Performance comparison of Apriori and FP-Growth algorithms in generating association rules

DANIEL HUNYADI
Department of Computer Science
"Lucian Blaga" University of Sibiu, Romania
daniel.hunyadi@ulbsibiu.ro

Abstract: In this article we present a performance comparison between Apriori and FP-Growth algorithms in generating association rules. The two algorithms are implemented in Rapid Miner and the result obtain from the data processing are analyzed in SPSS. The database used in the development of processes contains a series of transactions belonging to an online shop.

Key-Words: Apriori, association rules, data mining, FP-Growth, frequent item sets

1 Introduction

Having its origin in the analysis of the marketing bucket, the exploration of association rules represents one of the main applications of data mining. Their popularity is based on an efficiet data processing by means of algorithms. Being given a set of transactions of the clients, the purpose of the association rules is to find correlations between the sold articles. Knowing the associations between the offered products and services, helps those who have to take decisions to implement successful marketing techniques.

By means of the RapidMiner application we design several processes which generate frequent item sets, on the basis of which were then generated association rules. This article includes two processes, the first uses the Apriori algorithm and the second one uses the two algorithms *FP-Growth* and *Create Association Rules.*

Based on the obtained results and using the same work hypothesis and comparative statistical interpretations, we issued hypotheses referring to performance, precision and accuracy of the two processes created.

The article is organized as fallows: in section 2 we present Apriori algorithm, in section 3 we present the FP-Growth algorithm, in sections 4 we present two process developed for generating association rules, in section 5 we present the statistical interpretation of results and in section 6 we present conclusions of the research.

2 Apriori Algorithm

The first algorithm to generate all frequent item sets and confident association rules was the AIS algorithm by Agrawal et al. [1], which was given together with the introduction of this mining problem. Shortly after that, the algorithm was improved and renamed Apriori by Agrawal et al., by exploiting the monotonicity property of the support of item sets and the confidence of association rules [2, 7].

The items in transactions and item sets are kept sorted in their lexicographic order unless stated otherwise. The item set mining phase of the Apriori algorithm is given in Listing 1. I use the notation *X[i],* to represent the i^{th} item in *X*. The *k*-prefix of an item set *X* is the *k*-item set {*X[1], . . . ,X[k]*} [5].

Listing 1. Apriori algorithm – Item set mining

```
Input: D, minsupp
Output: F
  C1={{i}|i ∈ I};
  k=1;
  while Ck≠{} do{
    //Compute the supports of all
    //candidate itemsets
    forall transactions(tid,D) ∈ D
      forall candidate itemsets X ∈ Ck
        if ( X ⊆ I )
          X.support++;
    //Extract all frequent itemsets
    Fk = {X|X.support ≥ minsupp}
    //Generate new candidate itemsets
    forall X,Y ∈ Fk, X[i]=Y[i] for 1 ≤ i ≤ k-1,
      and X[k]<Y[k]{
      I = X ∪ {Y[k]};
      if (∀J ⊂ I,|J|=k, J ∈ Fk)
        Ck+1= Ck+1 ∪ I;
    }
    k++;
  }
```

The algorithm performs a breadth-first search through the search space of all item sets by

ISBN: 978-960-474-297-4

iteratively generating candidate item sets C_{k+1} of size $k+1$, starting with $k = 0$. An item set is a candidate if all of its subsets are known to be frequent. More specifically, C_1 consists of all items in I, and at a certain level k, all item sets of size $k+1$ are generated. This is done in two steps. First, in the *join* step, F_k is joined with itself. The union $X \cup Y$ of item sets $X, Y \in F_k$ is generated if they have the same $(k-1)$ - prefix. In the prune step, $X \cup Y$ is only inserted into C_{k+1} if all of its k-subsets occur in F_k.

To count the supports of all candidate k-item sets, the database, which retains on secondary storage in the horizontal database layout, is scanned one transaction at a time, and the supports of all candidate item sets that are included in that transaction are incremented. All item sets that turn out to be frequent are inserted into F_k.

