Fast String Matching Method Based on Histogram

(1) Ibrahem Amer Hameed, (2) Dr.Loay Edward George,

(1)College of Science, Computer Science Department, Baghadad University, Baghdad, Iraq; (2) College of Science, Computer Science Department, Baghadad University, Baghdad, Iraq
(1) Ibrahem_star_2005@yahoo.com
(2) Loayedward57@yahoo.com

ABSTRACT

This paper presents an efficient algorithm to locate all occurrences of any finite number of keywords in a string text, selectively in matching. It is a frequency-based method because it depends on the frequency of occurrence of the first letter of each keyword listed in the text. The idea of the search and matching is similar to that of searching a word in a dictionary. The mechanism of the proposed method is to alphabetically sort the keyword list, and the keywords have the same leading letter are considered as subsets and they may also be called as search intervals since the search of each input keyword is applied, only, within its corresponding alphabetical interval. In other words it is a sort of clustering words. The results show that applying this method has fastened the retrieval of information about 16.6% in an information retrieval, in contrast to the traditional exhaustive search.

Keywords : retrieval, subsets, alphabet, text, matching, sorting.

1 Introduction

In formal languages, which are used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols that are chosen from a set or alphabet [1] [2] In computer programming, a string is traditionally a sequence of characters, either as a literal constant or as some kind of variable. The latter may allow its elements to be mutated and/or the length changed, or it may be fixed (after creation). A string is generally understood as a data type and is often implemented as a byte (or word) array that stores a sequence of elements, typically characters, using some character encoding. A string may also denote more general array data types and/or other sequential data types and structures; terms such as byte string, or more general, string of datatype, or datatype-string, are sometimes used to denote strings in which the stored data does not (necessarily) represent text[3] Depending on programming language and/or precise datatype used, a variable declared to be a string may either cause storage in memory to be statically allocated for a predetermined max length or employ dynamic allocation to allow it to hold chronologically variable number of elements[4]. String searching is an important component of many problems, including text editing, data retrieval, and symbol manipulation. Despite the use of indices for searching large amounts of text, string searching helps reduce the run time in an information retrieval system. For example, it may be used for filtering of potential matches or for searching retrieval terms that will be highlighted in the output. [3] The string searching or string matching problem consists of finding all occurrences (or the first occurrence) of a pattern in a text, where the pattern and the text are strings over some alphabet. It is well known that to search for a pattern of length m in a text of length n (where n > m) the search time is O(n) in the worst case (for fixed m) [5] In many information systems it is necessary to be able to locate and match quickly some or all occurrences of user-specified patterns of words and phrases in text [6][1] exhaustive matching trail is a weak and trivial solution to be followed; because the system will try all of the possible choices until it reaches a positive match. The run time is wasted in the brute force search without taking into account the search efficacy. The proposed and applied search method is named frequency-based method because it depends on the frequency of occurrence of the leading letter of each keyword listed in one record. The idea of the search is similar to that of searching a word in a dictionary. The mechanism of the proposed method is to alphabetically sort the keyword list, and the keywords have same leading letter are considered as subsets and they may also be called as search intervals since the search of each input keyword is applied, only, within its corresponding alphabetical interval.

2 FREQUENCY BASED METHOD

- Capture the first letter of the query's keyword.
- Match this keyword with a convenient subset (the subset of keywords belongs to the examined database record) whose keywords begin with same letter of the query's keyword. In this way the search interval is shrunken to a small subset of keywords.

3 IMPLEMENTATION STEPS OF THE FREQUENCY BASED METHOD

The matching process is performed as follows:

1. Calculate the histogram of each letter. For example the histogram of the set {Cosmos, Comet, Uranus, Asteroids, Star, Spacecraft, Satellite} is equal to {(c,2), (u,1), (a,1), (s,3)}. The histogram determination implies the following steps:

a) Capture the first letter of each word,
b) Convert the letter to lower letter case,
c) Get the ASCII code of this letter,
d) Subtract it by the ASCII of the "a" to find its cardinal number.

The cardinal number is considered as a pointer of that letter in the histogram array. Whenever a keyword begins with the same initial letter the pointer goes to same index at the histogram array and adds one to the old content value (this means counting every occurrence of the alphabet letters). The histogram determination task is performed overall keywords listed in the tested database record. The size of histogram array is 26 and of byte data type, each histogram element represents the number of keywords begins with certain the alphabet letter whose cardinal number represents the index of the corresponding histogram element. The resulted histogram array is considered a key table in the next step.

