International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-9 Issue-4 November 2020

Discovering Periodic Patterns in Time Series from Twitter Data Set

Elavarasi D

Abstract: ***The important class of regularities that exist in a time series is nothing but the Partial periodic patterns. These patterns have key properties such as starting, stopping, and restartinganywhere— within a series. Partial periodic patterns areclassifiedinto two types: (i) regular patterns— exhibiting periodic behavior throughout a series with some exceptions and(ii) periodic patterns exhibiting periodic behavior only for particular time intervals within a series. We have focused primarily on finding regular patterns during past studies on partial periodic search. The knowledge pertaining to periodic patterns cannot be ignored. This is because useful information pertaining to seasonal or time-based associations between events is provided bythem. Because of the foll o wi n g two main reasons, finding periodic patterns is a non-trivial task.***

(i) Each periodic pattern is associated with time-based information pertaining to its durations of periodic appearances in a series. Since the information can vary within and across patterns, obtaining this information ischallenging.

(ii) As they do not satisfy the anti-monotonic property, finding all periodic patterns is a computationally expensive process. In this paper, periodic pattern model is proposed by addressing the above issues. Periodic Pattern growth algorithm along with an efficient pruning technique is also proposed to discover these patterns. The results through Experimentation have shown that Periodic patterns canbe really useful and it has also proven that our algorithm isnoteworthy.

Keywords: Categories and Subject Descriptors H.2.8 [Database Management]: Database Applications - Data Mining. General Terms Algorithms

I. INTRODUCTION

Time series is defined as a collection of events obtained from sequential measurements over time. Periodic pattern mining involves the finding of all patterns that unveil either complete or partial cyclic repetitions in a timeseries.

Finding *regular forms*, i.e., patterns displaying either whole or unfinished cyclic repetitions throughout a series [1, 2, 3, 4, 5, 6, 7, 8, 9] are focused in Past studies and it also dealtwith partial periodicsearch. In regular pattern of milk and groceries states that customers have been buying items '*milk*' and '*groceries*' almost daily during the course of the year can be considered as a classical example.

Partial periodic pattern which is considered as a useful related type is nothingbut the Periodic *patterns*, i.e., patterns that exhibits cyclic repetitions only for specific time intervals within aseries. In Periodic pattern of *Jackets, ttloves* states that customers have often purchased '*Jackets*' and '*ttloves*' from 10-October-2012 to 26-February-2013 and from 2-November-2013 to 2-March- 2014 is a perfect example.

The main purpose of this paper is to determine repetitive patterns by addressing mining tasks.Normally,Periodicpatternsarepervasiveinveryhuge datasets.Usefulin-formation could be provided concerning to cyclical or time-based relations between items, in many real- world applications. In marketing, a user may be concerned in describing periodic purchases for remarkable inventory controlling. Correspondingly, a social network data analyst may be concerned in attaining sequential data relating to bursts of hashtags, such as #earthquakes, #radiation and #floods. And also, a proficient in the health-care sector may be concerned in finding seasonal diseases in a geographical location. An administrator may be engrossed in attaining time-based information of profoundly visited web pages mainly to improve web site design andadministration. A set of huge stocks indices that rise occasionally for a specific time interval may be of special interest to enterprises and persons, in the stock market. An administrator might be concerned in discovering high sternness events (e.g. tumbling failure against regular mundane events (e.g. data backup), in the field of computer network.

Unfortunately, because of the following reasons, finding periodic patterns becomes a non- trivial task.

1. Each and every periodic pattern is related with time-based information relating to its durations of periodic appearances within the data. Becauseof theinformation thatcan vary within and acrosspatterns, attaining this information becomes challenging.
2. A symbolic sequence is taken into account a time series as in most current periodic pattern mining approaches; therefore, the actual time-based are not taken into account information of events bythem.
3. The *anti-monotonic property is not satisfied by the* Periodic patterns. That is, all filled subgroups of a periodic pattern might not be periodic patterns. Because of the increase in the search space, there will be increase in the computational cost of finding thesepatterns.

Revised Manuscript Received on October 10, 2020.
Manuscript Received On October 06, 2020
Elavarasi D., HOD, Department of Computer Science & Engineering, Mount Zion College of Engineering and Technology, Pudukkottai (Tamil Nadu), India.

Therefore, the challenging factor is to develop efficient pruning techniques to reduce the search space.

4. Since periodic behavior are exhibited by regular patterns through- out a series with some exceptions, time-based information pertaining to the intervals of periodic appearances of a pattern within the sequence are not obtained by regular pattern mining algorithms. As a result, these algorithms cannotbe stretched for finding periodic patterns.

5. In real-life, periodic patterns including rare items can be interesting to users. For example, the knowledge pertaining to rare events, such as regular events is not as important as cascading failures for a network administrator. Nevertheless, since rare items appear infrequently in the data, finding such patterns is difficult. Categorizing things into common or uncommon is biasedand depends on theuser and/or application requirements.

In this paper, a model that addresses all the above-mentioned issues while finding periodic patterns is proposed. In particular, time series is taken into account by our model as a time-based sequence and models it as a transactional database with connections ordered in respect to a particular times- tamp (without loss of generality).

Our model comprises of three novel measures, periodic-support, periodic-interval and recurrence, mainly to determine the lively periodic behavior of periodic patterns. The total times of consecutive
recurring repetitions of a form in a subset of data is determined by Periodic-support. The time interval (or window) pertaining to the periodic appearances of a pattern within a series is determined by Periodic- interval. The number of interesting periodic intervals of a pattern is determined by Recurrence. At last, a pattern- development algorithm along with a notable pruning technique is proposed to discern periodic patterns effectively. Our algorithm is called as periodic pattern-growth (RP-progression). Investigational results show that RP-progression is noteworthy and useful information can be provided by periodic patterns in many real-life applications.

The remaining section of the paper is systematized as follows: associated work on mining periodic patterns is described in Section 2. our model of periodic patterns is introduced in Section 3. RP-growth is presented in 4th Section. Investigational results are reported in 5th Section. Finally, in the 6th Section the paper concludes with future researchdirections.

II. RELATEDWORK

Since the overview of partial periodic patterns [5], the issue of discovering these patterns has acknowledged a great deal of consideration [6, 8, 10, 11, 12]. The model used in all these studies, nevertheless, remains the same. That is, a time series is taken into account as a symbolic sequence and discovers all patterns using the subsequent steps:

1. Segregating symbolic sequence into distinct subdivisions (or period-segments) of a fixed length (or period).
2. Determine all partial periodic patterns that gratify the user- described minimum support (minSup), which regulates the slightest number of period sections that a pattern must cover though the sequence. The foremost restriction of the above studies is that the actual time-based information of the events within a sequence is not taken into account. To address this issue, a time order as a time-based sequence was modeled b Ma and Hellerstein [7] and a model to discern a class of fractional periodic patterns known as p-patterns was proposed. A pattern is considered in this model as fractional periodic if its numberof
episodic appearances all the way through the sequence pleases the user- defined minSup. It should be noted that the concept of minSup is not the same in both regular pattern mining and unfinished periodic pattern mining.

