academicJournals

Vol. 8(16), pp. 640-648, 25 April, 2013
DOI 10.5897/SRE2013. 5373
ISSN 1992-2248 © 2013 Academic Journals
http://www.academicjournals.org/SRE

Scientific Research and Essays

Full Length Research Paper

A new search algorithm for documents using blocks and words prefixes

Khalid Thabit and Sumaia M. AL-Ghuribi*

Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia.

Accepted 5 April, 2013

Web has become the most enormous distributed databases on the Internet. The numbers of Web pages are growing in an amazing way. This leads to expanding the content information on the Web. With the rapid growing, it is easy to share information in web. At the same time, it is a hard task to search useful information effectively and accurately from vast amounts. This leads to create search methods that are more efficient. In this paper a new search algorithm for documents using blocks and words prefixes are presented. An implementation of the proposed algorithm is given also to check our algorithm performance and validity; we make a comparison between it and binary search. The experimental results show that the proposed algorithm outperforms the binary search in execution time and number of comparison parameters. Our algorithm using blocks and words prefixes is 66% faster than the binary search.

Key words: Search methods, binary search, blocks, words prefixes, execution time.

INTRODUCTION

The Internet has become a key communication and information medium for various organizations (Qingyu and Richard, 2008). It is a medium for accessing a great variety of information stored in different parts of the world. In recent years the growth of the World Wide Web exceeded all expectations. Today, there are several billions of HTML documents, pictures and other multimedia files available via Internet and the number is still rising (Bharanipriy and Prasad, 2011). With the rapid rising, it is very easy to publish and share information in web. At the same time, it is a hard task to search useful information effectively and accurately from vast amounts of data (XiangFeng and Quan, 2007). As a result, there is a need for efficient searching techniques to extract appropriate information from the web, as the users require correct and complex information from the web (Raymond and Hendrik, 2000). Many techniques for searching are available; a brief review of some related works about text (document) search techniques has been done.

XiangFeng and Quan (2007) suggested a semantic parsing model constructed above a symbolic system of concepts for understanding natural language. Based on the model a sentence can be mapped into the corresponding semantic category. A text search prototype based on the semantic model represented more accurate result by comparing the conceptual structures between search condition and text.

Kulekci (2007) introduced a new multi-pattern matching algorithm called Tara. It performs the searching process of fixed-length strings on text files with having the benefit of bit-parallelism which makes it works fast. Tara supports bounded gaps and character classes. The performance of Tara algorithm is compared with widely

*Corresponding author. E-mails: somaiya.ghoraibi@gmail.com, drthabit@gmail.com.

used GNU grep file search utility and also with 9 variants of Aho&Corasick and Comentz&Walter algorithms on natural language text. Tara algorithm outperforms them and be the fastest of them.

Holger et al. (2008) presented a demo of ESTER, a search engine that combines the ease of use, speed and scalability of full-text search with the powerful semantic capabilities of ontologies. They demonstrated the capabilities of ESTER on a combination of the English Wikipedia with the Yago ontology, with response times below 100 milliseconds for most queries, and an index size of about 4 GB. The system can be run both stand-alone and as a Web application.

Bharat et al. (2009) proposed an efficient word searching algorithm through splitting and hashing the offline text. In this algorithm the offline text is splitting into number of tables in the preprocessing phase. After splitting the text into number of tables, they match the hash value of the pattern P with the hash values of the words of same length in the text T. This algorithm reduces the search time of word searching algorithm WSA which is suggested by (Ibrahiem and Mohammed, 2008) for solving the word matching problem by splitting the offline text into number of tables in the preprocessing phase. Ishadutta et al. (2009) proposed multi-patterns word searching algorithm (MPWSA). This algorithm is extended to WSA algorithm which is proposed by (Ibrahiem and Mohammed, 2008) and made it for multi-patterns word matching by using the technique of bit-parallel proposed by (Baeza and Gonnet, 1992). In this algorithm the shift-or algorithm is applied to find the words with the same length in a text and by using the concept of classes of characters the set of multiple patterns is handled. Results show that MPWSA algorithm is faster than the proposed WSA algorithm.

Minnie and Srinivasan (2011) proposed intelligent search engine algorithms on indexing and searching of text documents using text representation. The indexing of text documents is done by classifying a text document based on its content and the presentation style of the content of the text document. Seven indexing algorithms and two searching algorithms are implemented.

