PARTITION-BASED PATTERN MATCHING APPROACH FOR EFFICIENT RETRIEVAL OF ARABIC TEXT

Saqib Hakak, Amirrudin Kamsin*, Palaiahnakote Shivakumara, Mohd.Yamani Idna Idris

Faculty of Computer Science and Information Technology
University Malaya, Kuala Lumpur, Malaysia

Email : saqibhakak@siswa.um.edu.my, amir@um.edu.my*, shiva@um.edu.my, yamani@um.edu.my

DOI: https://doi.org/10.22452/mjcs.vol31no3.3

ABSTRACT

Encoding for Arabic based on the Unicode Transformation Format (UTF) differs from encoding for English based on the American Standard Code for Information Interchange (ASCII) since the Arabic usage of diacritics, symbols and elongated characters makes searching more challenging in the field of information retrieval. In this paper, we propose a new partition-based pattern matching approach that divides the query words into two equal parts (sub-parts). The proposed approach treats the two divided sub-parts as independent query words and uses a parallel search to match the content in the database. In addition, the proposed approach modifies the conventional brute force pattern matching to speed up the searching process which results in efficient text retrieval from any database. The experimental results are used to evaluate the proposed approach in terms of processing time. The comparative analysis of the existing approaches and the proposed approach reveals that the proposed approach outperforms all other existing approaches in terms of computational time.

Keywords: Partition-based pattern matching, exact matching, Arabic texts, short patterns, digital Quran, information retrieval.

1.0 INTRODUCTION

Searching through pattern matching constitutes a widely-used technique including information retrieval for several applications such as Deoxyribonucleic Acid (DNS) sequence searching, image processing, medical image searching, etc [1-4]. In most applications, pattern matching is developed to search an English text in the database, yet hardly an Arabic text. According to the Web of Science statistics shown in Fig. 1, a number of algorithms have been proposed and evaluated using different datasets. The main tested datasets include natural language texts like Bible texts, protein sequences including DNA sequences, Rand texts and Snort datasets. All these texts apply the ASCII-based encoding scheme that uses seven bits to represent a particular character [1].

Elaborate Arabic script patterns are complex to search and retrieve compared to ASCII-based texts [2-4]. Searching and retrieving diacritical (vowelized) Arabic patterns are more complex than simple Arabic pattern [4]. The presence of symbols, diacritical signs, elongated characters and other such elements increase the complexity of searching Arabic texts and also the time complexity [5, 6]. Digital Qur'an text which is written in Arabic constitutes a very suitable example given its highly complex diacritical texts [7-10]. These diacritical texts contain many additional symbols that decrease the retrieval process and increase the search time. The search time can be improved by removing all symbols and diacritics, however, this will obscure the meaning of the Qur'anic Verses. Arabic Qur'an verses are very sensitive to the arrangement of diacritics. The removal of a single diacritic or symbol can change the meaning of the whole verse [11-14]. Thus, it is imperative to search for a diacritical pattern without removing any of the symbols or diacritics.

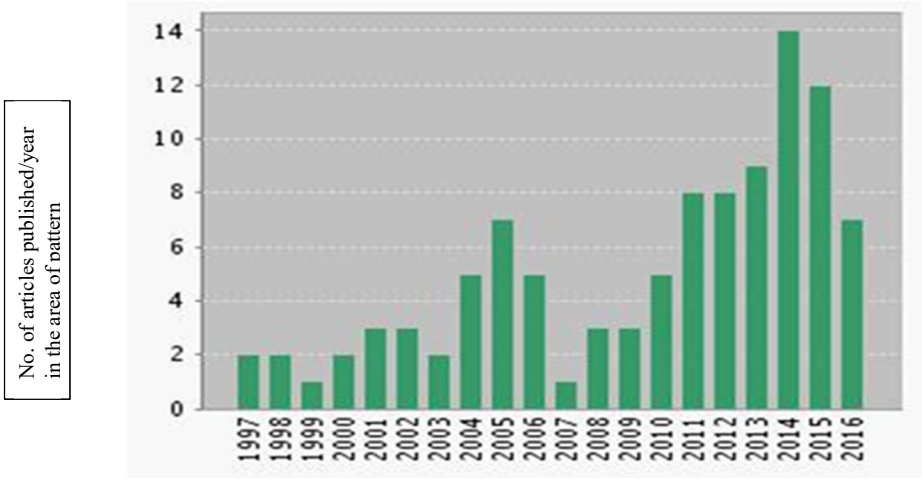

Fig. 1: No. of articles published in the area of pattern matching

The organisation of the paper is as follows: Section 2.0 presents the pattern matching approaches and our review on the existing methods, while the proposed methodology is explained in Section 3.0. The experimental results are discussed in Section 4.0, followed by the conclusion in Section 5.0.

