ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

SEQUENTIAL DATA MINING: EXPLORING THE REDUNDANT PATTERNS FROM SEQUENCES TO MINIMIZE THE OVERALL PATTERNS

Mr. TamboliSuraj Mubarak*, Prof. Prabhudev Irabashetti, Prof. Sutar Mushtaq
M.E Computer(II), Computer Department, Vishwabharati Academy's College of Engineering, Ahmednagar, Pune university,India(MH)
Assistant Professor, Computer Department, Vishwabharati Academy's College of Engineering, Ahmednagar, Pune university,India(MH)

ABSTRACT

Recent studies in discovering patterns from sequence data have shown the significant impact in many aspects of data mining. In this research, a novel method of finding the redundant pattern is proposed. To efficiently discover the redundant pattern, the focus is on developing new algorithms. Rapid increase of the sequential data has created the problem of discovering meaningful patterns from sequences. The most challenging problem is to find repeating patterns with gap constraints. In this work, we identify a new research for mining the redundant patterns with gap constraints. To solve the problem, we propose algorithm with components such as: (1) Data-driven pattern generation approach to avoid generating unnecessary candidates for validation. (2) Back-tracking pattern search process to discover approximate occurrences of a pattern under user specified gap constraints. (3) An Apriori-like deterministic pruning approach to progressively prune patterns and cease the search process if necessary. It is proposed to conduct experimental analysis on the synthetic and standard data sets. It is also proposed to conduct comparative analysis of the developed algorithms with the state of art algorithms.

KEYWORDS: Data mining, Back-tracking, prune patterns, RedundantPatterns, CFI, Gap Constraint,Delta Closed Pattern.

Introduction

Sequence data are the significant data in many forms. Discovering the patterns from the sequence data has great value in many sectors but, the discovery of patterns should be carried out in an efficient way. In a number of sequential data mining applications, the goal is to discover frequently occurring patterns. Many algorithms have been proposed to discover the sequential patterns producing large set of patterns which are highly redundant. Recently mining and analysis of sequence data has been studied in several fields. Sequential pattern mining remains one of the most important data mining tasks. With the ubiquity of sequential data, it is found broad applications in customer analysis, query log analysis, web click stream, custom purchase history, event sequences, financial stream data analysis and pattern discovery in genomic DNA sequences in bioinformatics.Today large amount of such data from genomic proteomic and business arenas has been acquired. The discovery of new interesting knowledge from these enormous sequence data has important application and great value in many sectors but the discovery process needs to be carried out in an efficient and effective way. Existing sequence mining algorithms mostly focus on mining for subsequence's. However the large class of applications require efficient mining of patterns that are not redundant. The work addresses the problem of mining the patterns from the sequences. We take multiple strings as input sequences and substrings as patterns which are referred to as contiguous patterns. Many algorithms have been proposed to discover the frequent patterns. Most of them overlook the output quality, producing the large set of patterns and many of the patterns are highly redundant.

Patterns have been used to implement efficient systems that can recommend based on previously observed patterns, help in making predictions, improve usability of systems, detect events, and in general help in making strategic product decisions. There are various applications of sequential data

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

mining in a variety of domains like healthcare, education, Web usage mining, text mining, and Bio-informatics, telecommunications, and intrusion detection. The discovery of new interesting knowledge from these enormous sequence data has important application and great value in many sectors but the discovery process needs to be carried out in efficient and effective way. A major problem with all the large genetic sequence databases is that records are deposited in them from a wide range of sources, from individual researchers to large genome sequencing centers. As a result, the sequences themselves, and especially the biological annotations attached to these sequences, vary tremendously in quality. Also there is much redundancy, as multiple labs often submit numerous sequences that are identical, or nearly identical, to others in the databases.

Many annotations are based not on laboratory experiments, but on the results of sequence similarity searches for previously-annotated sequences. Of course, once a sequence has been annotated based on similarity to others, and itself deposited in the database, it can also become the basis for future annotations. This leads to the transitive annotation problem because there may be several such annotation transfers by sequence similarity between particular database record and actual we lab experimental information. Therefore, one must always regard the biological annotations in major sequence databases with a considerable degree of skepticism, unless they can be verified by reference to published papers describing high-quality experimental data, or at least by reference to a human-curated sequence database.