Kwang et al. (2011) suggested an efficient string searching algorithm depending on characteristics of English alphabets combination in a pattern and a text like frequency and position of vowels in a pattern and a text. In this algorithm the pattern's length has no affect in the algorithm's performance, while pattern's location in a text influences on the algorithm's time. Using this algorithm un-matching vowels in a text and useless comparison between a pattern and a text can be avoided, which makes this algorithm useful and effective one in searching.

This paper gives a new search algorithm for documents using blocks and words prefixes. By using this algorithm in search engines the process of search will be more accurate and faster.

THE NEW SEARCH ALGORITHM FOR DOCUMENTS

Search algorithm using blocks and words prefixes

This algorithm depends on the concept of word prefix, the letters at the beginning of the word. The first four letters of the search word are used as indexes to determine where the search starts.

The idea of this algorithm is that words with length three or four are not stored in the search blocks and words with length more than four are stored in blocks ordered by alphabet. And when we made a search for a word, we do not make the search from the beginning of the block, we make it from the first word that has the same prefix as the search word in the suitable block. Ex. If we have in block < T> the following words “tabernacles, table, tableau, tables, tablet,……., thanks, thawed, theatre, their, theme, themselves, theory, there, thereabouts, thereby”, etc. And we want to search for, “there”, we will not search from the beginning of <T> block, “tabernacles”, we will search from the word *theatre* position which contains the first word with “the” prefix. This will reduce the number of comparisons and speed the process of searching.

The method of this algorithm is given below:

For any web text (document) you want to search from it, “book, story, articles, novel or any other texts", the following steps for preprocessing are done.

1. Tokenization where the input is taken as a plain text document and a set of tokens are produced. White space, punctuation marks and Full stops are used as word separators.
2. Delete repeated word.
3. Delete Empty tokens, numbers and critics mark.
4. Sort tokens by alphabet and stored them in a main array.
5. A 4 dimensional array is created, called prefix_array, with Boolean value which considers their indexes as the position of word prefix. Ex. for the word “the” the index of the prefix_array will be [20, 8, 5, 0]. Where t=20; h=8; e=5; the fourth dimension will be zero in case of words of length 3; and for the word "they" the index of the prefix_array will be [20, 8, 5, 25] Where y=25 and the same for all the alphabets letters starting from a =1 and ending with z = 26. The value of this array will be 1 or 0, if there is a word with the determined word prefix, the value for the array will be one otherwise zero. Ex. if the document contains the following words the, there, these, those; this will lead that the prefix_array in [20, 8, 5, 0] will equal 1. We put zero value as an initial value for the prefx_array. Also a 4 dimensional array is created, called flag array with Boolean value which indicates that the word in the determined indexing is full word (1) or prefix (0). Ex. they , those are two words in the document the flag array of they [20,8,5,25] =1 "complete word", while

the flag array of thos [20,8,15,19]= 0 "Just prefix".
6. A 4 dimensional array is created, called position_array, with integer value. This array is responsible for storing the index of the first occurrence of the prefix of the word if only the prefix_array for the word is holding the value 1 and the indexes of this array as same as the prefix_array. Ex. If we have in block <T> the following words tableau, tables, table, thanks, thawed, theatre, their, theme, themselves, theory, there, thereabouts, thereby, etc. The prefix_array in [20, 8, 5, 0] will be 1 so position_array in [20, 8, 5, 0] will be 6 which indicated to the index of the first word which has the prefix “the” in it.
7. The process of storing words in the appropriate blocks is as following:

7a. Words with length 3 and words with length 4 are not inserted to the block and insert the value 1 for the prefix_array in the correct index of the word. Also insert 1 to flag array to indicate that the word is complete and it is not a prefix. Ex. If the word “fan” is in the document, we will not insert it to the block and insert in prefix_array at index [6, 1, 14, 0] the value 1.

Then check the next word in the main array if its prefix is as same as the word which we add 1 for it in the prefix_array we put it in the appropriate block and put its index in the position array for the first occurrence of the prefix only. The process of checking next word prefix is continued, and each time the word is inserted to the appropriate block with no inserting in prefix array if the prefix as same as the word which we put its index in the position array, until the prefix of the word is become different.
7b. Words with length more than 4 are inserted to the block with two additional steps. First insert the value 1 for the prefix_array in the index of each word. Ex. If the word “these” is in the document, we will insert it to the block <T> and insert in prefix_array at index [20, 8, 5, 0] the value 1. Second, insert the index of the word in the block, where the first occurrence of the prefix of the word is occurred, in the position_array if and only if the prefix_array at that index is 1. Then as step 7a the process of checking next word prefix is continued, and each time the word is inserted to the appropriate block with no inserting in prefix array if the prefix as same as the word which we put its index in the position array, until the prefix of the word is become different.

