An Extensive Study of Frequent Mining Algorithms for Colossal Patterns

T. Sreenivasula Reddy and R. Sathya

Abstract During the last decade of research, a lot of focus has been placed on the subject of frequent pattern mining (FPM). A profitable data set with a large sum of transactions and only a few items in each transaction has been used to develop numerous FPM algorithms. Because of the rise of bioinformatics, a new sort of data set called a high-dimensional data set has emerged, with fewer transactions but a greater sum of elements in each. The execution time of classical algorithms grows with deal length. High-dimensional data sets can't be processed by existing algorithms. But when applied to large data sets with a lot of dimensions, mining algorithms generate a huge amount of data, much of which is useless to scientists because of the little and medium-sized patterns they include. As a way to lessen the number of output patterns for mining patterns, colossal pattern mining is discussed. Since small and mid-sized patterns aren't mined, mining algorithms for enormous patterns run faster. In this work, an extensive study of colossal patterns, existing mining algorithms with its drawback is mentioned. The definitions of FPM, high utility mining and relation of colossal patterns with others are also explained. Pattern-Fusion is the first algorithm, which is developed for colossal patterns that is described briefly in this work.

Keywords Frequent pattern mining · Commercial data set · High-dimensional data set · Frequent colossal patterns · Data mining · Machine learning

T. S. Reddy (✉)
Research Scholar, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University, Annamalai Nagar, Chidambaram, Tamil Nadu 608002, India
e-mail: seenu4linux@gmail.com

R. Sathya
Assistant Professor, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University, Annamalai Nagar, Chidambaram, TamilNadu 608002, India

R. Malhotra et al. (eds.), *High Performance Computing, Smart Devices and Networks*, Lecture Notes in Electrical Engineering 1087,
https://doi.org/10.1007/978-981-99-6690-5_16

1 Introduction

There are numerous applications for data mining, including the extraction of health computational biology, [1] detection of malicious online attacks, [2] web mining, [3] sentiment analysis and opinion mining in big data, [4] recommendation systems, [5] data warehousing, and [6] the use of data in decision-making. It is one of the most important practices for extracting from a data set that have occurred more than a user least threshold sum of transactions in the data set, and rule construction is the process of creating association rules using the patterns that have been extracted from the data set. A significant portion of research in association rule mining has been devoted to the FPM phase because of the enormous volume of computations, the lengthy execution time, and the enormous sum of memory it requires.

Various FPM algorithms in the literature have retrieved a variety of patterns. Patterns such as itemsets [7], sequences [8], and graphs [9] have been discovered from a variety of data sets. Novel approaches with humbler data constructions and more effective pruning algorithms have gradually superseded the initial motives of employing growth-based FP-based itemset mining methods due to their increasing complexity and vast number of created projected trees.

For efficient frequent itemset mining, several distributed tactics have been used [10]. Frequent itemset mining is an exponential problem, which means that mining methods produce a large number of extracted frequent itemsets as the outcome. The mining process may become less efficient as a result of the increased production. Research in the literature found that the best solutions to this problem were algorithms for closed pattern mining [11], maximum design mining [12], and colossal pattern mining [13]. The difficulty in mining frequent patterns is compounded by the fact that each pattern has an infinite number of sub-patterns, resulting in an enormous number of frequent patterns. Closed frequent pattern mining [14] and maximal frequent pattern mining (max-pattern) [15] have both been presented as solutions to this issue.

There are a limited number of common patterns in the whole pattern set, hence the term “closed pattern set” applies. A pattern is dubbed closed frequent if it occurs frequently in a data set but there is no super-pattern with the same support as it. There are fewer pattern super patterns than there are regular patterns in a database. In contrast to the more compact set of maximal frequent patterns, closed pattern mining tends to condense the set of frequently occurring patterns. As a result, maximally frequent patterns may not always include the full supporting material for their equivalent often occurring patterns.