2. After the histogram array is found it is, then, used to calculate a list of pointers, each list element points to the start position of the keywords begin with a certain letter. The histogram elements are used as keys to calculate the corresponding peers in the pointer list since the former elements store the sizes of each subset.

After the pointers generation step, each pointer is considered as the base address for the corresponding set of keywords which have a certain initial letter. The pointer moves next when a new keyword, which has the corresponding initial letter, appears.

3. Sort keywords: after the determination of the initial pointer list, then it used to sort the list of keywords of the tested database records; such that each keyword in the record is put in a new record at position index equal to the corresponding pointer value, and increment that pointer value by 1.

4. Search: the resulting initial pointer array is used to allocate the search area (i.e., the start and end positions) of each tested keyword belonging to a query record. In such case the matching trail will be bounded to a certain subset rather than searching the whole list of keywords belongs to the tested database record. In other words, the pointer list is used to provide the upper bound and the lower bound of each search interval (subset).

4 APPLICATION OVER THE INFORMATION RETRIEVAL SYSTEM

5. An information retrieval system is an application that stores and manages information about document contents, often textual documents but possibly multimedia [7] [8] the system assists users in finding the information they need. It does not explicitly return information or answer questions. Instead, it informs on the existence and location of documents that might contain the desired information. Information retrieval system can support three basic processes: the representation of the content of the documents, the representation of the user's information need, and the comparison (matching) of the two representations [7]. Figure (2) presents the main steps of a typical classical IR system.

6. The comparison of query against the document representations is called the matching process. The matching process usually results in a ranked list of documents [11]. Users can walk down the document list looking for the information they need. Ranked retrieval hopefully puts the relevant documents towards the top of the ranked list, minimizing the time the user has to invest in reading the documents. The simplest but effective, ranking algorithms use the frequency distribution of terms over documents, and they may use some statistics over other information (such as the number of hyperlinks that point to the document). Ranking algorithms based on statistical approaches easily halve the time the user has to spend on reading documents [8]. The frequency based method is used in an information retrieval system that matches two 40 keyword list with lists of size 90 whose numbers are 410 list, in such a system it is urgent to use a matching method performs the operation efficiently. So, the matching between the two lists of keywords in the information retrieval system one represents the query keywords list and the other is the keywords list stored in the database. So, this relationship is considered as a many to many relationship. Since matching aims to find out the common keywords listed in both records, so it requires mapping between two lists of strings whose elements are not sorted according to alphabetical criteria; instead they are ordered according to their relative significance weights. Taking into consideration that the retrieval operation should be performed as an online process, so the matching task should be done as quickly as possible in order to ensure an acceptable performance of the retrieval process. The user should not wait a long time to get the retrieval answer. So, another important issue must be taken seriously to reduce the time of matching the matching between the query and the database record.

7. In a simple one _to _one process, it is possible to convert the string to an integer number (hash number) according to the ASCII code of its characters, and then be matched with the hash values of the other side to see whether it is common or not. This matching mechanism could be adopted to handle the matching task between the user names and passwords. But this is not a practical solution to be applied in any system that deals with a large number of records with each contains many subrecords. This would clearly spend a long search time in conversion from one type to another.

8. On the other hand, exhaustive matching trail is a weak and trivial solution to be followed; because the system will try all of the possible choices until it reaches a positive match. The run time is wasted in the brute

force search without taking into account the search efficacy.

