

## Research Article

### Expertised String Mining in Outsized Databases and Hefty Files

K. Geetha Rani, Shobhanjaly P. Nair, P. Visu and S. Koteeswaran

Department of Computer Science and Engineering, Vel Tech. Dr. RR and Dr. SR Technical University,  
Chennai, Tamil Nadu, India

**Abstract:** In the last few decades data mining is one of the important research areas in data maintenance of computing. In computing world, plentiful algorithms are proposed for mining. The few applications of data mining are web mining, video mining, knowledge mining and string mining. In these applications, string mining is focused and concentrated to overcome the space allocation process. To overcome the drawback suffered by the space allocation process Efficient String Mining (ESM) algorithm is proposed. The ESM algorithm performs operation faster, helps in reducing the space allocation, which in turn improves the performance of string mining and it is also possible to locate patterns that recurrent in the string database or file with a given support.

**Keywords:** Data mining, string mining, suffix array, suffix tree, text mining

#### INTRODUCTION

Knowledge Engineering is a production restraint which involves desegregating knowledge into computer system so as to unravel advanced issues which normally requires a high level of creature capability. At present, it refers to the building, maintaining and extension of knowledge-based systems. It has an immense convention in common with software engineering and is employed in several computing engineering domains such as artificial intelligence, including databases, data mining, skilled systems, decision support systems and geographic data systems. Knowledge engineering is additionally associated to mathematical logic, further as robustly concerned in scientific discipline.

Data Mining is extortion of data or information from a data set and converts it into a clear structure for future use and it is a dominant innovative expertise, with immense effect to assist companies, focus on the most significant information in their data warehouse (Koteeswaran *et al.*, 2012a). Data mining tools predict upcoming trends and behaviors, permitting businesses to create sensible, knowledge-driven decisions. The mechanized, prospective analyses offered by data mining move beyond the analyses of past measures provided by demonstration tools typical of decision support systems.

Text Mining is intermittently and alternately cited as text data mining, approximately similar to text analytics and refers to the method of etymologizing premium information from text (Koteeswaran *et al.*, 2012b). High-quality data is naturally derived by making of patterns and propensity such as statistical

pattern knowledge. Text mining typically involves the process of configuration the input text, routinely parsing, besides the addition of some derived linguistic options and also removal of others and later insertion into a data set (Koteeswaran and Kannan, 2013).

The study explains ESM algorithm that helps in overcoming the space allocation problem by using less space while maintaining runtime efficiency similar to fastest available algorithms. Also spectacularly less memory is utilized than the other algorithms.

#### MATERIALS AND METHODS

Jasbir *et al.* (2012a) have proposed rising substring patterns from databases of string that have been exposed, which pass through an enhanced suffix array data structure. The data mining algorithms are usually lounged at efficiency spectrum at its extreme end; they are either fast and utilize massive amount of space or compact and orders of size slower. This study introduces an algorithm that achieves the best of both these boundaries, having runtime equivalent to the fastest available algorithms while using less space than the most space efficient ones. While we use spectacularly less memory than the other algorithms the same may be accepted and utilized worldwide.

Yun *et al.* (2003) have proposed a tree structures that are used extensively in domains such as computational biology, pattern recognition, computer networks and so on. The study present indexing technique for free trees and apply this indexing technique to the problem of mining frequent sub trees. They first define a new representation, the canonical

**Corresponding Author:** K. Geetha Rani, Department of Computer Science and Engineering, Vel. Tech. Dr. RR and Dr. SR Technical University, Chennai, Tamil Nadu, India

This work is licensed under a Creative Commons Attribution 4.0 International License (URL: <http://creativecommons.org/licenses/by/4.0/>).

form, for rooted trees and extend the definition to free trees. They also introduce another concept, the canonical string, as a simpler representation for free trees in their canonical forms. They then apply their tree indexing technique to the frequent sub tree mining problem and present Free Tree Miner, a computationally efficient algorithm that discovers all frequently occurring sub trees in a database of free trees. They performed and the scalability of their algorithms through extensive experiments based on both synthetic data and datasets from two real applications: a dataset of chemical compounds and a dataset of Internet multicast trees.