Figure 1 shows the steps of the preprocessing of the algorithm. So when the user wants to search about any word in the text (document) using this algorithm , first the length of the word is taken ,if its three or four only one step is made by searching in prefix_array and flag array in the index of the word, otherwise if the length is more than 4 the search will be made only from the index of the word that has the first occurrence of the prefix of the search word in the appropriate block, this leads to enhance the performance of the search process and make it faster as written in algorithm 1.

Figure 2 shows the mechanism of the algorithm in details. For example suppose a book contains of 500000 words and does not contain any word starting with **v** letter.

In the sequential search, the result “NOT FOUND” will appear after 500000 comparisons. But, in this algorithm, the prefix of the word is taken and search for it in prefix_array in the appropriate index, the result of it will be zero that means there is no word with this prefix. This is done only in one step.

IMPLEMENTATION OF THE NEW SEARCH ALGORITHM

The text search algorithm is implemented using Microsoft visual studio 2010 using c sharp language. The interface of the program is shown in Figure 3.

An English novel called MOBY DICK, which is available at (http://www.gutenberg.org/files/2701/2701-h/2701-h.htm), is copied into text file as Figure 4, the size of the novel is 1340 KB and it contains of 192608 words with 16017 distinct words, 392 of them are with length 3 and 1261 of them are with length 4.

The novel is considered as an input to the program to test the efficiency of our algorithms. By applying the first 3 steps of preprocessing, the number of the words is minimized from 192608 to 16017 which is a great difference. That is, 176591 words are deleted using our algorithm; they are repeated words, empty tokens, numbers, critics mark or words with length less than 3.

By applying the remaining steps, words will be arranged in blocks as illustrated in “the new search algorithm for documents” part of this work. Table 1 shows the 26 blocks of our algorithm and how many words in each block.

The total number of words in A-Z blocks is 14364 words. The 1653 words, which considered 13% of the total distinct words, with length 3 and 4 are not inserted.

Finally the user can insert the number of words that he/she wants to search for, in the text box, and then by pressing Load words from file button, all the search words are loaded to the program. Note, you must have all the search words in a txt file so the program can load them. Finally he/she presses applying text search algorithm button, the result will print in a text box containing the number of comparison that are made and the time that is taken to execute the searching process as Figure 5 shows.

COMPARISON BETWEEN OUR ALGORITHM AND BINARY SEARCH

In order to check the validity and the performance of our algorithm we compare the output results of the proposed algorithm (as explained in “the new search algorithm for documents” part of this work) with the binary search.