2.0 LITERATURE REVIEW

The methods involving pattern matching can be classified into two categories: 1) single pattern matching; and 2) multi-pattern matching. The single pattern matching approach identifies a single pattern from the whole database while the multi-pattern approach identifies multiple patterns from a single database [15, 16]. This study focuses exclusively on single pattern matching as multi-pattern matching extends beyond its scope.

The workable methods based on single pattern matching approaches can be classified into four categories: character-based, hash-based, automata-based and bit-parallelism approaches [17]. The character-based approach includes the two key stages of search and shift. Since character-based approaches work at the character level, the methods are computationally expensive. Hash-based approaches find hash values for the characters to match rather than the characters themselves. However, these methods possess certain implementation difficulties. Automata-based approaches involve automata theory for finding states as suffix for matching. These methods generally achieve better results yet consume more memory due to the necessary state diagram construction and traversal. The final approach, bit-parallelism, involves parallel processing to speed up the matching process [19, 20]. However, it requires dividing the task into sub-tasks.

It follows from the above discussion that character-based approaches are in general simpler and easier to implement than other approaches due to ease of manipulating shifts. The standard algorithm in the area of pattern matching following the character-based approach is Boyer-Moore (BM) [18]. In BM algorithm, searching starts from right to left of any given text. The scanning stops once the complete match has been found. However, in the case of a mismatch, it uses a shifting process popularly known as 'bad shift' and 'good shift' [21]. These shifting tables need a lot of pre-processing in terms of calculations such as *how many characters need to be skipped if there is a mismatch*, *how many characters need to be skipped if there is a match* and so on. Many attempts have been made to improve BM algorithms in terms of their required search time, for example the Horspool algorithm [22], the Tuned BM algorithm [23], and the Turbo BM algorithm [24]. The Turbo-BM Algorithm is based on the dynamic simulation technique. The time complexity of all of the above-mentioned algorithms is presented in the next section. Since our aim is to achieve time efficiency for searching Arabic text from the database, we review the time complexity of the existing methods in this section.

In this study, the worst-case scenario refers to Big O notation. Let p be the pattern to be searched with length m and t the source text of length n. These are the symbols used for reviewing time complexity of the existing methods as reported in Table 1.

Boyer-Moore [18]: It needs two rules for identifying and locating the pattern, namely good suffix rule and bad character rule. Let the necessary shift to be used in case of mismatch be denoted by s. Thus, in case of mismatch, the good shift rule aligns the pattern p of length m over text t in such a way that $t[s+i+1\ldots\ldots s+m-1] = p[i+1\ldots m-1]$. In order to calculate the number of shifts, the BM algorithm needs pre-processing that occupies $O(m)$ of space and in the worst-case scenario takes $O(mn)$ of time. Hence, the overall complexity of BM becomes $O(m)[n+1]$.

Tuned Boyer-Moore (BMT) [23]*:* This algorithm consists of two phases, last character localisation and matching. The first phase consists of the searching pattern $p(m-1)$ using three rounds of blind shifts based on bad character rule of the BM algorithm. In the matching phase, pattern $p(0\ldots m-2)$ is tried to obtain a match with the corresponding characters of the given text t. This algorithm has the quadratic time complexity denoted as $O(n^2)$ in the worst-case scenario and can increase further. Like the BM algorithm, BMT also needs $O(m)$ of space, thus taking overall complexity of $O(n^x+m)$, where x denotes the exponential power.