The pattern mining challenge is space intensive for large data sets, even though closed pattern mining dramatically abridged the sum of processing and output capacity. Small and medium-sized patterns are frequently generated by mining algorithms, yet this data is often useless in many applications. Finding methods for mining that only extract huge patterns and ignore tiny and medium-sized patterns makes sense because many applications benefit from only large patterns. This difficulty can be approached in a new way by mining large patterns instead of little ones.

A bottom-up strategy to finding patterns is used by all previous techniques of pattern mining. These techniques begin with little designs and work their way up to larger ones. Small and medium-sized patterns, however, often lack useful information and can only be obtained from large-sized patterns, known as colossal patterns, in specific applications. It was in 2007 when the core Pattern-Fusion (core-fusion) technique, the first real approach for mining enormous patterns, was published [16].

1.1 Motivation

In the case of huge and very large data sets, the solutions to pattern mining problems [17] take a long time and are completely inefficient when trying to solve more complex problems. Many optimization and high-performance figuring practices have been industrialised to increase the performance of the pattern mining systems [18–21]. However, when working with large databases, these solutions are ineffective since only a small number of useful patterns are showed to the end user.

1.2 Problem Scope

In general, the sub-patterns that make up a colossal pattern are expected to appear at about the same frequency as the main pattern, therefore they can be identified by counting the number of supporters for each sub-pattern. There would be ncr number of common sub-patterns of size r for a colossal pattern of size n. As a result, in order to get to the massive patterns, we must first study an immense number of smaller patterns. In order to swiftly find large patterns, a strategy has been proposed to traverse the search area in jumps, ignoring most of the mid-sized patterns.

1.3 Structure of the Paper

It gives an introduction to FPM and talks about the problem of a huge pattern in Sect. 1. Pattern mining and colossal pattern relationships are explained in Sect. 2; this is the beginning of the text. Section 3 gives an impression of the algorithms for colossal patterns. Section 4 explains how to do colossal pattern mining from a list of frequently used items. It comes to an end in Sect. 5.

2 Pattern Mining Problems

A general definition of pattern mining is initially presented in this part, followed by an explanation of the connections between large-scale patterns and smaller ones.

Definition 1 (pattern) Reflect $I = 1, 2, \ldots, n$ as a set of items, and $T = t1, t2, \ldots, m$ as a set of transactions with the sum of transactions. We create the function *a*, which reads $p =$ for the item I in transaction t j. (i, j).

Definition 2 (pattern mining) A pattern mining task involves discovering all of the patterns L that are relevant to a given problem

$$L = \{p|\text{Interestingness}(T, I, p) \geq \gamma\} \tag{1}$$

To analyse a pattern p among a set and a set of items I, the measure of Interestingness(T, I, p) is used. Existing pattern mining issues can be summarised using these two definitions.

Definition 3 (frequent itemset mining (FIM)) As an postponement of the pattern mining issue, we create a FIM problem (Def. 2)

$$L = \{p|\text{Support}(T, I, p) \geq \gamma\} \tag{2}$$

As described in Def. 1, the collection of transactions in a Boolean database, T, I, and p are all defined as a Boolean database, and Support (T, I, p) is the sum of transactions in T covering the pattern _(T.I)/(|T|).

Definition 4 (weighted folder) A weighted database is defined by defining the function

$$\sigma(i, j) = \begin{cases} w_{\text{ij}} \; if \; i \in t_j \\ 0 \quad \text{otherwise} \end{cases} \tag{3}$$

W_{ij} refers to the weight of Item I, which is part of the transaction Tj.

Definition 5 (weighted itemset mining (WIM)) WIM problems are an expansion of the pattern mining problems (see Def. 2) by extending the

$$L = \{p|WS(T, I, p) \geq \gamma\} \tag{4}$$

It is possible to have WS (T, I) = WS (T, I) p ($j = 1$)(|T|)W (tj, Ip). W ($t\,j$, I, p) is the least weight of the elements of the pattern p in the transaction $t\,j$, and is a least weighted threshold for the weighted database established in Def. 3.

Definition 6 (uncertain database) Setting the function (see Def. 2) as is how an uncertain database is defined.

$$\sigma(i, j) = \begin{cases} \text{Prob}_{i,j} \text{ if } & i \in t_j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Note that the transaction $t\,j$'s uncertainty value for I is Prob (i, j).

