computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

ASSOCIATION RULES MINING IN BIG DATA

Nataliya Shakhovska [1], Roman Kaminskyy [1], Eugen Zasoba [1], Mykola Tsiutsiura [2]

[1] Lviv Polytechnic National University, S.Bandera str, 12, Lviv, 79013, Ukraine
nataliya.b.shakhovska@lpnu.ua
[2] Kyiv National University of Construction and Architecture, Kyiv, Ukraine
tsiutsiura.mi@knuba.edu.ua

Paper history:
Received 16 October 2017
Received in revised form 27 February 2018
Accepted 14 March 2018
Available online 31 March 2018

Keywords:
Big data;
association rule;
data dependency;
Apriori;
Complexity;
parallel processing.

Abstract: The paper proposes a method for Big data analyzing in the presence of different data sources and different methods of processing these data. The Big data definition is given, the main problems of data mining process are described. The concept of association rules is introduced and the method of association rules searching for working with Big Data is modified. The method of finding dependencies is developed, efficiency and possibility of its parallelization are determined. The developed algorithm makes it possible to assert that the task of detecting association dependencies in distributed databases belongs to the class of P-tasks. The algorithm for finding association dependencies is well-solved with MapReduce. The low asymptotic complexity of the developed association rules mining algorithm and a wide set of data types supported for analysis allow to apply the proposed algorithm in practically all subject areas working with association dependencies in the data domain.

1. INTRODUCTION

Today, various uncoordinated information resources processing (such as search, system integration, etc.) is the problem that often arises. For example, for university needs the integration process is the formation of scientific reports, the definition of indicators of success and quality of training, the formation of the ranking of the department, etc.; for local authorities one of the purposes is to determine critical indicators of the region's development based on data obtained from the state and non-governmental organizations.

The processing of various types of uncoordinated data has been carried out by researchers since 1970s. Models and metalanguages for working out different types of data have been developed. However, existing models and methods today relate only to pre-known types of data (mostly relational databases or XML data) solving only part of the problem of processing different types of data, for example, indexing to speed up the search. But NoSQL databases and availability of mostly semistructural information require new methods and tools for data processing [1].

Big Data information technology is the set of methods and means of processing different types of structured and unstructured dynamic large amounts of data for their analysis and use for decision support. There is an alternative to traditional database management systems and Business Intelligence solutions. This class attributes of parallel data processing are the following ones: NoSQL, algorithms MapReduce, Hadoop [2].

Big Data features are:

- unstructured and structured information processing;
- orientation on the fast data processing;
- inefficiency of traditional query language while working with data.

One of the adapting concepts, not only for relational data, is NoSQL language. The followers of the NoSQL concept emphasize that it is not a complete negation of SQL and the relational model. Their concept comes from the fact that SQL is important and very useful tool, which cannot be considered as universal. One issue for the classical relational database is a problem of dealing with huge data and high-load projects. The main objective of the approach is to extend the database if SQL is

flexible enough, and not to displace it wherever it performs its tasks.

The main ideas of the NoSQL are the following [2]:

- non-relational data model,
- distribution,
- open output code,
- good horizontal scalability.

Therefore, there is a need to manage discriminating information, namely, its presentation in the form understood by users (even if they do not know the specifics of the organization of the structures of this data source) and the processing (search, integration, extraction of new knowledge, etc.).

2. RELATED WORKS

In [2, 3] it is determined that multidimensional and object models are used to represent the Big data. The multidimensional view of the data is well used for data visualization and analysis tasks, but due to the hypercube dissipation, the amount of data in this case is bigger than the relational representation that is not acceptable to the Big data. Object representation allows to store objects in the form of attributes, their characteristics and relationships between characteristics. With some modifications, it can be used for Big data. However, the problem of the transformation of various types of data (text, semi-structured data) into an object model remains unresolved.

So, most of existing data analysis methods are not suitable for Big data because:

- the data size is extremely large;
- we don't know the type of data dependence;
- according to its structure and due to various origin sources, the data are unclear, there are many deviations and outliers;
- it is necessary to use parallel processing of data.