Simple String Matching (SSM) algorithm) [26]: This algorithm represents a modification of the Horspool [22] algorithm that is in turn a modification of the BM algorithm where the good suffix rule is dropped. The increments in the shifts are based solely on the bad character rule. The algorithm is based on the observation of the first mismatch, i.e. if there occurs a mismatch between text $t\ [s\ldots.s+m-1]$ and pattern p occurs at $0 <= i < m$ and the occurrence of right-most character of $t[s+i]$ in p is at position of $j > i$ and the shifting process can be done backwards. This algorithm again involves pre-processing for computing all these shifts and takes $O(m)$ space with sub-linear time complexity. Thus, the overall time complexity is $O(m)[n+1]$.

Brute Force Algorithm: This is one of the simplest algorithms that do not involve any pre-processing. It searches whole pattern p from a text t character by character serially until the whole pattern is found. The only limitation of this approach is its time complexity. The overall time complexity of this algorithm is linear i.e. $O(mn)$. However, the same algorithm can perform faster on more advanced machines where instruction set and processing speed is massively increased. Our proposed approach is based on same assumption that brute force algorithms can perform better on modern machines for UTF based texts like Arabic [18-20].

Let t_L be the look up time taken by the above mentioned exact matching algorithms for shifting a pattern p from a given text t. For n mismatches, the existing exact matching algorithms need $n(t_L)$ of time to decide when to shift and when not. This changes the overall time complexity of the above-mentioned algorithms to:

$$O(m)[n+1] + t_L$$

where t_L can be negligible too in case p is found in the first attempt.

However, in the worst case, it will be $n_{\max}$ times. This factor of t_L that arises due to pre-processing can be avoided following the brute force approach as it does not involve any pre-processing. Although all these algorithms have competitive time complexities, the performance of these algorithms differs based on the different encoding techniques. To evaluate their performance, we conducted an experiment to make sure the different encoding technique results in the poorer performance of the existing matching algorithms as shown in Table 1. A sample Qur'anic verse was taken from corpus [27] and UTF encoding scheme was used to find a pattern from a given text. Similarly, an ASCII-based English text from the Bible was taken to compare the performance. As shown in Table 1, the initial assumption was proven to be correct, and the existing exact matching algorithms performance degrades for UTF based texts like Arabic. This is due to the extra symbols and characters used in these texts that occupy extra bits, thus increasing search time [28-29]. However, the brute force approach performed better for both types of texts. This factor motivated us to propose an algorithm that uses a character-based approach of query pattern for searching in the database.

Table 1: Effect of Encoding Schemes on Existing Pattern Matching Algorithms

Algorithm	Arabic texts (UTF Encoding)	Length of Text	Pattern Searched	Time (in milliseconds)	English texts (ASCII Encoding)	Length of Text	Pattern Searched	Time (in milliseconds)
BM	صِرَٰطَ ٱلَّذِينَ أَنۡعَمۡتَ عَلَيۡهِمۡ غَيۡرِ ٱلۡمَغۡضُوبِ عَلَيۡهِمۡ وَلَا ٱلضَّآلِّينَ	90	لَا	**12.6**	Thus, the heavens and the earth were finished and all the host of them on the seventh day God	90	day	**7.9**
BMT	صِرَٰطَ ٱلَّذِينَ أَنۡعَمۡتَ عَلَيۡهِمۡ غَيۡرِ ٱلۡمَغۡضُوبِ عَلَيۡهِمۡ وَلَا ٱلضَّآلِّينَ	90	لَا	**10.9**	Thus, the heavens and the earth were finished and all the host of them on the seventh day God	90	day	**6.3**
SSM	صِرَٰطَ ٱلَّذِينَ أَنۡعَمۡتَ عَلَيۡهِمۡ غَيۡرِ ٱلۡمَغۡضُوبِ عَلَيۡهِمۡ وَلَا ٱلضَّآلِّينَ	90	لَا	**13.9**	Thus, the heavens and the earth were finished and all the host of them on the seventh day God	90	day	**10.8**
Brute Force	صِرَٰطَ ٱلَّذِينَ أَنۡعَمۡتَ عَلَيۡهِمۡ غَيۡرِ ٱلۡمَغۡضُوبِ عَلَيۡهِمۡ وَلَا ٱلضَّآلِّينَ	90	لَا	**4**	Thus, the heavens and the earth were finished and all the host of them on the seventh day God	90	day	**4.6**

Based on the above discussion, it can be established that the primary focus of all exact matching algorithms is to reduce search time. It is also observed from the review that the existing methods use English text for searching. Therefore, the same approach may not be useful for searching Arabic text from the databases. Hence, we propose a new method for searching Arabic text, which consumes less time compared to the existing methods.