In item set mining [17] it is observed that frequent itemset (FI) often containsredundancies.Thus, closedfrequentitemsets (CFIs) [17] are proposed as a conciserepresentation of FIs.Although CFIs contain fewer redundanciesthan FIs, sometimes the definition of the closureistoo restrictive and the compression is fairly low as theyretain all the information. Hence,CFIs are further extendedto the delta-tolerance closed itemsets [18] aiming at giving amore concise representation. Delta closed itemsets provide acontrollable tight lossyapproximationto the closed itemsets.By allowing tuneable tolerance in delta closed itemsets, agreat number of redundant itemsets are pruned whileimportant information is retained. We employ the notionof delta closed itemset to discover delta closed patterns fromsequences.So this work explores the redundant patterns from the sequential patterns to minimize the highly redundant pattern from the overall pattern. To further shrink the output size, many methods attemptto extract statistically significant patterns from frequentpatterns. The motivation is that frequent expected patternsmight not be as interesting as statistically unexpected orsignificant patterns. Statistical significance is evaluatedthrough statistical hypothesis test which measures howmuch the frequency of a pattern deviates from the expectedone given the random model. It is hoped that patternsoccurring with significantly higher frequency will correspondto the functional units inherent in the sequences.Furthermore, the assessment of statistical significance canhelp in ranking output patterns, enabling experts to assess theresult. However, among the statistically significant patternssome are actually statistically redundant. They are consideredas significant merely because they contain very strongsignificant sub-patterns.The work presents two efficient algorithms developed to discover non-redundant patterns from sequences. One discovers the delta closed patterns and other finds the number of occurrences of the patterns from the given input string. The suffix tree is used to efficiently identify the proper super-pattern and sub-pattern and hence are able to discover delta closed patterns.

LITERATURE SURVEY

Finding the delta closed nodes from sequences was proposed by Wong, Zhuang[21], which finds the delta closed patterns from the given input sequences. Sequence synthesis and recognition of patterns for multiple sequences was proposed by Chan and Wong [14] in the early 1990s. Sequential pattern mining was introduced by Agrawal and Srikant [15] and many sequential pattern mining algorithms have been developed. Sequential pattern mining can serve as the general framework to mine frequent contiguous patterns from multiple strings. GSP [19] and PrefixSpan [20] are two representative methodsforApriori-based and pattern-growth-based approaches,respectively. Sequential pattern mining can serve as thegeneral framework to mine frequent contiguous patternsfrom multiple strings. We nextintroduce the generalized suffix tree T, the datastructure for representing strings. T can be constructed in O(L) time with O(L) space complexity via suffix array. Thedetails of the suffix tree and suffix array and their lineartime and space construction algorithms can be found in[16]. To use T ingeniously we establish the connectionbetween frequent contiguous patterns and the path labels. The general input for sequentialpattern mining is

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

transaction sequences where each elementin a sequence could contain multiple items. Multiple stringscan be considered as the special input where each sequenceelement only contains one item. Hence, sequential patternmining algorithms can be directly applied to such datawithout any modification. However, sequential pattern isquite different from the contiguous pattern considered here.There could be any gap between two adjacent elements insequential pattern while no gap is allowed in contiguouspattern. Therefore, gap constraint is needed for sequentialpattern mining algorithms to mine contiguous patterns.Mining sequential patterns with gap constraints is one of the recent focuses in sequential pattern mining. The gap constraint often implies strong correlations among pattern elements and hence is important in many applications. GenPrefixSpan [1] is one that mines sequential pattern with gap constraint and can be used to discover frequent contiguous patterns. Another recent focus in sequential pattern mining is to mine the complete set of closed sequential patterns instead of frequent patterns, achieving a more compact yet complete output set with better efficiency. A closed pattern is defined as a pattern with no super pattern of the same support. CloSpan [2] and BIDE [3] both mine closed patterns based on PrefixSpan. Gap-BIDE in [4] mines closed sequential patterns with gap constraint and can hence be used to discover closed contiguous patterns.

The authors in paper [5] discover association rules that are unexpected by the prior knowledge consisting of a set of beliefs. Surprising patterns are obtained in time series database using Markov chain as the random model [6]. Statistically significant rules are mined in [7]. Unexpected sequential patterns are discovered with respect to a given set of beliefs [8]. The paper [9] presents a method of mining surprising periodic patterns in sequence database. Significance is assessed through swap randomization in [10]. The paper [11] notices that both closed frequent item set and its non-closed sub item sets share the same level of statistical significance and hence it is safe to remove non closed item sets. Redundant patterns are identified as unexpected rules that can be inferred from other unexpected rules under monotonicity assumption [12]. The Blanchette and Sinha [13] observed that some significant sequence patterns are random variations of other significant patterns and hence redundant. They proposed a heuristic (Find-Explanator) to extract non-redundant patterns using a random model of order 3 Markov chain.

MEHODOLOGY

Sequence data may be Ordered set of elements such as s = $a_1,a_2,..a_n$. Where each element a_i could be Numerical, Categorical i.e. domain a finite set of symbols S, |S|=m, or Multiple attributes. In the sequence data the length *n* of a sequence is not fixed and the order determined by time or position and could be regular or irregular.