The minimum number of appearances of a pattern throughout the data is organized and controlled by minsupowever in frequent pattern mining, the minimum number of periodic appearances (or cyclic repetitions) of a pattern throughout the data is controlled by minSup controls in fractional periodic pattern mining. Therefore, the incomplete periodic patterns discovered in all the above studies represent regular patterns. On the other hand, our study was focused on discovering periodic patterns in a time-based sequence. Moreover, Ma and Heller stein's model cannot be extended for finding periodic patterns. The reasons are as follows:

1. Their model fails to attain the time-based information relating to the durations of periodic appearances of a pattern within thedata.
2. Finding p-patterns with a single minSup directs to the quandary known as the "sporadic itemproblem".

Those patterns that include rare items will not be found, if minSup is fixed excessively high. MinSup has to be set very low because to find forms comprising both recurrent and infrequent items. Nevertheless, this can lead to combinatorial blast producing too many patterns. In precise, many monotonous aperiodic patterns can be discovered as partial periodic patterns. For example, if we set a low minSup, sayminSup
= 3%, then we will be discovering an uninteresting aperiodic pattern that has only 3% of its periodically appearances throughout the data as a partial periodic pattern.

In recent past, researchers have been investigating the complete cyclic conduct of the recurrent patterns in a transactional database to determine a class of user-interest-based patterns known as periodic- recurrent patterns.

Nonchalantly, a recurrent pattern satisfying minSup is said to buy periodic-frequent if and only if all its inter-arrival times throughout the database satisfy the user-defined period thresh- old value. Thus, these studies were fixated on discovering regular patterns in a transactional databank. For our study, the partial cyclic behavior is investigated same the patterns to discover periodic patterns; consequently, make a sweeping statement regarding the current model of periodic- recurrent patterns. More importantly, none of the approaches presented in model time series data as a transactional database.

Instead, they are based on the implicit assumption that there are transactional databases with a sequentially ordered set of transactions. This paper fills the gap by describing the procedure to model time sequences data as a transactional database without loss ofgenerality. Yang et al. examined the change in sporadic behavior of patterns due to the intrusion of random noise and introduced a class of user-interest based patterns called as asynchronous periodic patterns. Even though asynchronous- periodic pattern mining is narrowly related to our work, it cannot be prolonged for finding periodic patterns.

The reason is asynchronous periodic pattern mining models a time series as a figurative sequence; as a result, it does not consider the actual time-based information of the events within a categorization.

The issue of locating progressive patterns and frequent episodes has received a great deal of attention. Nevertheless, it should be recorded that periodicity is investigated the problem of finding cyclic association rules. Nevertheless, that study is quite obstructive in finding the patterns that are present at every cycle.

Finding partial periodic designs motifs and periodic forms has also been calculated in time sequences; rather than figurative patterns.

On the whole, the offered model of finding periodic patterns is unique and is separate from current models. In the next section, we introduce our model of periodicpatterns.

III. PROPOSEDMODEL

At first, time series is defined in this section. Then, these series is represented as a transactional database and familiarize measures to discover periodic patterns.

DEFINITION 1. Let I be a set of items (or event types). An event is a pair (i,ts), where i I is an item and ts R is the timestamp of the event. Let X I be a pattern. An event sequence S isan

ordered collection of events, i.e., (i1, ts1), (i2, ts2), , (iN , tsN) , where ij I is an item at the j- th event. The term tsj represents the occurrence timestamp of the event, and tsh tsj for 1 h j N[7].

DEFINITION 2. A point sequence is a well-organized collection of happening times. Given an event sequence S = {(i1, ts1), (i2, ts2), ·· · , (iN , tsN)}, there is an implied point sequence, S = ts1, ts2, , tsN . An event sequence can be noticed as a combination of multiple point categorizations of each item.

Let TSD denote the time series data (or a set of events) being mined.

EXAMPLE 1. Figure 1 shows a TSD with a set of items I = a, b, c, d, e, f, g . In this figure, an item of each event is labeled above its occurrence timestamp. We should note that no item appears at the timestamps of 8 and 13. The item ‘a’ appears at the timestamps of 1, 2, 3, 4, 7, 11, 12and 14. Thus, the event sequence of ‘a’ is represented as Sa = {(a, 1), (a, 2), (a, 3), (a, 4), (a, 7), (a, 11), (a, 12), (a, 14)}.The point sequence of ‘a’ is represented as Sa = {1, 2, 3, 4, 7-, 11, 12, 14}. Likewise, the point sequences of ‘b’ and ‘ab’ are ^^TSab. Therefore, Sup(ab) = |{1, 3, 4, 7, 11, 12,14}|

An important role is played by the point sequence in measuring the periodic behavior of the patterns in a time series. The time-basedly ordered transactional database is described now which preserves the point sequence of items in the TSD.

Figure 1: Running example: time-based sequence consisting of items from ‘a’ to ‘g’

A transaction, tr = (ts, Y), is a tuple, where ts epitomizes the timestamp and Y is a pattern. A transactional database TDB over I is a set of transactions, TDB = tr1, , trm , m = T DB , where TDB is the size of the TDB in total number of transactions. For a transaction tr = (ts, Y), such that X Y , it is said that X oc- curs in tr and such a timestamp is denoted as tsX . Let TSX = {tsX , ⋯ , tsX }, where 1 ≤ k ≤ l ≤ m, denote an ordered set of timestamps at which X has occurred in the TDB. The TSX in theTDB

is the same as the point sequence of X in the TSD. Thus, any information pertaining to the time-based appearances of a pattern in the data is not missed.

Example 2. Table 1 shows the transactional database assembled by grouping the items appearing together at a specific timestamp in Figure 1. Each transaction in this database is exceptionally recognizable with a timestamp. All transactions have been well-organized with respect to their timestamps. It can be observed that the constructed database does not contain the transactions with timestamps 8 and 13. The reason is that no item appears at these timestamps in Fig- ure 1. In this database, the pattern ‘ab’ appears at the timestamps of 1, 3, 4, 7, 11, 12, and 14.Therefore,

TSab = {1, 3, 4, 7, 11, 12, 14}.

Table 1: Transactional database constructed from time-based sequence shown in Figure 1. The term ‘ts’ is an acronym for timestamp

ts	Items
1	*a, b, g*
2	*a, c, d*
3	*a, b, e, f*
4	*a, b, c, d*
5	*c, d, e, f,g*
6	*e, f, g*

ts	Items
7	*a, b, c, g*
9	*c, d*
10	*c, d, e ,f*
11	*a, b, e, f*
12	*a, b, c, d, e, f, g*
14	*a, b, g*

Definition 3. (Support of pattern X) The number of transactions containing X in the TDB is defined as the sup- port of X and denoted as Sup(X). That is, Sup(X) = |TS|.

Example 3. The support of ‘ab’ in Table 1 is the size of TSab. Hence, Sup(ab) = |{1, 3, 4, 7, 11, 12, 14}| = 7.

Definition 4. (Periodic appearance of pattern X)

Let tsX , tsX ∈ TSX , 1 ≤ j < k ≤ m, signify any two succeeding timestamps in TSX . The time difference between tsX and tsX is referred to as an inter-arrival time of X, and denoted as iatX . That is, iatX = tsX − tsX . Let IAT X = {iatX , iatX, ⋯ , iatX }, k = Sup(X) − 1, be set of all inter-arrival timesof X inT DB. An inter-arrival time of pattern X is said to be periodic (or interesting) if it is no more than the user- defined period threshold value. That is, a iatX ∈ IAT X is said to be periodic if iatX ≤ per, threshold value. Consequently, the following two definitions areintroduced.