3.0 PROPOSED METHODOLOGY

According to the proposed methodology presented in Fig. 2, there are two main phases for searching the Arabic texts which are, Tokenisation and Searching. In tokenisation phase, the whole input pattern *p* is split into individual character components as shown in Fig.2. The reason for splitting the pattern is to ease the search process involving diacritical texts. The proposed approach estimates total length of input pattern and it divides into two sub-patterns based on finding mid-point of the input pattern. The output of tokenisation is passed to the next phase, searching. In this phase, the two sub-patterns, *left halve* and *right halve* are processed simultaneously. Both halves i.e. *left* and *right* search their specific patterns simultaneously from the given text. From the result, it saves lots of comparisons and memory consumption compared to the existing pattern matching algorithms. The reason of consuming less memory compared to the existing character based approaches is that the proposed method does not require pre-processing such as information about shifting when their correct match is found and mismatch found during searching. This pre-processing leads to high memory consumption and requires more time as it uses look up for mismatch.

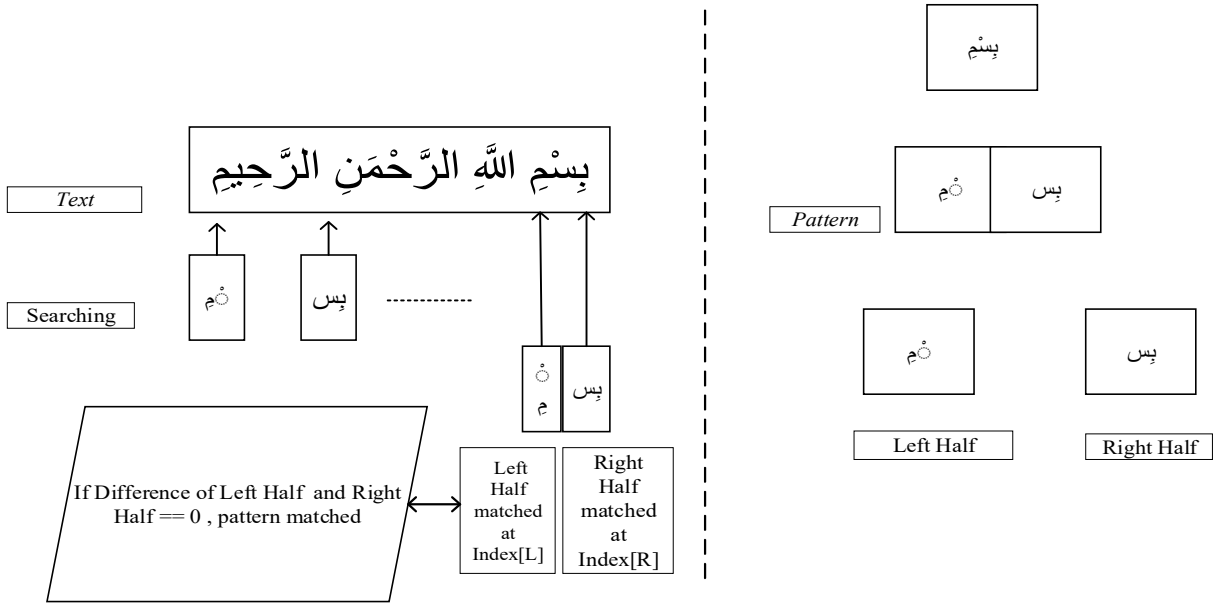

Fig. 2: Partition-based approach to search Arabic diacritical texts

The steps of the proposed approach are illustrated in Fig. 3 and Fig. 4. In Fig. 3, the given text is "SAQSAQI" and pattern needed to be found is "SAQI". According to the proposed approach, the whole input pattern will be split into two sub-patterns i.e. left halve having pattern "SA" and right halve having pattern "QI". After the split process, both patterns are searched simultaneously. In Fig 3, left halve is found at index (1) and right halve is found at index (6). In order to find the exact pattern, both halves must have zero difference between each other. However, in the present scenario, it can be seen that the difference between two halves is greater than zero indicating mismatch.

