ASSOCRULE –Association Rule Mining Tool

Aboubekeur Hamdi-Cherif
Computer College
Qassim University
Buraydah, Saudi Arabia
And Université Ferhat Abbas – Setif 1
Setif, Algeria
shrief@qu.edu.sa

Chafia Kara-Mohamed (*alias* Hamdi-Cherif),
Hessa S. Alshammeri, Lubna M.N. Al-Sawaf,
Computer College
Qassim University
Buraydah, Saudi Arabia

***Abstract*— An implementation of an association rule mining system (*ASSSOCRULE*) is described. The Unified Modeling Language (UML©) is used in order to offer the possibility of modeling a shareable, continuously upgradable and easily-maintainable software system. For portability reasons, Java™ programming language is used for development, supported by a friendly graphical user interface (GUI). *ASSOCRULE* can be used as a roadmap for association rule mining systems development since it follows the software design process from UML© diagrams to testing stage. Contrary to most available systems, *ASSOCRULE* offers an open-source framework easy to use and amenable to eventual upgrade. It also constructs multiple-consequent rules instead of one-consequent rules, as offered by some similar platforms.**

***Keywords—component;* Association rules, Apriori algorithm, UML© design, Open-source software.**

I. INTRODUCTION

Association rule mining represents one of the major methods used in data mining - the set of techniques used for extracting meaningful data from datasets. Specifically, association rule mining consists in finding useful correlations between items in a dataset. For instance, a medical doctor wishes to know eventual correlations between symptoms and diseases, based on data obtained from previous patients. This approach of automatic extracting rules from dataset was initially motivated by the so-called market basket problem related to companies' effort in understanding clients' purchasing behavior; the objective being the targeting of market audiences in a more efficient manner. One of the possible solutions to this issue is to establish the relationship between goods that are bought together (items) and to represent this knowledge as IF-THEN rules expressed in plain text. In addition to the market basket problem, association rule mining have been applied to diversified fields such as clinical medicine, risk analysis in commercial environments, crime avoidance, epidemiology, among others. In brief, we can assume that association rule mining is applicable to practically any area in which the relationship between entities in a dataset can produce additional relevant knowledge in the form of IF-THEN rules.

On the complexity side, in a given dataset, the number of rules that can be constructed from a given dataset grows exponentially with the number of attributes in this dataset. Given n distinct items, the search space lattice provides 2^n possible combinations of items (formally including the empty set) to search for. Given that n is often large, brute force search techniques are often infeasible [1].

Obviously, not all rules that can be extracted from the dataset are useful. The essential features of usefulness recommended in the literature are that the rules are nontrivial, unexpected, novel, actionable and externally significant [2]. In our context, usefulness boils down to finding rules with both high support (coverage) and high confidence (accuracy); although other metrics are also used, elsewhere. By *support* is meant the proportion of transactions which contains the set of items (itemset). *Confidence*, on the other hand, gives a measure of the estimate of the probability that the consequent of the rule is true if its antecedent is true.

Association rule mining method presents two stages. The first one identifies the itemsets within the dataset and the second produces the rules from these itemsets. Since the efficient discovery of itemsets is significantly more complex than rule generation, the majority of contributions have focused on the first stage and not on the construction of the rules *per se* [3].

Because of their ubiquity, association rule mining ought to have more open-source software than available to date. User friendly, reusable, thoroughly designed, interoperable and flexible software systems are essential to bridge the gap between academic settings and practical association rule mining applications. On the other hand, most public platforms and all commercial systems are closed systems and hence drastically reduce the chance of reusability and/or upgradability by developers. To address these issues and further assist in the open-source effort, a system (called *ASSOCRULE*) is proposed. It adheres to rigorous state-of-the-art software engineering practices and attempts to present a shareable, continuously upgradable software implementation of one association rule algorithm, namely the Apriori algorithm, as a prelude to the inclusion of other algorithms [4].

ASSOCRULE can be considered as a roadmap for association rule mining systems development as it maps the software design process from UML© diagrams to testing [5]. It also offers multiple-consequent rules construction instead of one-consequent rules, as in some traditional platforms such as WEKA[1].