Figure 2: Inter-arrival times of ‘ab’, IAT ab

Example 4. The pattern ‘ab’ has primarily appeared at the timestamps of 1 and 3. An inter-arrival time of ‘ab’ is given by the difference between these two time stamps. That is, iatab 1 = 2 (= 3 − 1). Similarly, other inter-arrival times of ‘ab’ are iatab 2 = 1, iatab 3 = 3, iatab 4 = 4, iatab 5 = 1 and iatab 6 = 2.

Therefore, the resultant IATab = {2, 1, 3, 4, 1, 2}. If the user-defined per = 2, then iatab 1 , iatab 2 , iatab 5 and iatab 6 are considered the periodic occurrences of ‘ab’ in the data. On the other hand, iatab 3 and iatab 4 are considered the aperiodic occurrences of ‘ab’ as iatab

3 per andiatab4 per. Figure 2 shows the set of all inter-arrival times forpattern

‘ab’, i.e., IATab. The thick lines represent the inter-arrival times that satisfy the period, while the dotted lines represent the inter-arrival times that fail to satisfy the period.

Most current partial periodic pattern mining approaches use minSup to measure the periodic interestingness of a form. This measure cannot be used for finding periodic patterns because it dictates the

minimum number of cyclic repetitions a pattern must have in all the data. Therefore, we introduce the following measures to define the fractional periodic behavior of periodic patterns.

Definition 5

(Periodic-interval of pattern X) Let TSX j = {tsXp, · · · , tsXq } ⊆ TSX, p ≤ q, be a set of timestamps such that 'tsXk ∈ TSX j , p ≤ k <q, tsXk +1 – tsXk ≤ per. The TSX j is a maximal set if there exists no superset in which an inter-arrival time between the two consecutive timestamps is no more than the period. A periodic-interval of X is presented by the range of timestamps in TSX j and is denoted aspiXj

. That is, piXj = [tsXp , tsXq].

Example 5. The maximal sets of timestamps in which ‘ab’ has appeared within the user-defined per = 2 are: TSab 1 = {1, 3, 4}, TSab 2 = {7}, and TSab 3 = {11, 12, 14}. So, the corresponding periodic-

intervals for ‘ab’ are piab 1 = [1, 4], piab 2 = [7, 7], and piab 3 = [11, 14]. Information pertaining to the duration (or window) of periodic appearances of a pattern in a database is obtained by the periodic- interval, as defined above. Most prominently, it can meritoriously determine the periodic durations that can vary within and across patterns. In very large databases, a pattern may have too many periodic-intervals. An efficient technique to reduce this number is to select only those periodic-intervals in which the number of cyclic repetitions of the corresponding pattern satisfies the user-defined threshold value. Thus, the following two definitions are introduced.

Definition 6. (Periodic-support of pattern X) The size of TSX j is defined as the periodic-support of X, and denoted as psXj . That is, psXj = |TSX j|.

Example 6. The periodic-support of ‘ab’ in piab 1 is the size of |TSab 1 |. Consequently, psab 1 = |TSab 1 | = 3. In the same way, the periodic-supports of ‘ab’ in piab 2 and piab 3 are 1 and 3, respectively.

In the real-world applications, some substances appear very recurrently in the data, while others rarely appear. Some rare items have been observed that it also exhibit periodic behavior in a portion of the data. The periodic-support, as defined above, facilitates the user to discover the knowledge concerning to those frequent and rare items that have unveiled adequate number of periodic repetitions in a portion of database. Each and every periodic-interval of a pattern will have only one periodic-support and vice- versa. In other words, there is one-to-one relationship between the periodic-intervals and periodic- supports of a pattern.

Definition 7. (Stimulating periodic-interval of pattern X) Let PIX = {piX1, · · · , piXk} and PSX = {psX1

,-· · · , psXk}, 1 ≤ k, be the complete set of periodic-intervals and periodic-supports of pattern X in the TDB, respectively. A piXk ∈ PIX is said to be an interesting periodic-interval if it’s corresponding psXk

∈ PSX has ps Xk≥ min PS. The user-defined minimum periodic-support is represented by the minPS. Example 7. If the user-defined min PS = 3, then piab1and piab3 are considered the interesting periodic- intervals of ‘ab’. This is because psab1 ≥ min PS and psab3 ≥ min PS. The piab2 is considered an unexciting periodic-interval of ‘ab’aspsab2

min PS. Since very large databases are generally composed over a very long time frame, it has been sensed that some users may stipulate a constraint on the minimum number of interesting periodic-intervals. As a result, the following definitions areintroduced.

Definition 8. (Recurrence of pattern X) Its number of interesting periodic-intervals in a database is represented by the recurrence count of a pattern . Let IPIX ⊆ PIX be the set of periodic-intervals of X such that for every piXk∈ IPIX,its corresponding psXk≥ minPS. The recurrence of pattern X is denoted as Rec(X) =|IPIX|.

Example 8. Continuing with the previous example, PIab = {[1, 4], [11, 14]}. The recurrence of ‘ab’ is the sizeof IPIab. That is, Rec(ab) = |IPIab| = 2.

Definition 9. (Periodic pattern X) Pattern X is a periodic pattern if Rec(X) ≥ min Rec, where min Rec is the user-specified minimum recurrence count. Periodic pattern X is expressed as follows:

X [Sup(X),Rec(X), {{piXk: psXk}|'piXk∈ IPIX}]. (1)

Example 9. If the user-defined minRec = 2, then ‘ab’ isa periodic pattern and is expressed as follows: ab [support = 7, recurrence = 2, {{[1, 4] : 3}, {[11, 14] : 3}}]

The above pattern informs that ‘ab’ has occurred in 7 transactions and its periodic occurrence behavior of once in every two transactions consecutively for at least three times has been observed at two distinct

subsets of a database whose time stamps are in the range [1, 4] and [11, 14]. Table 2 shows the complete set of periodic patterns revealed from Table 1.

Table 2: Periodic patterns in Table 1. Terms ‘Sup,’ ‘Rec’ and ‘IPI’ respectively denote support, recurrence, and interesting periodic-intervals along with their periodic-supports Pattern SupRec

Definition 10. (Problem Definition:) Given a time based sequence (i.e., a TSD), the issue of discovering periodic patterns include determining all those patterns that satisfy the user-defined per, min PS and min Rec constraints. The measures, support, period and periodic-support, can also be stated in percentage of |TDB|.

However, the former definitions for ease of explanation are used. Table 3 lists the taxonomy of different terms used in our model. Grouping the items appearing together at a particular timestamp and storing them in a linked hash table is involved in the construction of a transactional database from a time series. Since this process is simple and straight forward, it is not discussed in this paper.
As an alternative, findingthe
periodic patterns from the constructed database is focused focus.