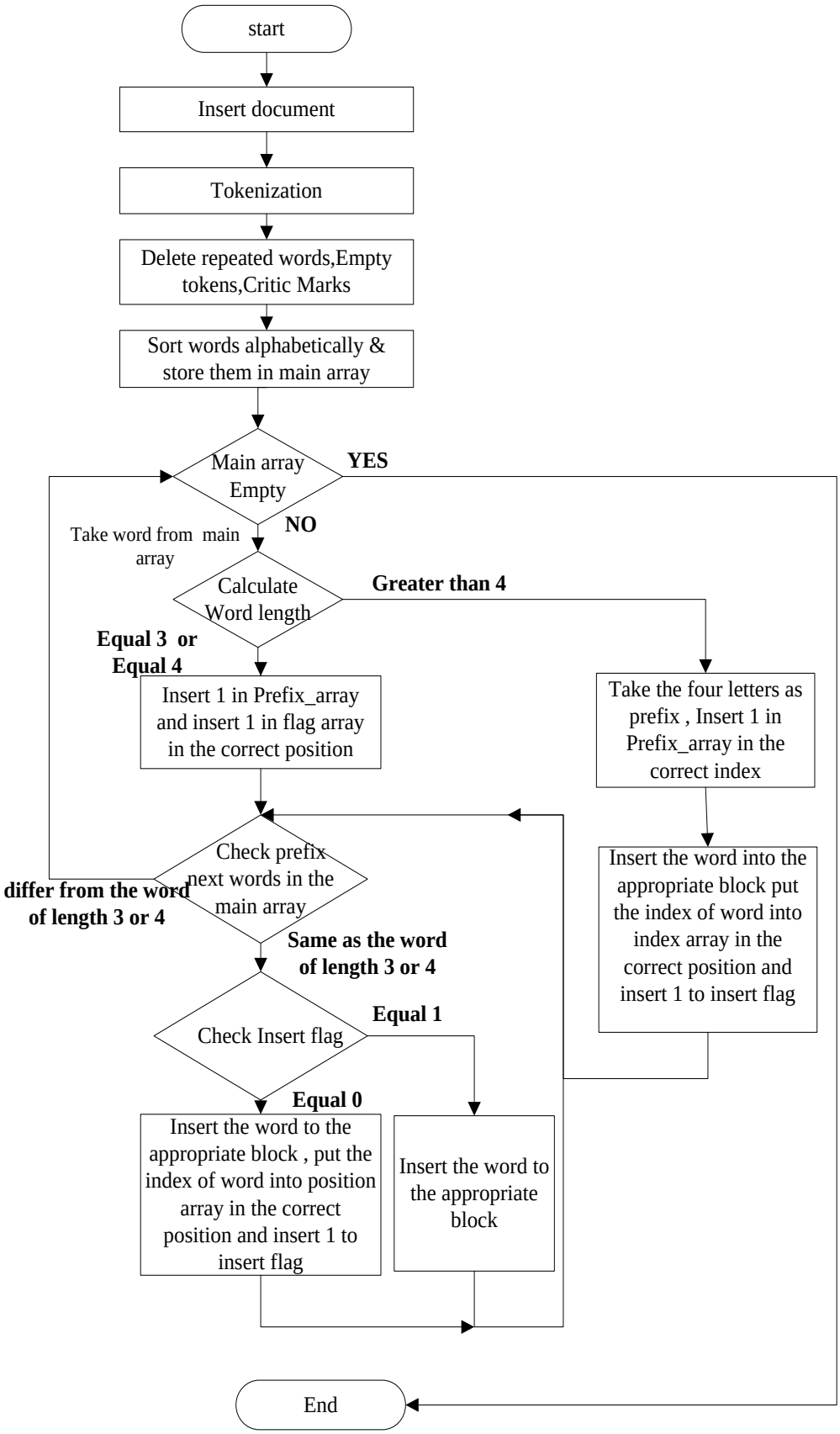

Figure 1. Preprocessing steps for algorithm.