Definition 7 (uncertain itemset mining (UIM)) A UIM problem is defined as an postponement of the pattern mining problem (see Def. 2) by

$$L = \{p | US(T, I, p) \geq \gamma\} \tag{6}$$

With $\text{US}(T, I, p) = \sum_{j=1}^{|T|} \prod_{i \in p} \text{Prob}_{\text{ij}}$. In the indeterminate database specified by Def. 5.

Definition 8 (utility database) Utility databases are defined by setting the function (Def. 2)

$$\sigma(i, j) = \begin{cases} \text{iu}_{\text{ij}} \text{ if } i \in t_j \\ 0 \quad \text{otherwise} \end{cases} \tag{7}$$

It is important to keep in mind that eu denotes the item's external utility; this is the value of i's internal utility in the transaction (i).

Definition 9 (high utility itemset mining (HUIM)) The pattern mining problem (Def. 2) is an extension of this problem, which is called a "HUIM problem."

$$\begin{aligned} L &= \{p | U(T, I, p) \geq \gamma\}, \text{with } U(T, I, p) \\ &= \sum_{j=1}^{|T|} \sum_{i \in p} \text{iu}_{\text{ij}} \times \text{eu}(i) \end{aligned} \tag{8}$$

According to Definition 7, T is the set of transactions that can be found in the database defined by Def. 7.

Definition 10 (sequence database) Shoulder that there is a total order on things, such as $1 > 2 > 3 > n\ldots$ Sequences are an ordered list of $s = [I1, I2, \ldots, I\_(|s|)]$, where s is an itemet. The function (see Definition 2) is defined as $(i, j) = i$, if $I\ t\ j$ for each itemet Ii.

Definition 11 (sequential pattern mining (SPM)) Mining is extended to include the SPM problem (see definition 2) by:

$$L = \{p | \text{support}(T, I, p) \geq \gamma\} \tag{9}$$

If the sequence database's total transaction count (T) is less than the minimum support criterion (), then the answer is.

2.1 Relationship of Colossal Patterns to Other Patterns

Figure 1 shows how a colossal pattern develops over time. Minimum support should be 50. A and B are combined, and the frequency of AB lowers to 198 when AB is extended. As a result, pattern A appears 198 times, with pattern B appearing only once. Closed, but not massive, is the pattern AB. It is impossible to stop the pattern's growth at C, since the addition of D provides the same level of support as the original design. Closed but not enormous ABCD pattern. The smaller pattern will not be called a monstrous pattern until there is a considerable difference in frequency when it is extended. When G is added to ABCDF, the frequency drops from 193 to 190, and we no longer regard ABCDF to be a huge pattern, but rather ABCDFG. There is a considerable decrease in the frequency of ABCDFG when it is extended with Q, therefore this extension is not regarded as massive. (Seen in the figure marked with a cross.2) Because of this, ABCDFG and ABCDFGQ are classified as enormous patterns, respectively. As a result, a maximum pattern does not always have to be huge. Large-scale patterns are depicted in Fig. 1 ABCDEH and ABCDFG. The following observations can be made, based on the case above:

- There is no requirement that every closed frequent itemset is also a huge pattern, however this is not always the case.
- A gargantuan pattern does not always have to be an itemset with the highest frequency possible.