IV. PROPOSEDALGORITHM

In this part, our pruning technique is introduced first to lessen the computational cost of defining periodic patterns. Next, our algorithm is presented to mine the whole set of periodic forms from the constructed database.
Basic Idea: Candidate patterns
The space of items in a database provides rise to a subset lattice. An item set lattice is a conceptualization of search space while searching user-interest-based patterns. The anti-monotonic property has been widelyusedtoreducethesearchspace. Regrettably, this propertyis notsatisfiedbyperiodicpatterns. The explorationspace,whichinturnupturns thecomputationalcostofminingperiodicpatterns isincreased.
Example 10. Consider the patterns ‘c’ and ‘cd’ in Table
Given the user-defined per = 2, min PS = 3 and min Rec = 2, the interesting periodic-intervals of ‘c’ and ‘cd’are {[2, 12]} and {[2, 5], [9, 12]}, respectively. Therefore, the Rec(c) = |{[2, 12]}| = 1 and Rec(cd) =
|{[2, 5], [9, 12]}| = 2. As the Rec(c) ! min Rec, ‘c’ is not a periodic pattern. Nevertheless, its superset ‘cd’ is a periodic pattern because Rec (cd) ≥ min Rec. Therefore, the anti-monotonic property is not satisfied by the periodic patterns. The same can be observed in Table 2.
The following pruning technique to reduce the computational cost of finding periodic patterns is introduced.
If E rec (X) <min Rec, then neither X nor its supersets will

$$\text{“Let } E_{rec}(X) = \sum_{i=1}^{|ps^X|} \left\lfloor \frac{ps_i^X}{minPS} \right\rfloor.$$

be periodic patterns” The upper bound of recurrence is symbolized by the E rec(X) as a superset of X can have in the database. Thus, Erec (X) is called as the predictable maximum recurrence of a superset of X. The correctness of our snipping technique is straight forward to prove from Properties 1 and 2, and illustrated in Example 11.
Property 1. For the pattern X, Erec(X) ≥ Rec(X).
Property 2. If X ⊂ Y , then TSX ⊇ TSY and Erec(X) ≥ Erec(Y).
Example 11. In Table 1, the item ‘g’ occurs in timestamps of 1, 5, 6, 7, 12 and 14. Therefore, TSg = {1, 5,
6, 7,-12, 14} and S(g) = 6. If per = 2, min PS = 3 and min Rec =2, then TSg1 = {1}, TSg2 = {5, 6, 7}, TSg 3 = {12, 14}, psg1 =|TSg1 | = 1, psg2 = |TSg2 | = 3 and psg3 = |TSg3 | = 2. For this item, Erec(g) = 1= $3i=1%psgi3&=%13&%33&+%23&(.That is, any superset of ‘g’ can at most have recurrence value equal to 1, which is less than the user-defined minRec.Henceforth, pruning ‘g’ will not result in missing of any periodicpattern.Based on our proposed pruning technique, we introducethe following definition. Definition 11. (Candidate pattern X.) Pattern X isa candidate pattern if Erec(X) ≥ minRec.A candidate pattern containing only one item is called acandidate item. The candidate patterns satisfy the antimonotonicproperty (Property 2). Therefore, we use candidate k-patterns to discover periodic (k + 1)- patterns.
RP-growth: Structure, Construction andMining
Traditional Frequent Pattern-growth algorithm [24] cannotbe used for finding periodic patterns. This is because the arrangement of FP-tree seizures only the frequency and neglects the periodic behavior of the patterns in a database. To address this issue, RP-growth introduces an alternative tree structure known as an Periodic Pattern-tree (RP-tree).Our RP-growth algorithm comprises the following two steps:
(i) construction of an RP-tree and (ii) recursive mining ofthe RP-tree to determine the complete set of periodic patterns. Before explaining the above two steps, the structure of an RP-tree is introduced.Structure of RP-tree
A prefix-tree and a candidate item list, known as the RP-list. It is comprised in the structure of an RP-tree.
The RP-list involvesof each distinct item (i) with support (s), expected maximum recurrence (Erec), and a pointer pointing to the first node in the prefix-tree carrying the item. The prefix-tree in an RP-tree bear a resemblance to the prefix-tree in FP-tree. Nevertheless, to attain both frequency and periodic behaviorof
the patterns, the nodes in an RP-tree obviously retain the existence information for each transaction by keeping an existence timestamp list, called as ts-list. To accomplish memory efficiency, the ts-list is sustained by only the last node of every transaction. Henceforth, two types of nodes are sustained in a RP- tree: ordinary node and tail-node.
The previous is a type of node alike to that used in an FP-tree, whereas the latter represents the last item of any organized transaction. For that reason, the arrangement of a tail-node is i[tsp, tsq, ..., tsr], 1 ≤ p ≤q
≤ r ≤ m, where i is the node’s item name and tsi, i ∈ [1,m], is the timestamp of a transaction containing the items from root up to the node i. The conceptual structure of an RP-tree is shown in Figure3. Like an FP-tree, each node in an RP-tree maintains parent, children, and node traversal pointers. Please note that no node in an RP-tree maintains the support count as in an FP-tree. To assist a high degree of compactness, items in the prefix-tree are arranged in support-descendingorder.

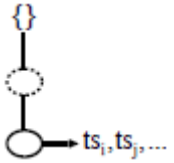

Figure 3: Theoretical structure of prefix-tree in RP tree. Dotted ellipse symbolizes ordinary node, while tail-node of sorted transactions with timestamps tsi, tsj ∈ R is signified by other ellipse.

One can take responsibility that the structure of the prefix-tree in an RP-tree may not be memory proficient since it unambiguously sustains timestamps of each transaction. On the contrary, it has been disputed that such a tree can attain memory efficiency by keeping transaction information only at the tail nodes and evading the support count field at each node.

Additionally, the problematical combinatorial explosion problem of candidate generation as in Apriori-like algorithms is eluded by an RP-tree. Keeping the information relating to ttransactional-identifiers in a tree can also be found in resourceful frequent pattern mining [25].

Algorithm 1 RP-List(TDB: Transactional database, I:Set of items, per: period, minPS: minimum periodic support,minRec: minimum recurrence)

1: Let idl be a temporary array that records the timestampof the last appearance of each item in the TDB. Let psbe another temporary array that records the periodicsupportof an item in a subset of a database. Let t ={tscur,X} denote the current transaction with tscur andX representing the timestamp of the current transactionand pattern, respectively.

2: for each transaction t ∈ TDB do

3: if an item i occurs for the first time then 4: Add i to the RP-list.

5: Set si = 1, eirec = 0, idil= tscur and psi = 1. 6: else

7: if (tscur − idil) ≤ per then

8: Set si ++, psi + + and idil= tscur. 9: else

10: eirec+ =%psiminPS&.

11: Set si ++, psi = 1 and idil= tscur. {Beginning of a new subset of a database.} 12: endif

13: endif

14: end for

15: To reflect the correct estimated recurrence value for each item in the RP-list, perform ei rec+ =[psi/minPS].

Algorithm 2 RP-Tree(TDB, RF-list)

1: Create the root of an RP-tree, T, and label it "null". 2: for each transaction t ∈ TDB do

3: Set the timestamp of the corresponding transaction as tcur.

4: Select and sort the candidate items in t according tothe order of CI. Let the sorted candidate item list in t be [p|P], where p is the first item and P is theremaining list.

5: Call insert tree([p|P], tcur, T).

