Model for Finding Frequent Sets in FP-growth for Multimodal Data

Nataliya Boyko, Oleksandr Tkachyk

Lviv Polytechnic National University, Profesorska Street 1, Lviv, 79013, Ukraine

Abstract

The article considers the process of generating candidates from a structured database and discusses the basic concepts and metrics (support, confidence, lift and conviction). The principle of operation of the FP-Growth algorithm is described in detail, both its mathematical part and the software part on the example of the Python language. Two primary levels of scientific knowledge are considered for research: empirical and theoretical. The advantages and disadvantages of the Apriori algorithm are presented. These shortcomings can be overcome with the FP-Growth algorithm. This algorithm is an improved method of Apriori. Frequent patterns are formed without the need to generate candidates. The study considers the main stages of the alternative algorithm FP-Growth search for associative rules. The process of transaction arrears in the database is considered in detail. The method of transforming multimodal data into a tree structure is thoroughly discussed. The example assumes a detailed iteration process for each transaction. The procedure for generating candidates is analyzed in such a way as to reduce the number of passes for a given data set. The paper examines the process of compressing transactions into a simple structure. The efficient and complete output of partial data sets is provided.

Keywords [1]

Machine Learning, Artificial Intelligence, Frequent Pattern-Growth Algorithm, Frequent Pattern-Growth Algorithm Tree, Apriori, Associative Rule, Associations Rules Learning, Netflix, Amazon.

1. Introduction

There is no doubt that in the last few years, machine learning (ML) / artificial intelligence (AI) has been gaining in popularity. Today, the use of machine learning in processing multimodal data is a multi-stage complex process that allows you to perform the necessary tasks for forecasting. Some of the most common ML algorithms are Netflix's algorithms for creating movie offers based on movies you've watched in the past [2, 12]. Another example is Amazon and its algorithms, which recommend books based on books you've previously purchased [3, 6, 11].

This work is based on one of these algorithms, the FP-Growth algorithm. FP-Growth (Frequent Pattern Growth) is a reasonably young algorithm. They were first described in 2000. FP-Growth offers a radical approach - to abandon the generation of candidates (generation of candidates is the basis of the Apriori algorithm) [1, 2, 9]. Theoretically, this approach will further increase the algorithm's speed and use even less memory. This is achieved by storing the prefix tree in memory, not from combinations of candidates but the transactions themselves [8, 16, 21].

There are several research methods. It is accepted to allocate two basic levels of scientific knowledge: empirical and theoretical. This division is because the subject can acquire knowledge experimentally (empirically) and through complex logical operations, i.e. theoretically [1, 3, 8].

The practical level of knowledge includes [13, 17]:

[1] The Fifth International Workshop on Computer Modeling and Intelligent Systems (CMIS-2022), May 12, 2022, Zaporizhzhia, Ukraine
EMAIL: nataliya.i.boyko@lpnu.ua (N. Boyko); oleksandr.a.tkachyk@lpnu.ua (O.Tkachyk)
ORCID: 0000-0002-6962-9363 (N. Boyko); 0000-0002-0728-4208 (O.Tkachyk)

- Observation of phenomena.
- Accumulation and selection of facts.
- Establishing ties between them.

The practical level is collecting data (facts) about social and natural objects. The object under study is reflected by external connections and manifestations at the functional level.

The theoretical level of cognition is associated with the predominance of mental activity, with the understanding of empirical material, it's processing.

The theoretical level reveals [2, 9]:

- Internal structure and patterns of development of systems and phenomena.
- Their interaction and conditionality.

2. Materials and methods

FP-Growth is an improvement on the Apriori method.

Disadvantages of the Apriori algorithm [11, 20]:

- To use Apriori, you need to generate sets of candidates. These elements can be significant if the set of elements in the database is also significant.
- Apriori requires multiple database scans to verify support for each generated set of items, resulting in high costs.

Unlike the Apriori algorithm, the alternative algorithm search algorithm avoids the multi-stage process of generating candidates while reducing the number of movements in the multimodal data set. This algorithm is an improved method of Apriori. Frequent regularity is formed without the need to generate candidates. The FP-Growth algorithm represents a database in a regular pattern tree or FPG tree [21, 13].

This tree structure will keep the link between the set of elements. A single frequent element fragments the database. This fragmented part is called a "pattern fragment". Sets of elements of these fragmentary patterns are analyzed. Thus, the search for frequent sets of items is relatively reduced with this method.