If the number of candidate $(k+1)$ - item sets is too large to retain into main memory, the candidate generation procedure stops and the supports of all generated candidates is computed as if nothing happened. But then, in the next iteration, instead of generating candidate item sets of size $k+2$, the remainder of all candidate $(k+1)$ - item sets is generated and counted repeatedly until all frequent item sets of size $k + 1$ are generated.

3 FP-Growth Algorithm

In order to store the data base in the primary storage and to calculate the support of all generated sets of articles, the FP/Growth algorithm uses a combination between the horizontal model and the vertical model of a database. Instead of saving the boundaries of each element from the database, the transactions of the database are saved in tree structure and each article has a pointer attached towards all transactions containing it. This new data structure, named FP-Tree was created by Han et al. [4].

The FP Growth algorithm is presented in listing 2.

Listing 2. FP-growth algorithm

```
Input: D, minsupp, J ⊆ I
Output: F[J]
  F[J]={};
  forall i ∈ I occurring in D{
    F[J]=F[J] ∪ {J∪ {i}};
    //Create D^i;
    D^i={};
    H={};
    forall j ∈ I occurring in D such that j>I
        if (support(J∪ {i,j})≥minsupp)
          H=H ∪ {j};
    forall (tid,X) ∈ D with i∈ X
        D^i = D^i ∪ {(tid,X ∩ H)};
    //Depth-first recursion
    Compute F[J ∪ {i}];
    F[J]=F[J]∪ F[J ∪ {i}];
  }
```

In the first step, the root of the tree is created and is labelled with „null”. For each transaction from the database, the articles are processed in reverse order. Each node from the structure will further contain a counter which saves the number of transactions that have to deal with to that node. More precisely, if we consider that a branch must be added for a transaction, the counter of each node along the common prefix will be labelled with 1 and the node related to the articles from the transaction which follows the prefix are created and linked accordingly. Additionally, a table head is created for that article, so that each article points towards its appearances in the tree by means of several links. Each article from this table head will memorize the support of the article, too. The transactions are saved in the FP-tree structure in reverse order because the aim is to have a rather small tree size, the most frequent articles within the transactions being saved as close as possible to the root.

4 Developing a series of processes for generating associations

The first process uses the *Apriori* algorithm to determine the frequent sets and to generate association rules based on the frequent sets discovered. The process is presented in figure 1.

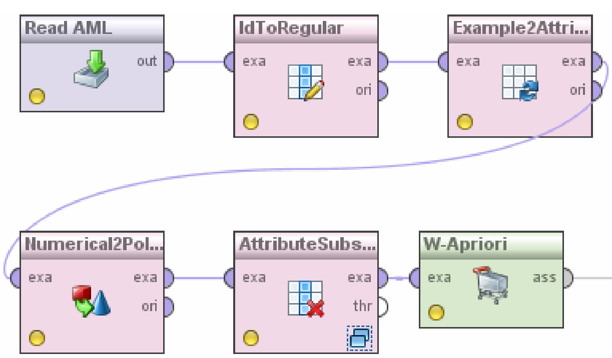

Fig. 1. Generating association rules by using the *W-Apriori* algorithm

The second process uses the *FP-Growth* algorithm to determine the frequent item sets and the *Create Association Rules* algorithm to generate association rules based on the frequent item sets discovered. The same data set was used as in the process presented in figure 1, namely the same values for minimum support and confidence. The process is presented in figure 2 [6].

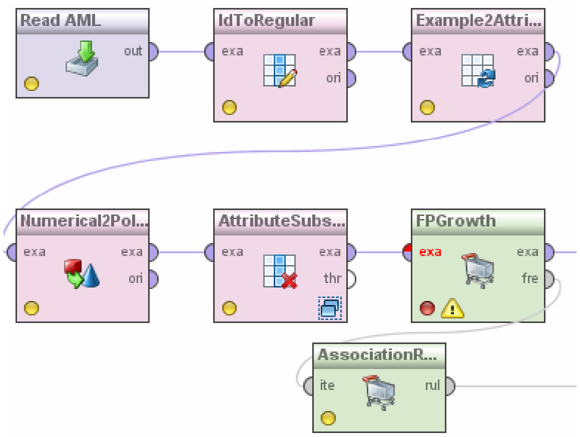

Fig. 2. Generating association rules using the *FP-Growth* and the *Association Rules* algorithms

The frequent sets were generated by means of the *FPGrowth* algorithm. This algorithm calculates all frequent item sets, building a FP-Tree structure from a database of transactions. The FP-Tree structure is a very compressed copy of data which are stored in the memory. All frequent sets of articles are obtained from this structure.