Association rules and rule generation are widely used and they are faced with many problems, the main of them is the availability of Big data and multivalued data sets [4]. The most popular algorithms for association rules mining are: Apriori, Apriori TID, Hybrid Apriori, FP-tree [7 – 14]. For small datasets the performance of Apriori is bigger than Apriori TID, but Hybrid Apriori can be used for big datasets too. FP-tree allows us to find another dependency than Apriori, because it creates relationships between levels in the tree.

However, it's very hard to use those algorithms for Big data analysis. For example, time complexity of Apriori is $O(d^2n)$, time complexity of FP-tree is $O(dn)$, where n is a number of records in dataset and d is a number of frequent item sets [19]. Single-processor systems with normal processor speed cannot handle such data volume, which makes the algorithm ineffective for the use. Recent developments in network technology and especially cloud platforms have provided some new ideas for generating multi-attribute rules using a parallel environment such as Hadoop [5]. MapReduce has been popular and more used to calculate a large amount of data since Google launched it on its platform. The distributed file system Google (GFS) and Amazon Web Services (AWS) use the Hadoop and MapReduce platform to provide their services [5].

In the case of generating association rules in MapReduce, Mapper is responsible for assigning various combinations of elements as "keys" and the "value" are used to track the number of inputs or sub-accounts of support. The task of the Reducer is to decrease the Mapper received per each key value and calculate the final support for all sets of candidate data.

In this way, association rules can be created with the maximum support and confidence.

The Apriori algorithm has big problems with large volumes of Big data, since it scans the entire database several times [6]. This means that the execution time is increased according to the number of transactions.

The purpose of the paper is to develop the association rules mining algorithm for Big data processing. To improve the Apriori algorithm authors proposed to use the both Spark and a hierarchical method for formulating rules.

3. PROPOSED METHOD FOR BIG DATA ANALYSIS

3.1. ASSOCIATION RULES DEFENITTION

Let's define the association rule given as dependence. We will search dependence on the relation r. This relation can be formed both for relational data sources and for non-relational (NoSQL) by forming a pair of values such as the name of the object and its characteristics.

That is why, the information model of Big data is triple [1]

$$BigD = \langle e, f, a \rangle, \quad (1)$$

where $e \in E$ is entity, $f \in F$ is characteristic, $a \to n_{e,f}$ is association between entity e and characteristic f.

The difference between relational model and model "entity-characteristic" is the association attribute $a \to n_{e,f}$ with a value ranging from 0 to 1.

The relational model is a subclass of the model "entity-characterization" with a value $a \to n_{e,f}$ equal to 1 for each entity and characteristics associated with it.

Let us r is a set of entities and characteristics with schema R. For different objects, the number of characteristics may be different. In this case, the ratio r will be formed as CROSS (r). The text below will only use the symbol r.

Association Dependence (AD) is a productive rule in the selection of the relation r, which is valid for a significant number of objects of this selection. The significance threshold must be determined by expert means, or output from the calculation of the probability of the false allocation of rule [7]:

$$\begin{aligned} F_I(S;T):(s;t) &= \left(r_S(R);r_T(R)\right), \\ i &= \overline{1..n_s}, \forall i: A_i \in R, \\ j &= \overline{1..n_T}, \forall j: A_j \in R, \\ s = r_S(R) &= \sigma_{S(\{A_i\})}\left(r(R)\right), \\ t = r_T(R) &= \sigma_{T(\{A_j\})}(s), \end{aligned} \quad (2)$$

where S, T are the predicates of the selection of the conditional and the resulting part, respectively, s, t are the results of selection operations for these predicates from the table. $S\left(\{A_i\}\right)$ means that predicate is used for the set of attributes $\{A_i\}$.