The example of sequence data may be,

- Classical applications-
  - Speech: sequence of phonemes
  - Language: sequence of words and delimiters
  - Handwriting: sequence of strokes
- Newer applications-

Bioinformatics:

Genes: Sequence of 4 possible nucleotides, |S|=4

Example: AACTGACCTGGGCCCAATCC

Proteins: Sequence of 20 possible amino-acids, |S|=20

Example:MAQQWSLQRLAGRHPQDSYEDST

Sequential pattern mining [2] methods have been found to be applicable in a large number of domains. Sequential data is omnipresent. Sequential pattern mining methods have been used to analyze this data and identify patterns.

Let Σ be a set of distinct elements {e1, e2, . . . ,$e_{|\Sigma|}$}, Σ is called the alphabet and |Σ| is itssize. A sequence s over Σisan ordered list of elements, denoted as s1s2 . . . sn, whereeach$s_{i\epsilon}$Σ.n is the length of S and S[i, j] the substring of S.Each element here contains only an item in contrast toelement of sequence used in general sequential patternmining containing multiple items.The sequence defined above is essentially a string. Manyimportant sequences such as DNA sequences can berepresented as strings.A contiguous pattern P is defined as a short sequence$p_1p_2 . . . p_m$ over Σ. m is the order (or the length) of thepattern. The order of P should be at least 2 to make itnontrivial. A pattern P`= $a_1a_2 . . . a_m$ is a sub-pattern ofanother pattern P = $b_1b_2 . . . b_{m`}$ or P a super-pattern of P` ifthere exists an integer j such that $a_i = b_{j+I}$for 1 ≤ i ≤ m. Forexample, ATCG is a super-pattern of TCG and TCG is asub-pattern of ATCG. We say P occurs at position i in S ifand only if P = S[i,i + m-1]. There might be multipleoccurrences of P which can be represented through a list ofpositions L_P= {i_1, $i_{2...}i_k$,}, where k is the number ofoccurrences of P in

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

S.In general, the input data might come as multiplesequences S_1, S_2,….,S_N, with lengths n_1, n_2, . . . ,n_N, respectively.

Let L be the overall length (input size) of the inputsequences. We define the number of occurrences and thesupport of P in multiple sequences as below,

1. Number of occurrences of P in multiple sequencesS_1, S_2, . . . , S_N.

The number of occurrences of P denoted by $k_P$inmultiple sequences is the sum of the number of occurrences ineach sequence S_i and the list of positions becomes L_P= {. . . ,(i, j),. . .} where (i, j) is a position denoting that Poccurs at the position j in the sequence i.

2. Support of P in multiple sequences S_1, S_2, . . . , S_N.

The support of P denoted by q_P in multiple sequences is thenumber of sequences in which P occurs at least once.

For example Let S1 = ATCGAT and S2 = TCGATC. Thenumber of occurrences of AT and GATC is 3 and 1 withrespective position list {(1,1), (1,5), (2,4)} and (2,3).Their support is 2 and 1 respectively.

In sequential pattern mining, frequent pattern is definedover the support. The support of a sequentialpattern is close to its number of occurrences. Here in order to deal with long sequences that maycontain multiple pattern occurrences, we define frequentpattern over the number of occurrences.

3. **(Frequent Pattern)** A pattern is said to befrequent if its number of occurrences $k_P \geq min_{occ}$, wheremin_{occ} specifies the minimum number of occurrences required.
4. **(Delta Closed Pattern)** Given a set of frequentpatterns, a delta closed pattern P is one that does not haveany delta closed super-pattern P` such that $k_{P`} \geq \delta.k_P$, where δ is the tolerance factor and $0 \leq \delta \leq 1$. In other words, P isnot delta closed if it has a delta closed super-pattern P` suchthat$k_P \geq \delta. k_P$.

Delta closed pattern is defined in a recursive way. Thepattern with the highest order is by definition a delta closedpattern as it does not have any super-pattern. However, weavoid using nondelta-closed patterns to check whetherother patterns are delta closed or not because they areredundant patterns not included in the output set. That is for example Suppose we have a set of frequent patterns{ATCG, ATC, TCG, TC, CG} with 15, 18, 15, 22, 20occurrences, respectively. With $\delta = 1$, the set of closedpatterns is {ATCG, ATC, TC, CG}. TCG is not deltaclosed with respect to its super-pattern ATCG which havethe same number of occurrences. With $\delta = 0.8$, the set ofdelta closed patterns is {ATCG, TC, CG} as TCG and

ATC are not delta closed with respect to ATCG (i.e., $15 \geq 0.8*15$ and $15 \geq 0.8*18$). Although TC has a super-patternATC satisfying $18 \geq 0.8*22$, ATC is not acandidate pattern for checking the delta closednessofTC as ATC is not delta closedwith respect to ATCG andnot included in the output. ATCG is the only candidatepattern for TC. Hence, TC is delta closed. CG is deltaclosed for the same reason.

