Grenze International Journal of Engineering and Technology, Jan Issue

Performance Analysis of Apriori and FP-Growth Algorithms on Online-Store Data

Bhavya Sharma[1] and Anuradha Rao[2]
[1]Student, I&CT department, Manipal Institute of Technology
Email: bhavya.sharma1505@gmail.com
[2]Faculty, I&CT department, Manipal Institute of Technology
Corresponding author: Email: anuradha.rao@manipal.edu

***Abstract*—Mining frequent item sets from transaction datasets is always being one of the most important problems of data mining. Apriori is the most popular and simplest algorithm for frequent itemset mining. Transaction data is a set of recording data resulting in connections with sales-purchase activities at a particular online store. In recent years transaction data has been used in research for discovering new information. One of the possible attempts is to design an application that can be used to analyze the existing transaction data. This paper focuses on the performance comparison of Apriori and FP-growth algorithms with respect to the execution time.**

***Index Terms*— Apriori, association rule mining, data mining, FP-growth, frequent item sets.**

I. INTRODUCTION

Data mining is the technique that finds hidden insights and unknown patterns from massive database, which are used as useful knowledge for decision making [1]. A well-organized data structure significantly reduces the time and space complexity. Apriori algorithm finds the frequent item sets by generating many candidate item sets. Candidates are the item sets containing all potentially frequent item sets. Data mining is used for "extracting knowledge from large amounts of data". Association rule mining (ARM) is one of the best one data mining technologies. ARM is a process for finding relations between data items in datasets. Association rule mining has been proven to be a successful technique for extracting knowledge. Frequent patterns are patterns item-sets that appear frequently in data set [2]. Data mining has long been used in relationship extraction from large amount of data for a wide range of applications such as consumer behavior analysis in marketing [3]. Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions. Knowing the associations between the offered products and services, helps those who must take decisions to implement successful marketing techniques [4].

This article uses python programming to design several processes which generates frequent item sets and hence association rules. Based on the implementation of two algorithms - Apriori and FP-growth, the time of execution for each is compared for different values of support and different sizes of the dataset.

The paper is organized as follows: In Section 2, we present the Apriori and FP-growth algorithms. In Section 3, design of the experimental study is presented with a preliminary experimental study that compares the performance between the two, on online store data. Finally, the paper is summarized with some concluding remarks and future research direction in Section 4.

Grenze ID: 01.GIJET.7.1.506_1

II. METHODOLOGY

Here we provide details on how python programming can be used to implement Apriori and FP-growth algorithms. The analysis shows the performance with respect to the time of execution with different values of support and different sizes of the dataset.

A. Apriori Algorithm

Apriori algorithm, also known as the level-wise algorithm, is the most popular algorithm to find frequent pattern sets. It uses the downward closure property. Before reading the dataset at each iteration, it prunes the sets which are not likely to be frequent item sets.

Downward Closure: It states that all subsets of a frequent pattern item-set will also be frequent item sets. For example, for a frequent item-set {x, y, z}, all subsets such as {x, y}, {x, z}, {y, z}, {x}, {y}, and{z} would all be frequent item sets under the same support threshold.

The first pass of the algorithm simply counts item occurrences to determine the frequent item sets. A subsequent pass, say k, consists of two phases. First, the frequent item sets L_{k-1} found in the $(k-1)^{th}$ step are used to generate the candidate itemset C_k. Next, the database is scanned and support for candidates in C_k are counted. Next, a pruning process ensures that all subsets in the candidate set are already known to be frequent item sets [5].

The prune function removes all item sets generated in the current pass, whose at least one subset is not part of the Frequent set in the previous pass.

gen_candidate_itemsets with the given L_{k-1}
    $C_k = \varphi$
    for all itemsets $l_1 \in L_{k-1}$ do
    for all itemsets $l_1 \in L_{k-1}$ do
        if $l_1[1] = l_2[2] \wedge l_1[2] = l_2[2] \wedge ... \wedge l_1[k-1] < l_2[k-1]$
        then $c = l_1[1], l_1[2], ..., l_1[k-1], l_2[k-1]$
        $C_k = C_k \cup \{c\}$

Prune (C_k)
    for all $c \in C_k$
    for all (k-1) subsets d of c do
        if $d \notin L_{k-1}$
        then $C_k = C_k - \{c\}$

Initialize k=1, C_1 = all 1-itemsets
read database to count support of C_1 to determine L_1
L_1 = {frequent 1-itemsets}
k=2
while (L_{k-1} != φ) do
begin
    C_k = gen_candidate_itemsets with given L_{k-1}
    Prune (C_k)
    for all transactions $t \in T$ do
        increment the count of all candidates in C_k that are in t
    L_k = All candidates in C_k with minimum support
    k= k+1
end
Answer: $\cup_k L_k$

B. FP-growth Algorithm

The FP-growth Algorithm, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). The FP-growth Algorithm is an alternative way to find frequent item sets without using candidate generations, thus improving performance. It uses a divide-and-conquer strategy.

