Development of Generic Search Method Based on Transformation Invariance

Fuminori Adachi, Takashi Washio, Hiroshi Motoda and *Hidemitsu Hanafusa
I.S.I.R., Osaka University, {adachi, washio, motoda}@ar.sanken.osaka-u.ac.jp
*INSS Inc., 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 under interactions among domain experts, data mining experts and data mining tools in active mining 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 information retrieval on multi-structured data stored in the computers is crucial to acquire relevant information under interactions among domain experts, data mining experts and data mining tools. 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. Any data is 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.

In this work, an approach to the 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 files to meet 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

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 respectively is applied, the computational time 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 number of practical methods for 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 sub-sequences contained in a large file where each part of the file indicates a specific meaning, the local characters of the byte sequence reflecting the meaning in the contents level will overlap with the local characters of the other 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.

Because this approach is quite novel, the basic feasibility and the characteristics of the proposed method have been checked though some numerical experiments on some pieces of byte sequences in advance. 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. Though each byte takes a value in the range of [0, 255], a number 128 is subtracted from the value to eliminate the bias of the FTT coefficient of order 0. 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 of the transformed coefficients. Next, the effect of the random replacement of some bytes are evaluated. Table 1 exemplifies the effects of the replacement in a basic sequence "26dy10mo" on the transformed coefficients. The distance in the table represent the Hamming distance, i.e., the number of the different bytes from the original. The coefficients from f_5 to f_8 are omitted due to the symmetry. 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 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 coefs.

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 replacements in this work is the discretization of the FFT coefficients. If the 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 discretization. The most efficient way to define the thresholds is that the coefficient obtained from an arbitrary sequence falls into an interval under an identical probability. To define the thresholds of the coefficient in every order for a certain length of byte sequences, i.e., the length n, we calculated coefficient value distribution for all 2^{8n} byte sequences. This computation is not tractable, however in practice, this is quite easily achieved by using the symmetric characteristics of FFT coefficients on various sequence patterns. Upon the obtained coefficient distribution for every order, (m-1) threshold values are defined where every interval covers the identical probability 1/m in the appearance of a coefficient of every order. 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. First, a feature vector of the byte sequence of a length n 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 enhance the

robustness of the similarity judgment among files. For example, the feature vectors of the first 8 bytes windows of "26dy10mo02yr" and "(LF)5dy10mo02 yr" 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 enables 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 is obtained from a given set of files more than a certain frequency threshold are registered as ineffectual vectors, and such ineffectual 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]. 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 data indexing from each feature vector to a set of files, that 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 1 outlines our retrieval system. The path represented by solid arrows is the aforementioned preprocessing. The "Data Extraction" part applies the moving window extraction of byte sequences to each files 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 "ineffectual vectors list".

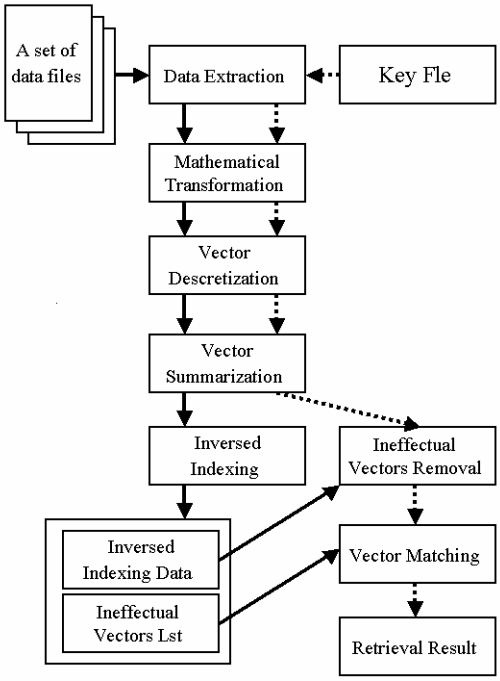

Figure 1 Outline of retrieval system

The file retrieval is conducted along the path represented by the dashed arrows. A retrieval key file having the content which is objective or similar to the objective is given to the "Data Extraction" part, and the identical information processing with the former paragraph derives the set of the feature vectors of the key file. Subsequently, the ineffectual vectors are removed from the set in the "Ineffectual 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. To focus the retrieval result to only files having strong relevance with the key files, a frequency threshold value is applied in the enumeration. If the frequency of the vector matching is less than the threshold for a file, 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 was 30KB and the standard deviation 10KB. The byte data in each file were generated by using the uniform random distribution. Next, 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 identical with these 5 sequences is confirmed. The parameters of the generation of the feature vectors are 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 ineffectual vector.