The level of trust (confidence) is the ratio of the number of objects presented in AD to the number of objects in the selection:

$$Conf(S \to T) = P(S \to T) = \frac{\left|\sigma_{S \wedge T}(r)\right|}{\left|\sigma_S(r)\right|}. \quad (3)$$

The level of support is the description of the predicate of the selection on the ratio calculated as the ratio of the number of objects satisfying the predicate P to the total number of objects in relation to

$$Supp(P) = \frac{\left|\sigma_P(r)\right|}{|r|}. \quad (4)$$

Level of improvement is calculated as the ratio of levels of trust and support to the AD:

$$Imp(S \to T) = \frac{Conf(S \to T)}{Supp(T)} = \frac{Supp(S \wedge T)}{Supp(S) \cdot Supp(T)}. \quad (5)$$

The level of support and the level of improvement are symmetrical, $Sup(X \to Y) = Sup(Y \to X)$ and $Imp(X \to Y) = Imp(Y \to X)$. The level of truth is ordered parameter, i.e. $Conf(X \to Y) \neq Conf(Y \to X)$.

3.2. ASSOCIATION RULES GENERATION

Let us build the method of AD mining [7].

Input data:

1. Relation CROSS(r), schema R is defined only for analysis selection.
2. Hash-function for each attribute in R: $h_j(A_j)$.
3. Threshold value of the confidence level of dependencies is sought as p_0. Instead of this parameter, the number of tuples for which the desired dependence must be determined can be used equally *minSupport*.
4. Threshold value of the confidence level of dependencies that are taken into account when new dependencies are formed = p^*.

Output data:

A set of association dependencies that meets the specified criteria: $\{S_i \to T_i\}, \forall i: Conf(S_i \to T_i) \geq p_0$

I. Data analysis:

1. Create a tree of hash tables of statistics for attribute value relationships. The hash table of each attribute A_j will match the values of this attribute to the structure (the number of occurrences of the value, the array of descriptions of the conditional values of other attributes). Description of the conditional values is a hash table of the attribute values A_k and the number of their repetitions on the set of tuples $X = R_{A_j = x_i[A_j]}$. The vertices of the even levels of the tree branch out by the name of the attribute on which the projection is carried out; odd levels are the value of the attribute of the parent's vertex. Hash tables of odd tree levels use predefined hash functions $h_j(A_j)$; for even levels, the internal hash function of comparing attributes to equality is used.

2. Fill in the data structure from step 1. Not the whole tree is fully filled up, but only branches corresponding to predefined criteria for the quality of dependencies. Additional refinement of statistics is possible using the principle of lazy calculations:

   a) For each tuple $x \in r(R)$;

   b) Initialize the current vertex of the statistics tree v with the root of the tree: $v = a$;

c) Initialize $startAttr = 0$ as number of attribute, from which we start branching of the tree;
d) For each attribute $A_j \in R, j \geq startAttr$;
e) If a branch of statistics tree does not exist: $v[j]\left[x\left[A_j\right]\right]$;
f) Create node $v[j]$ and $v[j]\left[x\left[A_j\right]\right]$;
g) Initialize attributes:
$v[j]\left[x\left[A_j\right]\right].count = 0$,
$v[j]\left[x\left[A_j\right]\right].childs = \varnothing$,
$v[j]\left[x\left[A_j\right]\right].nextAttrId = startAttr + 1$,
if $v.count > splitThreshold$;
h) Go to 2d.

Thus, the structure of data contains elemental dependencies $A_j = x_i\left[A_j\right] \rightarrow A_k = x_i\left[A_k\right]$, and also provides the ability to calculate the number of tuples of arbitrary projections $\left|\sigma_{A_{i_1}=v_{i_1} \wedge A_{i_2}=v_{i_2} \wedge \ldots \wedge A_{i_l}=v_{i_l}}(r)\right|$.

3. Declare a list of dependencies $z[l]\{S:set; T:set; NS:N; NT:N\}$. Each dependence is a structure with attributes: *S* is the set of values of the attributes of the conditional part of the dependence on which it is defined; *T* is the set of values of the attributes of the resulting part of the dependence on which it is defined; *NS* is the number of tuples for which the dependence is true; *NT* is a number of tuples for which the conditional part of the dependence is executed.

4. Add to the list all dependencies where $Conf\left(A_j = x_i\left[A_j\right] \rightarrow A_k = x_i\left[A_k\right]\right) \geq p_0$.

