Association rules mining with three-dimensional data structure

Evgenia O. Khramshina, Alexander V. Prutzkow

Abstract — **Association rules mining progresses year by year. There are many algorithms of association rules mining. The most popular are the Apriori algorithm and the FP-Growth algorithm. But these algorithms have disadvantages. The Apriori algorithm requires many transaction base passes. The FP-Growth algorithm uses a many-edged (non-binary) tree data structure. Algorithm is characterized by the data structure used in it. We discover an association rules mining algorithm using three-dimensional data structure. Algorithm needs only two transaction base passes. The first pass is to insert transactions in three-dimensional data structure. The second pass is to count support of extracted from three-dimensional data structure itemsets. The algorithm tested and compared with the Apriori algorithm and the FP-Growth algorithm. The algorithm is more effective by memory usage than the FP-Growth algorithm, when number of unique elements is between 10 and 868, and the Apriori algorithm, when number of unique elements is between 49 and 498.**

***Keywords* — algorithms, association rules mining, Apriori, FP-Growth, three-dimensional array.**

I. Introduction

More data is stored electronically. As before, data need to be processed with least amount of physical resources and time. The Data Mining subject is used to analyze patterns in data. The subject is a synthesis of few disciplines such as statistics, theory of algorithms and theory of databases, artificial intelligence, and others. Data Mining has such directions as classification, clustering, neural networks, association rules mining. The latter one is applied in trade [10], medicine [5, 12] (also see a survey in [13]), business [11] and Internet surfing [3]. Its purpose is to discover hidden patterns in data.

II. Problem Statement

A. Formulation of the problem

Let's formulate the problem of association rules mining in the terms of combinations. There is an ordered increasing sequence of non-repeating items named combination. There is a list of combinations named D. We need to find subcombinations with frequency of occurrence above minimal. The D list combinations are named transactions, final subcombinations are named itemsets, intermediate subcombinations – candidatesets, frequency – support.

Then every final subcombination S is arranged in rule as $X \Rightarrow Y$, where $X \cup Y = S$, $X \cap Y = \varnothing$. But we drop this action.

Combination items form the E set, $|E| = m$.

III. Research Questions

The following research questions guide the current study:

Question 1: Which data structure will decrease memory usage and, hence, decrease lead time of association rules mining?

Question 2: In which cases does the discovered data structure decrease memory usage and lead time?

IV. Purpose of the Study

The purpose of the study is to discover an algorithm, which will decrease memory usage and lead time of association rules mining.

V. Research Methods

Our research has the following stages:

- to analyze association rules mining algorithms with original data structures;
- to develop association rules mining method based on three-dimensional data structure.

VI. Findings

A. Analysis of structures of association rules mining algorithms

For the first time G. Piatesky-Shapiro raised the problem of association rules mining [6]. Various solutions to this problem have been proposed [2] by the authors R. Agrawal, T. Imielinski, A. Swami in [1] and [8].

Analysis of software for data mining (Statistica, Deductor, Loginom) was conducted. The most of software for association rules mining is based on Apriori and FP-Growth algorithms.

The Apriori Algorithm.

There are a lot of modifications of the Apriori algorithm.

Original algorithm consists of the following steps. Firstly, candidateset is formed; for this, a set of transactions is scanned and many i-element candidates are created, where i – the stage number. Support at this step is not computed. The next step is the computation of candidate support and the selection of those for which it is greater than or equal to

Manuscript received June 29, 2020.

E. O. Khramshina is with the Ryazan State Radio Engineering University named after V. F. Utkin, Ryazan, Russia, 390005 (e-mail: evgenia.khramshina@gmail.com).

A. V. Prutzkow is with the Ryazan State Radio Engineering University named after V. F. Utkin, Ryazan, Russia, 390005 and with Ryazan State Medical Univercity named after I. P. Pavlov, Ryazan, Russia, 390026 (e-mail: mail@prutzkow.com).

the given. The algorithm scans the transaction list several times.

Prefix tree can be used in the Apriori algorithm [9].

Benefits of prefix trees:

- the search time does not depend on the number of elements in the dictionary (depends on the key length);
- additional memory is used to store keys (keys are not stored in nodes);
- support bypass in an orderly manner;
- unlike hash tables, there are no collisions.