2.1. Analysis of the existing informational system

A Frequent Pattern-Growth tree is a tree-like structure made from initial sets of database elements. The purpose of the FPG tree is to remove the most common pattern. Each node of the FPG tree is an element of a set of elements [9, 14, 20].

To explain the tree structure of the alternative algorithm, the root node must be denoted by zero, and the leaves must acquire values from the data sets. The association of nodes with lower nodes, which are sets of elements, with other sets of elements is preserved during the formation of the tree.

FP-Growth Algorithm steps [7, 17]:

1) In the first stage, an FPG-tree is built that shows the data that is most common in the database. It is similar to the first step of the algorithm Apriori. FPG-tree is built based on ordered data sets, which are written in descending order of their support values. There is a rule for constructing an FPG tree. It should be formulated as follows: if a node of the same name occurs in a tree, a new node is not created, and the index of the corresponding node is increased by 1. Otherwise, a new node with index one is created.

2) In the second stage, the elements that are often found in the multimodal data set are removed from the tree.

3) In the third step, another transaction check is performed in the database. In this step, the components of the same name are removed from the multimodal data set with repeated force. The set of elements with the maximum number starts at the top, the next set of elements with the smaller number below. This means that a tree branch comprises sets of transaction elements in descending order.

4) In the fourth stage, any element from the original data set is selected, and all paths leading to this element are found. Then count the number of encounters of a given element on each way. Then the element itself (set suffix) is removed from the paths leading to it. As a result, the coincidence of

elements in the prefixes of the ways is calculated and recorded in descending order. Thus a new set of elements of groups is formed. On its basis, a new conditional tree is built, which is associated with one object. All nodes whose support is equal to or greater than the minimum value are present on this tree. Node indices are summed if there is an element with a frequency greater than 2. starting from the root, the paths to the nodes whose support is greater than or equal to the specified.

5) The paths from the root to the nodes are fixed in the next stage. the item that was deleted in the previous step is returned, and the final support values are calculated. From it is the path along the tree FPG. This path is called the conditional template base.

A conditional template database is a database that consists of configuration loops in the FPG tree that meet the lowest node (suffix).

6) A conditional tree FPG is constructed, which is formed by counting sets of elements on the path. The FPG tree considers the sets of elements corresponding to the threshold support.

7) Frequent templates are generated from the FPG tree.

Advantages of the FP-Growth algorithm [8, 19]:

- An alternative algorithm search algorithm does not require scanning each transaction on all iterations as it does in executing the algorithm Apriori. It must be performed twice.
- During the execution of the alternative algorithm, the same name elements are removed, and they are not combined as in the classical algorithm Apriori. It is this process that makes it work faster.
- In the process of building a FP-tree, it is reduced, and accordingly, the number of elements is smaller, which allows you to optimize the work of the database and optimize its dimensionality.
- The effectiveness of the alternative algorithm for finding alternative rules is due to the removal of long and short partial chains.

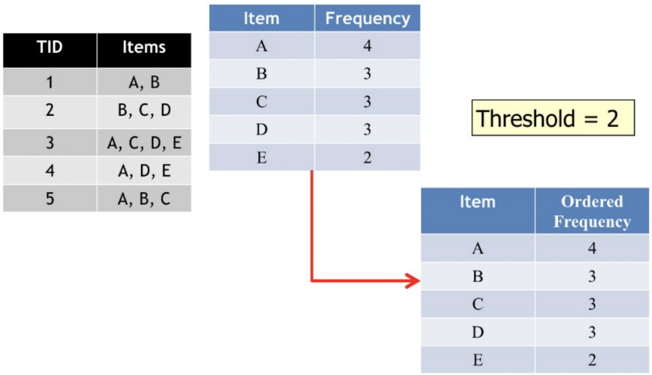
Disadvantages of the FP-Growth algorithm [5, 10, 16]:

- The process of building a FP-tree when working with an alternative algorithm for finding associative rules is cumbersome and complex, rather than the algorithm Apriori.
- In performing each stage of the alternative algorithm for finding associative rules, many resources are used because, algorithmically, it is more complex than Apriori.
- When building a tree using an alternative algorithm for finding associative rules on large sets of multimodal data, many nodes can occur. This can overflow the database.