A major advantage of the algorithm *FP Growth* compared to others of the same type is the fact that it uses only two scans of the data and it can be applied to larger data sets. The frequent sets of articles are searched for positive entries from the data base. The entry data set must contain only bionominal attributes. If the data contains other types of attributes preprocessing operators must be used to transform the data set. The necessary operators are the transformation operators which change the type of values from numerical attributes into nominal attributes and then from nominal attributes into binominal attributes.

The association rules were generated by means of the *CreateAssociationRules* operator.

The rule trust degree was used as generation degree. In RapidMiner the process of exploitation of frequent sets is divided into two parts, first are generated all frequent sets of articles after which are generated the association rules from the frequent sets.

5 Statistical interpretations for comparing results

By means of statistical interpretations, were compared the results of the two generation processes of the association rules set previously developed, using the same entry data set and the same parameter values.

For data processing, the minimum support (*min_support*) took the values of *0.1* first and next of *0.15*, respectively of *0.2*, and the confidence in the generated rules (*min_confidence*) took values from the set (*0.1*, *1.0*). Based on all these premises was determined the number of associations resulted on each of the two processes built.

After the execution of the process developed through the *FPGrowth* and *CreateAssociationRules* algorithms, no matter of the variables *min_support* and *min_confidence,* were obtained more frequent sets than after the execution of the *Apriori* algorithm. The graphs in figure 3 represent the average of results of these algorithms in the case of different values of the variables *min_support* and *min_confidence.*

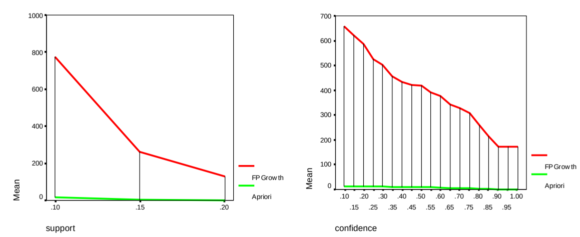

Fig.3 The average of the processed results

In figure 3 the medium values are much higher at using the *FPGrowth* / *CreateAssociationRules* algorithms than at using the *Apriori* algorithm

5.1 Distribution of values for the three values of the variable *min_support*

The statistical modeling requires to check for the state of normality of the used variables, this state being very important for the process of statistical inference. Thus, before performing the inference process, it is very important to determine whether the observed sample belongs to a normally distributed population, or not.

"*One Sample Kolmogorov-Smirnov Test*" is a formal method used to determine the distribution type of a variable (normal, uniform, exponential). Null hypothesis *H0* means „variable distribution is normal" and alternative hypothesis H1, „variable distribution differs from normal distribution".

For each of the three values of the variable *min_support* one can observe a normal distribution of the values *FPGrowth* / *CreateAssociationRules* (*p>0.05*) and a normal distribution of the values *Apriori* for *min_support* = 0.1. The distribution

ISBN: 978-960-474-297-4

differs from the normal one in the case of the *Apriori* values where *min_support*=0.15 or *min_support*=0.2.

support = .10

One-Sample Kolmogorov-Smirnov Test[c]

		FP Growth	Apriori
N		19	19
Normal Parameters[a,b]	Mean	774.7895	15.6316
	Std. Deviation	302.9306	9.3880
Most Extreme Differences	Absolute	.103	.153
	Positive	.103	.135
	Negative	-.084	-.153
Kolmogorov-Smirnov Z		.448	.666
Asymp. Sig. (2-tailed)		.988	.767
Exact Sig. (2-tailed)		.984	.712
Point Probability		.000	.000

support = .15

One-Sample Kolmogorov-Smirnov Test[c]

		FP Growth	Apriori
N		19	19
Normal Parameters[a,b]	Mean	260.4211	4.1579
	Std. Deviation	105.3625	2.4098
Most Extreme Differences	Absolute	.123	.357
	Positive	.105	.222
	Negative	-.123	-.357
Kolmogorov-Smirnov Z		.535	1.555
Asymp. Sig. (2-tailed)		.937	.016
Exact Sig. (2-tailed)		.904	.011
Point Probability		.000	.000

support = .20

One-Sample Kolmogorov-Smirnov Test[c]