II. Generate dependencies from existing dependencies:

1. Copy the list Z to the list *Aggr*.
2. Create hash-table h_{aggr} as a set of values from created dependencies. This will allow you to effectively search for AD, which can be aggregated by each specific dependence d_{aggr}.
3. For each dependence $z_1 \in Aggr$.
4. Create dependence $z_3 = z_1 + z_2$.
5. If $Conf(z_3) \geq p^*$, add z_3 to the end of list Aggr for the next dependences aggregation.
6. Add dependence z_1 to the hash-table h_{aggr} with key $h\left(z_1[\Pr T]\right)$.

III. Generate new dependencies:

1. Create the list of dependencies *y*.
2. Initialize $Y = Aggr$.
3. Announce the hash table of the resulting parts of the predicate *pr* to effectively search for the set of dependencies having the same resultant part of the predicate.
4. For each dependence $F_I \in Y$ add AD F_I to the list of hash-table: $pr\left[h\left(F_I[\Pr T]\right)\right]$.

The constructed algorithm of association dependencies mining makes it possible to conduct an effective analysis of the same type of data for the presence of AD, the total complexity of which in time is

$$t = O\left(minSupport \cdot \left(\frac{n}{minSupport}\right)^{1-\log_{avgD}(m)} + \frac{Z_{al}^2}{m \cdot D(A)} + \frac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{m \cdot D(A)}\right) =$$
$$= O\left(minSupport \cdot \left(\frac{n}{minSupport}\right)^{1-\log_{avgD}(m)} + \frac{Z_{aggr}^2 \cdot \log\left(sz_{aggr}\right)}{m \cdot D(A)}\right).$$

The memory complexity of the algorithm is equal

$$M = O\left(M_{stat} + M_{aggr} + M_{ma}\right) =$$
$$= O\left(\left(\frac{n}{minSupport}\right)^{1+\log_{avgD}(m)} + Z_{aggr} \cdot sz_{aggr} + Z_{ma} \cdot sz_{ma}\right) =$$
$$= O\left(\left(\frac{n}{minSupport}\right)^{1+\log_{avgD}(m)} + Z_{ma} \cdot sz_{ma}\right).$$

The second component is larger than the third component similarly to time complexity. It is impossible to compare the first and third component in the equation in the general case because their size depends on the amount of data.

Low asymptotic complexity and different data types supporting allow us to apply the developed AD mining algorithm for different subject areas.

4. EXPERIMENTAL RESULTS

In order to compare the effectiveness of the developed data analysis method, the three of the most promising methods were selected: Apriori, HybridApriori and the FP-tree method [8, 9, 14].

The quantitative criteria for methods comparison show useful dependencies, the percentage of useful

dependencies in the set of discovered dependencies, the time of data analysis.

Useful dependence is the dependence on the data, which has a usefulness factor of not less than 0.5 when evaluated by the experts of the problem area.

$$\frac{\left(\frac{1}{n_e}\sum_{i=1}^{n_e} m_i\right) - markMin}{markMax - markMin} \geq 0.5, \quad (6)$$

where m is i assessment of expert, n_e is amount of experts, *markMin* is minimum value of the assessment scale, *markMax* is maximum value of the assessment scale.

The useful dependencies are dependencies which experts consider useful for f the subject area.

Obviously, in order to determine this assessment, it is imperative for experts to analyze all the dependencies found automatically. The need to involve experts at this stage (interpretation of results) is not a problem, since the task of automated data analysis tools is precisely to suggest assumptions about important regularities in the data, rather than judging the importance of the relationships found. The expert should interpret the results of the analysis.

The percentage of useful dependencies is the ratio of the number of dependencies that experts consider to be useful to the total number of dependencies found by the method. This criterion is very important, since it defines the efficiency of the experts in interpreting data. Obviously, the data analysis method, known to thousands of important dependencies, but "lost" among millions of obvious facts, will not be useful and cannot be practical [10].

The data analysis speed is relatively simple to measure the criterion and is determined by the time required by the computing system to execute the data analysis request.