3. Results of research and experiments

Let's start with creating a tree [1, 12, 15]:

Select the table headings to build a FP-tree and determine its first instance. It speeds up access to all elements of the tree. You need to create an index plugin to keep track of the total number of certain types of items in the tree. You can use the createTree () function to build a tree based on minimal support and dataset arguments. During the first pass, all elements are checked. In the process, the number of identical on each path is counted. As a result in Figure 1 shows the title table.



TID	Items
1	A, B
2	B, C, D
3	A, C, D, E
4	A, D, E
5	A, B, C

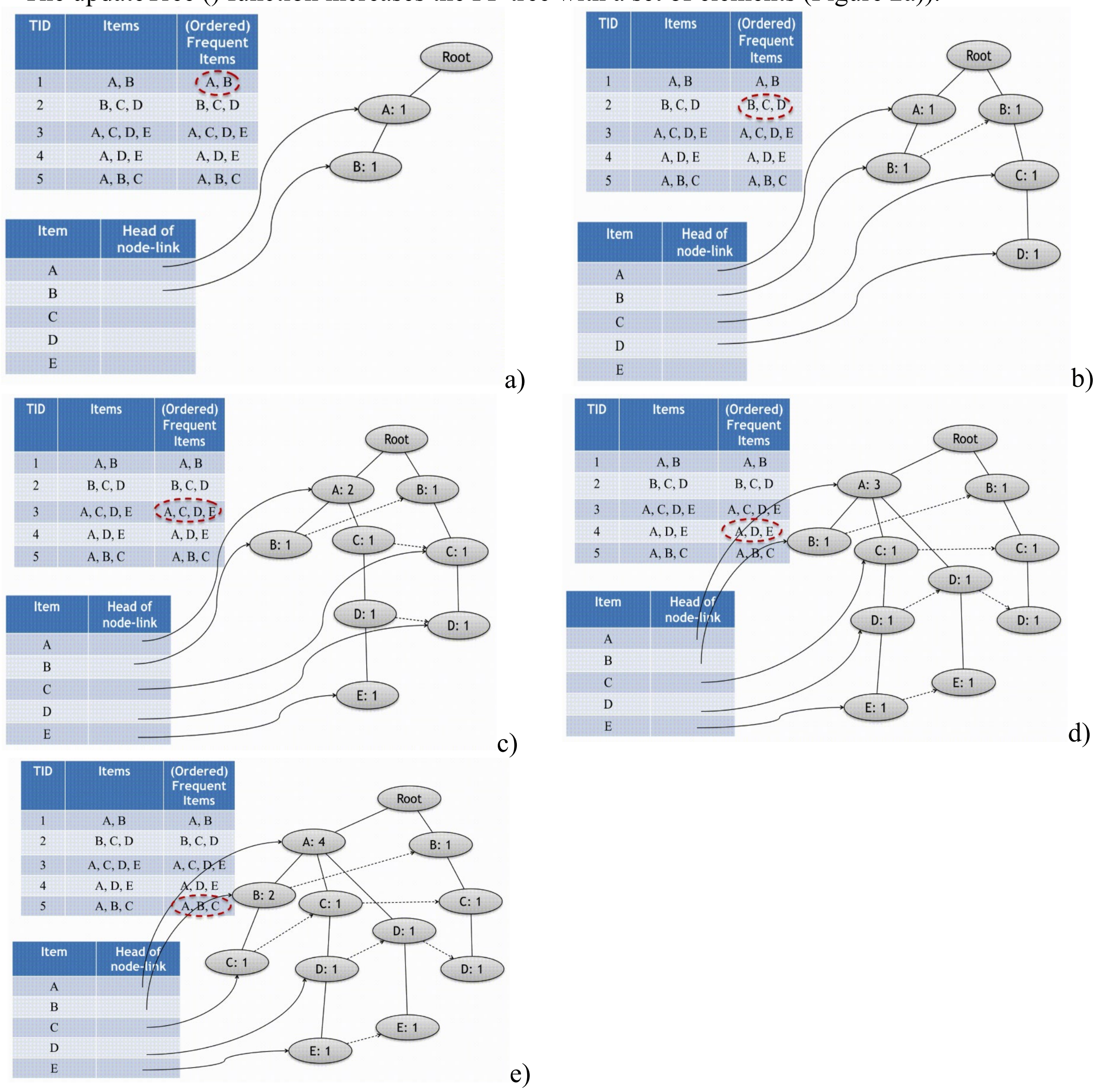
Item	Frequency
A	4
B	3
C	3
D	3
E	2

Threshold = 2

Item	Ordered Frequency
A	4
B	3
C	3
D	3
E	2

Figure 1: Result of the Frequent pattern tree

The updateTree () function increases the FP tree with a set of elements (Figure 2a)).



