Comparative Analysis of Effective Pattern Search Implementation

Vasudeva Reddy.P*, Kumar Vasantha #,Prof. C.Mohan rao $
Dept of CSE, Avanthi Institute of Engineering and Technology ,makavarapalem
*student M.Tech 2 nd Year ,Dept of CSE, Avanthi Institute of Engineering and Technology
HOD,Dept of CSE, Avanthi Institute of Engineering and Technology
$ Professor and Principal ,Dept of CSE, Avanthi Institute of Engineering and Technology

Abstract: In this paper we are introducing an efficient comparative analysis approach with different pattern search mechanism, A new algorithm to search for multiple patterns at the same time is presented.. In this paper we analyzed optimal pattern search mechanisms like quick large pattern matching algorithm and temporal pattern search algorithms. Quick large pattern search mechanism works with Shifted table and next array. TPS algorithm works with presence and absence function.

I. INTRODUCTION

Text-processing systems must allowth eir users to search for a given character string within a body of text. Database systems must be capable of searching for records with stated values in specified fields. Such problems are instances of the following string-matching problem: For a specified set ((X(i) ,Y (i)))o f pairs of strings, determine, if possible, an r such that X(r) = Y(r). Usually the set is specified not by explicit enumeration of the pairs, but rather by a rule for computing the pairs (X(i)Y, (i))f rom some given data.[1][2][3].

a general framework that can be specialized to yield several particular string-matching problems and algorithms. An instance of the general stringmatching problem is specified by Positive integers n and t. An index set R of cardinality t. 0 For each r E R, strings X(r) and Y(r) in (0, 1)". The problem is to decide whether there exists an index 7 such that X(r) = Y(r) and, if so, to find one such index. Particular string-matching problems lead to particular choices of R and particular rules for determining X(r) and Y(r) from the input data and the index r. We indicate two examples.

Pattern matching is one of the important concepts of Data Mining. In a standard formulation it is required to search for the string pattern in the string text. If the string pattern is present in the string text then we have to find the position of the first occurrence of pattern string in the string of text and this process of searching for a pattern string over streams of data is called Pattern Matching approach. [1][2][3].

Network security applications such as virus scan software, anti-spam software, and firewall use pattern matching algorithms to extract the threat from the network by properly tracking incoming traffic for suspicious contents. When Pattern matching algorithms are used to such applications the speed of the algorithm usually forms the bottleneck. Many algorithms have been designed in the literature as improvements of Brute Force algorithm each of which tries to avoid problems of existing algorithms. Still to determine which of the algorithms is the best depends on the application where the algorithm is to be used. A command line language such as SQL, and then review the results in an analysis tool, there are high costs associated with this approach. Because the multiple turnarounds are detrimental to the process and interfaces exist to circumvent the learning curve of query languages. Searching for temporal patterns is often unsupported by the interface, in favor of the simpler Boolean or conjunctive queries. Finally, searching temporal patterns on personal histories that have hundreds or thousands of events with tens of thousands of histories in a database can take a long time.

II. METHODOLOGY

Even though various traditional mechanisms available for pattern search mechanisms, they are not optimal, because either they work with suffix based approach or prefix based approaches, here time complexity issues takes place while searching the string in all patterns.

Temporal events are the time based events, it days interesting fields in the recent days of medical field event atterns,Various sql approaches are availble,but the problem with the number of join operations and one more disadvantage is we can not insist the analyst to learn the SQL language to process and visualize the patterns, traditional approaches are not feasible when we have more number of records

We allow analysts to specify a temporal pattern such that each item in the pattern can be a presence item or an absence item (negation). These items are selected from a list of combo boxes. The ordering of the combo boxes determines the temporal ordering. We do not allow other additional temporal constraints (e.g., a Stroke occurs within three days after a Heart Attack) in this interface. However,

this kind of temporal constraints can be specified via temporal summaries in our interface [28]. Once a pattern is specified, Lifelines2 finds all records that contain this pattern and visually filters out the rest. As these decisions were made to support existing features of the tool and before we designed our Temporal Pattern Search algorithm, the constraints represent interesting design challenges. In Lifelines2, each record is represented as a set of sorted arrays of events. There is one array for each event type. All events of the same type are sorted by their time stamps in their array. This decision comes from the following constraints:

Data constraint: It seems most straight forward to store all events on a single sorted array regardless of type. However, for events that have the same timestamp, this scheme can create conflicts, mislead analysts, and produce wrong results. In order to merge events that are in fact the same but come from two cylindrical depots data sources. While this assumption is practical and reasonable for personal records, it may not apply to all temporal event data.

Drawing constraint: Lifelines2 maintains a drawing order of events by event types. Lifelines2 maintains the z-order by event types to avoid visual inconsistencies that can potentially disrupt analytical tasks. The separated arrays would allow the drawing algorithm an efficient way to access events of the same type.