In the first step, the algorithm considers all 1-itemsets and counts the support for each. The frequent 1-itemsets are arranged in descending order (as per support) in the L set. Now for each transaction in the database, the FP-tree is constructed recursively.

The frequent item sets are generated from the FP-tree, by first counting the number of paths available to each item in the tree. From these paths, we find subsets that are frequent (crossing minimum support) [5].

Input: A transaction database DB and a minimum support threshold.

Scan the transaction database DB once. Collect L, the set of frequent items, and the support of each frequent item. Sort L in support-descending order as F_{list}, the list of frequent items.

Create the root of an FP-tree, T, and label it as null. For each transaction Trans in DB do the following:

> Select the frequent items in Trans and sort them according to the order of L List. Insert this Trans in the FP-tree along with the count 1 to each item.
>
> If path already exists:
>
> > increment the count of the items in the path by 1.
>
> If two Transactions have common items at the beginning:
>
> > increment count of those items by 1 and then branch out to the remaining uncommon items into new nodes.

Perform the above steps recursively for all transactions in the DB

III. Results and Observations

A. About the Dataset

The data set used is the Groceries Market Basket Dataset. The dataset contains 9835 transactions by customers shopping for groceries. The data contains 169 unique items. Table I shows the sample groceries data with the list of items bought by customers.

TABLE I. SAMPLE DATA

	Item(s)	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6	Item 7	Item 8	Item 9	...	Item 23	Item 24	Item 25	Item 26	Item 27	Item 28	Item 29	Item 30
0	4	citrus fruit	semi-finished bread	margarine	ready soups	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	3	tropical fruit	yogurt	coffee	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1	whole milk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	4	pip fruit	yogurt	cream cheese	meat spreads	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	4	other vegetables	whole milk	condensed milk	long life bakery product	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9830	17	sausage	chicken	beef	hamburger meat	citrus fruit	grapes	root vegetables	whole milk	butter	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9831	1	cooking chocolate	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9832	10	chicken	citrus fruit	other vegetables	butter	yogurt	frozen dessert	domestic eggs	rolls/buns	rum	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9833	4	semi-finished bread	bottled water	soda	bottled beer	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
9834	5	chicken	tropical fruit	other vegetables	vinegar	shopping bags	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

B. Pre-processing

The pre-processing involved converting the transaction data frame to a binary (true/false) matrix as shown in the Table II. This was achieved using numpy and TransactionEncoder.

TABLE II. TRANSACTION DATA AFTER PRE-PROCESSING

	Instant food products	UHT-milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef	...	turkey	vinegar	waffles	whipped/sour cream	whisky	white bread
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
9830	False	False	False	False	False	False	False	False	False	True	...	False	False	False	True	False	False
9831	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
9832	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
9833	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False
9834	False	False	False	False	False	False	False	False	False	False	...	False	True	False	False	False	False

C. Frequent Item Sets

Apriori and FP-growth algorithms return frequent item sets. With 9835 transactions and 169 unique items in the database, equal representation of every item yields a support value of 60. Hence the support value for the comparison is set at 60 for both the algorithms.

a. Apriori

The apriori algorithm generates 747 frequent item sets, set sizes from 1 to 4. This process executes in 788 ms wall time as shown in Table III.

TABLE III. SAMPLE EXECUTION TIME OF APRIORI ALGORITHM

	support	itemsets	length
0	0.008033	(Instant food products)	1
1	0.033452	(UHT-milk)	1
2	0.017692	(baking powder)	1
3	0.052466	(beef)	1
4	0.033249	(berries)	1
...	...	...	...
742	0.010880	(whipped/sour cream, yogurt, whole milk)	3
743	0.006202	(root vegetables, other vegetables, whole milk...	4
744	0.007016	(root vegetables, other vegetables, whole milk...	4
745	0.007829	(yogurt, root vegetables, other vegetables, wh...	4
746	0.007626	(yogurt, whole milk, other vegetables, tropica...	4

747 rows × 3 columns

b. FP-growth

The FP-growth algorithm also generates 747 frequent item sets, set sizes vary from 1 to 2. This process executes in 282ms wall time as shown in Table IV.