Table 2 Retrieval by key file No.1

Sequence No.	Thre-shold	Retrieved Files	Correct Files	Pre-cision	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

Sequence No.	Thre-shold	Retrieved Files	Correct Files	Pre-cision	Recall	Comp. 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	

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 levels of the feature vector matching to evaluate the similarity of the files in the "Vector Matching" part in Fig.1. The high value of the threshold retains only highly similar files. 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 shows the result of the retrieval where the key sequence No.1 is shifted randomly in circular manner. Because the length of the embedded sequences and the key sequence is 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 vector of the key sequence does not match with these of the embedded sequences. 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 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, very low values of the precision and the recall were obtained for every threshold levels. Table 4 represents the results for noisy data. 2 bytes randomly chosen in each original 16 bytes sequence are replaced by random numbers. Similarly to the former experiment, the excellent combination of the precision and the recall was obtained for every key sequence 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 replace, 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.	Thre-shold	Retrieved Files	Correct Files	Pre-cision	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 was less than 1 second due to the inverse indexing approach. Thus, the proposed method is highly practical for considerably large scale application. In short summary, the basic function of our approach subsumes the function of the conventional retrieval approach, because the conventional retrieval is performed by setting the frequency threshold of the feature vector matching at a high value. Moreover, this approach can retrieve the files having some generic similarity.

Table 5 Retrieval on semi-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	503	967	1497	2258
105	579	997	1496	1868
106	561	856	1503	2019
103	543	852	1504	1989
98	485	543	1506	1877
109	471	513	1508	2208
113	541	998	1501	2171
107	642	857	1495	2008
92	636	855	1507	2020
116	513	695	1505	2004
101	553	676	1511	1855
93	789	672	1513	2154
108	498	591	1510	2135
95	567	508	1509	2079
111	525	1084	1489	2068
110	504	1006	1492	2207
112	486	837	1491	2061
117	852	982	1515	2022
88	848	975	1493	2269
114	678	909	1488	2192
120	604	896	1512	2126
94	477	534	1519	1990
128	933	858	1486	1958
118	870	627	1483	1908
115	689	500	1518	2150
79	571	995	1485	2017
127	856	991	1514	1957
125	836	948	1487	1858
121	695	563	1528	1857
81	537	541	1490	2257
89	897	984	1521	2181
90	562	965	1516	2160
122	558	615	1482	2009
138	524	996	1517	2005
87	869	986	1524	1961
141	615	854	1525	2245
139	544	992	1484	2153
78	817	974	1520	2121
142	555	814	1527	2007
124	497	773	1526	1986
126	959	760	1480	2157
85	873	677	1522	2096
140	679	562	1476	2089
83	677	994	1523	2081
123	712	993	1481	1994
Ave. 108.22	Ave. 641.64	Ave. 823.60	Ave. 1503.44	Ave. 2058.80
t= -148.6	t= -70.8	t= - 44.2	t= 55.0	t= 136.1
0.642 sec	0.466 sec	0.422 sec	0.844 sec	0.370 sec

5. Evaluation on Word Processor Files

The practical performance of our proposed method is evaluated by using semi-real world data. The data is a set of 2253 word processor files having Microsoft Word doc format. Their average size is around 20KB, and each contains around 600 characters in form of a document. To evaluate the ability to retrieve similar content files within our proposing approach, the raw content data are converted to have some similar relations among some files. 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 in the file is selected from the file. Then, a randomly chosen part consisting of 16 character 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 a 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 consequence, 2253 files in total are generated where the files having close number have some similarity.

Based on this semi-real world data, the inversed indexing data and ineffectual 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.1. Table 5 show the result of the top 50 retrieved files in the order of the similarity in terms of the feature vector matching. The result clearly shows that the files having close number to the key file are retrieved. Some files are missing to be 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 in terms of numerical series data, and this

replacement significantly affects the coefficients of FFT to from the feature vectors. This effect has been already shown 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 on the judgment of similarity, the judgment is infected to some extent even under this approach. The third row from the bottom in the table indicates the average of the top 50 files' numbers, and the second row from the bottom shows the t-value on the deviation of the average of the top 50 files' numbers from the expected average of the uniformly sampled 50 files' numbers, i.e., 1126.5. According to the Central Limit Theorem, the average approximately follows the normal distribution N(1126.5, 105750) under the uniform sampling. The absolute t-value more than 3.3 indicates that the probability that the files retrieved follow the uniform distribution is less than 0.001. 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 ineffectual for each key file. For example, the number of the effective feature vector of the key fille No.1500 is 1474 while it is only 593 for the key file No. 2000. This difference is reflected to the retrieval time. The retrieval time is almost linear with the number of the effective feature vectors of each key file.

6. 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 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 practically fast retrieval [3]. One of the representative system is Namazu for Japanese documents [9]. Though this approach needs considerably large space for the indexing data storage, the recent increase of the capacity of the storage devices is alleviating this difficulty. However, this approach is for the complete matching on the files such as text documents.

In contrast, our proposing method applies a mathematical transform 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 transform, 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 transform is also achieved by applying the 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.

7. Conclusion

In this work, a generic retrieval approach for the data, where one dimensional byte sequences reflect the contents of the data, is proposed. The examples of the data are the ordinary text files, word processor files and sound data files. The proposed approach covers the most 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 straight forward manner.

Reference

[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] 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).

[7] Salton, G. and McGill, M.J.: Introduction to Modern Information Retrieval, McGraw-Hill Book Company (1983).
[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/