6: end for

7: call RP-growth (Tree, null);

Construction of RP-tree

Since the anti-monotonic property is not contented by the periodic currents, candidate 1-patterns (or items) will play a substantial role in actual mining of these patterns. The set of candidate items CI in a database for the user-defined per, min PS, and min Rec can be revealed by populating -list with a scan on the database. Figure 4 displays the construction of an RP-list using Algorithm 1. Due to page limitation, the key steps are offered in the construction of the RP-list. Please note that the per, min PS, and min Rec values have been set to 2, 3 and 2, respectively. The scan on the first transaction, "1 : a, b, g", with tscur = 1 sets items 'a', 'b', and 'g' in the RP-list and sets theirs, erec, idl, and ps values to 1, 0, 1, and 1, respectively (lines1 to 5 in Algorithm 1). Figure 4(a) indicates the RP-list produced after scanning the first transaction. The scan on the second transaction "2 : a, c, d" with tscur = 2 initializes the items 'c' and 'd' in the RP-list by setting their s, e rec, idl,

Algorithm 3 insert tree([p|P], tcur, T)

1: while P is non-empty do

2: if T has a child N such that p.itemName N. item Namethen

3: Create a new node N. Let its parent link be linked to T. Let its node -link be linked to nodes with the same item Name via the node-link structure. Remove p from P.

4: end if

5: end while

6: Add tcur to the leaf node.

Algorithm 4 RP-growth(Tree, α)

1: for each ai in the header of Tree do

2: Generate pattern β = ai ∪ α. Collect all of the a′is tslists into a temporary array, TSβ, and calculate Eβ rec.

3: if Eβrec ≥ minRec then

4: Construct β's conditional pattern base then β'sconditional RP-tree Treeβ. Call getRecurrence(β, TSβ).

5:ifTreeβ Ø then

6: call RP-growth(Treeβ, β); 7: endif

8: endif

9: Remove ai from the Tree and push the ai's ts-list to its parent nodes.

10: end for

and ps values to 1, 0, 2, and 1, respectively. In addition, the s, erec, idl, and ps values of an already existing item 'a' are updated to 2, 0, 2, and 2, respectively (lines 7 to 9 in Algorithm 1). Figure 4(b) displays the RP-list made after scanning the second transaction. Figure 4(c) indicates the RP-list built after perusing the seventh transaction. It can be observed that the 'erec' of 'a' and 'b' have been restructured from 0 to 1. This is because their [ps/min PS]= 1. The ps value of 'a' and 'b' is set to 1 because they seemed periodically once again in the database (line 11 in Algorithm 1). Figure 4(d) express the-list assembled after scanning every transaction in the database. The estimated recurrence (erec) value foralltheitemsintheRP-listisonceagaincomputedtoreflectthecorrectness(line15inAlgorithm1).

Figure 4(e) demonstrates the updated erec value for all items in the RP-list. Using our pruning technique, 'g' is removed from the RP-list as its ecur < min Rec. The remaining items are organized in descending order of their support values (line 16 in Algorithm 1).Figure 4(f) illustrates the sorted list of candidate items in the RP-list. After finding candidate items, another scan on the database is conducted and it constructs the prefix-tree of the RPtree, as in Algorithms 2 and 3. These techniques are the same as those for constructing an FP-tree. On the other hand, the major difference is that no node in an RF-tree sustains the support count, as in an FP-tree. The first transaction{1 : a, b, g} is skimmed and a branch is constructed in the RPtree with only the candidate items 'b' and 'a.' The tail node 'b : 1' transmits the timestamp of the transaction. The RP-tree formed after scanning the first transaction isshown in Figure

5(a). A similar process is repeated for the remaining

Figure 4: Construction of RP-list:

(a) after scanning first transaction, (b) after scanning second transaction,(c) after scanning seventh transaction, (d) after scanning every transaction, (e) after computing actual 'erec' values, and (f) sorted list of candidate items transactions and the tree is updated accordingly. Figure5(b) displays the RP-tree made after scanning the complete database. For ease, the node traversal pointers in trees is not shown; however, they are preserved like an FP treedoes.

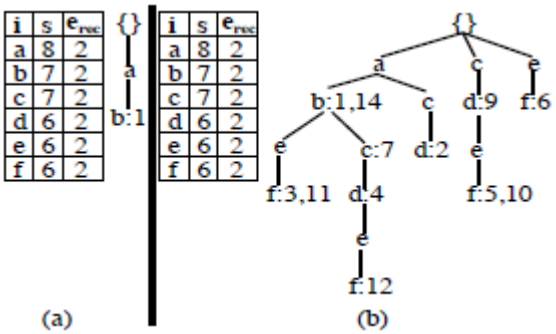

Figure 5: Construction of RP-tree: (a) after scanning first transaction and (b) after scanning entire transactional database

The whole information of all periodic patterns in a database is sustained by the RP-tree. The precision is based on Property 3 and shown in Lemmas 1 and 2. For each transaction t ∈ the TDB, CI(t) is

the set of all candidate items in t, i.e., CI(t) = item(t) ∩ CI, and is known as the candidate item projection of t.

Property 3. a complete set of candidate item projections is preserved by an RP-tree for each transaction in a database only once.

Lemma 1. Given a TDB and user-defined per, (minPS),and minRec values, the complete set of all periodic item projections of all transactions in the TDB can be derived from the RP-tree.

Proof. Based on Property 3, each transaction t ∈ TDB is mapped to only one path in the tree, and any path from the root up to a tail node conserves the complete projection for exactly n transactions (where n is the total number of entries in the ts-list of the tail node).

Lemma 2. The size of the RP-tree (without the root node) on a TDB for user-defined per, minPS, and min Rec is bounded by $t∈ TDB|CI(t)|.

Proof. According to the RP-tree construction process and Lemma 1, each transaction t donates at most one path of size |CI(t)| to an RP-tree. As a result, the total size support of all transactions can be$ t∈ TDB|CI(t)| at best. On the other hand, since there are usually many common prefix patterns amongthetransactions,thesizeofanRP-treeisnormallymuchsmallerthan∑t∈ TDB|CI(t)|.

Algorithm 5 get Recurrence(X: pattern, TSX: ts-list of pattern X)

1: Let idl be a variable that records the timestamp of the last transaction containing X. Let sub DB be a list of pairs of the form (start TS, end TS), where start TS and end TS respectively represent the starting and ending timestamps of periodic appearances of a pattern in a subset of data. It is used to record the periodic-intervals of a pattern. Let current PS be a variable to measure the periodic-support of X in a periodic-interval.

2: for each timestamp tscur ∈ TSX do 3: if (tscur is X's first occurrence) then 4: currentPS = 1, startTS = tscur;

5: else

6: if (tscur − idl ≤ per) then

7: currentPS ++;

8: else

9: if (currentPS ≥ minPS) then 10: subDB.insert(startTS, idl);

11: endif

12: currentPS = 1, startTS =tscur; 13: endif

14: endif

15: idl = tscur; 16: end for

17: // To reflect correct recurrence of X. 18: if (currentPS ≥ minPS) then

19: subDB.insert(startTS, idl);

20: end if

21: return ((subDB.size() ≥ minRec)?true:false);

Mining Periodic Patterns

Despite the fact that, an RP-tree and FP-tree gather items in support descending order, FP-growth mining on an RP-tree is not directly applied. The reasons are as follows: (i) the support count at each node an RP tree does not sustain and the ts-lists is controlled at the tail nodes and (ii) periodic patterns do not gratify the anti-monotonic property. Another pattern growth-based bottom-up mining technique to mine the patterns is formulated. The basic operations in mining an RP-tree includes: (i) counting length-1 candidate items, (ii) constructing the prefix tree from each candidate pattern, and (iii) constructing the conditional tree from each prefix tree. The length-1 candidate items. I is given by the RP-list. Before the prefix-tree construction process, the following important property and lemma of an RP-tree are reconnoitered.