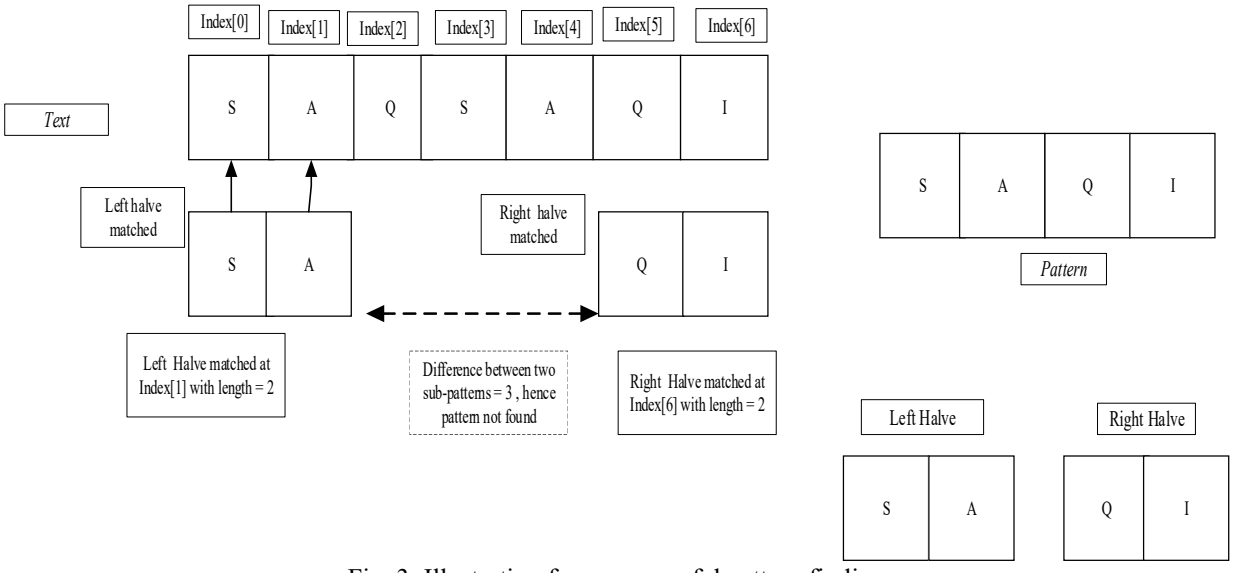

Fig. 3: Illustration for unsuccessful pattern finding

In the first scenario, there was no match. The same procedure is repeated as shown in Fig. 4. However, in this case there is no difference between left halve and right halve, thus indicating a complete match.