Next section defines the research and development contexts of the proposed system. Section 3 reviews the design and implementation steps followed. Section 4 is dedicated to test the system by presenting the results in a GUI. The paper ends with a conclusion highlighting the main contributions and describing some possible future enhancements.

II. *ASSOCRULE* BACKGROUND

A. Ubiquity of association rule mining

Since the seminal work presented in [4], the contributions have been growing over the last two decades, or so. At present, association rule mining are used in a wide range of industrial and scientific applications. On the other hand, it is a fact that the efficiency of association rules generation is affected by both the data structure used and the algorithms searching for frequent itemsets generation. In [6], the authors use a temporary root in the frequent pattern tree (FP-tree) to avoid the conditional FP-trees generation bottleneck. In [7], and for the same purpose, researchers used a stable table for generating the FP-trees. For speeding up the process of frequent items generation while freeing memory, we find [8], relying on aggregate chain, [9] using graph painting and [10] based on new header table structure.

B. Related algorithms and systems

The main algorithms for association rule mining are described in [11]. In addition to WEKA, the main platforms include, RapidMiner™[2], and KNIME™[3]. The main other systems include ARtool[4], ELKI and SPMF[5] are Java™ implementation of Apriori, Eclat and FPGrowth and other variations of Apriori. The arules[6] is an R package containing Apriori and Eclat. Orange[7] is an open-source data mining suite; with version 3.3 released in 2016.

C. ASSOCRULE motivation and characteristics

1) Development objectives

The present effort involves the design and development of an association rule mining system to strengthen the open-source trend. This tool lets users store datasets using a friendly graphical user interface (GUI) and apply it for mining association rules.

2) ASSOCRULE characteristics

ASSOCRULE characteristics are:

- *Software Standards: ASSOCRULE* is designed using state-of-art software engineering standard methods embodied by the Unified Modeling Language (UML©) which plays an important role in the readability of the software, its continuous improvement and its future maintenance.
- *Intuitive input GUI*: Availability of a GUI for introducing data and reading results.
- *Readability: ASSOCRULES* provides rules that are offered to the user in plain text.
- *Upgradability:* Developers are able to add the insertion of their own codes. Therefore *ASSOCRULE* is easily upgradable.
- *ASSOCRULE* is free*:* no license is required to use it.
- *Limitations:* At present, *ASSOCRULE* offers just one association rule mining algorithm. The expandability is possible since it is open-source.

III. *ASSOCRULE* DESIGN AND IMPLEMENATATION

A. Apriori algorithm implementation

Algorithm 1 represents steps of Apriori algorithm. It needs κ dataset traversals. It has two main parts, candidate generation and validation [11].

```
        //* Apriori Itemset Generation *//
1: κ = 0
2: repeat
3:      κ++
4:      if κ > 1 then
5:         {Cκ = generate_candidates(Vκ−1)}
6:         else
7:         {C1 = E}
8:      endif
9:      for all o do
10:         Co ={c | c ⊆ o ∧ c ∈ Cκ ∧ o ∈ O}
11:         for all Co do
12:            Ciκ.count++ | Co ∈ Cκ
13:         end for
14:     end for
15:     Lκ = {Cκ | Ciκ.count ≥ minsup}
16: until Vκ = ∅
17: return L = ⋃_{k=0}^{n} Lκ
```

Algorithm 1(b): Apriori Item Generation

The algorithm finds candidate itemsets C_κ of size κ from $V_{\kappa-1}$ for $\kappa > 1$, using support to reduce $|C_\kappa|$, and thus decreasing the exploration. Then, V_κ is found through a scan of the dataset, adding counts for each $c \in C_\kappa$. Therefore, given a *support* threshold *minsup*, $V_\kappa = \{C_\kappa : \sigma(C_\kappa) \geq minsup\}$.