Hiroki *et al.* (1998) have discussed about study of data mining problem in a large collection of in structured texts based on association rules over sub words of texts. A two-word association rule is an expression such as (TATA, 30, AGGAGGT) C that expresses a rule that if a text contains a sub word followed by another sub word with distance no more than k then a condition C will holds with a probability. They present an efficient algorithm for computing frequent patterns that optimizes the confidence with respect to a given collection of texts. The algorithm runs in time  $O(mn^2 \log^2 n)$  and in space  $O(km n \log n)$ , where m and n are the number and the total length of classification examples, respectively and k is a small constant around 30-50. The algorithm employs the suffix tree data structure from string pattern matching and the orthogonal range query techniques from computational geometry. They also give a faster version that runs in time and in space.

David and Marcel (2008) have proposed that a general approach for frequency based string mining, which has many applications, e.g., in contrast data mining. Their contribution is a novel algorithm based on a deferred data structure. Despite its simplicity, their approach is up to 4 times faster and uses about half the memory compared to the best-known algorithm of Fischer *et al.* (2006). Applications in various string domains, e.g., natural language, DNA or protein sequences, demonstrate the improvement of our algorithm.

Johannes *et al.* (2005) have explained about Mining frequent strings in databases that has many interesting applications, e.g., in computational biology. They focus on a special kind of constraint-based frequent string mining, namely computing all strings that are frequent in one database and infrequent in another. They also present a method to find such strings by using the suffix-and LCP-arrays, which can be computed extremely fast and space efficiently and further exhibit a good locality performance. They tested their method on several biologically relevant data sets and demonstrated that it outperforms existing methods in terms of time and space.

Mocian (2012) have focused on the algorithms and data structures used in string mining and their

applications in bioinformatics, text mining and information retrieval. More specific, it studies the use of suffix trees and suffix arrays for biological sequence analysis and the algorithms used for approximate string matching, both general ones and specialized ones used in bioinformatics, like the BLAST algorithm and PAM substitution matrix. Also, an attempt is made to apply these structures and algorithms for text mining and information retrieval.

Existing algorithms are making two passes from start to finish of the LCP array to calculate the support of all substrings in the database (Jasbir *et al.*, 2012b). The primary pass computes and stores substrings and repeats it within a single string. Theoretical and Practical Performance of Previous String Mining Algorithms, Theoretical space is in bits, practical space is in bytes and also practical time is given relative to the original frequent linear algorithm are all completely existing. They build the Suffix Array for multiple databases in a similar way by concatenating the string for each database. We have a tendency to call these correction factors and store them in an array C. The second pass then computes how often all substrings occur within the whole database. By subtracting the correction factor from this number, you get the support for each substring.

Adrian and Enno (2008) contribution helps in enhancing the existing linear-time algorithm for frequency constraint in such a way that the maximum memory utilization is a constant factor of the size of the largest database of strings. Also an algorithm based on two suffix arrays to calculate the intersection of the results of different databases are proposed. According to Zhan *et al.* (2010), databases of strings are used to find all strings that fulfill certain constraints of all string databases. A suffix and LCP table are constructed which can reduce the total space consumption of string mining efficiently.

**Proposed algorithm and approaches:** In this proposed research study we are planning to present a innovative algorithm and approach for mining substrings from a information or file of strings that's going to be very efficient. While we use outstandingly less memory than the other algorithms the same may be measured and utilized internationally. We are planning to use the following techniques and methods Auxiliary Reducing Space, Civilizing Runtime, Parallelization and Multi Course of Accomplishment and also Resilient Processing. ESM Longest Common String and Longest Repeated string algorithms are successfully implemented and achieved the effective results.

**Algorithm 1: ESM-longest repeated string:**

```
package project.sm;
public class LongRS extends Base {
public void fLRS ()
{
String input = readInputFile ();
```

```

input = input.replaceAll ("\\s+", " ");
SuffixArray ss = new SuffixArray (input);
int saLen = ss.length ();
long sTime = System.currentTimeMillis ();
String lrs = "";
for (int i = 1; i<saLen; i++)
{
    int length = ss.lcp (i);
    if (length>lrs.length ())
        lrs = ss.select (i) .substring (0, length);
}
long eTime = System.currentTimeMillis ();
System.out.println ("LongRS : " + lrs);
System.out.println ("Time Taken (ms): "+ (eTime -
sTime));
System.out.println ("Memory (bytes): "+saLen);
}
public static void main (String [] args)
{
    LongRS lrs = new LongRS ();
    lrs.fLRS ();
}
}

```

The Efficient String Mining (ESM) algorithm one and two are effectively analyzed, designed, implemented and tested to get the efficient string mining in the large database or large files.