We tested all methods on the same data set. The data set used for this application is the Adult data set in the Machine Learning Repository UCI [10]. It is widely used for association rules mining [10 – 11]. We compared built-in Apriori and FP-tree algorithms with the proposed method.

The data set contains about 2320000 observations with 14 variables. Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0)). Prediction task is to determine whether a person makes over 50K a year: >50K, <=50K:

1. age: continuous.
2. workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
3. fnlwgt: continuous.
4. education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.
5. education-num: continuous.
6. marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.
7. occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
8. relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
9. race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
10. sex: Female, Male.
11. capital-gain: continuous.
12. capital-loss: continuous.
13. hours-per-week: continuous.
14. native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

The results of visualizing the generated association dependencies are given in Fig. 1 and shown below.

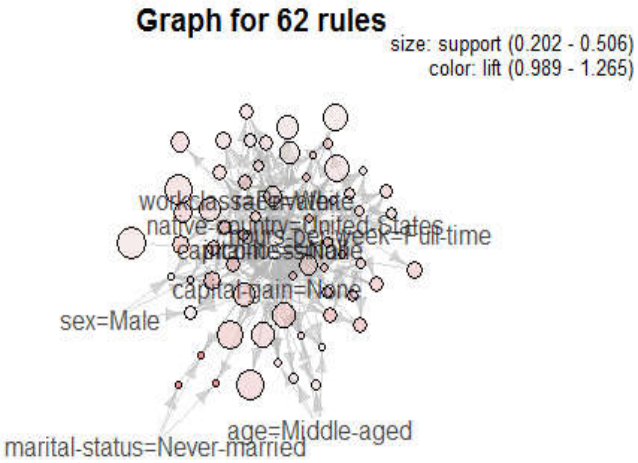


Fig. 1 – Visualization of association dependencies.

```
install.packages("arules")
library(arules)
library(Matrix)
data("Adult")
## find only frequent itemsets which do not contain  small or
large income
is <- apriori(Adult, parameter = list(support= 0.1,
target="frequent"),
        appearance = list(none = c("income=small",
"income=large"),
                          default="both"))
itemFrequency(items(is))["income=small"]

## find itemsets that only contain small or large income and
young age
is <- apriori(Adult, parameter = list(support= 0.1,
target="frequent"),
        appearance = list(items = c("income=small",
"income=large", "age=Young"),
                          default="none"))
inspect(head(is))
## find only rules with small or large income in the right-hand-
side.
rules <- apriori(Adult, parameter = list(support=0.2, confidence
= 0.5),
                 appearance = list(rhs = c("income=small",
"income=large"),
                                   default="lhs"))
inspect(head(rules))
install.packages("arulesViz")
library(arulesViz)
plot(rules, method="graph", control=list(type="items"))
plot(rules)
plot(rules, method="paracoord", control=list(reorder=TRUE))
```

The comparison of proposed methods and another association rules mining algorithms is done (Table 1). As it can be seen from the Table 1, the number of founded dependencies does not exponentially increase, depending on the size of the input data, as we would have guessed. Moreover, this grandfather seems to be limited above some boundary. This can be explained by the fact that the data analyzed was rather homogeneous. Therefore, in each subsequent block of data, more and more dependencies were repeated from previous blocks. These data respectively did not determine the generation of new dependencies, but increased only the level of support of existing data. The dependencies that match the search criteria were added to the resulting set.

Table 1. The number of useful dependencies found by different methods from the volume of the analyzed data.

Amount of records	Proposed method	FP-tree	Apriori	Hybrid-Apriori
20000	2856	2199	720	849
40000	5220	3627	888	1008
60000	6657	4530	1011	1158
80000	7656	5136	1053	1278
100000	8043	5274	1104	1329
125000	8184	5382	1125	1377

Another factor that could affect the number of dependencies was sorting the data by the date of birth of the driver. Thus, each subsequent block of data, in the meantime, introduced really new dependencies that could not be detected in the previous data blocks, depending on the age of the driver. Therefore, we can assume that the number of dependencies is unlimited from above with a clear boundary, but goes to a directly proportional hall with a fairly small angular coefficient of the number of tuples of data.