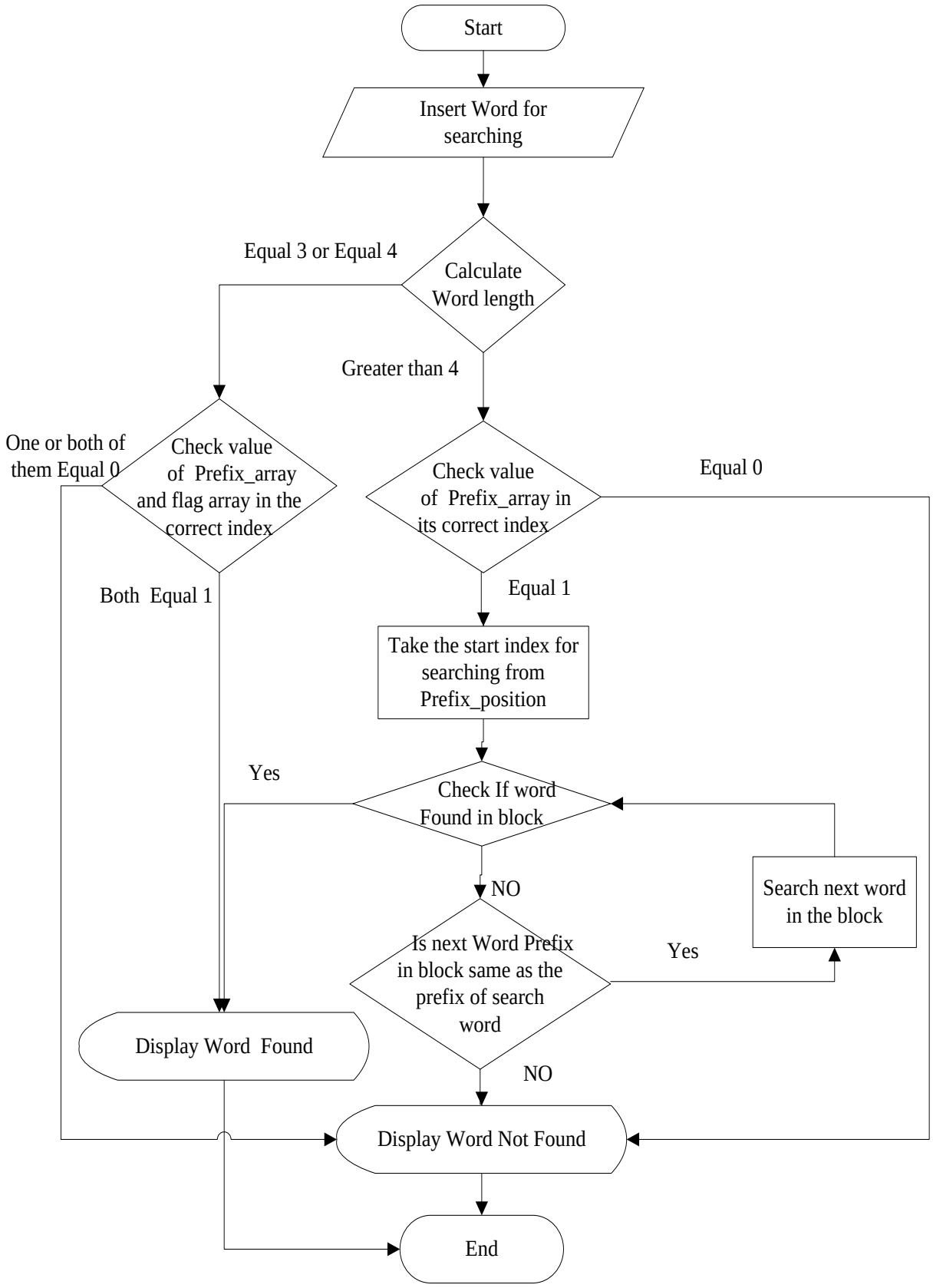

Figure 2. Search algorithm using blocks and words prefixes.

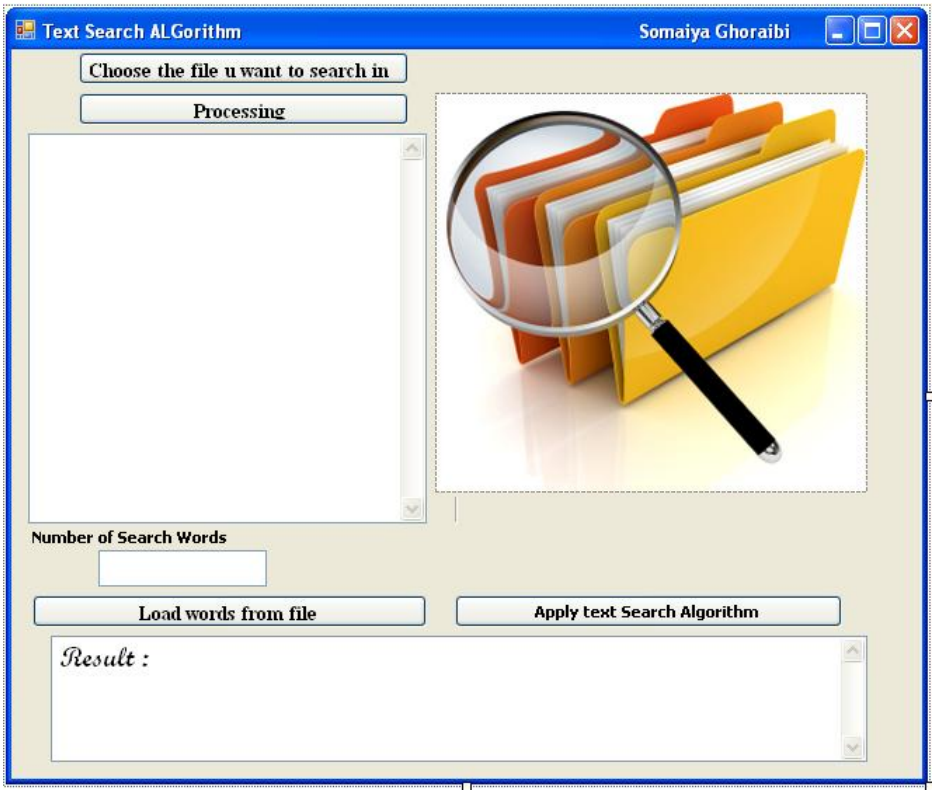

Figure 3. The interface of the implementation of the suggested algorithm.