**Algorithm 2: ESM-longest common string:**  
package project.s.m;

```

for (int i = 1; i<saLen; i++)
{

```

```

    if (sa.select (i) .length () <= N2 && sa.select
        (i-1) .length () <= N2) continue;
    if (sa.select (i) .length () >N2+1 && sa.select (i-1)
        .length () >N2+1)
        continue;
    int length = sa.lcp (i);
    if (length>substring.length ())
        substring = sa.select (i) .substring (0, length);
}
long endTime = System.currentTimeMillis ();
System.out.println ("LongCS: "+substring);
System.out.println ("TimeTaken (ms): "+ (eTime-
sTime));
System.out.println ("Memory (bytes): "+saLen);
}
public static void main (String [] args){
    LongCS lcs = new LCS ();
    lcs.fs ();
}
}

```

The final results achieved through the proposed algorithms are very much important and so much in demand.

**EXPERIMENTAL RESULTS AND DISCUSSION**

Hardware's Used: CPU: Intel i5, RAM: 4 GB RAM, Memory: 8 MB L3 Cache, 500 GB Hard Disk. Software Used: Java. When used the suffix array, to find LCS (Longest Common String) and LRS (Longest Repeated String) achieved the following (Fig. 1 to 3):

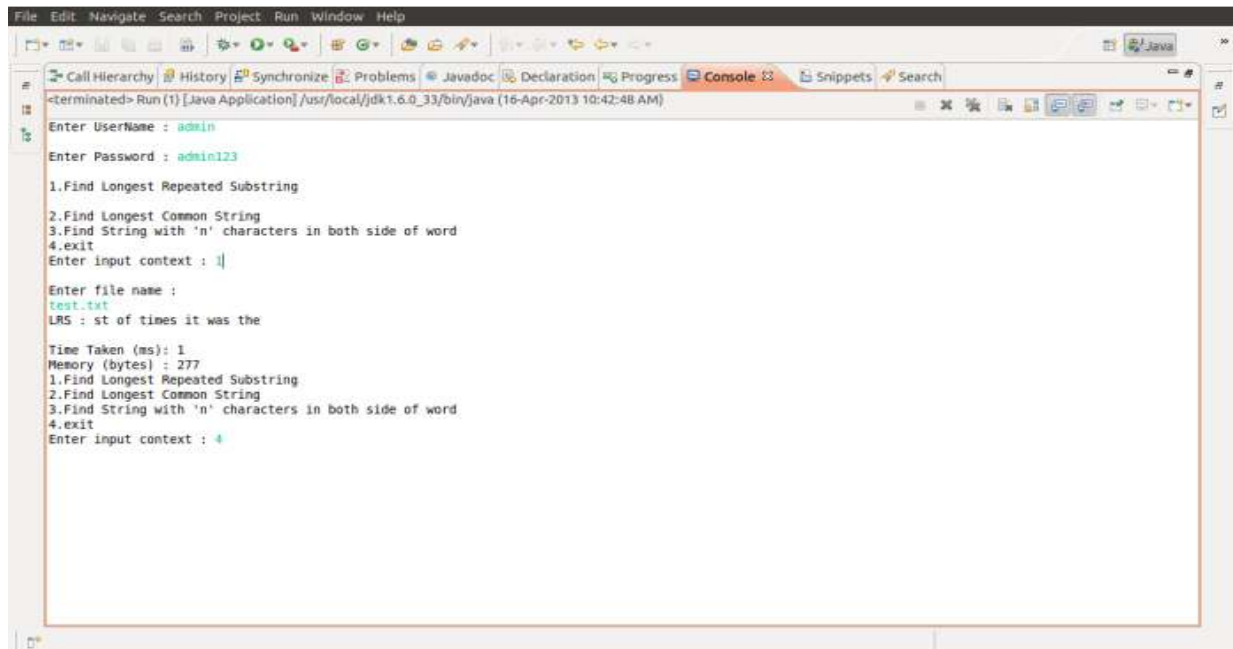
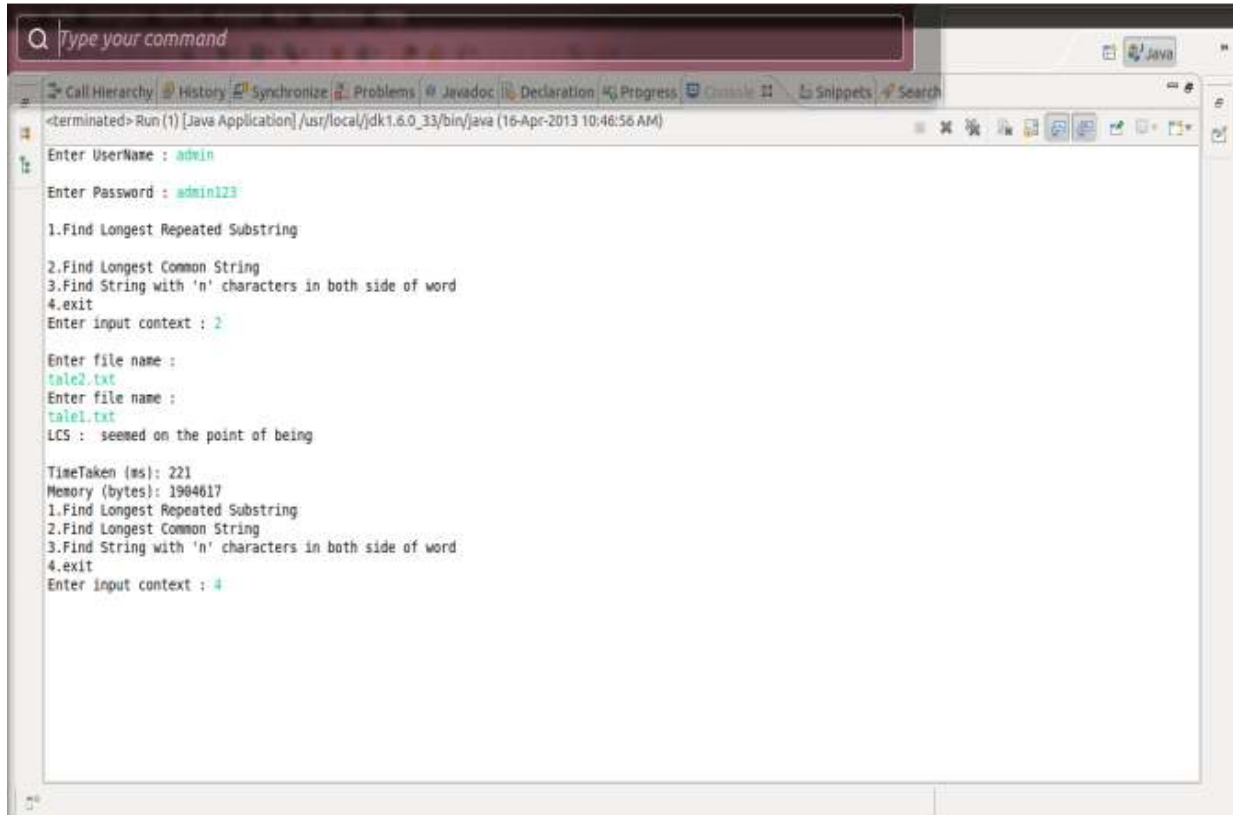
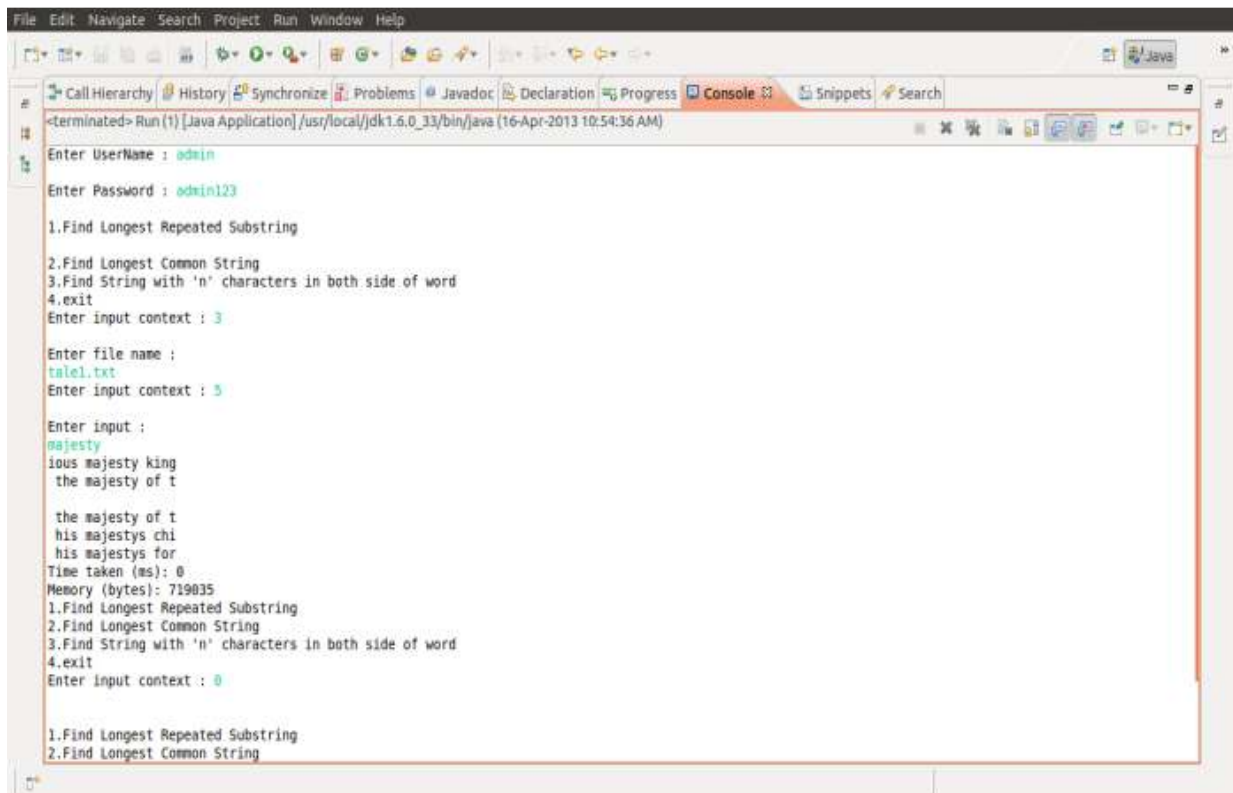