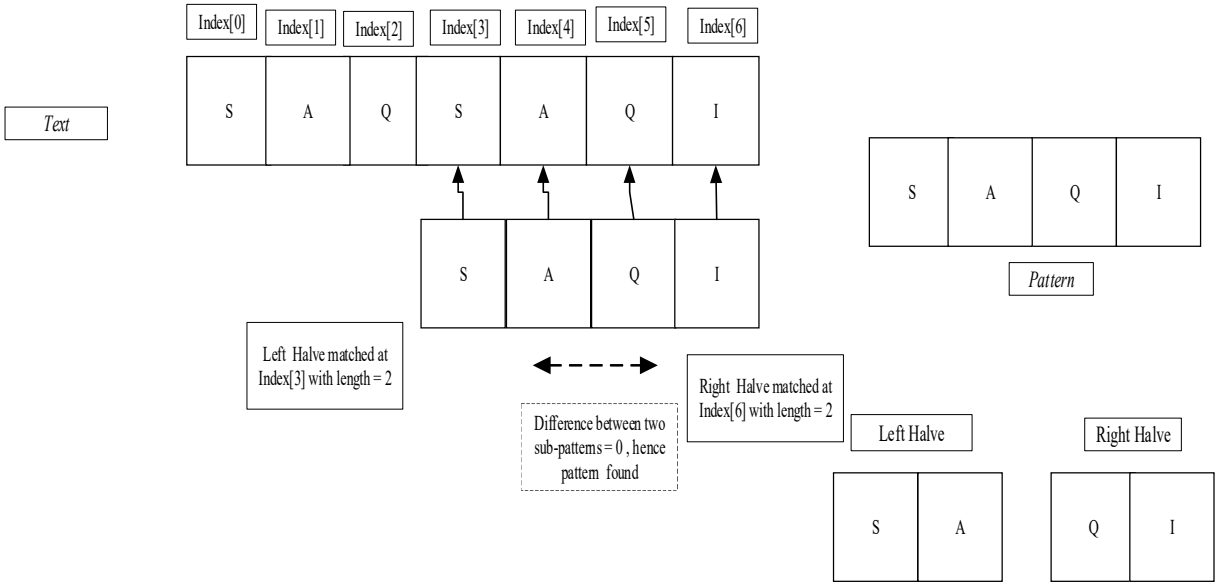

Fig. 4: Illustration for successful pattern finding

The pseudo-code of the proposed algorithm is shown in Fig. 5.

```
1.Input Pattern= " "
2.Given text = " "
3.Compute the length(L) of pattern
4.Compute the length(T) of text
5.Divide the pattern into two equal halves i.e. left and right with respective length l and r
6.Compute l of left part and r of right part
7.Search right part
      for ( i = 0; i <= T - r; i++)
         for (j = 0; j < r; j++) {
            if (text.charAt(i+j) != r.charAt(j))
               break;
         if (j ==r)
Display (Right Pattern Found)
8. Search left part
      for ( i = 0; i <= T -l; i++)
         for (j = 0; j <l; j++) {
            if (text.charAt(i+j) != l.charAt(j))
               break;
         if (j == l)
Display (Left Pattern Found)
9.Compute the differences between locations of two parts i.e. left and right part.
If difference == 0 (pattern found)
```

Fig. 5: Pseudocode of Partition-based Algorithm

The logical steps of the proposed approach are shown in Fig. 5 where the lines 1-4 denote the length of input pattern and the given text is calculated. In the lines 5-6, the input pattern is split into a left and right half respectively, and the length of each sub-part is calculated. The search phase of both halves is represented in lines 7 and 8 respectively. Finally, if there is a difference of zero between the right and the left half, we will get the expected result as represented in line 9. If not, the whole process will be repeated (lines 7-9).

4.0 EXPERIMENTAL RESULTS

To evaluate the proposed algorithm, we consider a standard dataset of Qur'anic scripts from Tanzil.net [25]. To evaluate the proposed method in terms of time complexity, we use processing time in milliseconds as the performance measure. To demonstrate the effectiveness and usefulness of the proposed algorithm, we compare the results of the proposed algorithm with the results of the well-known existing algorithms in the Qur'anic dataset. The existing algorithms are: (1) Boyer-Moore (BM) algorithm that uses good suffix rule and bad character shift for matching a specific pattern; (2) an improved version of the BM algorithm known as BMT which combines the strengths of the BM and the KMP (Knuth-Morris) algorithms. Recently, a new character-based approach, namely the SSM has been proposed by [26]. This algorithm compares the pivot character with the corresponding character and shifts the pattern either through Horspool shift or hybrid shift.

For the experiment, we consider a very short query pattern (1-4-character length) and medium query pattern (>=4-character length) to test the time efficiency of the proposed and the existing algorithms on the Qur'anic dataset. The reason for selecting a short pattern is due to the nature of the Arabic text, particularly Qur'anic verses. The Qur'anic verses are usually medium in length ranging from four characters per word (medium) to a minimum of two characters (short) along with diacritics. Each algorithm is run 10 times, and the running times were calculated by taking the mean of 10 running times. The implementation is done using the NetBeans 8.02 on i-5 Intel Processor with 4 MB caches, 4 GB RAM using Windows 10.

The quantitative results of the proposed and existing algorithms for query pattern lengths on different datasets are reported in Table 2 and Table3. It is noticeable that the proposed algorithm outperforms the other existing algorithms in all patterns. Therefore, it can be argued that the proposed algorithm is effective for Arabic texts in terms of time efficiency.