For data of practical value, the number of dependencies Z_{aggr} is no more than several thousand, while the number of tuples n can be calculated in billions. Therefore, in the case of parallel computing on a large number of computers, the second member of the function t_n can be neglected. For the same reasons $\log_2(n) = o(minSupport)$.

Thus, the asymptotic estimation of the execution time of the algorithm on a system with k computers is

$$t_n = O\left(minSupport^{-\log_{avgD}(m)}\right). \qquad (7)$$

The estimated time of execution of the algorithm is sub-polynomial, and therefore, the developed algorithm is an effective parallel algorithm.

The developed algorithm makes it possible to assert that the task of detecting association dependencies in distributed databases belongs to the class of P-tasks. So, the algorithm for finding association dependencies is well-solved with MapReduce [16, 17].

In addition to several successive implementations, parallel processing of Big data are not widely available. One example of serial implementation is the well-known statistical computation with the package R, called “arules”. Parallel implementation of the FP-Groth program is available in the library for studying the open source computer (MLlib) Apache Spark and Apache Mahout [18, 19].

Association rules are used widely in a large number of applications. A developed algorithm can be used in many cases when traditional algorithms are not viable because of the huge amount of processed data. This can be very beneficial, for example, in sensor networks that generate a huge amount of data at short intervals or in social networks which have millions of users. Moreover procedures of the developed algorithm can be generalized to other methods of data mining using the rules of the association, such as exceptions and

anomalies [11], gradual dependencies [12, 13, 20], and some others [21, 22].

6. CONCLUSION

The important scientific problem of associative dependencies mining in relational and NoSQL databases is solved in this paper. The received results allow us to improve the quality of decision-making in applied information systems by discovering new dependencies in the data.

A new method and algorithm for the associative dependencies mining were developed. It allowed obtaining a polynomial estimation of algorithm time complexity as well as carrying out the parallel execution of the algorithm on Hadoop systems using the map-reduce. It should be noted that the most suitable method among existing ones is the FP-tree method. The last one has exponential time-bound complexity estimation and requires memory that also exponentially depends on data attributes number. Moreover the FP-tree method is not designed for execution in distributed storage environments.

Comparison of the developed method with existing analogues (Apriori, FP-tree, Hybrid Apriori) showed its advantages, in particular: in 27% higher values of usefulness and time characteristics, a robustness to data errors, the possibility of algorithm parallelization and its execution in distributed databases, the autonomy work and the ability to operate in the streaming mode are observed. This affirms the prospect of wide spreading the proposed method in various domains.

7. REFERENCES

[1] N. Schahovska, "Datawarehouse and dataspace – information base of decision support system," in *Proceedings of the IEEE 11th International Conference on CAD Systems in Microelectronics (CADSM'2011)*, 2011.

[2] N. Shakhovska, M. Medykovsky, P. Stakhiv, "Application of algorithms of classification for uncertainty reduction," *Przeglad Elektrotechniczny*, vol. 89, no. 4, pp. 284-286, 2013.

[3] M. J. Zaki, "Scalable algorithms for association mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, issue 3, pp. 372-390, 2000.

[4] J. Han, J. Pei, Y. Yin, "Mining frequent patterns without candidate generation," in *ACM Sigmod Record*, pp. 1-12, 2000.

[5] J. Woo, "Apriori-Map/Reduce algorithm," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 2012, pp. 1.

[6] X. Y. Yang, Z. Liu, Y. Fu, "MapReduce as a programming model for association rules algorithm on Hadoop," in *Proceedings of the IEEE 3rd International Conference on Information Sciences and Interaction Sciences (ICIS'2010)*, 2010, pp. 99-102.

[7] R. Agrawal, T. Imieliński, A. Swami, "Mining association rules between sets of items in large databases," in *ACM Sigmod Record*, pp. 207-216, 1993.

[8] O. Yu. Pshenychnyj, "Data dependencies mining," *Mathematical Machines and Systems*, vol. 1, no. 1, 2012. (in Ukrainian).