Fig. 1: ESM-LRS implementation result



```
Q Type your command
<terminated> Run (1) [Java Application] /usr/local/jdk1.6.0_33/bin/java (16-Apr-2013 10:46:56 AM)
Enter UserName : admin
Enter Password : admin123
1.Find Longest Repeated Substring
2.Find Longest Common String
3.Find String with 'n' characters in both side of word
4.exit
Enter input context : 2
Enter file name :
tale2.txt
Enter file name :
tale1.txt
LCS : seemed on the point of being
TimeTaken (ms): 221
Memory (bytes): 1904617
1.Find Longest Repeated Substring
2.Find Longest Common String
3.Find String with 'n' characters in both side of word
4.exit
Enter input context : 4
```

Fig. 2: ESM-LCS implementation result



```
File Edit Navigate Search Project Run Window Help
<terminated> Run (1) [Java Application] /usr/local/jdk1.6.0_33/bin/java (16-Apr-2013 10:54:36 AM)
Enter UserName : admin
Enter Password : admin123
1.Find Longest Repeated Substring
2.Find Longest Common String
3.Find String with 'n' characters in both side of word
4.exit
Enter input context : 3
Enter file name :
tale1.txt
Enter input context : 3
Enter input :
majesty
ious majesty king
the majesty of t
the majesty of t
his majestys chi
his majestys for
Time taken (ms): 0
Memory (bytes): 719035
1.Find Longest Repeated Substring
2.Find Longest Common String
3.Find String with 'n' characters in both side of word
4.exit
Enter input context : 0
1.Find Longest Repeated Substring
2.Find Longest Common String
```