Interface constraint: it is useful to analysts to hide event types that are not of interest. These interface features involve finding event data of a specific type. Separating events into different arrays by type would allow Lifelines2 to afford these features most efficiently.

Index:	0	1	2	3	4	5	6	7	8	9
Time:	0	10	20	10	40	50	60	70	80	90
Record **R=A**	A	C	E	B	C	A	C	D	C	F

Index:	0	1	2	3	4	5
Pattern	A’	B’	C	D’	E’	F

	Current index χ and current time τ	**Binary Search performed in matching pattern with data**	**Code Line #**	**β**	**T**	**Δ**	**Φ**	**π**	**χ**	**τ**	**δ**
	Initialization			False	[ ]	[ ]	[ ]	[ ]	0	-∞	-1
1	$\chi = 0, \tau = -\infty$	Matching P[0]=A…R[0]	(73-78)	False	[0]←0	[0]←-1			1	0	0
2	$\chi = 1, \tau = 0$	Matching P[1]=B’...R[3] Matching P[2]=C....R[1]	(48-53)	True	[2]←10	[2]←0	[0]←30	[2]←True	3	10	2
3	$\chi = 3, \tau = 10$	Matching P[3]=D’...R[7] Matching P[4]=E’...R[2] Matching P[5]=F....R[9]	(42-46)	True				[5]←True	2	19	0
4	$\chi = 2, \tau = 19$	Matching P[1]=B’...R[3] Matching P[2]=C....R[4]	(68-70)	False					0	29	-1
5	$\chi = 0, \tau = 29$	Matching P[0]=A....R[5]	(73-78)	False	[0]←50	[0]←-1			1	50	0
6	$\chi = 1, \tau = 50$	Matching P[1]=B’..NIL	(30-34)				[1]←∞		2		2
7	$\chi = 2, \tau = 50$	Matching P[2]=C....R[6]	(73-78)	False	[2]←60	[2]←0			3	60	0
8	$\chi = 3, \tau = 60$	Matching P[3]=D’...R[7]	(42-46)	True				[5]←True	2	69	

		Matching P[4]=E'...NIL Matching P[5]=F....R[9]									
9	$\chi = 2, \tau = 69$	Matching P[2]=C...R[8]	(73-78)	False	[2]←80	[2]←0			3	80	2
10	$\chi = 3, \tau = 80$	Matching P[3]=D'...NIL Matching P[4]=E'...NIL	(30-34)				[2]←∞		5		
11	$\chi = 5, \tau = 80$	Matching P[5]=F....R[9]	(73-78)	False	[5]←90	[5]←5			6	90	5

Fig. 1

III. TPS ALGORITHM

We present Temporal Pattern Search (TPS), a novel algorithm for searching for temporal patterns of events. The traditional method of searching for such patterns uses an automaton-based approach over a single array of events,. Instead, TPS operates on a set of arrays, where each array contains all events of the same type. TPS searches for a particular item in the pattern using a binary search over the appropriate arrays. Although binary search is considerably more expensive per item, it allows TPS to skip many unnecessary events.

This algorithm using efficient back tracking with the status flags and uses IS_A_MATCH(),PRESENCE() and ABSENCE METHODS . IS_A_MATCH takes one pattern at a time for search in the records. When processing an item in the pattern, if it is a presence item, IS_A_MATCH(R,P) calls HANDLE_PRESENCE(R,P), which attempts to find an event that satisfies the current item. If it is an absence item, TPS calls HANDLE_ABSENCEEVENT(R, P), which finds the next absence event, and checks to see if that absence event occurs between the previous presence item match and the next presence item match. If it does, then a constraint is violated, and the algorithm backtracks. Backtracking means TPS tries to look for an alternative to one or more of its previously made matches. The algorithm increments the variables x (the current item on pattern) and T (the current time) when processing the search. When backtracking occurs, TPS rolls back these variables, among others, appropriately to restart a previous search.

STARTING FUNCTION AND INITIALIZATION OF GLOBAL VARIABLES

```
IS_A_MATCH(R, P){
    β FALSE      //backtrack flag
    T[P. Length]          //matching times
    Δ[P .Length]          //last pos item
    Φ[P. Length+1]        //next neg item time
    Π[P. Length]          //if has neg item before
    χ 0                   //current index
    T -∞                  //current time
    Δ-1                   //last pos item
    While(χ<P.Length)
        If(χ == -1)
            Return False
          If(P[χ.isNeg])
             HANDLE_ABSENSE(R,P)
           Else
             HANDLE_PRESENSE(R,P)
    Return TRUE;
}
```