There couldbe several candidate patterns for checking whether apattern P is delta closed or not. Among them, we denotethe one with the largest number of occurrences as theproper pattern for such checking.

Suffix Tree

Suffix tree is the datastructure for representing strings. Suffix tree can be constructed inO(L) time with O(L) space complexity via suffix array. The information of the suffix tree andsuffix array and their lineartime and space construction algorithms can be found in [16]. For making the use of suffix tree ingeniously we establish the connectionbetween frequent contiguous patterns and the path labels.

Given a collection of multiple strings S_1, S_2, . . . , S_N over Σ, the generalized suffix tree T representing them is arooted directed tree with the following properties:

1. Each leaf node is labelled by a position (i, j) indicatinga suffix of string Si starting at the position j.
2. Each internal node has at least two outgoing edgeseach of which is labeled with a nonempty substringin the input strings. No two edges going out of anode can have the edge label starting with the samecharacter.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Most often, a termination character $ is appended toeach string to ensure that T exists for these multiple strings.
The work is carried out that is generating the suffix tree [2] on the sequential data.A suffix tree is a fundamental data structure for string searchingalgorithms. Unfortunately, when it comes to the use ofsuffix trees in real-life applications [3], the current methods forconstructing suffix trees do not scale for large inputs.

In computer science, a suffix tree (also called PAT tree or, in an earlier form, position tree) is a data structure that presents the suffixes of a given string in a way that allows for a particularly fast implementation of many important string operations.

The suffix tree for a string *S* is a tree whose edges are labeled with strings, such that each suffix of *S* corresponds to exactly one path from the tree's root to a leaf. It is thus a radix tree (more specifically, a Patricia tree) for the suffixes of *S*. A suffix tree is a special kind of a Trie.

Constructing such a tree for the string *S* takes time and space linear in the length of*S*. Once constructed, several operations can be performed quickly, for instance locating a substring in *S*, locating a substring if a certain number of mistakes are allowed, locating matches for a regular expression pattern etc. Suffix trees also provided one of the first linear-time solutions for the longest common substring problem. These speedups come at a cost: storing a string's suffix tree typically requires significantly more space than storing the string itself.

The suffix tree for the string *S* of length 'n' is defined as a tree such that:

- The paths from the root to the leaves have a one-to-one relationship with the suffixes of *S*,
- Edges spell non-empty strings,
- And all internal nodes (except perhaps the root) have at least two children.

Since such a tree does not exist for all strings, *S* is padded with a terminal symbol not seen in the string (usually denoted $). This ensures that no suffix is a prefix of another, and that there will be n leaf nodes, one for each of the n suffixes of *S*. Since all internal non-root nodes are branching, there can be at most $n - 1$ such nodes, and $n + (n - 1) + 1 = 2n$ nodes in total (n leaves, $n - 1$ internal non-root nodes, 1 root).

Suffix links are a key feature for older linear-time construction algorithms, although most new algorithms, which are based on Farach's algorithm [4][5], dispense with suffix links. In a complete suffix tree, all internal non-root nodes have a suffix link to another internal node. If the path from the root to a node spells the string $\chi\alpha$, where χa single is character and α is a string (possibly empty), it has a suffix link to the internal node representingα.

A suffix tree for a string *S* of length n can be built inθ(n) time, if the letters come from an alphabet of integers in a polynomial range (in particular, this is true for constant-sized alphabets). For larger alphabets, the running time is dominated by first sorting the letters to bring them into a range of size*O(n)*in general, this takes *O(n log n)* time. The costs below are given under the assumption that the alphabet is constant.

Applications of suffix tree

Suffix trees can be used to solve a large number of string problems that occur in text-editing, free-text search, computational biology and other application areas. Primary applications include

- String search, in *O(m)* complexity, where *m* is the length of the sub-string (but with initial *O(n)* time required to build the suffix tree for the string)
- Finding the longest repeated substring
- Finding the longest common substring
- Finding the longest palindrome in a string

Suffix trees are often used in bioinformatics applications, searching for patterns in DNA or protein sequences (which can be viewed as long strings of characters). The ability to search efficiently with mismatches might be considered their greatest strength. Suffix trees are also used in data compression; they can be used to find repeated data, and can be used for the sorting stage of theBurrows–Wheeler transform. A suffix tree is also used in suffix

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

tree clustering, a data clustering algorithm used in some search engines.