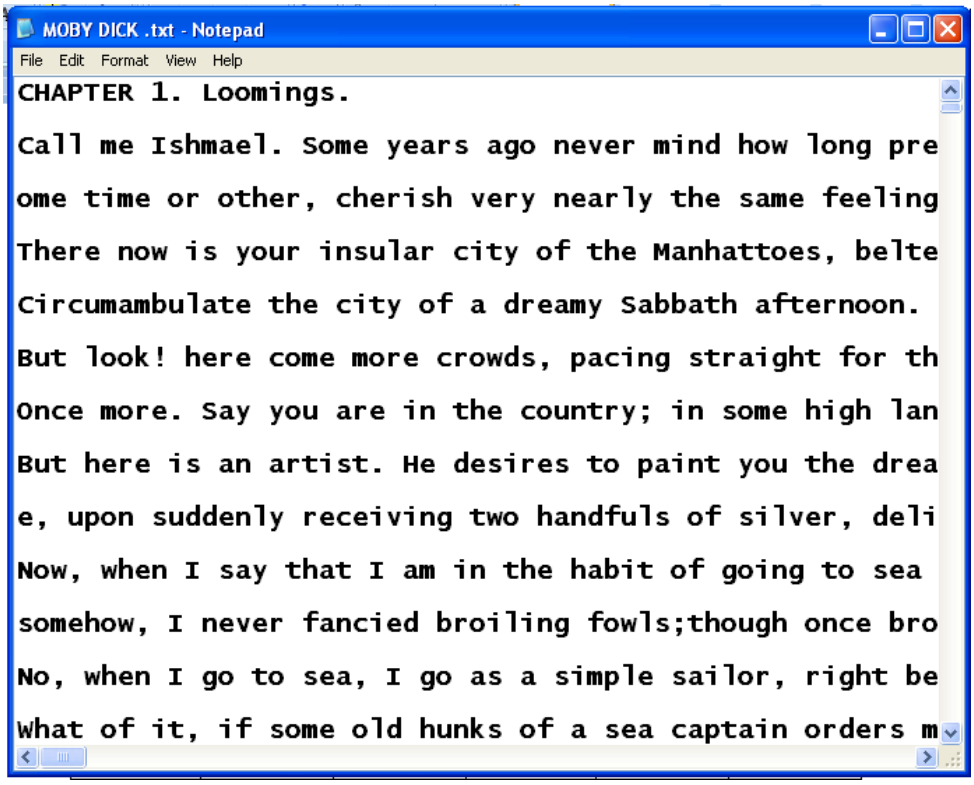

Figure 4. The Moby Dick novel for testing.

Table 1. Showing A-Z blocks in the two suggested algorithms.

Block name	# of words in our algorithm	Block name	# of words in our algorithm
A	829	N	210
B	863	O	307
C	1428	P	1137
D	823	Q	74
E	620	R	710
F	666	S	1844
G	424	T	686
H	527	U	465
I	669	V	236
J	121	W	476
K	83	X	1
L	431	Y	37
M	688	Z	9