		FP Growth	Apriori
N		19	19
Normal Parameters[a,b]	Mean	127.8947	1.4737
	Std. Deviation	46.1446	.7723
Most Extreme Differences	Absolute	.226	.384
	Positive	.226	.248
	Negative	-.137	-.384
Kolmogorov-Smirnov Z		.987	1.673
Asymp. Sig. (2-tailed)		.284	.007
Exact Sig. (2-tailed)		.244	.005
Point Probability		.000	.000

Fig. 4: *One Sample Kolmogorov-Smirnov Test* for the values *min_support*

The result of this test is interpreted according to the value *„Asymp". Sig (2-tailed)*" thus:

- if this value is smaller than 0.1, the test is 90% reliable, i.e. the null hypothesis can be rejected at a trust level of 90% (this means that the distribution of the variable differs significantly from the normal distribution)
- if this value is smaller than 0.05, the test is 95% reliable, i.e. the null hypothesis can be rejected at a trust level of 95% (this means that the distribution of the variable differs significantly from the normal distribution). This is the standard criterion;
- if this value is smaller than 0.01, the test is 99% reliable, i.e. the null hypothesis can be rejected at a trust level of 99% (this means that the distribution of the variable differs significantly from the normal distribution).

If the value "*Asymp. Sig (2-tailed)*" is higher than 0.05, null hypothesis is admitted, considering that the variable distribution is normal.

Before effectively applying this test we represented the histogram graph in the figures 5 and 6 for the results of the two techniques applied in the construction of processes for the three different values of the variable *min_support*.

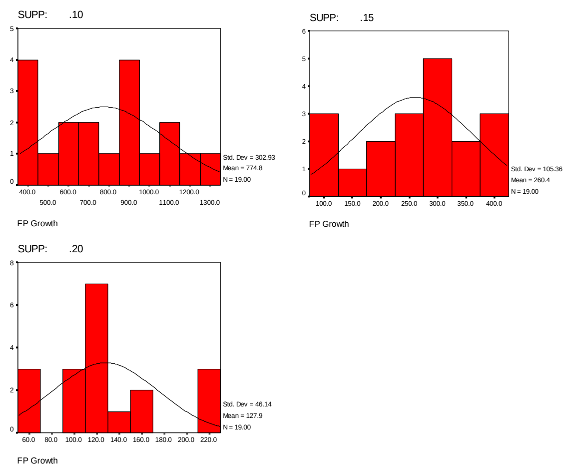

Fig. 5: The histogram for the results of the process using the ***FPG / AR*** technique for the *min_support* values

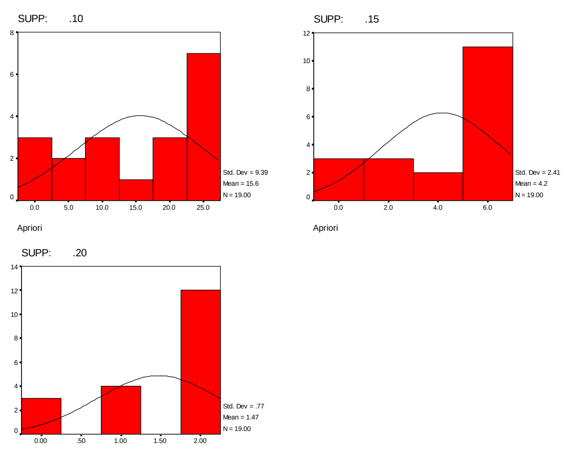

Fig. 6: The histogram for the results of the ***Apriori*** technique for the *min_support* values.

5.2 Comparison of the medium values of *FPGrowth/CreateAssociationRules*

"*Anova Test*" is a procedure applied to the independent samples (more than two samples with normal distribution) to verify if the average of several groups is equal.

It is considered null hypothesis *H0: "*there are no significant differences among the averages of the groups" and alternative hypothesis H1: "there is significant difference among the averages of the groups".