Frequent Patterns in a suffix tree is a contiguous pattern has its unique path inthe suffix tree, we need to find a way to obtain its number ofoccurrences which is important for defining frequentpattern. The number of occurrences of a pattern is thenumber of positions found under its path in the suffix tree.More efficiently, we first store into eachnode x the number of positions of k(x) in the subtree rooted byit. Then the number of occurrences of P whose path ends ator above node x can be easily obtained by k(x). Hence, frequent patterns are represented by labels of paths that end at or above a node x with $k(x) \geq$ minocc.

The suffix tree which we have constructed has an advantage that it performs the fast searching of the strings and the tree gives the various patterns from it.

MATHEMATICAL MODEL

Algorithm for Finding Delta Closed Patterns

Delta closed patterns are frequent patterns and hence represented by path labels as well. However, the path of a delta closed patterns does not end within an edge but at a node. A pattern whose path ends within an edge can be further extended by at least one character to the right without decreasing the number of occurrences and hence cannot be delta closed.
The algorithm contains various steps such as constructing the generalized suffix tree 'T' for whatever the input sequence given. After the tree is generated finding the number of positions under each node 'x' of 'T'.

The suffix tree contains various patterns in it; also it contains different frequent patterns. A pattern is said to be frequent pattern if its number of occurrence i.e. (k_p) is greater than or equal to the minimum occurrence. For defining the frequent patterns in a suffix tree there is a need to find a way to its number of occurrences. Number of occurrences of a pattern is number of position found under its path in a suffix tree. Position means all the leaf nodes under a node (basically an internal node). We have to find the position under each node x i.e. k(x). After finding the position under each nodes extracting the set of nodes whose $k(x) >= Min_{occ}$

The next step is to find the position level of x, that is number of characters found on a node called as pl(x). And sorting the nodes in descending order according to the length of pl(x).

After sorting the nodes in descending order taking each node for the processing and checking whether it is delta closed node or not. After processing a node if it is found that the node is delta closed then the patterns found on that node is also called as delta closed patterns. For calculating a node as delta closed or not first of all we have to find the cover node of it.

Cover node is the node which covers the given node using the path labels. Let 'y' be the cover node of 'x'.For example we have an input sequence which contains an alphabets as {A, T, C, G} if the node v1 is to be processed and node v1 contains a string as ATC, then checking the other node which contains the string as GATC or TATC or AATC or CATC etc. if any of the string from this is found on any node say that node v3 contains the string as GATC then this node v3 is called as the cover node of v1. From the figure 1 the node v1 has the string as ATC, then by adding any of the one character to the left of ATC from the given input string and checking whether we get any node in the tree, i.e. we can find the node v6which contains the path labels as GATC, then this node v6 is the cover node of v1.

Also we have to set the suffix node and the parent node of x. The suffix node means suppose v6 has path labels as GATC then by removing the first character 'G' from the string GATC we get ATC which is found on the node v1, so suffix node of v6 is v1.

After that checking whether the node 'x' has cover node or not, if it has no cover node then annotate 'x' as delta closed node. If 'x' has a cover node then the pl(x) is delta closed if and only if $k(y) < \delta.k(x)$, if then condition satisfies then annotate 'x' as delta closed node else annotate 'x' as not delta closed. The value of δ can be in the range as $0 \leq \delta \leq 1$.

Discovery of Delta Closed Patterns (DDCP) Algorithm

1. Construct a Suffix Tree T for the Input Sequences
2. Annotate k(x) the number of positions under each nodex of T
3. Extract a set of nodes whose $k(x) \geq min_{occ}$
4. Sort the above nodes in descending order according tothe length of pl(x) using counting sort.
5. For each node x

a) Let y be the cover node of x

b) Let x_s and x_P be the suffix node and parent node ofx, respectively

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

c) If x has no cover node
i. Annotate x as delta closed
d) Else
i. If pl(x) is delta closed (k(y)<δ*k(x))
Annotate x as delta closed
ii. Else
Annotate x as not delta closed
iii. End if
e) End if
6. End For
7. Output path labels of nodes annotated as delta closed.

For the input string given in the Figure 1 after applying the delta closed algorithm we get the various delta closed nodes as v6, v7, v1, v4.