THE HANDLE ABSENSE FUNCTION

```
HANDLE_ABSENCEEVENT(R,P)
{   minTimeNIL
   numAbs0
For(iχ to P.Length)
ItemP[i]
If(NOT item.isNeg) break
numAbsnumAbs+1
absEventNEXT(R[item,Type],τ)
If(absEvent==NIL)
        Continue
minTime MIN(absEvent.Time,minTime)
If(minTime==NIL)
If(minTime==NIL)
If(δ>-1)
          Φ[δ] ∞
         else
Φ(ϕ.Length-1]∞
χχ+numAbs
     Else
If(χ+numAbs<P.Length)
nItemP[χ+numAbs]
nEventNEXT(R[nItem.Type],τ)
Π[χ+numAbs]TRUE
If(nEvent==NIL OR nEvent.Time>minTime)
nEvent.Time>minTime
Xδ
If(χ>0)
δΔ[χ]
```

```
                    TminTime-1
                BTRUE
            Else
T[χ+minAbs]nEvent.Time
            Φ[δ]minTime
                Δ[χ+numAbs] χ-1
                δ χ+numAbs
                X χ+numAbs+1
                TnEvent.Time
        Else
            χδ
            ΔΔ[χ]
            TminTime-1
            BTRUE
}

THE HANDLE_PRESENCE FUNCTION
HANDLE_PRESENCE(R,P)
    {  EventNEXT(R[P[χ].Type,τ)
    If(event==NIL)
        X-1
    Else
        backtrackingMoreFALSE
        If(β AND π[χ])
            If(Δ[χ]<0)
                badTimeϕ[ϕ.Length-1]
            Else
                badTimeϕ[Δ[χ]]
            If(badTime<event.Time)
                χΔ[χ]
                τϕ[χ]-1
                Δ Δ[χ]
                backtrackingMoreTRUE
        If(backtrackingMore) return
        τevent.Time
        T[χ]event.Time
        Δ[χ]δ
        Δχ
        X χ+1
        BFALSE
}
```

TPS searches for each and every event in the record until the pattern is completed. For each and every, iteration it checks whether the event is negative or not, if the event is negative event it makes a call to absence function otherwise it makes a call to presence function.

If it is a presence event in the record it maintains the current index and position of the presence item in the respective positions. If it is an absence event and presents in previous presence item and next present item, it needs to back track. No need to back track when there is no absence event.

Consider an example of pattern AB'CD'E'F and the record as R = ACEBCACDCF, Initially declare the variable for current index of event, Matched time stamp, item last position, time stamp of the next negative item, Boolean array for status, if it contains the negative item before.

In our example Initially it starts the search with initial event A at P[0] and available at record R[0] at time stamp updated to 0, it simply increments the current index, β is false, because there is no back tracking. In the second iteration now the current event B is an absence event and presence between the previous present event A and next immediate presence event B and makes a call next function of event in the record, currently no violation and no need to back track, simply maintains the time stamps and current positions or index of the current event.

In the next phase it again deals with the absence block DE presents between the previous presence Event C and immediate next F and now set the back tracking flag to True. Reduce a minimal time stamp during the backtrack, the current time stamp is 20(i.e. R[2]= E),it reduces to 20-1=19 for smallest time granularity in consideration

In the next iteration it finds the third position event C in R[4] again it violates the constraint again there is a back track and increments the current position.

In the next iteration it finds the p[0]=A in R[5] and sets the back tracking flag t false, while processing B, there are no more B but next negative event position is maintained and sets the negative event to NIL for further process

In the Seventh iteration C available at R[6] and sequentially D available at R[7] and next available in R[9] another back track with D and E and again set them to Nil as above process and initialize the other variables, new C found in R[8] in iteration 9,no violations, finally matches the last items and returns True.

1) The seat shifted table:

Use the parallel technology for the establishment of a chain. The establishing rules of the seat shifted table are as follows

a) Handling alphabet :

According to alphabet size, definite first level size of the seat shifted table. Assuming that the size of an alphabet is to SIZE, and then the size of the first level is to SIZE. Each character uses its value of the decimal base corresponding with its ASCII to mark the first level of the position

b) Handling the Pattern:

First mark the location of the characters in the pattern from left to right, and then the positions of each

character which appears in the pattern string according to the decrease order, in turn enter the position which is indicated with its ASCII, which would constitute a chain of other levels

Next Array:

Next array holds the pattern individual characters with their presence bits, if the character occurs previously we need to mention the previous character position, this mechanism maintains window size approach for individual search with 2m-1.

Pattern Search:

Now consider window size content from the large pattern, select a character with respect to the next array size and search that character in the shift table if it is present get the position and if it is not zero go to the next position that decides the number of characters, which are divided from the left side and right side. If it is not present leave the first next array sized characters and move to next window sized character until the pattern found.