The results of this test are presented in two tables. The first table presents descriptive statistical elements of the variable for the two groups:

- number of cases
- averages
- standard deviations
- standard average error

The test results are interpreted according to the probability value "*Sig*", from the second table:

- if a value is smaller than 0.05, the test is 95% reliable, this means the null hypothesis can be rejected at a trust level of 95% (the difference between the average of the two groups is statistically significant).
- If a value is higher than 0.05, the null hypothesis is admitted: the difference between the averages of the two groups is not statistically significant.

Descriptives

FP Growth

	N	Mean	Std. Deviation	Std. Error	95% Confidence Interval for Mean Lower Bound	95% Confidence Interval for Mean Upper Bound	Minimum	Maximum
1.00	19	774.7895	302.9306	69.4970	628.7816	920.7974	357.00	1342.00
2.00	19	260.4211	105.3625	24.1718	209.6380	311.2042	98.00	424.00
3.00	19	127.8947	46.1446	10.5863	105.6538	150.1357	60.00	210.00
Total	57	387.7018	336.1324	44.5218	298.5138	476.8897	60.00	1342.00

ANOVA

FP Growth

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	4437202	2	2218601.175	63.390	.000
Within Groups	1889956	54	34999.177		
Total	6327158	56			

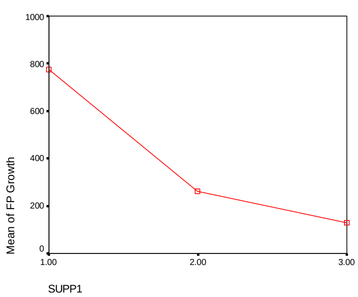

Fig. 7: *Anova* test results

In this situation one can observe a significant difference among the averages of the values *FPGrowth / CreateAssociationRules* considering the three values of the variable *min_support* (p<0.05).

5.3 Comparison of the medium values of the *Apriori* technique

If there are more than two independent samples, which do not have a normal distribution, the "*Kruskal-Wallis*" test will be used, the test results being interpreted according to the probability value "*Sig*", like the *Anova* test.

Here too, one can observe a significant difference among the average values resulted from the *Apriori* technique, regarding the three values of the *min_support* (p<0.05) variable.

Descriptive Statistics

	N	Mean	Std. Deviation	Minimum	Maximum
Apriori	57	7.0877	8.2922	.00	26.00
SUPP1	57	2.0000	.8238	1.00	3.00

Kruskal-Wallis Test

Ranks

	SUPP1	N	Mean Rank
Apriori	1.00	19	43.53
	2.00	19	28.82
	3.00	19	14.66
	Total	57	

Test Statistics[a,b,c]

	Apriori
Chi-Square	29.962
df	2
Asymp. Sig.	.000

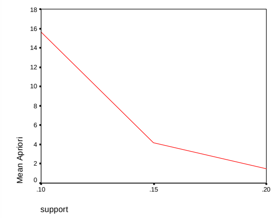

Fig. 8: *Kruskal-Wallis* test results

5.4 Correlations between the result values of the processes generated through the *FPGrowth/ CreateAssociationRules technique* and the *Apriori technique*

Interpreting the graph in figure 9 one can observe a significant correlation between the *FPGrowth / CreateAssociationRules* values and the *Apriori* values, i.e, when the Apriori values rise, the *FPGrowth / CreateAssociationRules* values (p<0.05) increase as well.

Correlations

		FP Growth	Apriori
FP Growth	Pearson Correlation	1.000	.970**
	Sig. (2-tailed)	.	.000
	N	57	57
Apriori	Pearson Correlation	.970**	1.000
	Sig. (2-tailed)	.000	.
	N	57	57

Descriptive Statistics

	Mean	Std. Deviation	N
FP Growth	387.7018	336.1324	57
Apriori	7.0877	8.2922	57

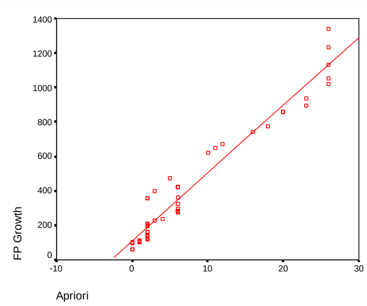

Fig. 9: Correlations between the *FPGrowth / CreateAssociationRules* values and the *Apriori* values

