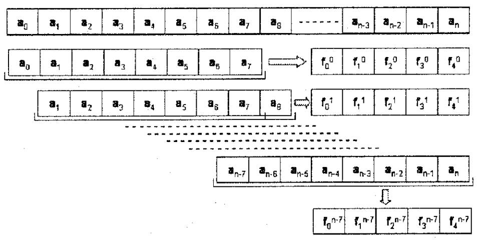
The method taken to enhance the robustness against the replacement noises in this work is the discretization of the absolute value of the FFT coefficients. If the absolute coefficients are discretized in an appropriate manner, the slight differences of the coefficient vales do not affect the similarity 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

Table 1 FFT 係数に関するバイトデータ変更の影響
Table 1 Effect of byte replace on FFT coefficients.

| Sequneces | 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 |

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



$a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_6$ $a_7$ $a_8$ ------ $a_{n-3}$ $a_{n-2}$ $a_{n-1}$ $a_n$
$f_0^0$ $f_1^0$ $f_2^0$ $f_3^0$ $f_4^0$
$f_0^1$ $f_1^1$ $f_2^1$ $f_3^1$ $f_4^1$
$a_{n-7}$ $a_{n-6}$ $a_{n-5}$ $a_{n-4}$ $a_{n-3}$ $a_{n-2}$ $a_{n-1}$ $a_n$
$f_0^{n-7}$ $f_1^{n-7}$ $f_2^{n-7}$ $f_3^{n-7}$ $f_4^{n-7}$

Figure 1 移動窓に対する FFT
Fig. 1 FFT on moving windows.

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.

## 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

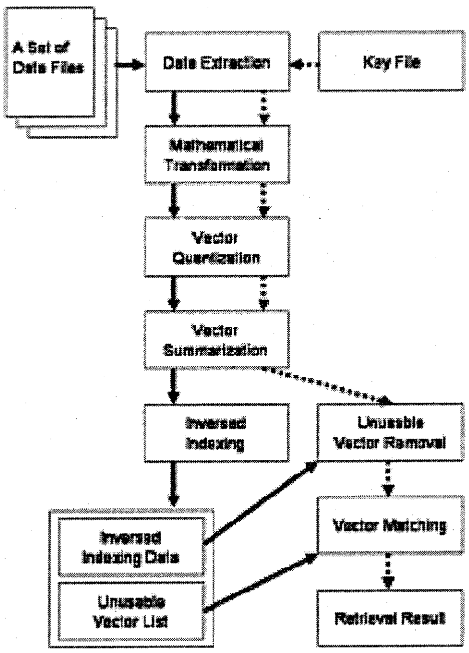


Figure 2 検索手法の概要
Fig. 2 Outline of retrieval approach.

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".

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. Evaluation on Real-World Word Processor Files

The practical performance of our proposed method is 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 text document consisting of 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 our proposing 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 word processor files and numbered as No.1. Then, another file $X$ is randomly chosen from the raw file set, and a sequence consisting 16 characters is selected from the file. Then, a randomly chosen part consisting of 16 characters sequence in the original file No.1 is overwritten by the sequence selected from the file $X$, and the new file is numbered as No.2. Starting from this stage, a part of 16 characters randomly chosen in the file No. $n$ is overwritten by the sequence of 16 characters selected from another randomly chosen file $X$, and the new file is numbered as No. $n+1$. This process is repeated 2253 times to gradually and randomly change the original seed file and newly generate similar files. As a result, 2253 files in total are generated where the files having close number labels have some similarity.

Based on this real world data, the inversed indexing data and unusable vector list are generated in the preprocessing stage of our approach. Subsequently, 5 key files arbitrary chosen from the 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 2 show the result of the top 10 retrieved files in the order of the similarity in the feature vector matching for the 5 key files. The result clearly shows that the files having close number to the key file are retrieved. Some files are missed to be retrieved even when their numbers are close 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 under this approach when the character sequence for the replacement is much different from the overwritten sequence. The third row from the bottom in the table indicates the standard deviation of the number labels of the top 50 retrieved files, and the second row from the bottom shows the chi-squared value on the difference of the distribution from the expected distribution of the number labels of randomly sampled 50 files. The numbers from 1 to 2253 are divided into the 50 sections, then the expected number of files in each section is 1 when 50 files are sampled randomly. At this time, the chi-squared value follows chi-squared distribution which flexibility 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. Therefore, 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 files. 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 is 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. This difference is reflected on the retrieval time. The retrieval time is almost linear with the number of the effective feature vectors of each key file.

Table 2 実データに関する検索結果

Table 2 Retrieval on real world data.

| Key File No.100 | Key File No.500 | Key File No.1000 | Key File No.1500 | Key File 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 | 508 | 967 | 1497 | 2258 |
| 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 |
| 0.642 sec. | 0.466 sec. | 0.422 sec. | 0.844 sec. | 0.370 sec. |

## 5. Summary

In this work, a generic retrieval approach for the data, where one dimensional byte sequences reflect the contents of the data, is proposed. The proposed approach covers most of the advantage of the conventional approaches. The next issue is to extend this approach to multi-dimensional data such as image data and 3D data where the information of the contents are not reflected in the byte sequences in straightforward manner.

## Acknowledgement

This work was partially supported by the grant-in-aid for scientific research on priority area "Active Mining" (No. 13131101, No. 13131206) funded by the Japanese Ministry of Education, Culture, Sport, Science and Technology.

## 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/