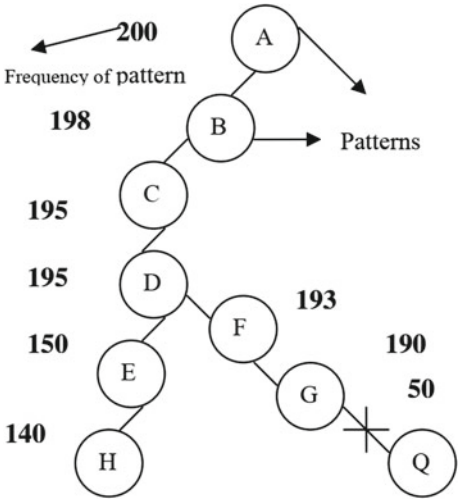

Fig. 1 Pattern tree growth process

3 Brief Explanation of Existing Techniques

3.1 Colossal Pattern Miner (CPM) Overview

Step-Wise Method

- There is an initial pool of 1 or 2 itemsets that we begin with (can be obtained using any frequent mining algorithm).
- We use any standard clustering algorithm to divide the patterns in the initial pool into groups based on their frequency. Frequency bands is what we call them.
- Choose a random seed pattern from the greatest frequency band and create a neighbourhood of -core seed patterns (depending on distance), startin–g with the present band and moving down through the other bands in declining order of their size.
- To remove a -core pattern () from its parent frequency band if its frequency is less than 1-, use the following procedure.
- Steps 3 and 4 should be repeated until all patterns have been picked or the desired number of neighbourhoods has been reached.
- To create one or more super patterns, we merge the patterns in each neighbourhood (colossal patterns).
- The new pool of super patterns is now the new initial pool, and the process repeats itself until we reach colossal patterns.

3.2 Bit-Wise Vertical Bottom-Up Colossal (BVBUC)

The enormous mining algorithm shown in Algorithm 1 is bit-wise vertical bottom-up. To begin with, it takes as input m, which is an index of the first row in the parent rowset, as well as the maximum number of rows in the bitmatrix, l, the level of processing a tree is now at, as well as S, the input rowset for the method.

This is a recursive algorithm with two major components. The algorithm initially determines if it has reached level minsup in the main if block. The method stops expanding the current branch if the level of three is the minimum level. This node's pattern and its support are written to the output file if the new gargantuan pattern is calculated and does not already exist in it. It is a basic function called pattern (S) that calculates the appropriate pattern of the rowset S by performing a "AND" operation on the bit vectors of the row-ids t–here are in S and selecting the items that have a

value of 1 in the result "AND" vector. The support of pattern (S) provided by function support is the sum of row ids in rowset S (S).

To begin, m is added to the S rowset, and then the else block builds the node's matching rowset. The algorithm will stop increasing this third branch if the pattern does not reach a user-specified threshold if it is not massive. Mining can continue only if the pattern (S) is large. This child node's rowsets will be generated by extending this node and creating its children for each row-id in the range of $m + 1$ to max if the branch of consistent child reaches the minsup level. If $p + m{-}a$ p is less than or equal to min sup, an algorithmic loop is in place. As a result of the closeness requirement method, we can now define a novel version of BVBUC that searches for closed, gigantic frequent patterns in data sets. BVBUC's closed version will be depicted in the algorithm's first component (the main if-block):

```
If(l = min sup)then
Begin
 If (Pattern(S)is colossal)then
  If(pattern(S)is not in file)then
   If(pattern(S)is closed)then
    Output(File, (Pattern(S), support(S)));
End
```

Only one condition has changed between the closed and primary versions of BVBUC, and that is the condition that tests for pattern proximity before adding it to output file. Since the algorithm has been altered, it can now mine closed, massively recurring patterns.

Algorithm 1. BVBUC Algorithm

```
Procedure BVBUC
Input
M: integer;// current, row id
Max: integer;// number of rows
1: integer;// current level (also number of row ids in this node)
S; string;// input rowset
Minsup; integer;// minimum support
Var
    i; integer;
Begin
   If(1 = minsup) then
Begin
  If (Pattern(S) is colossal) then
    If (pattern(s) is not in file) then
     Output (File, (Pattern(s), Support(S)));
End
Else
Begin
   S: = S + InToStr(m) + ";
   Begin
       For i:=m+1 to max do
         Begin
         BVBUC(I,max, 1+1,S, min sup);
         End
        End
      End
     End
End
```

3.3 LCCP: A Length Constraints with Algorithm for Mining Colossal Patterns

Here, two theorems are presented to establish the theoretical foundation of the suggested approach, which is cost effective in mining enormous patterns with length limits. Candidate patterns that do not meet the min-length restriction can be swiftly discarded using Theorem 1. For candidates with a max-length restriction, Theorem 2 has been found to be a time-saving tool.