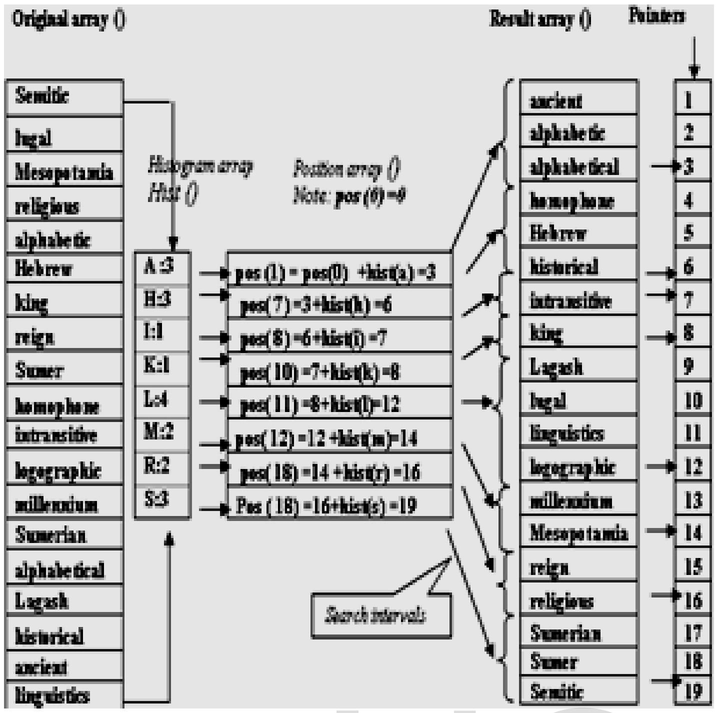

Fig. 1. The proposed frequency based method

5. FREQUENCY-BASED SEARCH VS. EXHUSTIVE SEARCH

The method used to match two keyword list each of sizes 40 and 90 respectively, firstly by applying the brute force as a search method which implies trial of all choices, and then secondly the same set of sample queries are tested using the frequency based method.

The test is performed to assess the efficacy of this method, as illustrated in table (1) and figure (3) The results show that applying this method has fastened the retrieval about 16.6%.

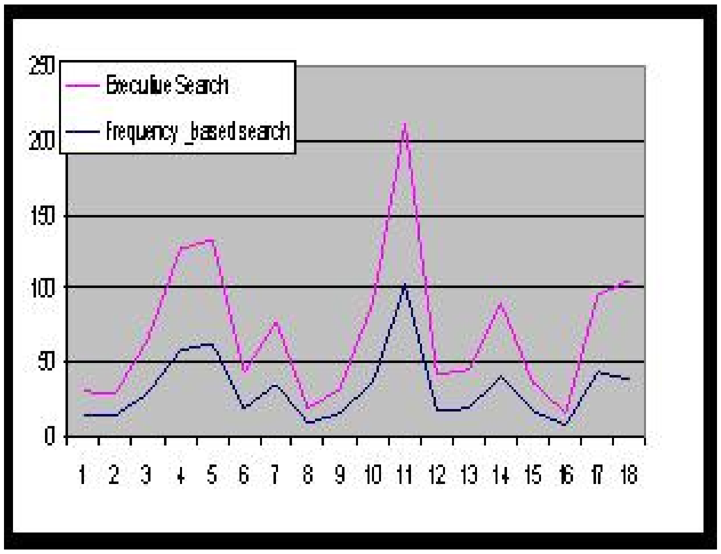

Fig. 1. The test result of the frequency based method

6. CONCLUSIONS

The string matching is a time consuming step in systems that deals with large numbers of strings. In the many to many mappings it is urgent to use fast matching methods rather than the classical matching (exhaustive). In this paper a matching method is conducted to perform the matching efficiently, the search is a dictionary like method. The retrieval of the system that matches two list of sizes 40 and 90, respectively, was 16% faster than using the brute force search.