Table2: Processing time of the Proposed and Existing Methods for Very Short Patterns in Seconds

Words	Boyer-Moore	BMT	SSM	Brute-Force	Proposed
بِسْمِ	1135	974	1015	1619	**855.3**
فِي	981	968	963	1290	**768**
هُوَ	1057	1013	894	1447	**789**
يَا	1241	1235	1229	1818	**777**
وَ	1257	1209	1248	1856	**796**
نَ	1135	**974**	1015	1619	1011.5

Table I: Processing time of the Proposed and Existing Methods for Medium Patterns in Seconds

Verses	BM	BMT	SSM	Brute force	Proposed
بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ	993	1004	998	1870	**884.3**
إِذْ قَالَ لَهُمْ أَخُوهُمْ هُودٌ أَلَا تَتَّقُونَ	898	908	865	1338	**818.9**
فَبِأَيِّ آلَاءِ رَبِّكُمَا تُكَذِّبَانِ	**841**	867	851	1385	922.4
فِيهِمَا عَيْنَانِ نَضَّاخَتَانِ	876	901	**838**	1285	900
قُلْ يَا أَيُّهَا الْكَافِرُونَ	792	780	804	1214	**750**

Although the time complexity of the proposed algorithm is linear meaning *O(n)*, the proposed algorithm performed better compared to the existing algorithms due to the simplicity of steps involved in the pattern matching. In the existing pattern matching algorithms, the shifting process is the major cause of time consuming. This proposed approach does not include shifting phase; thus, the T_L factor is negligible and the search time considerably reduced. Besides than that, the use of modern appliances like *i-5* or *i-7* processors pose another advantage that processes bits much faster and further improves the searching process.

5.0 CONCLUSION

This paper proposes a simple and fast algorithm for exact pattern matching to achieve better time efficiency regardless of query pattern length, dataset size and script. The proposed algorithm divides the given pattern length into two halves and searches for both halves simultaneously in a given text. Once the locations of both halves are located, the proposed approach computes the differences between them. The difference of zero between two halves indicates that the complete location of the pattern is being searched. In order to demonstrate the usefulness of the proposed approach, we conducted several experiments on an Arabic dataset by varying query word lengths. The experimental results of the proposed and the existing algorithms on the Arabic dataset for different query pattern length show that the proposed algorithm outperforms most of the existing algorithms in terms of time efficiency. Thus, we can establish that the proposed algorithm is in terms of script and query pattern length, suitable for Arabic texts. In the future, we aim to extend this approach for multiple patterns and implement the same technique using Graphical Processing Units (GPU). In addition, we also consider extending the same method for both Arabic and English.

Acknowledgement

This research work was supported by Faculty of Computer Science and Information Technology, University of Malaya under a special allocation of Post Graduate Fund and IPPP research fund (PG017-2015B)

REFERENCES

[1] A. McEnery and R. Xiao, "Character encoding in corpus construction," *Developing Linguistic Corpora: a Guide to Good Practice*, AHDS, Oxford, 2005, pp. 47-58.

[2] B. Elayeb and I. Bounhas, "Arabic Cross-Language Information Retrieval: A Review," *Acm Transactions on Asian and Low-Resource Language Information Processing,* Vol. 15, No. 3, Mar 2016, p. 18.

[3] A. Alfaifi and E. Atwell, "Comparative evaluation of tools for Arabic corpora search and analysis," *International Journal of Speech Technology,* Vol. 19, No. 2, 2016, pp. 347-357.

[4] A. S. Metwally, M. A. Rashwan, and A. F. Atiya, "A multi-layered approach for Arabic text diacritization," in *Cloud Computing and Big Data Analysis (ICCCBDA), 2016 IEEE International Conference on*, 2016, pp. 389-393.

[5] A. Abdelali, J. Cowie, and H. S. Soliman, "Arabic information retrieval perspectives," in *Proceedings of the 11th Conference on Natural Language Processing, Journes d'Etude sur la Parole-Traitement Automatique des Langues Naturelles (JEP-TALN)*, 2004, pp. 391-400.

[6] K. Darwish and W. Magdy, *Arabic information retrieval*, Now Publishers, 2014.

[7] A. Farghaly and K. Shaalan, "Arabic natural language processing: Challenges and solutions," *ACM Transactions on Asian Language Information Processing (TALIP),* Vol. 8, No. 4, 2009, pp.1-22.