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] https://rapidminer.com/

[3] http://www.knime.org/products/product-matrix

[4] http://www.cs.umb.edu/~laur/ARtool/

[5] http://www.philippe-fournier-viger.com/spmf/

[6] https://cran.r-project.org/web/packages/arules/index.html

[7] https://orange.biolab.si/

When $\kappa = 1$, the set of candidates is formed by the set of items, E. For $\kappa > 1$, the generation is done using a merge function comprising members of $V_{\kappa-1}$. Successive accrual gives the candidate itemset support. Those itemsets meeting the chosen threshold *minsup* are appended to the valid set $V\kappa$; with: $V_1 = \{e \mid e \in E \wedge \sigma(e) \geq minsup\}$.

Next, L_κ (frequent itemset of size κ) for $\kappa > 1$ is constructed on $V_{\kappa-1}$, with the addition of support. An itemset can only be valid if all subsets are valid. For instance, for a valid itemset a such that $a \in V$, all subsets of a exist in V. On this basis, C_κ is built from the constrained merge of $V_{\kappa-1}$ pairs where the pairs only differ by a single item, and all other $\kappa - 1$ permutations of the new itemset also exist in $V_{\kappa-1}$.

B. Overall architecture

1) UML© class diagram

The object-oriented approach in software development is generally used for interactive systems, such as *ASSOCRULE*. The UML© has become a standard for object-oriented language modeling and is therefore used as a modeling tool. A class diagram, in the UML© settings, is a static structural picture of the system describing the different classes, the attributes, and the various operations they can perform with the relationships between objects [5]. Figure 2 shows the UML© Class Diagram of *ASSOCRULE*.

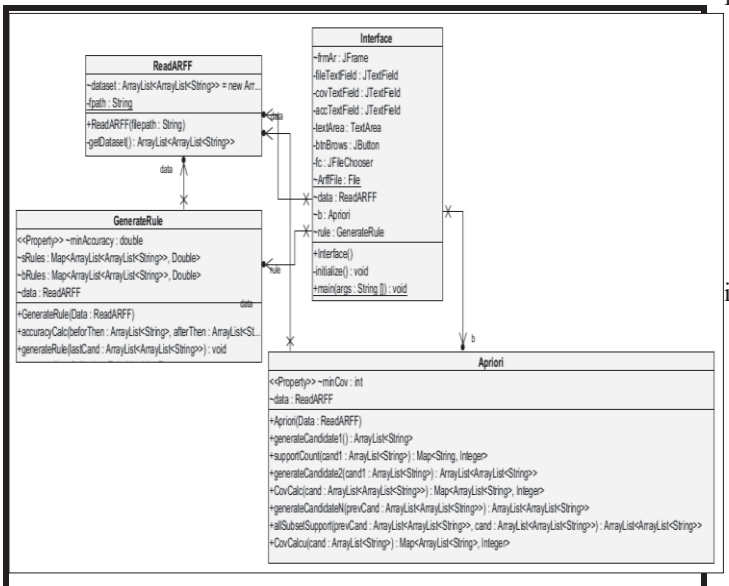

Figure 2 – *ASSOCRULE* class diagram

a) ReadARFF class

This class reads **`file.arff`** and store all attributes in order, using "**`ArrayList<String>.`**

b) Apriori class

This class calculates coverage and generates candidates. To generate **`candidate_1`**, take each item alone and check whether its coverage is at least equal to minimum coverage. To generate **`candidate_2,`** take **`candidate_1`** use a join operation in lexical order, then check its coverage. Recursively generate **`candidate_N,`** and check coverage.

c) Other classes class

`GenerateRule` is a class that that generates rules by calculating the coverage and accuracy while **`AssociationRuleInterface`** is a class that represents the GUI, connected to all other classes for input/output.

2) Java™ implementation

Figure 2(a) shows a screenshot of the classes as they appear in the implementation.

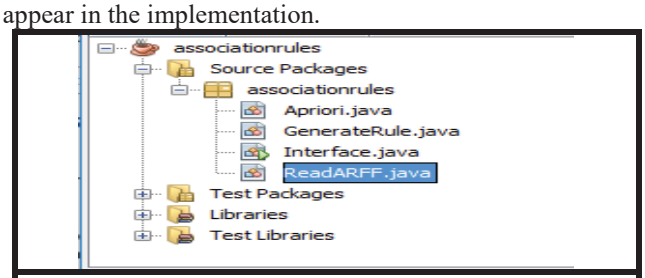

Figure 2(a) – Classes in *ASSOCRULE*

Figure 2(b) shows the class for reading the data. The format is ARFF (Attribute-Relation File Format).