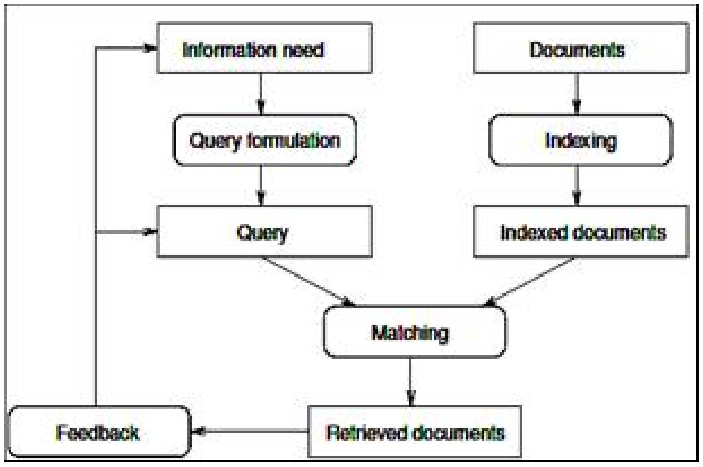

Fig. 1. The Classical information retrieval system

TABLE 1

FREQUENCY-BASED MATCHING AND SEARCH VS. THE TRADITIONAL EXECUTIVE SEARCH

Frequency _based search	Executive Search	Frequen-cy _based search	Executive Search
14.92	15.33	15.27	15.66
14.08	15.23	7.27	7.48
29.52	35.64	45.56	50.11
58.67	67.34	9.45	10.81
63.75	69.05	48	56.14
18.78	25.73	17.5	15.59
35.52	42.98	20.17	28.09
9.17	9.52	22.91	44.98
16.5	16.27	6.75	7.13
37.45	52.78	50.61	58.89
101.66	109.08	9.86	9.06
17.28	25.67	13.25	19.45
19.64	26.08	7.5	7.8
40.55	49.91	10.06	10.69
18.66	19.2	10.66	11.06
7.67	7.91	29.75	36.36
44.63	51.47	24.3	24.39
38.31	66.69	51.47	58.17
Average Time		27.42	32.72

REFERENCES

[1] J.S. Bridle, “Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition,” *Neurocomputing – Algorithms, Architectures and Applications,* F. Fogelman-Soulie and J. Herault, eds., NATO ASI Series F68, Berlin: Springer-Verlag, pp. 227-236, 1989. (Book style with paper title and editor)

[2] W.-K. Chen, *Linear Networks and Systems.* Belmont, Calif.: Wadsworth, pp. 123-135, 1993. (Book style)

[3] H. Poor, “A Hypertext History of Multiuser Dimensions,” *MUD History,* http://www.ccs.neu.edu/home/pb/mud-history.html. 1986. (URL link *include year)

[4] K. Elissa, “An Overview of Decision Theory," unpublished. (Unplublished manuscript)

[5] R. Nicole, "The Last Word on Decision Theory," *J. Computer Vision,* submitted for publication. (Pending publication)

[6] C. J. Kaufman, Rocky Mountain Research Laboratories, Boulder, Colo., personal communication, 1992. (Personal communication)

[7] D.S. Coming and O.G. Staadt, "Velocity-Aligned Discrete Oriented Polytopes for Dynamic Collision Detection," *IEEE Trans. Visualization and Computer Graphics*, vol. 14, no. 1, pp. 1-12, Jan/Feb 2008, doi:10.1109/TVCG.2007.70405. (IEEE Transactions)

[8] S.P. Bingulac, “On the Compatibility of Adaptive Controllers,” *Proc. Fourth Ann. Allerton Conf. Circuits and Systems Theory*, pp. 8-16, 1994. (Conference proceedings)

[9] H. Goto, Y. Hasegawa, and M. Tanaka, “Efficient Scheduling Focusing on the Duality of MPL Representation,” *Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS '07)*, pp. 57-64, Apr. 2007, doi:10.1109/SCIS.2007.367670. (Conference proceedings)

[10] J. Williams, “Narrow-Band Analyzer,” PhD dissertation, Dept. of Electrical Eng., Harvard Univ., Cambridge, Mass., 1993. (Thesis or dissertation)

[11] E.E. Reber, R.L. Michell, and C.J. Carter, “Oxygen Absorption in the Earth’s Atmosphere,” Technical Report TR-0200 (420-46)-3, Aerospace Corp., Los Angeles, Calif., Nov. 1988. (Technical report with report number)

[12] L. Hubert and P. Arabie, “Comparing Partitions,” *J. Classification,* vol. 2, no. 4, pp. 193-218, Apr. 1985. (Journal or magazine citation)

[13] R.J. Vidmar, “On the Use of Atmospheric Plasmas as Electromagnetic Reflectors,” *IEEE Trans. Plasma Science*, vol. 21, no. 3, pp. 876-880, available at http://www.halcyon.com/pub/journals/21ps03-vidmar, Aug. 1992. (URL for Transaction, journal, or magzine)

[14] J.M.P. Martinez, R.B. Llavori, M.J.A. Cabo, and T.B. Pedersen, "Integrating Data Warehouses with Web Data: A Survey," *IEEE Trans. Knowledge and Data Eng.*, preprint, 21 Dec. 2007, doi:10.1109/TKDE.2007.190746.(PrePrint)