Algorithm for Finding the Redundancy of the Delta Closed Patterns After the execution of the first algorithm we get the various delta closed patterns/nodes, these patterns are provided as an input for the second algorithm. The second algorithm then finds that how many times the delta closed nodes are occurring in the given input string. It then returns the node name and its number of occurrences based upon the specified threshold value. The threshold value is the numeric value which matches with the length of the delta closed nodes and gives the output with the string and the number i.e. its number of occurrence of it. Also the discovery of delta closed pattern algorithm and redundancy finding algorithm find the redundant patterns with the gap constraints.

The Threshold value is set which matches with the length of the processing node or processing pattern and gives the output as how many times the pattern is occurring in the given input string.

For the second algorithm we have implemented the steps to get the number of occurrences of node string from the main string. For this purpose we took the output of first algorithm that is the nodes which came after checking for the delta nodes. We took the node string (processing node) and created an array of string by passing one single value, this is done because C#.net does not allow direct string in system defined string functions for that we need array. So we created an array with single value that is a node string. After that we used one string function named as split, which splits the whole given string with given character or string array. For this purpose we have used 2 parameters, the first is our main given string in textbox, and the second is the one array created so that the split function returns me one array of substring after that one condition is added where we have checked whether the main string is starting with the node string or not, if yes then split the result array length is the number of occurrence for that node string, if no then the split result array length -1 is a number of occurrence for that node string. By this way we get the number of occurrences of the pattern in the given input string.

RESULTS AND DISCUSSION

Experimental Setup

Explain the machine configuration on which experimentation is performed. Processor, RAM, Software Tools used, etc….

Results

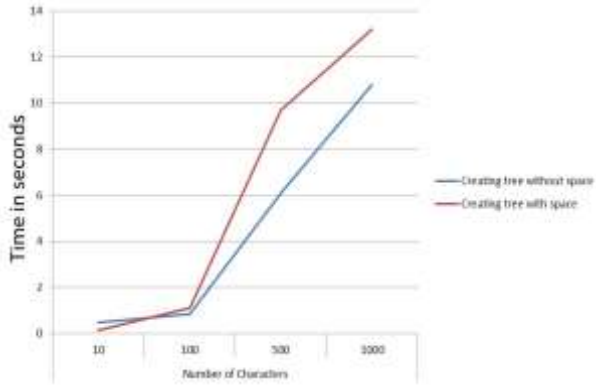
To calculate the performance of the algorithms, the algorithm is tested on three different sets of input. It uses real data sets to investigate the effect of δ and the output quality of the algorithms. The first set contains the number of characters 10 as an input string, also the string contains gap into it. We have handled the gap in two ways that is the first method draws the tree by deleting the gaps from the given input string, and the second method draws the tree by considering the gaps into it. That is based upon the gaps present into the tree, that much of different trees are been drawn. The minimum occurrences is set to 2 and the δ value is set to 1, after that we get the various delta closed patterns. Then these delta closed patterns are provided as an input to the second algorithm and the occurrences of these delta closed patterns are found in it. Based upon the different threshold value we get the different patterns and the count of their occurrences. Similarly we have tested the second set which contains the number of characters as 100 as an input. Then we have calculated the delta closed nodes and then the redundancy of it. There are total four sets on which the algorithms have been tested and we get the exact output within less time. We have tested the result based upon the time factor and recorded the time required for the different sets, which are as follows,

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Table: Runtime Comparison on different sets

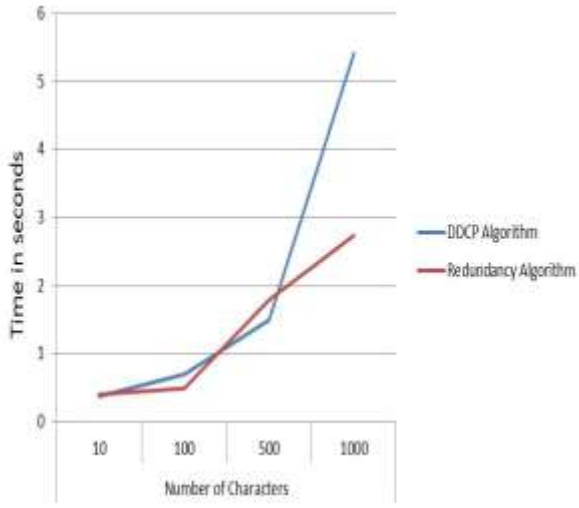
	Number of Characters			
	10	100	500	1000
Creating tree after removing spaces from it	0.8365991 sec	0.9541278 sec	12.077387 sec	19.7910042 sec
Creating tree based upon the spaces	0.8047431 sec	2.1078254 sec	6.5896805 sec	25.1869728 sec
DDCP Algorithm	0.9170033 sec	0.9001391 sec	5.4963596 sec	9.4124911 sec
Redundancy Algorithm	0.8097097 sec	0.9091547 sec	3.7858681 sec	6.7428847 sec

The following graph shows the execution time required for the generation of the suffix tree by considering the spaces and by not considering the spaces.