TID	Items	(Ordered) Frequent Items
1	A, B	A, B
2	B, C, D	B, C, D
3	A, C, D, E	A, C, D, E
4	A, D, E	A, D, E
5	A, B, C	A, B, C

Item	Head of node-link
A	
B	
C	
D	
E	

Figure 2: a) Sets of items found in the database after scanning; b) The root is represented as a zero value; c) A tree branch is built from sets of transaction elements in descending order; d) The branch of the transaction has a common prefix to the root; e) Increase by 1 total node and the number of new nodes.

The updateHeader () function allows you to fix connections with nodes, so Fig. 2 b shows an FP-tree with instances of the required elements.

In fig. 2c shows the process when it is impossible to obtain list entries using the createTree () function. In the process of processing, a dictionary is expected, in which there is a set of elements and the frequency of their use.

The createInitSet () function does this transformation for us (Figure 2d)).

Finally, we can easily generate all frequent sets (Figure 3).

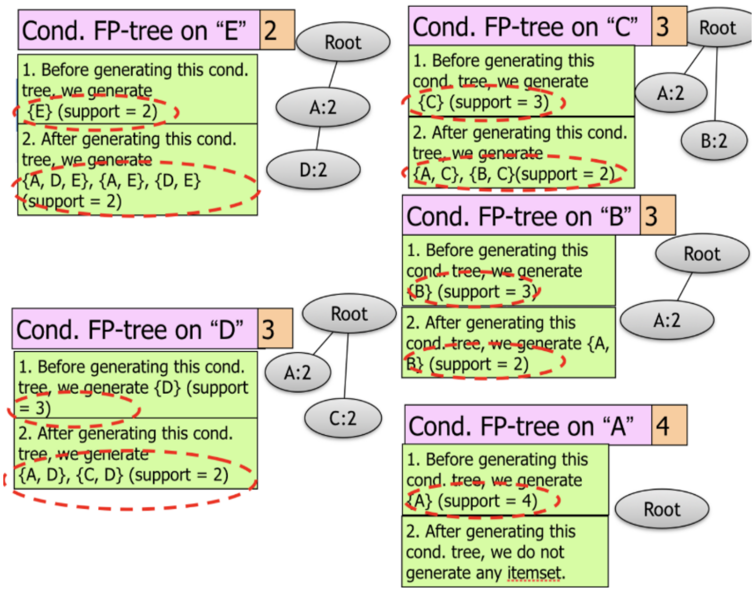

Cond. FP-tree on "E"
2
1. Before generating this cond. tree, we generate {E} (support = 2)
2. After generating this cond. tree, we generate {A, D, E}, {A, E}, {D, E} (support = 2)
Root
A:2
D:2
Cond. FP-tree on "C"
3
1. Before generating this cond. tree, we generate {C} (support = 3)
2. After generating this cond. tree, we generate {A, C}, {B, C}(support = 2)
Root
A:2
B:2
Cond. FP-tree on "B"
3
1. Before generating this cond. tree, we generate {B} (support = 3)
2. After generating this cond. tree, we generate {A, B} (support = 2)
Root
A:2
Cond. FP-tree on "D"
3
1. Before generating this cond. tree, we generate {D} (support = 3)
2. After generating this cond. tree, we generate {A, D}, {C, D} (support = 2)
Root
A:2
C:2
Cond. FP-tree on "A"
4
1. Before generating this cond. tree, we generate {A} (support = 4)
2. After generating this cond. tree, we do not generate any itemset.
Root

Figure 3: Generated frequent sets

3.1. Development of algorithms for solving a functional problem

Let's start creating the algorithm itself. There are three main steps to finding frequent sets of elements in an FP tree:

1) Get a database of conditional templates from the FP-tree.
2) The next step is to create a conditional tree based on conditional templates.
3) To build a node, repeat steps 1 and 2 until one element remains.