Fig. 3: ESM-SM implementation result

Table 1: Time

Tested input	Derived output
Time_Taken (msec)	Time_Taken (msec)
1-3 msec	7-10 msec
2-5 msec	0.5-1 sec

Table 2: Space

Tested input	Derived output
String_Size	String_Size
1-3 KB	10 KB
3-6 KB	1 MB

Better results could be achieved when multiple cores and high RAM configuration are used. The time taken to repossess data would be in Micro Seconds/Nano Seconds. As we have used to JAVA which is high level language, it takes some memory to load the runtime classes so the memory taken by overall process is more. If we use low level languages (like C, Pascal), the memory usage of the overall process will be less. Storing the suffix array of large data in RAM would be memory consuming job. Instead we can use the other algorithms like (LRU-Least Recently Used, LCU-Less Commonly Used, LFU-Less frequently used), to store the data in second level cache. The second level cache shall be disk or database. So with multiple cores, data reclamation shall be made better and with our proposed algorithms and approach the memory usage also shall be optimized.

### CONCLUSION

In this research study we achieved very measurable results in string mining through ESM algorithm and this could be updated and enhanced with more advanced techniques to deliver extra fast and to use less amount of space. The Time and Space efficiently notated through the Mille Seconds and Kilo bytes respectively are mentioned in Table 1 and 2. The same work should be proceeded to achieve a better efficient string mining.

### REFERENCES

Adrian, K. and O. Enno, 2008. A space efficient solution to the frequent string mining problem for many databases. *Data Min. Knowl. Disc.*, 17(1): 24-38.

David, W. and H.S. Marcel, 2008. Efficient string mining under constraints via the deferred frequency index. *Proceedings of the 8th Industrial Conference on Advances in Data Mining: Medical Applications, E-Commerce, Marketing and Theoretical Aspects (ICDM '08)*, pp: 374-388.

Fischer, J., V. Huen and S. Kramer, 2006. Optimal string mining under frequency constraints. *Proceeding of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 4213: 139-150.

Hiroki, A., W. Atsushi, F. Ryoichi and A. Setsuo, 1998. An efficient algorithm for text data mining with optimal string patterns. *Proceeding of the ALT'98, LNAI, Vol. 247*.

Jasbir, D., J.P. Simon and T. Andrew, 2012a. Practical efficient string mining. *IEEE T. Knowl. Data En.*, 24(4).

Jasbir, D., J.P. Simon and T. Andrew, 2012b. Trends in suffix sorting: A survey of low memory algorithms. *Proceedings of the 35th Australasian Computer Science Conference (ACSC'12)*.

Johannes, F., H. Volker and K. Stefan, 2005. Fast frequent string mining using suffix arrays. *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM '05)*, pp: 609-612.

Koteeswaran, S. and E. Kannan, 2013. Analysis of Bilateral Intelligence (ABI) for textual pattern learning. *Inform. Technol. J.*, 12(4): 867-870.

Koteeswaran, S., P. Visu and J. Janet, 2012a. A review on clustering and outlier analysis techniques in data mining. *Am. J. Appl. Sci.*, 9(2): 254-258.

Koteeswaran, S., J. Janet and E. Kannan, 2012b. Significant term list based metadata conceptual mining model for effective text clustering. *J. Comput. Sci.*, 8(10): 1660-1666.

Mocian, H., 2012. Applications of string mining techniques in text analysis. *Sci. Bull. Petru Maior Univ., Targu Mures*, 9(1): 5.

Yun, C., Y. Yirong and R.R. Muntz, 2003. Indexing and mining free trees. *Proceeding of the 3rd IEEE International Conference on Data Mining (ICDM, 2003)*, pp: 509-512.

Zhan, X.G., X.M. Zhi, S.X. Yu and L. Li, 2010. An Optimized LCP table based algorithm for frequent string mining. *Appl. Mech. Mater.*, 20-23: 653-658.