Disadvantages:

- stores strings or characters, which means that the key value will have a type restriction (strings, characters, or numbers represented as strings);
- if you implement on its basis an associative array with keys of the string type, then too much memory will be used.

The FP-Growth Algorithm.

The algorithm consists of the following steps [8]. The first step is scanning transactions and finding support for each item. Then the elements are sorted in the transaction in descending order of support for these elements in the entire set. The next step is filtering - removing those elements for which support is less than given by the user. Based on the received data, an FP-tree is built from the remaining elements. And finally, the final step is the extraction of the result sets.

The FP-Growth algorithm uses a tree as a data representation structure. A tree can be described recursively: either it consists of a node and several subtrees (with leaf ends), or it is empty. Another definition assumes that a tree is a connected graph without loops. Decision trees have certain features. Let's start with the benefits:

- simplicity of interpretation and clarity;
- the ability to work both with categories and quantitative values;
- universality in terms of problem solving and classification, and regression;
- the ability to work with data gaps (empty attribute values) and fill in the gaps with the most probable value;
- good performance in the classification process according to an already constructed tree (since the search algorithm in the tree is effective even for large data sets).

Disadvantages:

- process instability (small changes in the data set can lead to the construction of a completely different tree);
- the difficulty of controlling the size of the tree;
- inadequacy of distribution elements into classes in complex cases;
- the criterion for the growth of information is characterized by a tendency to prefer attributes that have a large number of different values.

B. General steps of association rule mining algorithms

As a result, we can summarize, all algorithms have common groups of operators. The general groups can be represented as follows:

- at the first step, all transactions are read and recorded in an intermediate representation;
- at the second step, itemstes are extracted from the intermediate representation.

See more detail analysis of data structure in association rules mining in [7].

C. Association rules mining algorithm based on a three-dimensional data structures

There is a hypothesis that it is possible to improve the algorithm by changing steps or by changing the structure used in it. An array, including three-dimensional one, can be perspective in association rules mining algorithms. We discovered an algorithm for association rules mining, which is based on a three-dimensional array.

The advantages of the structure are following:

- access to elements by index without enumerating all the elements;
- the same access time to all elements;
- the indexing time of elements does not depend on the size of the array, in contrast to linked lists;
- calculation of the address of an element by its index (due to the sequential arrangement of array elements);
- the small size of the memory occupied by the elements (by storing only the information field).
- In addition to the positive aspects of using arrays, there are also negative aspects:
- for a static array – the lack of dynamics, the inability to delete or add an element without shifting others;
- an array with pointers without additional means of control – the danger of exceeding the boundaries of the array and data corruption.

These negative aspects do not manifest themselves in a particular task, so the structure can be used in the task of association rules mining.

We named the algorithm as three-dimensional array association rules mining algorithm (3DAARM = 3D2ARM algorithm).

Since a three-dimensional array is an one-dimensional array of two-dimensional arrays, it is important to arrange its dimensions in ascending order, for programming. In the general case, the maximum transaction length will be much less than the number of unique elements. Size of the three-dimensional array is

$$1/2 \cdot \sum_{i=1}^{t_{\max}-1} (m-i)\cdot(m-i+1) \approx m^2/2 \cdot (t_{\max}-1),$$

where tmax – the maximum length of itemset. According to the Big O theory we can neglect the ($t_{\max}$–1) component.

The name of the array is A.

The array has levels. For example, if we have candidate {1, 2, 3}, we will write it on first level like A[1, 2, 1] and on the second level like A[2, 3, 2], where the last index indicates level. But there can be false itemsets, for example, we have transactions {3, 5, 6} and {2, 5}. There are (A[3, 5, 1] > minimal support) and (A[2, 5, 1] > minimal support) on the first level and (A[5, 6, 2] > minimal support) on the second level. We get itemsets {2, 5, 6} and {3, 5, 6}. But {3, 5, 6} is false itemset. We should analyze it.

The 3D2ARM algorithm consists of the following steps:

0) Initialization of algorithm. Creating and initialization of the three-dimensional array.

1) Iterating over all transactions and inserting them in the three-dimensional array (the first pass).

1.1) Reading a transaction.

1.2) Generating of two-element combinations from unique elements of the transaction.

1.3) Inserting two-element combinations to the three-dimensional array. On each level add 1 to array cell, where first index is equal to or greater than level index. For example, if we have combination {2, 3}, we can add 1 to array element only on the first level and the second level. As a result, we get the array with transactions.