The ascendTree () function, which climbs our tree and collects the names of the elements it encountered (Figure 4):

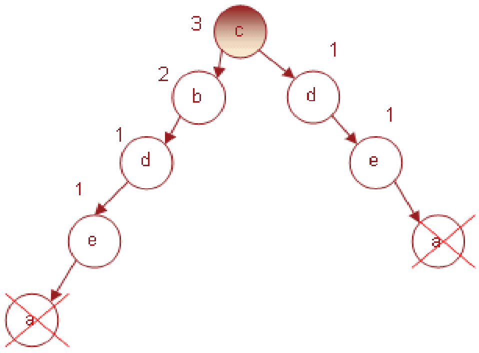


Figure 4: AscendTree () function results

The findPrefixPath () function sorts through the list until it reaches the end. For each element it encounters, it calls the ascendTree () function.

Figure 5 shows the process when condPats is returned to the list and added to the template dictionary.

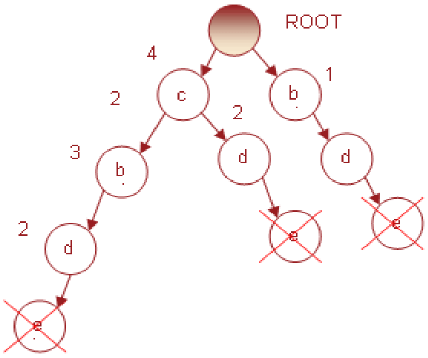


Figure 5: FindPrefixPath () function results

Giving search parameters for our algorithm, it finds the necessary data for us.
Command for the algorithm: indPrefixPath('x', myHeaderTab['x'][1]).
Results: {frozenset({'z'}): 3}.

3.2. Definition and estimation of qualitative indicators of algorithms, comparing with existing ones

Data preprocessing is required to apply any algorithms, including the classical Apriori algorithm and the alternative search algorithm. The data must be without extra attributes, precise, not zeros, etc. Reprocessing should be treated with all responsibility because the best result depends on their effective processing.

Two data sets will be used for the experimental study. Datasets were obtained from the UCI machine learning databases (Table 1).

Table 1
Data set details

Dataset name	Number of Instances	Number of Attributes
Mushroom	8124	22
Super Market	4627	217

We are analyzing the data in the Table 1, according to the sets of input elements Mushroom and Supermarket, we can conclude that the efficiency of the alternative algorithm for finding associative rules is better than the classical algorithm a priori. This difference in the execution time of transactions is presented in Fig. 6-9. Execution time is when to mine frequent data patterns with different transactions (Figures 6-9).

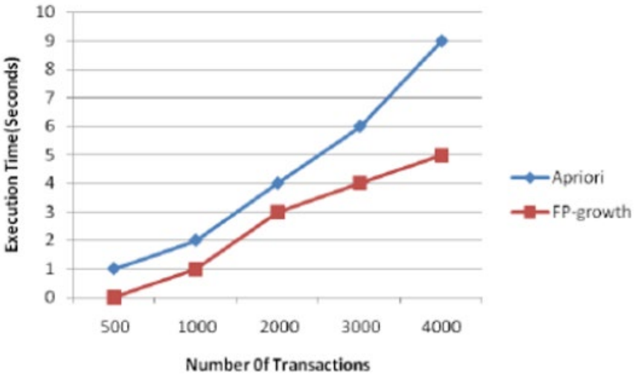


Figure 6: Execution time with increased transactions for the Mushroom date set (support = 0.1)

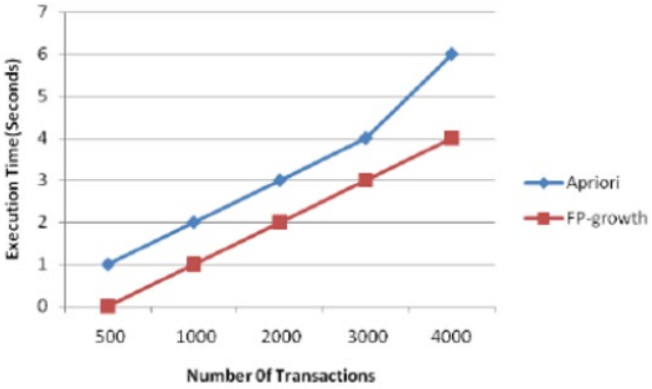


Figure 7: Execution time with increased transactions for the Mushroom date set (support= 0.5)