Theorem 1 Nodes in the CP-tree that do not meet the min-length restriction are not allowed to have any children that do meet the constraint.

Theorem 1 states that if a node does not meet the least length requirement, it does not need to be expanded. As a result, mining operations can reduce the size of the exploration zone.

Theorem 2 Each child node in the CP-tree that meets the max-length criteria is considered to be a member of the parent node.

Rapid extraction of enormous patterns with length limits is made possible by the effective method LCCP (min-length and max-length). In order to produce and prune enormous pattern candidates, LCCP relies on PCP-Miner [22]. Using Theorems 1 and 2, it is possible to reject candidates who do not meet the min-length constraint and those who do not require the max-length condition to be tested.

4 From Frequent Itemset Mining to Colossal Itemset Mining

Itemsets that don't fulfil the minimum support are pruned from the search space. These prior row-enumeration algorithms, even using the support measure, failed to uncover many interesting closed patterns. Since the support-based strategy filters out patterns with low support but high confidence, this is the case. Support-based algorithms are discussed as alternatives in this part, which also examines approximately of the existing approaches.

4.1 Alternative User-Defined Threshold

MAXCONF [23] presents an algorithm for discovering interesting gene interactions from microarray data sets. In order to successfully narrow the search field, minsup is a measure of confidence rather than a requirement. For perturbation microarray data, which includes both common and rare (infrequent) correlations, this strategy was inspired by the view that support-based trimming is inappropriate. Multiple microarray data sets were used to evaluate MAXCONF and RERII [24], two improved versions of CARPENTER. For the purposes of categorising genes according to their molecular function, biological activity, and cell component, the extracted rules were subjected to the Gene Ontology's international standard for gene annotation (GO). In order for a rule to be considered biologically relevant, it needs be annotated with GO annotations. Support pruning is not ideal for mining gene expression data sets, according to this study, which found that MAXCONF rules are more biologically relevant than those identified by RERII. Though research has shown support-based algorithms remove many intriguing rules, the authors did not recognise the importance of pattern size. In association mining tasks, longer sequences are generally more essential than shorter sequences [25, 26].

4.2 Current Colossal Pattern Mining Process

First developed in an algorithm called Pattern-Fusion, the notion of colossal pattern was used to find an efficient approximation to gigantic patterns. In order to find the enormous pattern, Pattern-Fusion fused all of the smaller patterns into one, saving time and effort compared to manually traversing the pattern tree level by level. The Pattern-Fusion algorithm has been shown to be able to approximate large patterns in real data sets in several investigations.

Randomly picked sub-patterns are fused to form the enormous pattern, then the support is counted utilising individual database scans in Pattern-Fusion. However, Colossal Pattern Miner (CPM) [27] proposed a more intelligent technique to combine and separate the sub-patterns, which avoided the vast number of mid-sized patterns. To further reduce the number of database scans, vertical data format is used. Unfortunately, this technique has not been compared to Pattern-Fusion in terms of performance.

The data set is represented and compressed using a bit matrix by the BVBUC algorithm [28]. A bottom-up row list search tree is developed up to the minsup level since the greatest pattern of each branch is formed there. As a result, the time and memory requirements of most pattern mining algorithms are increased while searching for patterns with low minsup values. This is not the case in BVBUC, where when minsup drops, it also reduces the number of levels in the tree, removing branches that don't meet the minimum. However, as we'll see in a moment, this approach resulted in fewer purportedly mined itemsets. Another pruning strategy is to cease extending a node when the pattern has less elements than the minimum allowable quantity in a huge pattern. BVBUC beats both CPM and Pattern-Fusion in real data sets and microarray data sets, according to the authors.

DisClose [29] is an algorithm that first enumerates high cardinality itemsets and then constructs smaller itemsets to extract enormous closed itemsets. The row-enumeration tree is searched from the bottom-up like in BVBUC. Starting with a table that is transposed, the algorithm creates a compact row tree to hold the transposed table's itemsets. The CR-Tree is used during the search phase. Many rowset values are shared by a single node, making the suggested data structure compact and able to express itemsets with a minimum cardinality (mincard). Other methods employing a tree-based data structure, on the other hand, require a large amount of memory because of the length of the itemsets as well as the number of transactions that take place.