After applying the regression analysis, this relation will take the form of:

$$FPGrowth / CAR = 39.334 * Apriori + 108.991 \quad (1)$$

The definite relation in (1) indicates the fact that we can preview the result of the *FPGrowth / CreateAssociationRules* (*CAR*) algorithm if we know the result of the *Apriori* algorithm. More precisely, if the result value of the analysis is 50 for the *Apriori* algorithm, the result of the *FPGrowth/CreateAssociationRules* technique can be calculated according to the formula given below:

$$FPGrowth\ /\ CAR = 39.334 * 50 + 108.991 \quad (2)$$

A significant correlation between the values *FPGrowth/CreateAssociationRules* and the *Apriori* values exists also in the case of the three values of the variable *min_support,* with the observation that together with the rising of the values of the variable *min_support* the correlation becomes weaker.

support = .10

Descriptive Statistics[a]

	Mean	Std. Deviation	N
FP Growth	774.7895	302.9306	19
Apriori	15.6316	9.3880	19

a. support = .10

Correlations[a]

		FP Growth	Apriori
FP Growth	Pearson Correlation	1.000	.963**
	Sig. (2-tailed)	.	.000
	N	19	19
Apriori	Pearson Correlation	.963**	1.000
	Sig. (2-tailed)	.000	.
	N	19	19

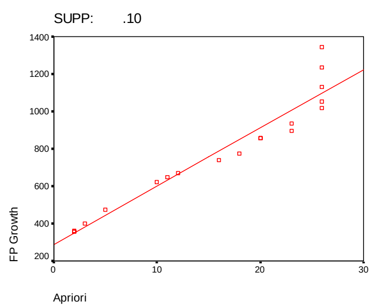

Fig 10: Correlations between the result values on the two processes for the minimum support 0.1

In the above situation, the equation of the regression line is following:

$$FPGrowth\ /\ CAR = 31.007 * Apriori + 289.004 \quad (3)$$

6 Conclusion

The association rules play a major role in many data mining applications, trying to find interesting patterns in data bases. In order to obtain these association rules the frequent sets of articles must be previously generated. The most common algorithms which are used for this type of actions are the Apriori (which generate both frequent sets and association rules) and the FP-Growth / Create Association Rules (FP-Growth generates frequent sets of articles, which are then used by Create Association Rules to generate association rules).

Although the Apriori algorithm processes data in a different manner from the algorithms *FPGrowth* and *Create Association Rules*, eliminating the sets of articles which are not frequent (with a minimum support smaller than the minimum support specified), there is a significant correlation ($p<0.05$) between the results of the generated processes through the respective algorithms, made evident through the regression line, in the case support independent, respectively through the regression lines, in the case of the three variants of the *min_support* values.

References:

[1] R. Agrawal, T. Imielinski, and T. Swami. Mining association rules between sets of items in large databases. In *Proc., ACM SIGMOD Conf. on Manag. of Data*, pages 207–216, Washington, D.C., 1993.
[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proc. Int. Conf. on Very Large Data Bases*, pages 478–499, Santiago, Chile, 1994.
[3] Craus M., Archip A., *A Generalized Parallel Algorithm for Frequent Itemset Mining*, Proceedings of the 12th WSEAS International Conference on Computers, Heraklion, Greece, 2008, pg. 520-523
[4] Han J., Pei J., Yin Y. and Mao R., *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*, Data Mining and Knowledge Discovery, 2003.
[5] Daniel Hunyadi, *Improvements of Apriori Algorithms*, First International Conference on Modelling and Development of Intelligent Systems – MDIS, October 22-25, 2009, Sibiu, Romania, "Lucian Blaga" University Publishing House
[6] Daniel Hunyadi, *Rapid Miner E-Commerce*, The 12th WSEAS International Conference on AUTOMATIC CONTROL, MODELLING & SIMULATION (ACMOS '10), Catania, Italy, 2010 pg. 316-321
[7] R. Srikant and R. Agrawal, Mining generalized association rules, 1999, pg. 407–419.

ISBN: 978-960-474-297-4