Graph 1: Combined graph for with space and without spaces for different length dataset

It is seen that with the growing data sets, if the sequential data set contains the gaps in it, then creating the tree by removing the gaps takes less time than the tree created by considering the gaps.
The following graph shows the execution time required for the redundancy finding algorithm.



Graph 2: Execution time for Redundancy Algorithm for different length dataset.

With growing dataset length, it isobserved that the execution time for redundancy finding algorithm also increases. The execution time is recorded in sec for both the graphs.

RESULT SNAPSHOT

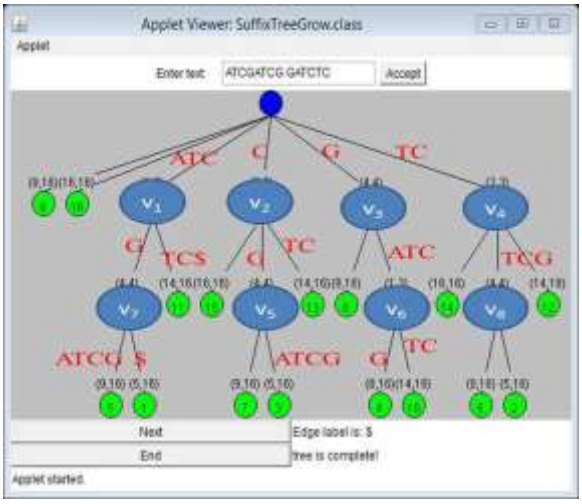
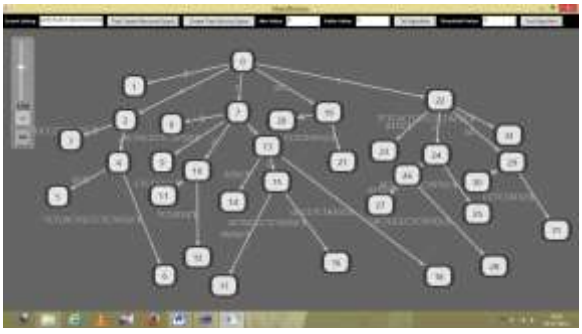


Fig. 1 Suffix Tree for the String S1= TCGATCGGATCTC



ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Figure 2: Suffix Tree by Deleting Gap S1= ATTCTCACT GCCCTCTATGC"

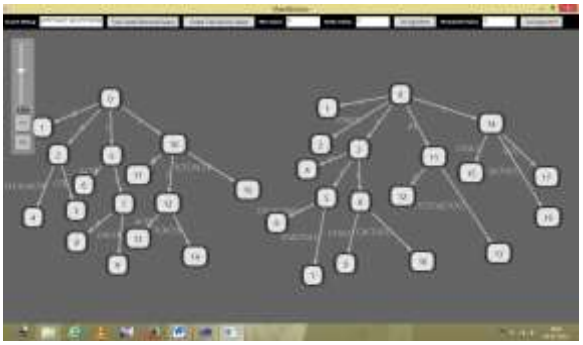


Figure 3: Suffix Tree by Considering Gap S1="ATTCTCACT GCCCTCTATGC

Figure 4: Based upon threshold value finding redundant patterns.

CONCLUSION

This work gives the method of finding the delta closed nodes and their redundancies for the sequence patterns. The work presents the first algorithm that is delta closed algorithm and second finding the redundancies of the delta closed algorithm. Also the use of suffix tree is made in an effective manner for identification of the patterns effectively in the linear time. Effectiveness of finding the redundant patterns significantly decreases the runtime. The experiments show the algorithms efficiencies as well as their ability to find the redundant patterns in smaller set of time.

ACKNOWLEDGEMENTS

We express great many thanks to Prof.PrabhudevIrabashetti,forhisgreateffortofsupervisingandleading ,toaccomplish thisfine work. Also I thanks Prof. Salve B. S., for their help. To collegeanddepartmentstaff,they were agreatsourceofsupport andencouragement.To my friendsand family, for theirwarm, kind encouragesand loves. To every person gave ussomethingtoo light my pathway,I thanks forbelievingin me.