Colossal pattern mining using the -core ratio has been proposed using the newly presented algorithms CP-Miner and its upgraded counterpart, PCP-Miner [30], the pattern is a -core pattern. If there are no supersets in the database, then a -core pattern is considered gigantic. Instead of requiring the usage of a -core ratio, BVBUC proposes an acceptable minimum threshold and an acceptable minimum sum of itemsets in an itemset. However, in terms of runtime, they've outperformed BVBUC.

Using biological data sets, DPMine [13] proposes a new method for identifying enormous Colossal Pattern Sequences (CPS). In the DPT + tree, a vector

intersection operator is used to build enormous pattern sequences by identifying Doubleton Patterns. DPMine uses a new integrated data structure called "D-struct," which combines a doubleton data matrix with a one-dimensional array pair set to dynamically discover Doubleton Patterns from Biological data sets. DPT + trees are constructed using a bit-wise Top down Column enumeration tree. Constrained and predictable, D-main struct's memory can execute at a phenomenally high rate if memory is limited. It simply takes one scan of the database to detect enormous colossal pattern sequences thanks to the algorithm's construction. DPMine surpasses Colossal Pattern Miner (CPM) and BVBUC in a variety of biological data sets, according to an empirical investigation.

The decision-making process will be hampered because not all of the association's regulations will be generated. Most BVBUC enormous closed itemset support information is incorrect. This generates inaccurate association rules and has an impact on the decision-making process. Large cardinality itemsets, also known as enormous itemsets, are of particular interest to ARM because they may be used in applications using High-Dimensional Biological Data sets (HDBD) [31]. Colossal itemsets are significantly relevant and influential in many applications [32]. Naulaerts et al. [33] and Alves et al. [34] proved the importance of mining enormous item collections from high dimensional.

Since the time required to extract short and average-sized itemsets is exponential, [35] are inefficient for extracting FCCI from HDBD. The Pattern-Fusion approach was the first algorithm to come up with a large number of things [16]. In the Pattern-Fusion approach, the approximation of gigantic closed itemsets aids in the mining of big cardinality itemset. This bottleneck prevents the Pattern-Fusion approach from extracting a high number of FCCI. As a result of an insufficient collection of association rules, making decisions become more complicated. Even though HDBD contains a significant number of FCCI and frequently enormous itemsets, the BVBUC algorithm is unable to extract any significant number of these. Association rules influence decision-making in part because they are created. Even for the vast majority of mined FCCI, the BVBUC delivers incorrect support info. As a result, incorrect association rules are formed, which has a negative effect on decision-making.

The DisClose algorithm described by Zulkurnain et al. [36] mines FCCI from HDBD using a CompactRowtree (CR-tree). Before beginning the process of extracting FCCI from HDBD, the existing works do not have the ability to remove all unimportant characteristics and rows. As the number of rows enumerated in the mining search space grows exponentially, the algorithm becomes less efficient in mining FCCI. An efficient trimming methodology to minimise the row enumerated search space and an well-organised rowset testing mechanism are missing from current FCCI mining methods. The best way to extract the FCCI from HDBD is to traverse the row enumerated tree. There is an inherent imbalance in the row enumerated tree since the number of nodes in each row enumerated tree branch varies. Row enumerated trees must be distributed evenly across compute nodes in order to mine the FCCI efficiently. The compute nodes should be equally burdened when it comes to traversing.

Recently, a technique known as LCCP for mining massive patterns has been developed [25]. To begin, the issue of mining enormous patterns while enforcing length restrictions was raised. To swiftly determine if a huge pattern satisfies the length limitations, two new theorems were presented. Theorems based on these were used to develop a real algorithm for mining huge patterns with length limitations, eliminating those patterns that do not meet the length constraints in order to reduce mining durations. The min- and max-length limitations for mining gargantuan patterns were the focus of this paper. However, in order to mine the huge patterns, which necessitate the ideal selection of threshold values, it takes a long time to train the system.