2) Extracting results from the three-dimensional array.

2.1) Generating combinations of unique elements by 2.

2.2) For each level, while the condition, that new candidatesets have been added at the previous level, is true, it performs the following actions. At the first level, we go through all the generated combinations and check if the support is greater than or equals to the minimal support specified. At the next levels, we go through all the previous level candidatesets, adding unique elements to them one by one and checking the support. Unique elements are considered only those that have index larger compared to the last item of the candidateset. As a result, we get candidatesets with support greater than minimal support.

2.3) Checking for the occurrence of the found candidates in the transactions (the second pass). If it is included, we increase support by 1. As a result, we get itemsets.

D. An example of the algorithm run

For example, transactions are shown in the table below (table 1).

TABLE 1 – TRANSACTION DATA

INDEX	TRANSACTION
1	1, 2, 3, 4, 5, 6
2	2, 4, 5, 7
3	1, 3, 6, 7
4	3, 5, 6, 7
5	3, 5, 7
6	1, 2, 4, 6
7	1, 5, 6
8	3, 4, 6
9	2, 5
10	1, 3, 5, 6, 7

We assume the minimum support value to be 0.4, that is, the resulting "popular" sets must occur in at least 4 out of 10 transactions.

The first step is to read the data. Then initialize the three-dimensional array. With the initial data (every element is 0), the array will have the following measurement values: $t_{\max} = 6$, $m = 7$, that is, its size will be

$$1/2 \cdot \sum_{i=1}^{t_{\max}} (m-i)\cdot(m-i+1) =$$

$$= 1/2 \cdot \sum_{i=1}^{5} (7-i)\cdot(7-i+1) = 55$$

To write data to an array, it is necessary to generate combinations for each transaction. For example, for the 1st transaction: {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {3, 4}, {3, 5}, {3, 6}, {4, 5}, {4, 6}, {5, 6}.

For the 2nd transaction: {2, 4}, {2, 5}, {2, 7}, {4, 5}, {4, 7}, {5, 7}.

For the 3rd transaction: {1, 3}, {1, 6}, {1, 7}, {3, 6}, {3, 7}, {6, 7}.

For the 4th transaction: {3, 5}, {3, 6}, {3, 7}, {5, 6}, {5, 7}, {6, 7}.

For the 5th transaction: {3, 5}, {3, 7}, {5, 7}.

For the 6th transaction: {1, 2}, {1, 4}, {1, 6}, {2, 4}, {2, 6}, {4, 6}.

For the 7th transaction: {1, 5}, {1, 6}, {5, 6}.

For the 9th transaction: {2, 5}.

For the 10th transaction: {1, 3}, {1, 5}, {1, 6}, {1, 7}, {3, 5}, {3, 6}, {3, 7}, {5 , 6}, {5, 7}, {6, 7}.

Then, for each level we add 1 to the value of the array cell at the address corresponding to the combination. For example, for the 1st transaction: at level 1, 1 are added to the elements with the following addresses – A[1, 2, 1] = 1, A[1, 3, 1] = 1, A[1, 4, 1] = 1, A[1, 5, 1] = 1, A[1, 6, 1] = 1, A[2, 3, 1] = 1, A[2, 4, 1] = 1, A[2, 5, 1] = 1, A[2, 6, 1] = 1, A[3, 4, 1] = 1, A[3, 5, 1] = 1, A[3, 6, 1] = 1, A[4, 5, 1] = 1, A[4, 6, 1] = 1, A[5, 6, 1] = 1.

For the 2nd transaction: A[2, 4, 1] = 1 + 1, A[2, 5, 1] = 1 + 1, A[2, 7, 1] = 1, A[4, 5, 1] = 1+ 1, A[4, 7, 1] = 1, A[5, 7, 1] = 1. Thus, after processing the 2nd transaction in the elements A[2, 4, 1], A[2, 5, 1], A[4, 5, 1] the value 2 is stored, which means that these combinations have met twice.

For the 3rd transaction: A[1, 3, 1] = 2, A[1, 6, 1] = 2, A[1, 7, 1] = 1, A[3, 6, 1] = 2, A[3 , 7, 1] = 1, A[6, 7, 1] = 1. Thus, after processing the 3rd transaction in the elements A[1, 3, 1], A[1, 6, 1], A[3, 6, 1] the value 2 is stored, which means that these combinations have met twice. And so on for all transactions.