Property 4. The occurrence information for all the nodes in the path (from the tail node to the root) at least

in the transactions in its ts-list.

Lemma 3. Let Z = {a1, a2, · · · , an} be a path in an RP tree where node an is the tail node carrying the ts- list of the path. If the ts-list is pushed-up to node an−1, then an−1 remains the occurrence information of the path Z′ = {a1, a2, · · · , an−1} for the same set of transactions in the ts-list without any loss. Proof. Based on Property 4, an maintains the occurrence information of path Z′ at least in the transactions in its ts-list. Therefore, the same ts-list at node an−1 conserves the same transaction information for Z′ without any loss.

The procedure to discern periodic patterns from RP-tree is displayed in Algorithm 4. The functioning of this algorithm is as follows.

The prefix tree for each candidate item in the RP-list, starting from the bottom most item, say I is to be made by proceeding. To build the prefix-tree for i, the prefix sub-paths of nodes i are gathered in a tree-structure, PTi. As i is the bottom-most item in the RP-list, each node labeled i in the RP-tree must be a tail node. While building PTi, based on Property 4, the ts-listof every node of i to all items are deliberated in the respective path explicitly in the temporary array (one for each item). This temporary array assists the calculation of support and erec of each item in PTi (line 2 in Algorithm 4). If an item j in PTi has support ≥ minSup and erec ≥ minRec, then wemake

its conditional tree and mine it recursively to learn the periodic patterns (lines 3 to 8 in Algorithm 4). Additionally, to allow the prefix-tree's construction for the next item in the RP-list, based on Lemma 3, the ts-lists are pushed up to the respective parent nodes in the original RP-tree and in PTi as well. All nodes of i in the original RP-tree and i's entry in the RP-list are erased thereafter (line 9 in Algorithm 4). Using Properties 1 and 2, the conditional tree CTi for PTi is constructed by removing all those items from PTi that have erec < minRec. If the deleted node is a tail node, its ts-list is pushed-up to its parent node.

The contents of the temporary array for the bottom item j in the RP-list of CTi represent TSij (i.e., the set of all timestamps where items I and j have appeared together in the database). As a result, using Algorithm 5, the recurrence of "ij" is intended and it is determined whether "ij" is a periodic pattern. The same process of creating a prefix-tree and its corresponding conditional tree is repeated for further extensions of "ij". The whole method of mining for each item is repeateduntilRP-list

∅. Consideritem

'f', which is the last item in the RP-list in Figure 4(e). The prefix-tree for 'f', PTf , is built

from the RP-tree, as shown in Figure 6(a). There are five items, 'a, b, c, d', and 'e' in PTf . Only item 'e' satisfies the condition Erec(e) ≥ minRec. Hence, the conditional tree CTf from PTf is constructed with only one item 'e', as displayed in 6(b). The ts-list of 'e' in CTf ' produces TSef . The "recurrence" of 'ef' is measured using Algorithm 5. Since Rec(ef) ≥ minRec, 'ef' will be made as a periodic pattern. A similar process is repeated for the other items in the RP-list. Next, 'f' is pruned from the original RP-tree

Table 4: User-defined *per*, *minPS* and *minRec* values in different databases

	per			*minPS*			*minRec*		
	1	2	3	1	2	3	1	2	3
T10I4D100k	360	720	1440	0.1%	0.2%	0.3%	1	2	3
Shop-14	360 (=6 hr.)	720 (=12 hrs.)	1440 (=24 hrs.)	0.1%	0.2%	0.3%	1	2	3
Twitter	360 (=6 hr.)	720 (=12 hrs.)	1440 (=24 hrs.)	2%	5%	10%	1	2	3

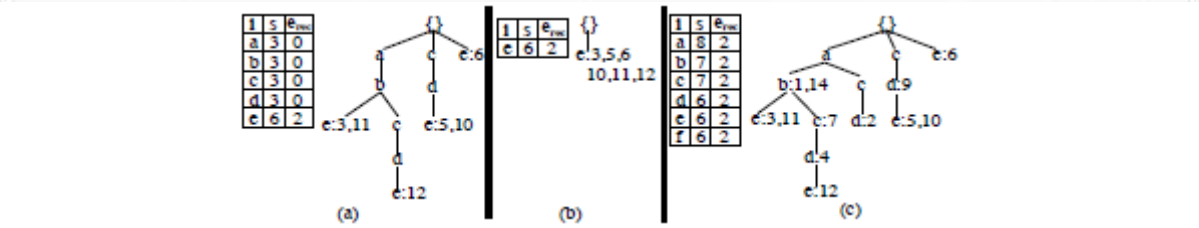

Figure 6: Finding RP-forms for suffix item 'f' in RP-tree: (a) prefix-tree for item 'f' (i.e., PTf),

(b) conditional tree for item 'f' (i.e., CTf), and (c) RP-tree after pruning item 'f' and its ts-lists are pushed to its parent nodes, as shown in 6(c). All the above procedures are once again repeateduntil

theRP-list ∅.The above bottom-up mining technique on a support descending RP -tree is efficient, because it reduces the search space radically as the mining processdevelops.

V. EXPERIMENTALRESULTS

The performance of RP growth is evaluated first in this section. Next, the usefulness of periodic patterns is discussed by matching them against p-patterns and periodic frequent patterns. It should be noted that our periodic patterns are the generalization of periodic-frequent patterns, as the latter patterns reveal complete (rather than partial) cyclic repetitions in the entire database. There are only two Apriori-like algorithms, periodic-first and association first, to discern p-patterns. The periodic-first algorithm isused

to notice p-patterns as it is comparatively faster than the association-first algorithm. The Periodic- Frequent pattern-growth++ algorithm (PF-growth++) is used to determine periodic-frequent patterns. The performance of RP-growth against the periodic-first and PF-growth++ algorithms is not compared. The reason is that the other algorithms ascertain regular patterns; therefore, different measures are used to calculate the periodic interestingness of a form.

Experimental setup

The algorithms, RP-growth, periodic-first and PF-growth++, were written in GNU C++ and run with Ubuntu 14.4 on a 2.66 GHz machine with 8 GB of memory. To the best of our knowledge, there are no publicly obtainable time-based sequences. And so, experiments are conducted using the following databases. • T10I4D100K database. This database is a synthetic transactional database produced using the procedure given by [23]. This database contains 100,000 transactions and 941 distinct items. Shop-14 database. Clickstream data of seven online stores was provided by a Czech company in the ECML/PKDD 2005 Discovery challenge. The click stream data of product categories visited by the users in "Shop 14" was well-thought-out

(www.shop4.cz), and formed a transactional database with each transaction indicating the set of web pages visited by the people at a particular minute interval.

