Reframing in Frequent Pattern Mining

Chowdhury Farhan Ahmed*, Md. Samiullah†, Nicolas Lachiche*, Meelis Kull‡, Peter Flach‡
*ICube Laboratory, University of Strasbourg, France
farhan@du.ac.bd, nicolas.lachiche@unistra.fr
†Department of Computer Science & Engineering, University of Dhaka, Bangladesh
samiullah@cse.univdhaka.edu
‡Intelligent Systems Laboratory, University of Bristol, United Kingdom
{Meelis.Kull, Peter.Flach}@bristol.ac.uk

***Abstract*—Mining frequent patterns is a crucial task in data mining. Most of the existing frequent pattern mining methods find the complete set of frequent patterns from a given dataset. However, in real-life scenarios we often need to predict the future frequent patterns for different tasks such as business policy making, web page recommendation, stock-market behavior and road traffic analysis. Predicting future frequent patterns from the currently available set of frequent patterns is challenging due to dataset shift where data distributions may change from one dataset to another. In this paper, we propose a new approach called reframing in frequent pattern mining to solve this task. Moreover, we experimentally show the existence of dataset shift in two real-life transactional datasets and the capability of our approach to handle these unknown shifts.**

***Keywords*-Data Mining, Frequent Pattern Mining, Dataset Shift, Machine Learning, Adaptation.**

I. Introduction

Data mining techniques can discover hidden and potentially useful knowledge from databases. Frequent pattern mining [1], [2] plays an important role in data mining tasks such as association rule mining, classification, clustering, time-series mining, graph and web mining. By mining the frequent patterns, several important business area decisions like maximizing revenue, minimizing marketing and/or inventory costs can be made; and knowledge about the interesting customers/itemsets can be discovered. In addition to market basket data analysis, frequent pattern mining can also be useful in several real-life domains including web click streams, biological gene databases, social network databases, stock-market databases, road traffic data analysis, data feeds from sensor networks, telecommunication call records, and so on.

Most of the existing frequent pattern mining methods extract frequent patterns when the full dataset is available. It is a challenging task to predict the future frequent patterns based on the currently available set of frequent patterns since the future set of frequent patterns may vary due to dataset shift [3]. Dataset shift is a natural event in real-life datasets where data distributions and/or decision functions may change from one dataset to another [3]. None of the existing frequent pattern mining methods is applicable to predict the future frequent patterns based on the currently available set of frequent patterns under the presence of dataset shift. However, it is a very necessary task in data mining and machine learning as we often do not have enough data in several situations to discover the complete set of frequent patterns.

Consider a real-life scenario, where we have the transactions of a retail store in City-1 for the month of June and we move to City-2 in the beginning of July where we have very few transactions. According to the normal cases, some frequent patterns may be common between these two cities and some are not. For example, City-1 may have {bread}, {meat}, {milk}, and {bread, meat} as frequent patterns while City-2 may have {rice}, {meat}, {milk}, and {rice, meat} as frequent patterns. Therefore, it would always be a challenging problem in the domain of frequent pattern mining as mentioned earlier. In this situation, the existing methods would wait until the end of July for all the transactions of City-2 to arrive and then they mine the exact frequent patterns. But, we have some knowledge here such as the set of frequent patterns of another city of the previous month and a few transactions of this city for this month. If we could predict a good approximation in the beginning of the month with this available knowledge, it would be great beneficial for the businessmen of City-2 to make their business decisions/policies for that month. This real-life scenario motivates us to propose a new approach called reframing in frequent pattern mining. The main objective of reframing is to reuse the existing model/knowledge in several deployment contexts and handle dataset shift even though a small amount of knowledge is given at each deployment. To the best of our knowledge, our proposed approach is the first solution to predict future frequent patterns based on the currently available set of frequent patterns with the capability of handling dataset shift.

The main contributions of our approach are as follows

- We develop a new idea of reframing in frequent pattern mining, and design an efficient algorithm, called Reframing Frequent Patterns (RfmFP), to perform the task. The proposed RfmFP algorithm can discover a very good set of approximate frequent patterns even though only a few transactions are available in the deployment with the presence of dataset shift.

- The efficiency and effectiveness of the proposed approach have been shown experimentally. We present the existence of dataset shift in two real-life transactional datasets [4], [5] and capability of our algorithm to approximate these unknown real-life shifts accurately.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we describe our proposed reframing approach for frequent pattern mining. In Section 4, our experimental results are presented and analyzed. Finally, in Section 5, conclusions are drawn.

II. Related Work

The support/frequency of a pattern is the number of transactions containing the pattern in the transactional database. The problem of frequent pattern mining is to find the complete set of patterns satisfying a minimum support in the transactional database. The Apriori [1] algorithm is the initial solution of frequent pattern mining problem and very useful in association rule mining [1], [5]. However, it is based on the level-wise candidate generation-and-test mechanism which may create the problems of several database scans and huge candidate pattern generation. Han et al. [2] solved these problems by introducing a prefix tree (FP-tree)-based algorithm without candidate set generation and testing. This algorithm is called the frequent-pattern growth or FP-growth algorithm and needs only two database scans.