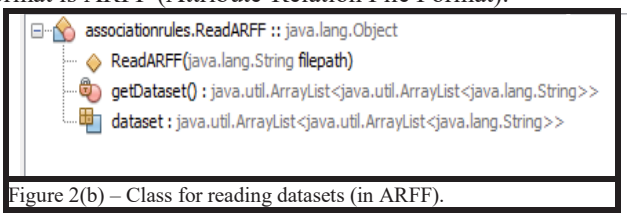

Figure 2(b) – Class for reading datasets (in ARFF).

Figure 2(c) shows the Apriori class as it appears in the implementation.

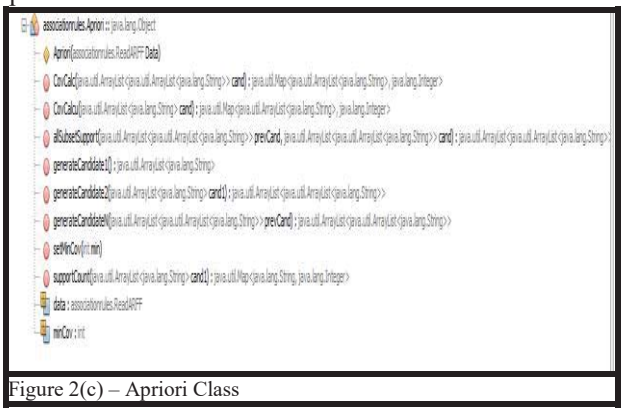

Figure 2(c) – Apriori Class

IV. *ASSOCRULE* TEST

A. *Interaction with ASSOCRULE*

For example, consider the weather dataset usually tested in [1]. There are 5 **<attribute:value>** pairs, namely **<Outlook: sunny, overcast, rainy>; <Temperature: hot, mild, cool>; <Humidity: normal, high>; <Windy: false, true>**; the target function is: **< Play: yes, no>.** The objective is to identify associations between different values on the basis of a labelled dataset.

a) Input Window

Figure 3 shows the input window, allowing the user to upload the dataset. **Choose File/Browse** for uploading. **Minimum Coverage** and **Minimum Accuracy** for selecting the user's values. **Run** for obtaining the rules.

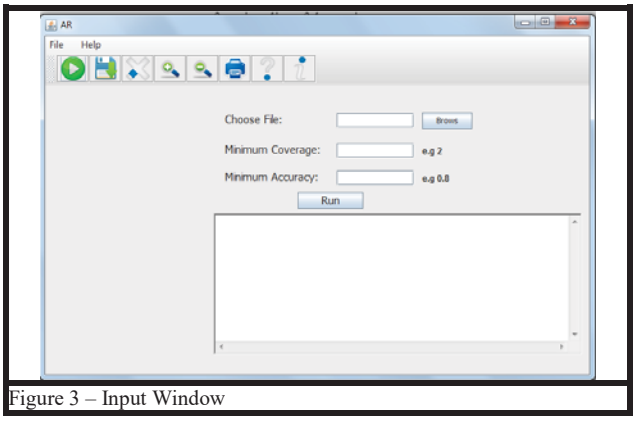

Figure 3 – Input Window

B. *Output in ASSOCRULE*

Figure 4 shows the **Output Screen** showing the results. Only part of the rules is shown. Note the rule 11 has two consequents (**Temperature=hot** and **Play=yes**).