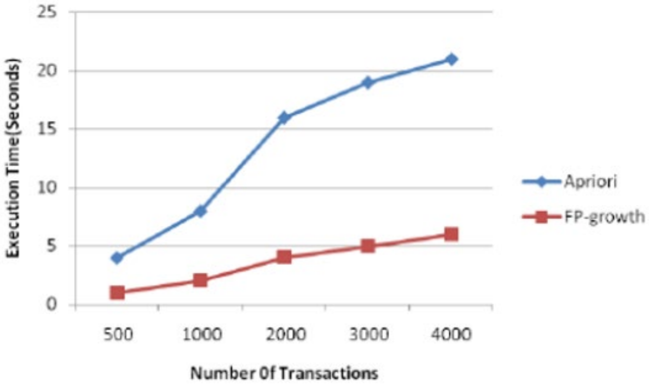


Figure 8: Execution time with increased transactions for the SuperMarket date set (support = 0.1)

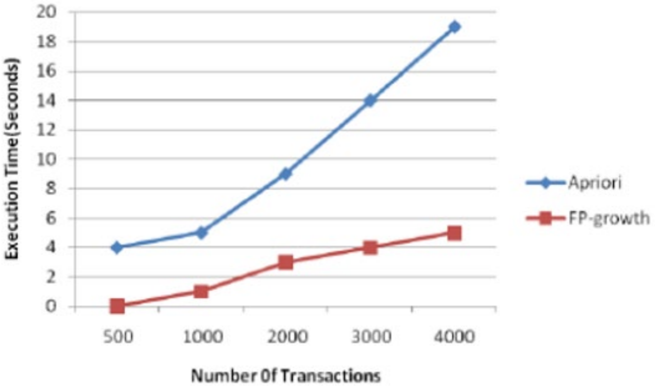


Figure 9: Execution time with increased transactions for the SuperMarket date set (support = 0.5)

A comparison of the curves in Figures 10-13 showed that when using an alternative algorithm search algorithm, as the number of transactions increases, the execution rate also increases, but support levels are minimal. This means that the FPG- algorithm is more efficient in terms of saving time than the Apriori.

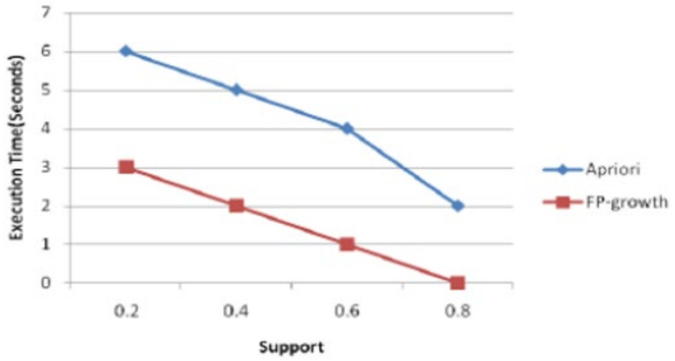


Figure 10: Execution time with different levels of support for the Mushroom set date (2000 transactions)

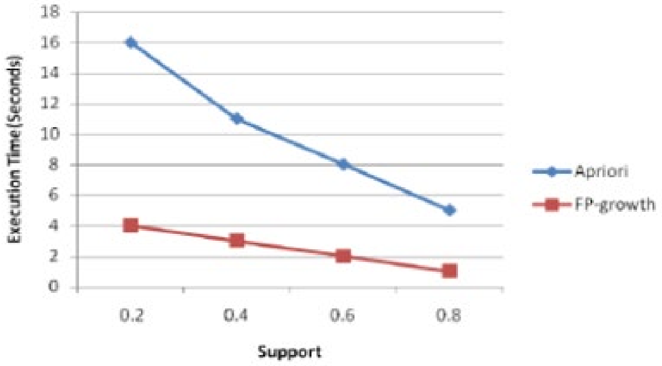


Figure 11: Execution time with different levels of support for the Mushroom set date (4000 transactions)