At the next level, the three-dimensional array is filled in the same way. It is worth noting that now not all combinations are used, but only with index of the first element is greater than the level number.

As a result, we get an array whose values reflect the frequency of occurrence in transactions (table 2 – table 6). Horizontally it is the first index, vertically it is the second index.

TABLE 2 – THE FIRST LEVEL OF THE ARRAY

	1	2	3	4	5	6
2	2					
3	3	1				
4	2	3	2			
5	3	3	3	2		
6	5	2	5	3	4	
7	2	1	3	1	3	3

TABLE 3 – THE SECOND LEVEL OF THE ARRAY

	2	3	4	5	6
3	1				
4	2	1			
5	1	2	1		
6	2	3	3	4	
7	0	2	1	3	3

TABLE 4 – THE THIRD LEVEL OF THE ARRAY

	3	4	5	6
4	1			
5	1	1		
6	0	2	2	
7	0	0	2	2

TABLE 5 – THE FOURTH LEVEL OF THE ARRAY

	4	5	6
5	1		
6	1	1	
7	0	0	1

TABLE 6 – THE FIFTH LEVEL OF THE ARRAY

	5	6
6	1	
7	0	0

For example, if an element of an array with indices A[1, 6, 1] = 5, this means that the element {1, 6} will be a candidateset. If at the next level there are elements for which the first index is 6, for example, A[6, 7, 2] = 3, this means that there isn't a candidateset {1, 6, 7}, because the support is less than minimal support. And so on through the levels.

When the transactions are inserted in the array, we can start to extract them as candidatesets. To do this, we will generate combinations of unique elements of 2. In total, in the example there are 7 unique elements: 1, 2, 3, 4, 5, 6, 7. Based on them, the following two-element combinations will be generated: {1, 2}, {1, 3}, {1, 4}, {1, 5}, {1, 6}, {1, 7}, {2, 3}, {2, 4}, {2, 5}, {2, 6}, {2, 7}, {3, 4}, {3, 5}, {3, 6}, {3, 7}, {4, 5}, {4, 6}, {4, 7}, {5, 6}, {5, 7}.

We sort through all the values of the levels. At the first level, we go through all the generated combinations and check that the support is greater than or equal to the minimal. At the next levels, we go through all the selected candidates of the previous level, adding unique elements to them one at a time and checking support. Unique elements are considered only those that have index larger compared to the last element of the candidate. As a result, we get many candidates with minimal support. For example, on the first level we will find next itemsets: {1, 6}, {3, 5}, {3, 6}, {3, 7}, {5, 6}, {5, 7}. On the second level we will add to itemsets unique elements – {1, 6, 7}, {3, 5, 6}, {3, 5, 7}, {5, 6, 7}, but their minimal support is less than 0.4.

After the set of prospective candidatesets are selected, it is necessary to check them for occurrence in the transactions. If it is included, then increase support by 1. As a result, we get a list of rules. For example, itemset {1,6} is a part of transactions: {1, 2, 3, 4, 5, 6}, {1, 3, 6, 7}, {1, 2, 4, 6}, {1, 5, 6}, {1, 3, 5, 6, 7}. So its support count is 5 or support – 0.5. It's bigger than minimal support, so the itemset became the rule. In result we have 6 rules: {3, 6} with support count 5, {5, 7} with support count 4, {3, 7} with support count 4, {1, 6} with support count 5, {3, 5} with support count 4, {5, 6} with support count 4.

E. Algorithms comparison results

Comparison of the algorithms (3D2ARM, Apriori, FP-Growth) was carried by number of generated data. For 3D2ARM it was number of cells, for Apriori – number of candidatesets, for FP-Growth – count of tree nodes. We sign these parameters as y.

Several data sets were used, differing in the number of transactions, number of items. For testing data was generated with the Arules R library. We fixed the minimal support on value 0.1 and the number of transactions. To build a graph were generated 15 datasets with number of transaction 100 000 and number of unique elements from 10 to 50 000.

Based on the obtained data, an approximating function for dependence of the number of candidates on the number of unique elements was calculated for FP-Growth algorithm – hyperbolic regression:

$$y = 381916.06 - 3933100.52/m$$

Function for the 3D2ARM algorithm is $y = m^2 / 2$.