REFERENCES

[1] *C. Antunes and A.L. Oliveira, "Generalization of Pattern-Growt Methods for Sequential Pattern Mining with Gap Constraints," Proc. Int'l Conf. Machine Learning and Data Mining, pp. 239-251, 2003.*

[2] *X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Databases," Proc. Third SIAM Int'l Conf. Data Mining, pp. 166-177, 2003.*

[3] *J. Wang and J. Han, "BIDE: Efficient Mining of Frequent Closed Sequences," Proc. 20th Int'l Conf. Data Eng., pp. 79-90, 2004.*

[4] *C. Li and J. Wang, "Efficiently Mining Closed Subsequences with Gap Constraints," Proc. Eighth SIAM Int'l Conf. Data Mining, pp. 313-322, 2008.*

[5] *B. Padmanabhan and A. Tuzhilin, "A Belief-Driven Method for Discovering Unexpected Patterns," Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining, pp. 94-100, 1998.*

[6] *E. Keogh, S. Lonardi, and B. Chiu, "Finding Surprising Patterns in a Time Series Database in Linear Time and Space," Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD), pp. 550-556, 2002.*

[7] *W. Hamalainen and M. Nykanen, "Efficient Discovery of Statistically Significant Association Rules," Proc. IEEE Eighth Int'l Conf. Data Mining, pp. 203-212, 2008.*

[8] *D.H. Li, A. Laurent, and P. Poncelet, "Mining Unexpected Sequential Patterns and Rules," Technical Report RR-07027, Laboratoired'Informatique de Robotiqueet de Micro'electronique de Montpellier, 2007.*

[9] *J. Yang, W. Wang, and P. Yu, "InfoMiner: Mining Surprising Periodic Patterns," Data Mining and Knowledge Discovery, vol. 9, no. 2, pp. 189-216, 2004.*

[10] *A. Gionis, H. Mannila, T. Mielika¨inen, and P. Tsaparas, "Assessing Data Mining Results via Swap Randomization," ACM Trans. Knowledge Discovery from Data, vol. 1, no. 3, pp. 167-176, 2007.*

[11] *J. Li, G. Liu, L. Wong, "Mining Statistically Important Equivalence Classes and Delta-Discriminative Emerging Patterns," Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 430-439, 2007.*

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

[12] *B. Padmanabhan and A. Tuzhilin, "On Characterization and Discovery of Minimal Unexpected Patterns in Rule Discovery," IEEE Trans. Knowledge and Data Eng., vol. 18, no. 2, pp. 202-216, Feb. 2006.*

***[13]** M. Blanchette and S. Sinha, "Separating Real Motifs from Their Artifacts," Bioinformatics, vol. 17, suppl. 1, pp. S30-S38, 2001*

[14] *S.C. Chan and A.K.C Wong, "Synthesis and Recognition of Sequences," IEEE Trans. Pattern Analysis Machine Intelligence, vol. 13, no. 12, pp. 1245-1255, Dec. 1991.*

[15] *A.K.C. Wong, D.K.Y. Chiu, and S.C. Chan, "Pattern Detection in Biomolecules Using Synthesis Random Sequence," J. Pattern Recognition, vol. 29, no. 9, pp. 1581-1586, 1995.*

[16] *S. Aluru and P. Ko, "Lookup Tables, Suffix Trees and Suffix Arrays," Handbook of Computational Molecular Biology, CRC Press, 2006.*

[17] *N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemset for Association Rules," Proc. SeventhInt'l Conf. Database Theory, pp. 398-416, 1999.*

[18]*J. Cheng, Y. Ke, and W. Ng, "δ-Tolerance Closed FrequentItemsets," Proc. Sixth Int'l Conf. Data Mining, pp. 139-148, 2006.*

[19] *R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizationsand Performance Improvements," Proc. Fifth Int'l Conf.Extending Database Technology, pp. 3-17, 1996.*

[20] *J. Pei and J. Han, "PrefixSpan: Mining Sequential PatternsEfficiently by Prefix-Projected Pattern Growth," Proc. 17th Int'lConf. Data Eng., pp. 215-224, 2001.*

[21]*Andrew K.C. Wong, Fellow, IEEE, Dennis Zhuang, Gary C.L. Li, Member, IEEE, andEn-Shiun Annie Lee "Discovery of Delta Closed Patterns andNoninduced Patterns fromSequences" IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 24, NO. 8, AUGUST 2012*

Authors

Mr.Tamboli Suraj M..received his B.E.degree in Computer Science (FirstClass with Distinction)in the year 2011 from BAMU, Aurangabad and persuing in M.E.Degree in ComputerEngineering fromPuneUniversity.He has 04 years of teaching experience at undergraduate level.His reaserch lies in Data mining.

Prof. Prabhudev Irabashetti
He is Currently working as Assistant Professor in Department of Computer Engineering of VACOE, Ahmednagar,Pune University.His reaserch lies in Data mining.

Prof. Sutar Mushtaq Shakil
He is Currently working as Assistant Professor in Department of Information Technology of ADCOEAT,Ashta.His reaserch lies in Data mining.