According to this assessment, the greatest patterns may be found in the rowset, which has a support of 1. A criterion for mining large itemsets from high-dimensional data involving the item's support value is therefore unnecessary. As an alternative, a minimal permissible number of elements in a collection is known as the minimum cardinality of a collection.

5 Conclusion

To mine patterns in databases with a great number of characteristics and values, mining enormous patterns is utilised, although the number of occurrences in each database is limited. It is possible to extract gigantic patterns using efficient methods, but these methods cannot be applied to the case of constraint-based colossal pattern mining. It is the goal of this survey to investigate the extraction of enormous itemsets from biological data sets with high dimensionality. Massive mined itemsets of an average length do not include sufficient and useful info for making decisions. As a result, an enormous sum of time is spent mining a large number of short and medium-sized itemsets. The high-dimensional data set was created because of the increased interest in bioinformatics research and the abundance of data from a range of sources. These data sets have a great sum of features and a short number of rows. Bioinformatic applications, e.g. rely heavily on colossal pattern itemsets, which have a major impact on decision-making. From the enormous amount of information and knowledge that can be extracted, it is not an easy task. A high-dimensional data set's sequential and computationally expensive sequential mining algorithms include gigantic closed itemsets (CCI).

References

1. Xu J, Zhang Y, Zhang P, Mahmood A, Li Y, Khatoon S (2017) Data mining on ICU mortality prediction using early temporal data: a survey. Int J Inf Technol Decis Mak 16:117–159
2. Sohrabi MK, Karimi F (2018) Feature selection approach to detect spam in the facebook social network. Arabian J Sci 43:949–958

3. Kapusta J, Munk M, Drlik M (2018) Website structure improvement based on the combination of selected web structure and web usage mining methods. Int J Inf Technol Decis-Making. https://doi.org/10.1142/S0219622018500402
4. Hemmatian F, Sohrabi MK (2018) A survey on classification techniques for opinion mining and sentiment analysis. Artific Intell Rev. 10.1007/ s10462-017-9599-6
5. Sohrabi MK, Azgomi H (2017) Parallel set similarity join on big data based on localitysensitive hashing. Sci Comput Program 145:1–12
6. Liao S, Chang H (2016) A rough set-based association rule approach for a recommendation system for online consumers. Inf Process Manage 52:1142–1160
7. Sohrabi MK, Roshani R (2017) Frequent itemset mining using cellular learning automata. Comput Hum Behav 68:244–253
8. Huynh B, Vo B, Snasel V (2017) An efficient parallel method for mining frequent closed sequential patterns. IEEE Access 5:17392–17402
9. Cheng X, Su S, Xu S, Xiong L, Xiao K, Zhao M (2018) A two-phase algorithm for differentially private frequent subgraph mining. IEEE Trans Knowl Data Eng 30:1411–1425
10. Sohrabi MK, Taheri N (2018) A haoop-based parallel mining of frequent itemsets using N-Lists. J Chin Inst Eng 41:229–238
11. Rodríguez-González AY, Lezama F, Iglesias-Alvarez CA, Martínez-Trinidad JF, Carrasco-Ochoa JA, de Cote EM (2018) Closed frequent similar pattern mining: reducing the number of frequent similar patterns without information loss. Expert Syst Appl 96:271–283
12. Fasihy H, Shahraki MHN (2018) Incremental mining maximal frequent patterns from univariate uncertain data. Knowl-Based Syst 152:40–50
13. Prasanna K, Seetha M (2015) Efficient and accurate discovery of colossal pattern sequences from biological datasets: a doubleton pattern mining strategy (DPMine). Proc Comput Sci 54:412–421
14. Pasquier N, Bastide Y, Taouil R, Lakhal L (1999) Discovering frequent closed itemsets for association rules. In: Proceeding of the 7th international conference on database theory (ICDT'99). Israel, pp 398–416
15. Burdick D, Calimlim M, Gehrke J (2001) MAFIA: a maximal frequent itemset algorithm for transactional databases. In: Proceeding of the 2001 international conference on data engineering (ICDE'01). Heidelberg, Germany, pp 443–452
16. Zhu F, Yan X, Han J, Yu P, Cheng H (2007) Mining colossal frequent patterns by core pattern fusion. In: Proceeding of the 2007 Pacific-Asia conference on knowledge discovery and data mining
17. Fournier-Viger P, Lin JC-W, Vo B, Chi TT, Zhang J, Le HB (2017) A survey of itemset mining. Wiley Interdisc Rev: Data Mining Knowl Disc 4(7):e1207
18. Zhang L, Fu G, Cheng F, Qiu J, Su Y (2018) A multi-objective evolutionary approach for mining frequent and high utility itemsets. Appl Soft Comput 62:974–986
19. Djenouri Y, Comuzzi M (2017) Combiningapriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem. InfSci 420:1–15
20. Xun Y, Zhang J, Qin X, Zhao X (2017) FiDoop-DP: data partitioning in frequent itemset mining on hadoop clusters. IEEE Trans Parallel Distrib Syst 28(1):101–114
21. Djenouri Y, Lin JC-W, Nørvåg K, Ramampiaro H (2019) Highly efficient pattern mining based on transaction decomposition. In: IEEE international conference on data engineering, pp 1646–1649
22. Nguyen T, Vo B, Snásel V (2017) Efficient algorithms for mining colossal patterns in high dimensional databases. Knowl-Based Syst 122:75–89
23. McIntosh T, Chawla S (2007) High confidence rule mining for microarray analysis. IEEE/ ACM Trans ComputBiolBioinform 4:611–623
24. G. Cong, K.-L. Tan, A. Tung, and F. Pan, "Mining Frequent Closed Patterns in Microarray Data," Proc. Fourth IEEE Int"l Conf. Data Mining (ICDM), vol. 4, pp. 363–366, 2004.
25. Le T, Nguyen TL, Huynh B, Nguyen H, Hong TP, Snasel V (2021) Mining colossal patterns with length constraints. Appl Intell 51(12):8629–8640