So, let's fix the minimal support 0.01, and build a graph (fig. 1). Here x is m.

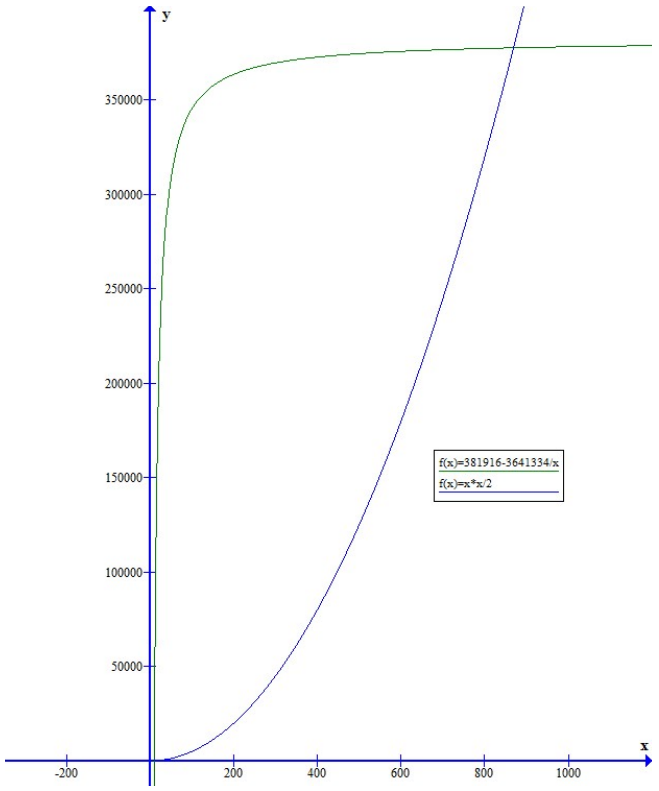

Fig. 1 – The graph for FP-Growth and three-dimensional algorithm

Thereis a certain interval three-dimensional algorithm has advantage. Data with number of unique elements from 10 to 868 the 3D2ARM algorithm is more effective.

Do this for Apriori, too. An approximating function for dependence of the number of candidates on the number of unique elements was calculated for Apriori algorithm – quadratic regression:

$$y = 0.08x^2 + 230.19x - 10362.64$$

So, let's fix the minimal support 0.01, and build a graph (fig. 2).

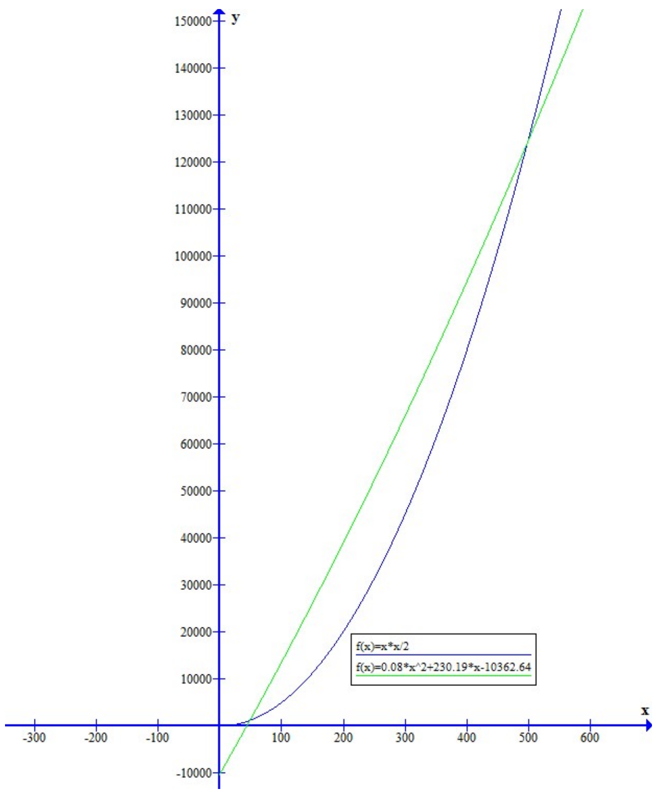

Fig. 2 – The graph for Apriori and the 3D2ARM algorithm

There is a certain interval three-dimensional algorithm has advantage. Data with number of unique elements from 49 to 498 three dimensional algorithm is more effective than Apriori.

VII. Possible enhancements

We plan to improve the algorithm.