TABLE IV. SAMPLE EXECUTION TIME OF FP-GROWTH ALGORITHM

	support	itemsets	length
0	0.082766	(citrus fruit)	1
1	0.058566	(margarine)	1
2	0.017692	(semi-finished bread)	1
3	0.139502	(yogurt)	1
4	0.104931	(tropical fruit)	1
...	...	...	...
742	0.007524	(whole milk, soft cheese)	2
743	0.007117	(other vegetables, soft cheese)	2
744	0.009964	(meat, other vegetables)	2
745	0.006914	(meat, rolls/buns)	2
746	0.009964	(meat, whole milk)	2

747 rows × 3 columns

D. Association Rules

Association rules are generated from frequent item sets, using a threshold metric-confidence. Confidence of a rule describes the chance to find the consequent in a transaction, given the antecedent is present. The confidence for the comparison here is set at 50%.

a. Apriori

The frequent item sets generated from Apriori, yield 67 association rules at a 50% confidence threshold. It takes 282ms wall time to execute the process. The sample execution result is as shown in Table V.

TABLE V. SAMPLE EXECUTION TIME OF APRIORI ALGORITHM

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(baking powder)	(whole milk)	0.017692	0.255516	0.009253	0.522989	2.046793	0.004732	1.560725
1	(beef, rolls/buns)	(whole milk)	0.013625	0.255516	0.006812	0.500000	1.956825	0.003331	1.488968
2	(yogurt, beef)	(whole milk)	0.011693	0.255516	0.006101	0.521739	2.041904	0.003113	1.556648
3	(brown bread, other vegetables)	(whole milk)	0.018709	0.255516	0.009354	0.500000	1.956825	0.004574	1.488968
4	(root vegetables, butter)	(other vegetables)	0.012913	0.193493	0.006609	0.511811	2.645119	0.004110	1.652039
...	...	...	...	...	...	...	...	...	...
62	(root vegetables, whole milk, tropical fruit)	(other vegetables)	0.011998	0.193493	0.007016	0.584746	3.022057	0.004694	1.942201
63	(yogurt, root vegetables, other vegetables)	(whole milk)	0.012913	0.255516	0.007829	0.606299	2.372842	0.004530	1.890989
64	(yogurt, root vegetables, whole milk)	(other vegetables)	0.014540	0.193493	0.007829	0.538462	2.782853	0.005016	1.747433
65	(yogurt, whole milk, tropical fruit)	(other vegetables)	0.015150	0.193493	0.007626	0.503356	2.601421	0.004694	1.623913
66	(yogurt, other vegetables, tropical fruit)	(whole milk)	0.012303	0.255516	0.007626	0.619835	2.425816	0.004482	1.958317

67 rows × 9 columns

b. FP-growth Tree

Association rules generated from the frequent item sets of FP-growth within a 50% confidence threshold. It yields 67 rules in 26.1ms wall time as shown in Table VI.

TABLE VI. SAMPLE EXECUTION TIME OF APRIORI ALGORITHM

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(citrus fruit, root vegetables)	(other vegetables)	0.017692	0.193493	0.010371	0.586207	3.029608	0.006948	1.949059
1	(citrus fruit, root vegetables)	(whole milk)	0.017692	0.255516	0.009151	0.517241	2.024301	0.004630	1.542145
2	(rolls/buns, margarine)	(whole milk)	0.014743	0.255516	0.007931	0.537931	2.105273	0.004164	1.611197
3	(yogurt, other vegetables)	(whole milk)	0.043416	0.255516	0.022267	0.512881	2.007235	0.011174	1.528340
4	(yogurt, tropical fruit)	(whole milk)	0.029283	0.255516	0.015150	0.517361	2.024770	0.007668	1.542528
...	...	...	...	...	...	...	...	...	...
62	(baking powder)	(whole milk)	0.017692	0.255516	0.009253	0.522989	2.046793	0.004732	1.560725
63	(other vegetables, frozen vegetables)	(whole milk)	0.017794	0.255516	0.009659	0.542857	2.124552	0.005113	1.628559
64	(root vegetables, frozen vegetables)	(whole milk)	0.011591	0.255516	0.006202	0.535088	2.094146	0.003241	1.601343
65	(root vegetables, frozen vegetables)	(other vegetables)	0.011591	0.193493	0.006101	0.526316	2.720082	0.003858	1.702627
66	(whole milk, onions)	(other vegetables)	0.012100	0.193493	0.006609	0.546218	2.822942	0.004268	1.777303

67 rows × 9 columns

```
CPU times: user 24.2 ms, sys: 4.02 ms, total: 28.2 ms
Wall time: 26.1 ms
```

E. Execution Time

The two algorithms are compared in-terms of their execution time on two fronts. Firstly, the change in the support value, ranging from 60 to 90. Secondly the size of dataset, starting from 5000 transactions to the complete database, 9835 transactions.

Fig. 1(a) shows the trend as the support value increases, the execution time of frequent item-set mining, reduces exponentially for Apriori dropping from >800ms to close to 400ms. FP-growth algorithm shows stability in execution time throughout the range, clocking between 300ms to 200ms.