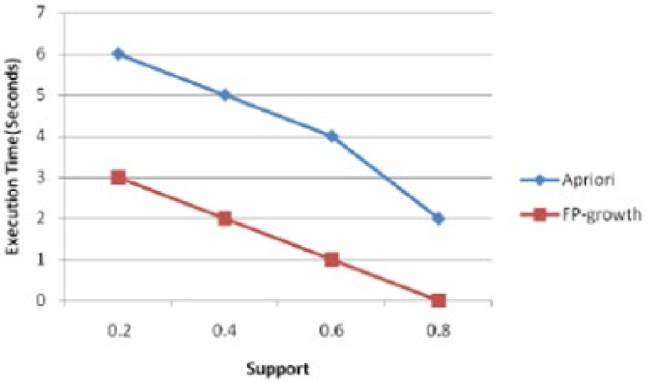


Figure 12: Execution time with different levels of support for the SuperMarket date set (2000 transactions)

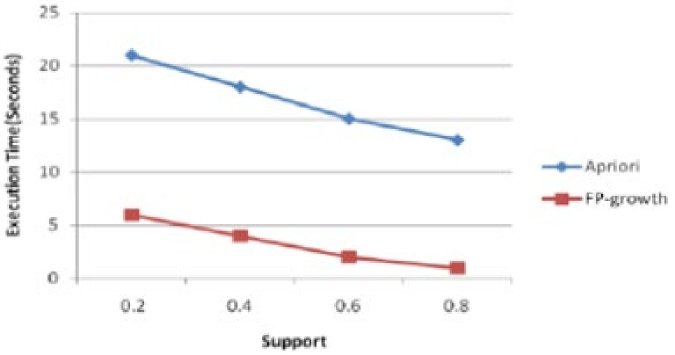


Figure 13: Execution time with different levels of support for the SuperMarket date set (4000 transactions)

Figures 10-13 show the performance analysis of execution time Apriori algorithm and FP-Growth algorithm with different levels of support. This shows the time required to be completed.

The FP-Growth algorithm is significantly smaller compared to the Apriori algorithm.

4. Conclusions

So, we got acquainted with the basic theory of ARL ("who bought x, also bought y") and the basic concepts and metrics (support, confidence, lift and conviction).

The principle of operation of the FP-Growth algorithm was described in detail, both its mathematical part and the software part on the example of the Python language.

FPG is more effective than a priori. It is convenient because the set of multimodal source data is represented as a tree. If each element occurs several times, then on the FP-tree, it is defined as a node, and its index indicates the number of its repetitions in the set.

A comparison of the work of the two algorithms showed that with the increase in the size of the original data set, the time spent searching for the most common element. (Figure 14).

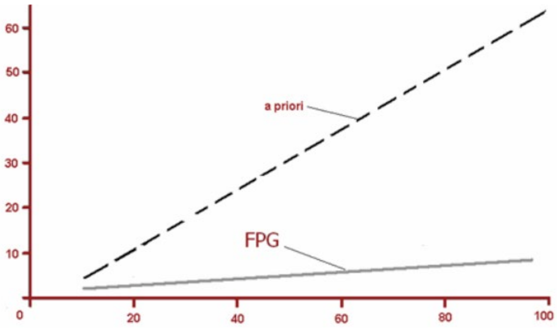


Figure 14: Graph of comparison results of Apriori and FP-Growth algorithms

The work of an alternative algorithm search algorithm provides the most efficient and complete extraction of frequent subject sets to compress database transactions. After all, the divide and conquer technique is used when constructing an FP tree. With its help, it is allowed to perform decomposition of one complex task into several simple ones. in the process of completing the stages of the algorithm, it is possible to avoid the procedure of losses in the generation of candidates.

As a result of the study of the two algorithms on the sets of sizeable multimodal data, the efficiency of searching for associative rules in favor is proved by the FP-Growth algorithm. In the example, the efficiency was determined in the accelerated search and informativeness of the found associative regulations and the ability to predict decisions.

5. References

[1] H. Alshammari, S.A. El-Ghany, A. Shehab, Big IoT Healthcare Data Analytics Framework Based on Fog and Cloud Computing. Journal of Information Processing Systems, Vol. 16, 2020, pp. 1238–1249.

[2] J. Qi, P. Yang, M. Hanneghan, S. Tang, B. Zhou, A Hybrid Hierarchical Framework for Gym Physical Activity Recognition and Measurement Using Wearable Sensors. IEEE Internet of Things Journal, Vol. 6, 2019, pp. 1384–1393.