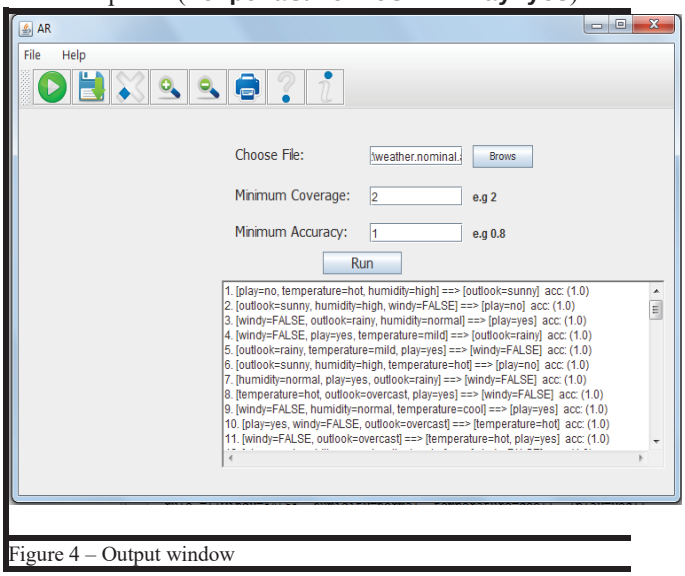

Figure 4 – Output window

Note that the same example run on WEKA will produce rules with one consequent only.

V. CONCLUSION

We have designed and developed *ASSOCRULE*; a novel association rule mining system tool. We have followed the important methodological software steps required in design and we have implemented this system from scratch. In this regard, *ASSOCRULE* offers a ready-made association rule mining method for solving practical problems in the same manner as WEKA and RapidMiner platforms. The main contribution is that *ASSOCRULE* is completely open and all internal components are readable and upgradable. The proposed system is ready for practical use to solve diversified association rule mining problems. As it stands now, the system implements one association rule algorithm but can be used as a blueprint for further developments. It is proposed to use *ASSOCRULE* and develop it further by integrating additional relevant approaches such as the mining of rulesets to make it more powerful.

VI. REFERENCES

[1] Witten, I.H., E. Frank and M.A. Hall, Data Mining Practical Machine Learning Tools and Techniques, 3rd Ed., Morgan Kaufmann Publishers, 2011.

[2] R.J. Hilderman and H.J. Hamilton, "Evaluation of interestingness measures for ranking discovered knowledge," In Proc. of the 5th Pacific-Asia Conf. on Knowledge Discovery and Data Mining, (PAKDD'01), Hong Kong, Vol. 2035. Springer, 247–259, 2001.

[3] S. Kotsiantis, D. Kanellopoulos. "Association rules mining: A recent overview," GESTS Int. Trans. on Comp. Sci. and Eng., 32(1):71-82, 2006

[4] R. Agrawal, R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," Proc. of the 20th Int. Conf. on Very Large Data Bases, VLDB '94, 1994, pp. 487-499

[5] O.M.G., Unified Modeling Language™, Version 2.4.1, Object Management Group, Inc.

[6] Y. Qiu, Y. Lan and Q. Xie, "An improved algorithm for mining from FP-Tree," in Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, 26-29 August 2004

[7] A. B. M. R. Islam and T. S. Chung, "An Improved Frequent Pattern Tree Based Association Rule Mining Technique," 2011 International Conference on Information Science and Applications, Jeju Island, 2011, pp. 1-8.

[8] J. Minghai, Y. Ping, J. Huiyan, "Research and Application on Web Information Retrieval Based on Improved FP-Growth Algorithm", WUJNS Wuhan University Journal of Natural Sciences, Vol. 11 No. 5, 2006, pp. 1065-1068.

[9] Yi Zeng, Shiqun Yin, Jiangyue Liu, and Miao Zhang, "Research of Improved FP-Growth Algorithm in Association Rules Mining," Scientific Programming, vol. 2015, Article ID 910281, 6 pages, 2015. doi:10.1155/2015/910281.

[10] L. Deng and Y. Lou, "Improvement and Research of FP-Growth Algorithm Based on Distributed Spark," 2015 International Conference on Cloud Computing and Big Data (CCBD), Shanghai, 2015, pp. 105-108.

[11] A. Ceglar, J.F. Roddick. "Association mining," ACM Computing Surveys (CSUR), vol. 38, no. 2, Article no. 5, 2006, pp.1-42.