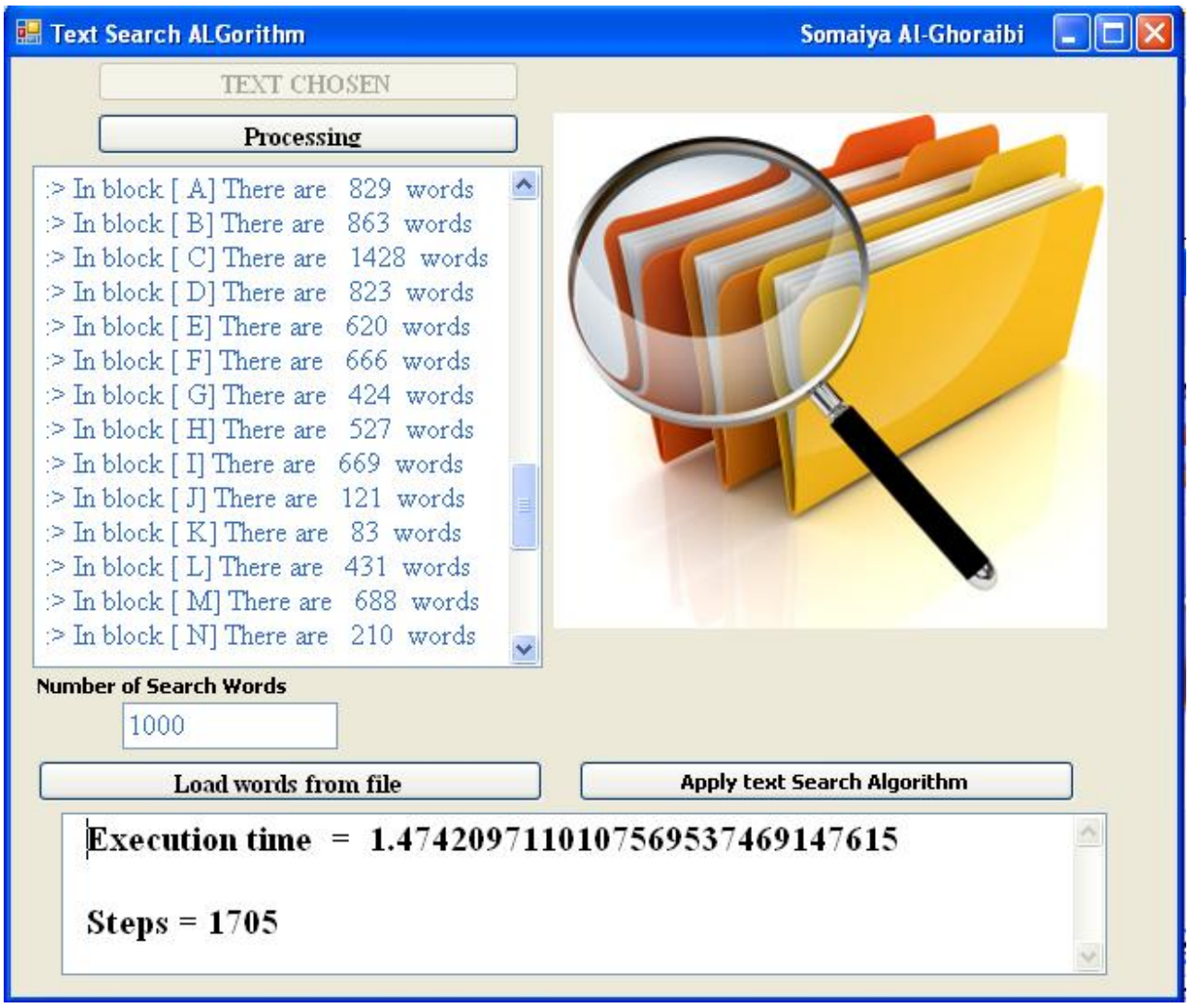

Figure 5. The Interface of the program showing the result of searching for 1000 words.

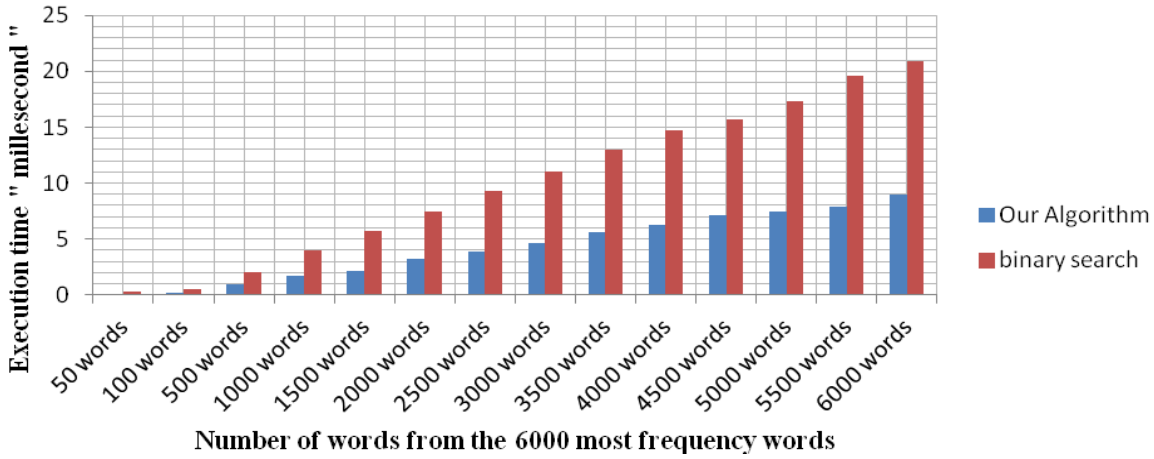

Figure 6. Execution time of sequential search and the proposed algorithms.