Similarly, Fig. 1(b) shows the trend when compared on the size of data, the apriori execution times show a steady increase from close to 450ms to >700ms. Whereas FP-growth again shows stability between 200ms to 300ms.
Apriori and FP-growth algorithms when compared based on execution times, show opposite results. Change in support values and size of dataset have an impact on the execution times of Apriori, whereas FP-growth boasts stable results, with near equal execution time in all iterations.

IV. CONCLUSIONS

The association rules play a major role in many data mining applications, trying to find interesting patterns in databases. In order to obtain these association rules, the frequent sets of transactions must be previously generated. The most common algorithms are Apriori and FP-growth. Although both have different manner of processing the data, FP-growth shows stability and near equal execution times for change in support values and scale in data, made evident through the two visualizations. It can be thus concluded that FP-growth algorithm is invariant to scaling of data and change in support values. Future enhancements with respect to improve the performance of the Apriori algorithm and FP-growth may be checked for considering the various data structures used to store the data.

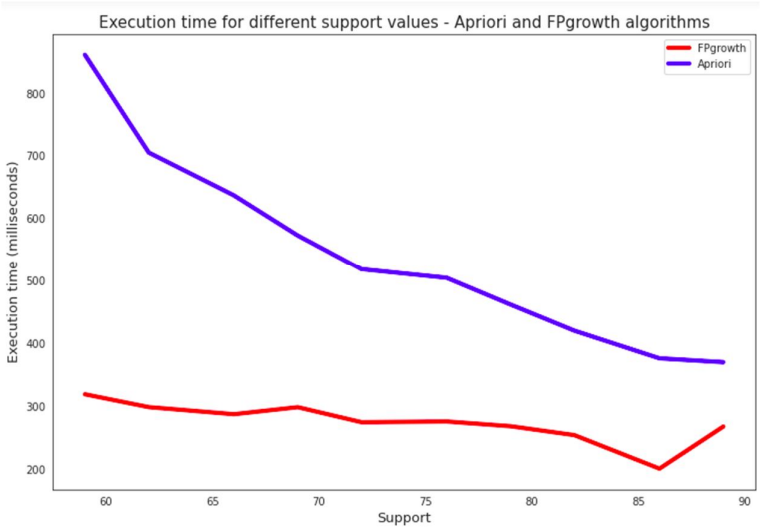

Figure 1(a). Variation in execution time with respect to support

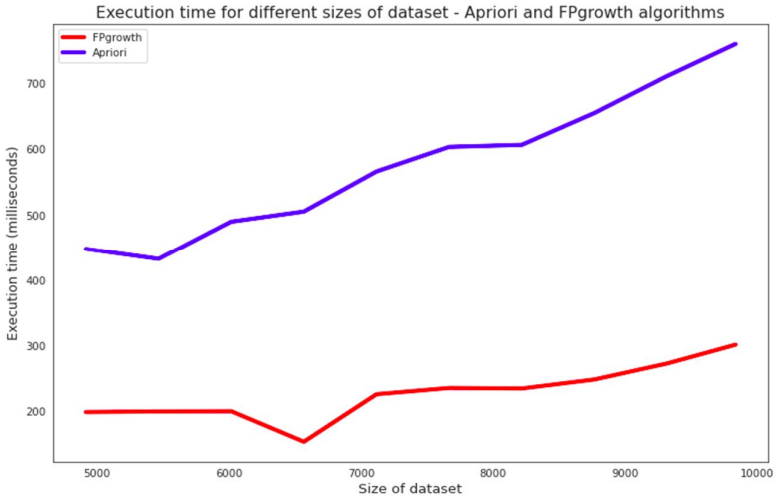

Figure 1(b). Variation in execution time with respect to size of dataset

REFERENCES

[1] Sudhakar Singh, Rakhi Garg, P.K. Mishra, "Performance Analysis of Apriori Algorithm with Different Data Structures on Hadoop Cluster", International Journal of Computer Applications (0975 – 8887) Volume 128 – No.9, October 2015

[2] Apurwa Sahu, Mradul Dhakar, Pushpi Rani, "Comparative Analysis of Apriori Algorithm based on Association Rule", International Journal of Computer Science and Communication, Volume 6, Issue 2, September 2015.

[3] Hau Ling Chan, King Wah Pang, Ka Wing Li, "Association Rule Based Approach for Improving Operation Efficiency in a Randomized Warehouse", Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management Kuala Lumpur, Malaysia, January 22 – 24, 2011.

[4] Fachrul Kurniawan, Binti Umayah, Jihad Hammad, Supeno Mardi Susiki Nugroho, Mochammad Hariadi, "Market Basket Analysis to Identify Customer Behaviors by Way of Transaction Data", Knowledge Engineering and Data Science (KEDS), Vol 1, No 1, January 2018, pp. 20–25.

[5] A. K. Pujari, "Data Mining", 2001.