The transactional database comprises 59,240 transactions (i.e., 41 days of page visits) and 138 distinct items (or productcategories). Twitter database. This database was formed by bearing in mind the top 1000 English hashtags appearing in 44 million tweets/re tweets from 1-May-2013 to 31-August- 2013 (i.e., 123 days). The measure, term frequency inverse document frequency, is used to rank the hash tags. A minute starting from 00:00 hours of 1-May-2013 to 24:00 hours of 31-August-2013 is demonstrated by the timestamp of each transaction. The resultant transactional database has 177,120 transactions with 1000 distinct items (or hashtags). More details on the collection of data process and the worth of this data in finding attention-grabbing events have beenpresented. To mine p-patterns and periodic patterns, all the above databases are converted into time-based sequences using the timestamps of each transaction. As this conversion process is a rather simple and straight-forward approach, we do not discuss it for concision.

Table 4 lists the different per, min PS and min Rec values used in above the databases. The periodic interval (i.e., per value) in both Shop-14 and Twitter databases varied from six hours to one day. Correspondingly, the min Rec in these databases varied from 1 to 3. In this paper, we do not present the results for min Rec values greater than 3 because very few periodic patterns were getting produced at min Rec > 3. The min PS values in T10I4D100K and Shop-14 databases varied from 0.1% to 0.3%. The reason for choosing low min PS values is to find out the forms comprising both common and uncommon items. In the Twitter database, we set min PS at 2%, 5% and 10%. The reason for choosing a comparatively high min PS values as compared with the other two databases is that very low min PS values are resulting in a combinatorial explosion producing too manypatterns.

Generation of periodic forms

Table 5 lists the numbers of periodic patterns discovered in T10I4D100K, Shop-14 and Twitter databases at different per, minPS and minRec values. The partial results of Table 5 are shown in Figure 7. The following interpretations can be drawn from this figure:

- At a stable per and minRec, the increase in minPS can decrease the number of periodic patterns. The purpose is that many forms were unsuccessful while appearing periodically for longer timeperiods.

Table 5: Number of periodic patterns generated at different per, minPS and minRec threshold values Dataset minPS Number of periodic patterns

Table 5: Number of recurring patterns generated at different *per*, *minPS* and *minRec* threshold values

Dataset	*minPS*	Number of recurring patterns								
		minRec = 1			*minRec* = 2			*minRec* = 3		
		per=360	per=720	per=1440	per=360	per=720	per=1440	per=360	per=720	per=1440
T10I4-D100k	0.1%	428	1254	7193	255	436	1036	194	160	27
	0.2%	339	757	3205	168	103	39	72	0	0
	0.3%	296	622	2148	109	32	2	21	0	0
Shop-14	0.1%	593	1885	4977	447	1339	3198	338	266	9
	0.2%	342	1077	1906	257	750	1470	118	14	0
	0.3%	251	744	933	195	534	760	48	3	0
Twitter	2%	14736	36354	42319	8718	17982	19746	4551	7749	8103
	5%	1655	11268	26341	595	6847	7010	337	3713	5123
	10%	511	714	1190	11	34	912	6	17	98

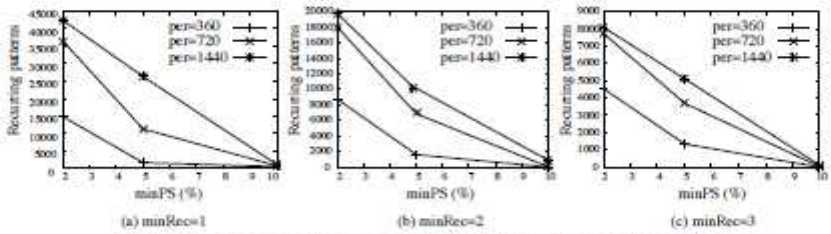

Figure 7: Recurring patterns discovered in Twitter data

Figure 7: Periodic patterns discovered in Twitter data

- At a fixed min PS and per, the increase in min Rec can decrease the number of periodic patterns. This is for the reason that many patterns failed to fulfill the increased min Recvalues.
- At a fixed min PS, the upsurge in per can have different impact on the generation of periodic patterns for the values min Rec = 1 and min Rec > 1. At min Rec = 1, increase in per can increase thenumber

of periodic patterns. Because of the very reason that the inter arrival times of the patterns that were considered as aperiodic at low per values were considered as periodic with the increase in the per value. For minRec > 1, increase in per can either increase or decrease the number of periodic patterns. The reason for decrease is due to the unification of exciting periodic-intervals discovered at low pervalues.

Table 6 lists some of the periodic forms discovered from the Twitter database at per = 360, minPS = 2% and min Rec = 1. Figures 8 (a) and (b) show the frequencies of the terms present in patterns {yyc, uttarakhand} and {nuclear, hibaku} on a daily basis. It can be observed that “uttarakhand” is a relatively rare term as compared with other terms, and our model has meritoriously revealed the knowledge pertaining to this term. Another interesting observation from Table 5 is that even at low min PS values, our ideal has generated only a limited number of periodic patterns in each database. Thisobviously displays that the knowledge pertaining to rare terms can be discovered by our model without producing too many uninteresting patterns. In other words, our model is tolerant of the “rare item problem”.

Performance of RP-growth

Table 7 lists the runtime required using RP-growth to discover periodic patterns in T10I4D100K, Shop-

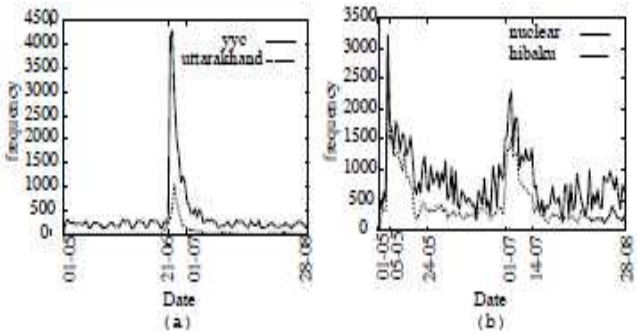

Figure 8: Frequency of hashtags at different days in database. Date is of form 'dd-mm'. Year of this date is 2013

Twitter databases. The time taken to transform the time-based sequence into a transactional database and mining of periodic patterns is involved in the runtime.

Figure 9 illustrations the runtime required by RP-growth while mining the periodic patterns in Twitter database. The deviations in the per, minPS and minRec inception values displays an akin effect on runtime ingesting as in the generation of periodic patterns. The complete set of periodic patterns at a equitable runtime even at low minPS thresholds was discovered by the proposedalgorithm.

Comparison of p-patterns, periodic patterns and periodic-frequentpatterns

P-patterns, periodic patterns and periodic frequent patterns at different per and minPS values are compared. For brevity, the results discovered when period is set to 1 day, i.e., per = 1440 is presented. The minSup and minPS values are set to 0.1% and 2% respectively for Shop-14 and Twitterdatabases.

The p-pattern mining requires another parameter known as window length (w). We set w = 1 for our experiment.