First, at 2.3 step of algorithm. We denote the remaining number of transactions for analysis by the letter X. If the condition

$$candidateSupport + x < \min Support$$

is false, it means, that the candidate can't be "popular" at all, and we can delete it from our list of candidates, and not to check it for other transactions.

Second, the three-dimenional array is a sparse array in which most of the elements are zero. We plan to compress the array.

VIII. Conclusions

We have concluded the following:

1. We analyzed data structures in algorithms of association rules mining.
2. We discovered three-dimensional array can decrease memory usage and lead time of association rules mining (RQ1). We proposed the algorithm of association rule mining based on the array. We called the algorithm as three-dimensional array association rules mining algorithm (3D2ARM algorithm). The algorithm has only two passes through the transaction list.
3. The algorithm is the most effective, when number of unique items is from 10 to 868 comparing the FP-Growth algorithm and from 49 to 498 comparing the Apriori algorithm (RQ2).

The algorithm will be improved, by cutting off impossible candidates and squeezing the array.

References

[1] Agrawal R., Imielinski T., Swami A. Mining Association Rules between Sets of Items in Large Databases. SIGMOD Conference, 1993. (DOI: 10.1145/170035.170072)

[2] Brin S., Motwani R., Ullman J.D. Dynamic Itemset Counting and Implication Rules for Market Basket Data. SIGMOD'1997, «ACM Press», 1997. (DOI: 10.1145/253260.253325)

[3] Gallo G., Signorello G., Farinella G. M., Torrisi A.. Exploiting Social Images to Understand Tourist Behaviour. Image Analysis and Processing - ICIAP 2017: 19th International Conference, Catania, Italy, September 11-15, 2017, Proceedings – Springer, 2017, pp 707-717. (DOI: 10.1007/978-3-319-68548-9_64)

[4] Han, J., Pei, J., & Yin, Y. Mining Frequent Patterns without Candidate Generation. In Proc. ACM SIGMOD Intl. Conference on Management of Data, 2000, pp. 1-12. (DOI: 10.1145/342009.335372)

[5] Hristovski D. Supporting Discovery in Medicine by Association Rule Mining of Bibliographic Databases. Principles of Data Mining and Knowledge Discovery: 4th European Conference, PKDD, 2000, Lyon, France, September 13-16, 2000, pp 1344-1348.

[6] Piatetsky-Shapiro G. Discovery, analysis and presentation of strong rules. Knowledge Discovery in Databases. «AAAI Press», 1991.

[7] Prutzkow A. Algorithms and Data Structures for Association Rule Mining and its Complexity Analysis. In ICPE 2018 – International Conference on Psychology and Education, «The European Proceedings of Social & Behavioural Sciences EpSBS» Serie, 2018, pp. 558-568. (DOI: 10.15405/epsbs.2018.11.02.62)

[8] Srikant R., Agrawal R. Fast algorithms for Mining Association rules in large database. In VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases, 1994.

[9] Zaki M.J., Meira W. Data Mining and Analysis: Fundamental Concepts and Algorithms – Cambridge University Press, 2020.

[10] Zhang C., Zhang S. Association Rule Mining: Models and Algorithms – Springer, 2002. (DOI: 10.1007/3-540-46027-6)

[11] Zhang H., Zhao Y., Cao L., Zhang C. Combined Association Rule Mining. Advances in Knowledge Discovery and Data Mining: 12th Pacific-Asia Conference, PAKDD 2008 Osaka, Japan, May 20-23, 2008 – Springer Science & Business Media, 2008, pp 1069-1074. (DOI: 10.1007/978-3-540-68125-0_115)

[12] Billig V.A., Tsaregorodtsev N.A., Ivanova O.V. Programmnye produkty i sistemy. №2. Postroenie assotsiativnykh pravil v zadache meditsinskoi diagnostiki. – Tver, «Faktor i K», 2016. [In Rus]

[13] Khramshina E.O. Primenenie metoda poiska assotsiativnykh pravil v meditsine // Materialy IV Vserossiiskoi nauchnoi konferentsii molodykh spetsialistov, aspirantov, ordinatorov s Mezhdunarodnym uchastiem. Innovatsionnye tekhnologii v meditsine: vzgliad molodogo spetsialista. – Riazan, RiazGMU, 2018. – s. 7-9. [In Rus]