26. Alves R, Rodriguez-Baena DS, Aguilar-Ruiz JS (2009) Gene association analysis: a survey of frequent pattern mining from gene expression data. Brief Bioinform 11(2):210–224
27. Madhavi D, Mogalla S (2010) An efficient approach to colossal pattern mining. IJCSNS Int J Comput Sci Netw Secur 10(1)
28. Sohrabi MK, Barforoush AA (2012) Efficient colossal pattern mining in high dimensional datasets. Knowl-Based Syst 33:41–52
29. Zulkurnain NG (2012) DisClose: discovering colossal closed itemsets from high dimensional datasets via a compact row-tree
30. Nguyen TL, Vo B, Snasel V (2017) Efficient algorithms for mining colossal patterns in high dimensional databases. Know-Based Syst 122(C):75–89
31. Sohrabi MK, Barforoush AA (2012) Efficient colossal pattern mining in high dimensional datasets. Knowl-Based Syst 33:41–52
32. Yoon Y, Lee GG (2012) Subcellular localization prediction through boostingassociation rules. IEEE/ACM Trans Comput Biol Bioinform 9(2):609–618
33. Naulaerts S, Meysman P, Bittremieux W, Vu TN, Berghe WV, Goethals B, Laukens K (2015) A primer to frequent itemset mining for bioinformatics. Brief Bioinform 16(2):216–231
34. Alves R, Rodriguez-Baena DS, Aguilar-Ruiz JS (2009) Gene association analysis: a survey of frequent pattern mining from gene expression data. Brief Bioinform bbp042
35. Zaki MJ, Hsiao C-J (2005) Efficient algorithms for mining closed itemsets andtheir lattice structure. IEEE Trans Knowl Data Eng 17(4):462–478
36. Zulkurnain NF, Haglin DJ, Keane JA (2012) Disclose: discovering colossal closeditemsets via a memory efficient compact row-tree. In: Emerging trends in knowledge discovery and data mining. Springer, pp 141–156