Table 6: Some of the interesting periodic patterns discovered in Twitter database

Table 6: Some of the interesting recurring patterns discovered in Twitter database

S.No	Pattern	Periodic duration	Cause for the events
1	{yyc, uttarakhand}	[2013-06-21 01:08, 2013-07-01 04:27]	On June 20, Uttarakhand, a state in India and Alberta, a province in Canada have witnessed heavy floods.
2	{nuclear, hibaku} (In Japanese, hibaku means radiation.)	[2013-05-06 22:33, 2013-05-24 22:13], [2013-07-01 06:17, 2013-07-14 06:21]	(*i*) A Japanese minister has visited Chernobyl, Ukraine to learn from the recovery from the severe nuclear accident. (*ii*) People were tweeting about detection of Plutonium at a point 12 KM from Fukoshima nuclear reactor.
3	{pakvotes, nayapakistan}	[2013-05-09 16:15, 2013-05-15 14:11]	The general elections were held in Pakisthan. on May 11, 2013.
4	{oklahoma, tornado, prayforoklahoma}	[2013-05-21 11:52, 2013-05-24 21:38]	Oklahoma was struck with a tornado on May 20, 2013.

Table 7: Runtime of RP-growth at different *per*, *minPS* and *minRec* threshold values

Dataset	*minPS*	Runtime of RP-growth								
		minRec = 1			*minRec* = 2			*minRec* = 3		
		per=360	*per*=720	*per*=1440	*per*=360	*per*=720	*per*=1440	*per*=360	*per*=720	*per*=1440
T10I4-D100k	0.1%	14.8	150.9	366.5	3.8	10.7	40.1	3.5	3.9	6.3
	0.2%	7.7	45.9	99.6	3.6	5.4	9.6	2.7	3.1	3.1
	0.3%	3.7	11.6	21.3	3.2	3.4	4.2	2.5	2.4	2.6
Shop-14	0.1%	47.7	55.6	67.3	43.5	47.7	52.3	42.4	45.1	48.2
	0.2%	42.9	46.1	51.3	41.7	43.4	45.0	41.4	42.1	43.8
	0.3%	42.4	44.0	47.3	41.6	42.1	43.6	41.1	41.5	41.7
Twitter	2%	55.1	190.0	290.5	42.9	154.9	248.4	41.3	139.2	226.1
	5%	37.9	134.3	225.6	33.0	105.3	181.9	31.5	96.1	159.7
	10%	32.3	108.3	190.9	30.4	89.2	151.3	29.9	66.9	124.1

The numbers of periodic-frequent patterns, periodic patterns and p-patterns is listed in Table 8 and it is discovered in the Shop- 14 and Twitter databases. The column labeled 'II' in this table refers to the length of the longest pattern discovered in each of these databases. The following observations can be drawn from this table.

First, the total numbers of periodic-frequent patterns revealed in both databases were relatively less than the number of periodic patterns and p-patterns. The reason is that the severe restraint that a frequent pattern has to display complete cyclic repetitions throughout the data has failed to identify many interesting partial periodically appearingpatterns.

In addition, this strict constraint also resulted in finding the very short periodic-frequent patterns (see columns labeled 'II' in Table 8). This happens because longer patterns generally fail to exhibit complete cyclic repetitions throughout the data. Setting a high period verge value can enable a user to determine long periodic-frequent patterns. On the contrary, this high period can result in ascertaining occasionally look as if patterns as periodic-frequent patterns. Accordingly, it is essential to ease this strict restriction without changing the period threshold value. As our model enables a user to relax this strict constraint, periodic patterns have been found more interesting than the periodic-frequent patterns for a given per threshold.

Table 8: Number of patterns discovered in Shop-14 and Twitter databases. Terms '*I*' and '*II*' represent *total number of patterns* and *maximum length of pattern* found in each database, respectively.

	Shop-14		Twitter	
	I	*II*	*I*	*II*
PF patterns	22	3	466	2
Recurring patterns	4,977	9	42,319	7
p-patterns	156,7001	12	442,076	16

Second, the total number of p-patterns discovered in both databases was much higher than the periodic patterns and periodic-frequent forms.

It is because of the usage of a low minSup has flattened Ma and Hellerstein's model to discover not only all our periodic patterns as p-patterns, but also resulted in a combinatorial explosion of frequently appearing items fabricating too many p-patterns. Most importantly, many of the p-patterns discovered at low minSup were uninteresting to the users. The reason is that frequent
items were merging with one another in all possible ways and generating p-patterns by nourishing a low minSup value. On the contrary, our model has reduced the combinatorial explosion of frequent items by assessing their interestingness with respect to their number of consecutive periodic appearances
in a portion of data.

VI. CONCLUSIONS ANDFUTUREWORK

A new-fangled class of fractional periodic patterns called periodic patterns was introduced and it also discussed the effectiveness of these patterns in various real-world applications. A model for noticing such patterns was also proposed. The patterns discovered with the anticipated model do not please the anti- monotonic property. Consequently, a novel clipping procedure was proposed to lessen the computational cost of finding these patterns. A pattern-growth algorithm was also projected to discover the periodic patterns commendably. Investigational results suggest that the model is tolerant to the "rare item
problem" and that the algorithm is well-organized. The efficacy of the periodic patterns was discussed by comparing them against the periodic-frequent and p-patterns. In our current study, we did not considered noisy data and the phase-shifts of the items within the data.

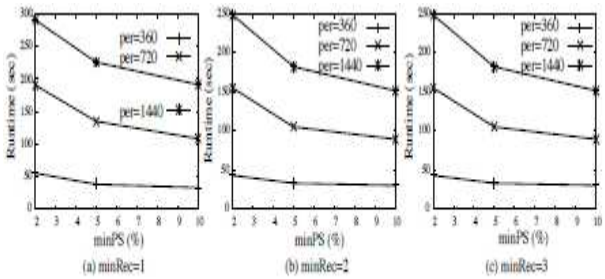

Figure 9: Runtime of RP-growth

For future work, methods for handling these two Scenarios will be handled. Another interesting future work will be out spreading our model to increase the performance of a relationship rule-based recommender system.

REFERENCES

1. R. Uday Kiran, J.N. Venkatesh, Masashi Toyoda, Masaru Kitsuregawa, P. Krishna Reddy. "Discovering artial periodic-frequent patterns in a transactional database" , Journal of Systems and Software, 2017
2. "Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXVIII" , Springer Science and Business Media LLC, 2018
3. R. Uday Kiran, Haichuan Shang, Masashi Toyoda, Masaru Kitsuregawa. "Discovering Partial Periodic Itemsets in Temporal Databases" , Proceedings of the 29th International Conference on Scientific and
4. Statistical Database Management - SSDBM '17, 2017
5. "Advances in Knowledge Discovery and Data Mining" , Springer Science and Business Media LLC, 2017
6. Syed Khairuzzaman Tanbeer. "Discovering Periodic-Frequent Patterns in Transactional Databases" Lecture Notes in Computer Science, 2009
7. Shimpli Dhale, Sagar Badhiye. "Review on towards efficient mining of recurrent patterns in time series data" , 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), 2017

AUTHORS PROFILE

Mrs. D.Elavarasi, Assistant Professor, Department of Computer Science and Engineering, Mount Zion College of Engineering and Technology. I am having 13+ years of teaching experience. My area of interest is Data Mining. My keen interest in research focus on social media mining.