Table 2. Number of comparisons of binary and the proposed algorithm.

Number of word	No. of comparisons our algorithm	No. of comparisons binary search
50	77	697
100	162	1397
500	856	6987
1000	1705	13977
1500	2563	20968
2000	3420	27950
2500	4253	34936
3000	5073	41926
3500	5900	48917
4000	6721	55909
4500	7553	62892
5000	8385	69880
5500	9212	76872
6000	10060	83853

Binary Search algorithm is one of the best algorithms for searching. We use execution time and number of comparisons as performance parameters to compare between the performances of them.

We make the test for the previous novel, Moby Nick, which is shown in Figure 4 and download the most 6000 frequency words (http://www.wordfrequency.info/). We put the 6000 words in a text file to be called from our program. The search starts from 50 words until 6000 words.

Figure 6 shows the comparison between our algorithm and binary search in the execution time (in millisecond) parameter. From this Figure, we can see that our algorithm takes less time compared to binary search. Our algorithm is 66% faster than binary search.

Also the comparison parameter proves the great performance of our algorithm as Table 2 shows.

Table 2 presents the number of comparisons for words when the search is made for the previous novel. The very big difference between our algorithm and binary search concludes that our algorithm works efficiently.

CONCLUSION AND FUTURE WORK

With the rapid increasing of the document in World Wide Web and everywhere, a need for search algorithm becomes an important issue. In this paper a new search algorithm for documents using blocks and word prefixes is presented. An implementation for the proposed algorithm is given. Finally to find that our algorithm works well and to prove the benefit of our new algorithm we made a comparison between it and binary search algorithm. Our algorithm outperforms binary algorithm in

the execution time and number of comparisons parameters, which prove its efficiency. For the future work, we plan to make this algorithm applicable for Arabic language. We also plan to develop the program to be extracting text from web page instead of copying the web text into a text file.

REFERENCES

Baeza R, Gonnet G (1992). A new approach to text searching. Commun. ACM, pp. 74-82.

Bharanipriy V, Prasad V (2011). Web Content Mining Tools: A Comparative Study. Int. J. Inf. Technol. Knowl. Manage. 4(1):211-215.

Bharat S, Ishadutta Y, Suneeta A, Rajesh P (2009). An Efficient Word Searching Algorithm through Splitting and Hashing the Offline Text. Recent Technologies in Communication and Computing, 2009. ARTCom '09. International Conference 27-28 Oct. 2009, pp. 387-389.

Holger B, Fabian S, Ingmar W (2008). Semantic Full-Text Search with ESTER: Scalable, Easy, Fast. Data Mining Workshops, 2008. ICDMW '08. IEEE International Conference 15-19 Dec. 2008, pp. 959-962.

Ibrahiem E, Mohammed J (2008). A New Approach for Solving String Matching Problem through Splitting the Unchangeable Text. World. Appl. Sci. J., pp. 626-633.

Ishadutta Y, Bharat S, Suneeta A, Rajesh P (2009). An Efficient Bit-Parallel Multi-Patterns Word Searching Algorithm through Splitting the Text . Recent Technologies in Communication and Computing. ARTCom '09. International Conference 27-28 Oct. 2009, pp. 406-410.

Kulekci M (2007). Tara: An algorithm for fast searching of multiple patterns on text files. Comput. Information Sciences (ISCIS) 2007. 22nd International Symposium, pp. 1-6.

Kwang C, Heon Y, Sung J (2011). An Efficient String Searching Algorithm Based on Vowel Occurrence Pattern. Commun. Comput. Inf. Sci. 185:379-386.

Minnie D, Srinivasan S (2011). Intelligent Search Engine algorithms on indexing and searching of text documents using text representation. Information Systems (ReTIS), 2011 International Conference, pp. 121-125.

Qingyu Z, Richard S (2008). Web mining: A survey of current research, Techniques, and software. Int. J. Inf. Technol. Decis. Making 7(4):683-720.

Raymond K, Hendrik B (2000). Web Mining Research: A Survey. SIGKDD Explor. 2(1):1-15.

XiangFeng W, Quan Z (2007). Approach of Text Search Based on Semantic Parsing Model. Fuzzy Systems and Knowledge Discovery (FSKD) 2007. Fourth International Conference, pp. 355-359.