[8] E. F. Khalaf, K. Daqrouq and A. Morfeq, "Arabic Vowels Recognition by Modular Arithmetic and Wavelets using Neural Network," *Life Science Journal,* Vol. 11, No. 3, 2014, pp. 33-41.

[9] Y. M. Alginahi, O. Tayan and M. N. Kabir, "Verification of Qur'anic Quotations Embedded in Online Arabic and Islamic Websites," *International Journal on Islamic Applications in Computer Science And Technology,* Vol. 1, No. 2, 2013, pp. 41-47.

[10] A. Al Gharaibeh, A. Al Taani and I. Alsmadi, "The usage of formal methods in Quran search system," in *Proceedings of international conference on information and communication systems, Ibrid, Jordan*, 2011, pp. 22-24.

[11] A. Alshareef and A. E. Saddik, "A Quranic quote verification algorithm for verses authentication," in *Innovations in Information Technology (IIT), 2012 International Conference on*, 2012, pp. 339-343.

[12] A. Mohammed, M. S. Sunar and M. S. H. Salam, "Quranic Verses Verification using Speech Recognition Techniques," *Jurnal Teknologi,* Vol. 73, No. 2, 2015, pp. 99-106.

[13] N. J. Ibrahim, "Automated TAJWEED checking rules engine for Quranic verse recitation," *Doctoral dissertation*, Computer Science, University of Malaya, 2010.

[14] A. Arslan, "DeASCIIfication approach to handle diacritics in Turkish information retrieval," *Information Processing & Management,* Vol. 52, No. 2, 2015, pp. 326-339.

[15] A.Hudaib, D. Suleiman, M. Itriq, and A. Al-anani, "A fast pattern matching algorithm with two sliding windows (TSW)," *Journal of Computer Science,* Vol. 5, No.4, 2008, pp. 393-401.

[16] K. M. Alhendawi and A. S. Baharudin, "String Matching Algorithms (SMAs): Survey & Empirical analysis," *Journal of Computer Sciences and Management,* 2013.

[17] J. Yuan, J. Zheng and S. Ding, "An improved pattern matching algorithm," in *Intelligent Information Technology and Security Informatics (IITSI), 2010 Third International Symposium on*, 2010, pp. 599-603.

[18] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Communications of the ACM,* Vol. 20, No. 10, 1977, pp. 762-772.

[19] S. Faro and M. O. Külekci, "Fast packed string matching for short patterns," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, 2013, pp. 113-121.

[20] S. Faro and T. Lecroq, "The exact online string matching problem," *ACM Computing Surveys,* Vol. 45, No. 2, 2013, pp. 1-42.

[21] M. K. Ahmad, "An Enhanced Boyer-Moore Algorithm" *Doctoral dissertation*, Middle East University, 2014.

[22] R. N. Horspool, "Practical fast searching in strings," *Software: Practice and Experience,* Vol. 10, No. 6, 1980, pp. 501-506.

[23] D. M. Sunday, "A very fast substring search algorithm," *Communications of the ACM,* Vol. 33, No. 8, 1990, pp. 132-142.

[24] M. Crochemore, Czumaj, A., Gasieniec, L., Jarominek, S., Lecroq, T., Plandowski, W. and W. Rytter, "Speeding up two string-matching algorithms," *Algorithmica,* Vol. 12, No. 4-5, 1994, pp. 247-267.

[25] http://tanzil.net/#2:1. (2016, 2nd January).

[26] A. M. Al-Ssulami, "Hybrid string matching algorithm with a pivot," *Journal of Information Science,* Vol. 41, No. 1, 2014, pp. 82-88.

[27] B. Rahima, S. Zidat and F. Marir. "An analytical study on the holy Quran based on the order of words in Arabic AND conjunction." *Malaysian Journal of Computer Science*, Vol. 31, No. 1, 2018, pp. 1-16.

[28] H. Saqib, A. Kamsin, O. Tayan, M. Y. I. Idris and G. A. Gilkar, "Approaches for preserving content integrity of sensitive online Arabic content: A survey and research challenges." *Information Processing & Management*, 2017.

[29] H. Saqib, A. Kamsin, J. Veri, R. Ritonga and T. Herawan, "A Framework for Authentication of Digital Quran.", *Information Systems Design and Intelligent Applications*, Springer, 2018, pp. 752-764.