[9] M. Delgado, M. D. Ruiz, & D. Sánchez, "New approaches for discovering exception and anomalous rules," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 19, issue 2, pp. 361–399, 2011.

[10] M. Hahsler, C. Buchta, B. Grün, K. Hornik, *arules: Mining Association Rules and Frequent Itemsets*. R package version 1.0-3., 2010, [Online]. Available: http://CRAN.R-project.org/.

[11] F. Berzal, et al., "A new framework to assess association rules," in *Advances in Intelligent Data Analysis*, Springer Berlin: Heidelberg, pp. 95–104, 2001.

[12] E. Hüllermeier, "Association rules for expressing gradual dependencies," in *Principles of Data Mining and Knowledge Discovery,* Springer, Berlin: Heidelberg, pp. 200–211, 2002.

[13] H. Srivastava, V. Kumar, S. Shiwani, "An efficient enhancement of mining top-K association rule," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, issue 6, June 2014.

[14] D. Hunyadi, "Performance comparison of Apriori and FP-Growth algorithms in generating association rules," in *Proceedings of the European Computing Conference*, 2011, pp. 376-381.

[15] A. O. Ogunde, O. Folorunso, A. S. Sodiya, "A partition enhanced mining algorithm for distributed association rule mining systems," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 297-307, 2015.

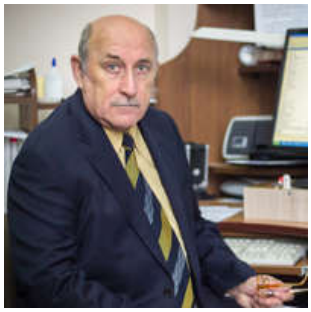
[16] R. Porkodi, B.L Shivakumar, "An improved association rule mining technique for xml data using Xquery and Apriori algorithm," pp. 1510-1514, March 2009.

[17] S. Rao, P. Gupta, "Implementing improved algorithm over Apriori data mining association

rule algorithm", *IJCST*, vol. 3, pp. 489-493, 2012.
[18] V. K. Shrivastava, P. Kumar, K. R. Pardasani, "FP-tree and COFI based approach for mining of multiple level association rules in large databases," *arXiv preprint arXiv:1003.1821*, 2010.
[19] K. Khurana, and S. Sharma, "A comparative analysis of association rule mining algorithms," *International Journal of Scientific and Research Publications*, vol. 3, issue 5, May 2013.
[20] N. Shakhovska, "Consolidated processing for differential information products," in *Proceedings of the IEEE VIIth International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH'2011)*, 2011.
[21] J. Chen, D. Dosyn, V. Lytvyn, A. Sachenko, "Smart data integration by goal driven ontology learning," in *Advances in Big Data. Proceedings of the 2nd INNS Conference on Big Data*, Thessaloniki, Greece, October 23-25, 2016, pp. 283-292.
[22] I. Perova, Y. Bodyanskiy, "Fast medical diagnostics using autoassociative neuro-fuzzy memory," *International Journal of Computing*, vol. 16, issue 1, pp. 34-40, 2017. Retrieved from http://computingonline.net/computing/article/view/869.

***Nataliya Shakhovska**, doctor of sciences, Prof., Head of Department of Artificial Intelligence, Lviv Polytechnic National University.*

Research interests: Big data mining, data-warehouses, intelligence systems.



***Roman Kaminskyy**, doctor of sciences, Prof., Professor of Department of Artificial Intelligence, Lviv Polytechnic National University.*

Research interests: Numerical optimization, Times series, intelligence systems.

***Zasoba Eugen,** assistant of Department of Artificial Intelligence, Lviv Polytechnic National University.*

Research interest: robotics, complexity of algorithms

***Mykola Tsiutsiura** has graduated from Kiev National University of Civil Engineering and Architecture. Now he is working as Associated Professor at Information Technologies Department of Kiev National University of Civil Engineering and Architecture.*

Research interests: Object-Oriented Programming; Software Design; Algorithmization and programming; Cloud technologies; IT project management; Standardization and certification in information systems; Models and methods of project management.