[3] T. Wang, L. Qiu, A.K. Sangaiah, A. Liu, Z. Alam Bhuiyan, Y. Ma, Edge-Computing-Based Trustworthy Data Collection Model in the Internet of Things. IEEE Internet of Things Journal, Vol. 7, 2020, pp. 4218–4227.

[4] N. Boyko, K. Kmetyk-Podubinska and I. Andrusiak, Application of Ensemble Methods of Strengthening in Search of Legal Information, in: Lecture Notes on Data Engineering and

Communications Technologies, Vol. 77, 2021, pp. 188-200. https://doi.org/10.1007/978-3-030-82014-5_13

[5] A. Al-Shargabi, F. Siewe, A Lightweight Association Rules Based Prediction Algorithm (LWRCCAR) for Context-Aware Systems in IoT Ubiquitous, Fog, and Edge Computing Environment, in: Proceedings of the Proceedings of the Future Technologies Conference (FTC), 2020.

[6] C.H. Lee, J.S. Park, An SDN-Based Packet Scheduling Scheme for Transmitting Emergency Data in Mobile Edge Computing Environments. Journal Series: Human-centric Computing and Information Sciences, Vol. 11, 2021.

[7] Y. He, Z. Tang, Strategy for Task Offloading of Multi-user and Multi-server Based on Cost Optimization in Mobile Edge Computing Environment. Journal of Information Processing Systems, Vol. 17, 2021, pp. 615–629.

[8] J.-H. Lee, Next Task Size Prediction Method for FP-Growth Algorithm. Journal Series: Human-centric Computing and Information Sciences, Vol. 11, 2021.

[9] Y.A. Ünvan, Market basket analysis with association rules. Journal Communications in Statistics - Theory and Methods, Vol. 50, 2021, pp. 1615–1628.

[10] Z. Mar, K.K. Oo, An Improvement of Apriori Mining Algorithm using Linked List Based Hash Table, in: Proceedings of the 2020 International Conference on Advanced Information Technologies (ICAIT), 4–5 November 2020.

[11] Z. Pan, P. Liu, J. Yi, An Improved FP-Tree Algorithm for Mining Maximal Frequent Patterns, in: Proceedings of the 2018 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), 10–11 February 2018.

[12] N. Boyko, A look trough methods of intellectual data analysis and their applying in informational systems, in: XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT), 2016, pp. 183-185.

[13] M. Yin, W. Wang, Y. Liu and D. Jiang, An improvement of FP-Growth association rule mining algorithm based on adjacency table, in: 2nd International Conference on Material Engineering and Advanced Manufacturing Technology (MEAMT 2018), Vol. 189, 2018. https://doi.org/10.1051/matecconf/201818910012.

[14] J. Hao, M. He, A Parallel FP-Growth Algorithm Based on GPU, in: IEEE, International Conference on E-Business Engineering. IEEE Computer Society, 2017, pp. 97-102.

[15] H.Y. Chang, J.C. Lin, M. L. Cheng, A Novel Incremental Data Mining Algorithm Based on FP-growth for Big Data, in: International Conference on NETWORKING and Network Applications, 2016, pp. 375-378.

[16] J. Heaton, Comparing dataset characteristics that favor the Apriori, Eclat or FP-Growth frequent itemset mining algorithms, in: Southeastcon. IEEE, 2017, pp. 1-7.

[17] J. Hao, M. He, A Parallel FP-Growth Algorithm Based on GPU, in: IEEE, International Conference on E-Business Engineering. IEEE Computer Society, 2017, pp. 97-102.

[18] B. Subbulakshmi, B. Dharini, C. Deisy, Recent weighted maximal frequent itemset Mining, in: International Conference on I-Smac. IEEE, 2017, pp. 391-397.

[19] T. Willhalm, FPTree: A Hybrid SCM-DRAM Persistent and Concurrent B-Tree for Storage Class Memory, in: International Conference on Management of Data, 2016, pp. 371-386.

[20] Y. Zeng, Y. Shiqun, L. Jiangyue, M. Zhang, Research of Improved FP-Growth Algorithm in Association Rules Mining. Scientific Programming, Vol. 5, 2015, pp.124-136.

[21] C. Jun, G. Li, An improved FP-growth algorithm based on item head table node. Journal of Information Technology, Vol. 12, 2013, pp. 34–35.