Future Enhancement:

Main drawback in this approach is, Even though the initial event or any event in the pattern not present in the record pattern proceeds with the next event, it leads to the useless computation of search over the all patterns. We can solve this issue by ignoring the process of searching when the event (i.e. initially or middle event of the pattern) not present in the record.

We can enhance the procedure by initially checking the all the negative items, if not available in the search pattern, we need not to declare and initialize the negative array initializes; it preserves the space and obviously gives the optimal time complexity.

Integration of new pattern helps the future record patterns when the search pattern not available in the record set, specifically in the field of medical events.

IV. CONCLUSION

Even though various pattern search algorithms available for searching the patterns from the large set of patterns. Time complexity and performance are important factors during the searching. Traditional approaches are suffers from the various problem we explained above ,our quick large pattern and temporal pattern search approach gives an efficient search performance than the traditional suffix and prefix based approaches.

REFERENCES

[1] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman, “Efficient Pattern Matching over Event Streams,” Proc. ACM SIGMOD Int’l Conf. Management of Data, pp. 147-160, 2008.

[2] R.S. Boyer and J.S. Moore, “A Fast String-Searching Algorithm,” Comm. ACM, vol. 20, no. 10, pp. 762-772, 1977.

[3] R. Cox, “Regular Expression Matching Can Be Simple and Fast,” http://swtch.com/rsc/regexp/regexp1.html, 2007.

[4] DataMontage, http://www.stottlerhenke.com/datamontage/2011.

[5] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White, “Towards Expressive Publish/Subscribe Systems,” Proc. 10th Int’l Conf. Extending Database Technology (EDBT), pp. 627-644, 2006.

[6] J. Fails, A. Karlson, L. Shahamat, and B. Shneiderman, “A Visual Interface for Multivariate Temporal Data: Finding Patterns of Events across Multiple Histories,” Proc. IEEE Symp. Visual Analytics Science and Technology (VAST ’06), pp. 167-174, 2006.

[7] D. Ficara, S. Giodano, G. Procissi, F. Vitucci, G. Antichi, and A.D. Pietro, “An Improved DFA for Fast Regular Expression Matching,” ACM SIGCOMM Computer Comm. Rev., vol. 38, no. 5, pp. 29- 40, 2008.

[8] L. Harada and Y. Hotta, “Order Checking in a CPOE Using Event Analyzer,” Proc. ACM Int’l Conf. Information and Knowledge Management (CIKM), pp. 549-555, 2005.

[9] L. Harada, Y. Hotta, and T. Ohmori, “Detection of Sequential Patterns of Events for Supporting Business Intelligence Solutions,” Proc. Int’l Database Eng. and Applications Symp. (IDEAS ’04), pp. 475-479, 2004.

[10] J.E. Hopcroft, R. Motwani, and J.D. Ullman, Introduction to Automata Theory, Languages, and Computation. Addison-Wesley,2000.

[11] R.M. Karp and M.O. Rabin, “Efficient Randomized Patter
Matching Algorithms,” Technical Report TR-31-81, Aiken Computation Laboratory, Harvard Univ., 1981.

[12] D.E. Knuth, J.H. Moris, and V.R. Pratt, “Fast Pattern Matching in Strings,” SIAM J. Computing, vol. 6, no. 2, pp. 323-350, 1977.

[13] S. Kumar, B. Chandrasekaran, J. Turner, and G. Varghese, “Curing
Regular Expressions Matching Algorithms from Insomnia, Amnesia, and Acalculia,” Proc. Third ACM/IEEE Symp. Architecture for Networking and Comm., Systems (ANCS), pp. 155-164, 2007.

[14] H. Lam, D. Russell, D. Tang, and T. Munzner, “Session Viewer: Visual Exploratory Analysis of Web Session Logs,” Proc. IEEE Symp. Visual Analytics Science and Technology (VAST ’07), pp. 147- 154, 2007.

BIOGRAPHIES

P.Vasudeva Reddy: He has obtained B.Tech in Information Technology from Acharya Nagarjuna University Guntur and pursuing M.Tech in Computer Science and Engineering at Avanthi Insistute Of Engneering and Technology Makavarapalem, interested in Java and DBMS

Mr. kumar vasantha: He has obtained M.Tech in Software Engineering from jawaharlal nehru technological university Kakinada, He has published 10 papers in National and International journals his interested areas are Web Technologies,Network Security

Dr. C P V N J Mohan Rao : He has obtained M.Tech in Computer Science and Technology from Andhra University College of Engineering and awarded Ph.D by Andhra University during 2000. He has 18 years of teaching and research experience and guided number of M.Tech students for their projects. He has published 23 papers in National and international journals. He is guiding research scholars for Ph.D. He received Best Teacher award from JNTU, Kakinada during 2009.