Some methods have been proposed to extend FP-tree to capture all the information with one database scan for incremental databases [6] and data streams [7]. These methods wait for the new parts/batches of transactions to be arrived completely before mining. Accordingly, they can mine frequent patterns in the new time periods when all the new transactions are available. Stated in other words, they are unable to predict the future frequent patterns based on the currently available frequent patterns. In order to generate a limited number of frequent patterns when the total number of frequent patterns becomes very high, several methods have been proposed which can approximate the actual frequent patterns. For example, closed [8] and maximal frequent patterns [9] to generate only the longest supersets of frequent patterns, Top-K frequent patterns [10] to generate the top-most K frequent patterns according to frequency, K-summarized patterns [11] to generate K representative frequent patterns, and approximate frequent patterns [12] in the presence of noise (i.e., absence/presence of an item in a transaction is altered by noise). However, these methods also need all the transactions to find the frequent patterns and are not applicable to predict the future frequent patterns based on some available knowledge. Moreover, they cannot handle dataset shift.

On the other hand, research areas such as transfer learning [13] and domain adaptation [14] proposed solutions to perform classification and regression tasks where training and test data follow different distributions. A score-based method [15] has been proposed to handle classification problem in different deployment scenarios. It is also possible to adapt regression outputs when the cost function changes. A tuning method has been proposed to adjust the average misprediction cost of a cost-sensitive regression model [16]. This method adds/subtracts a constant shift to all the predicted values of the original regression model. Research has also been done to tackle different kinds of dataset shift [3] such as covariate shift, prior probability shift and concept shift. An input transformation based approach, called GP-RFD, has been proposed recently to handle general dataset shift [17].

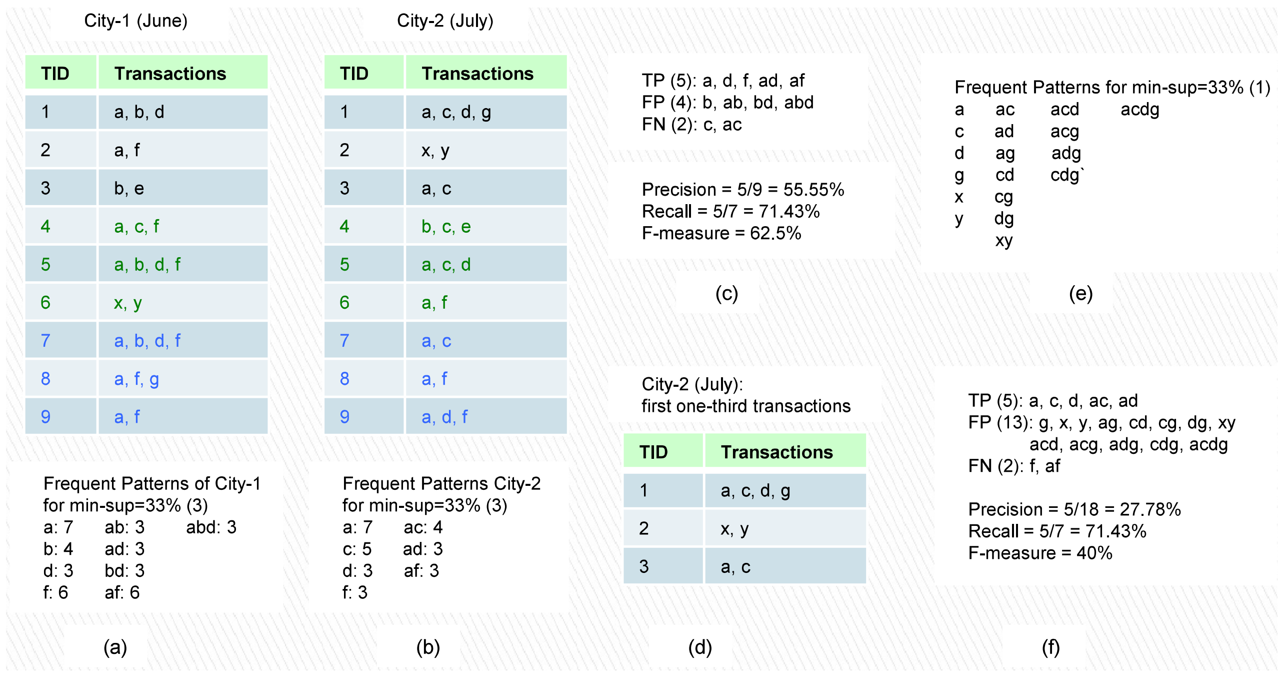
However, the existing dataset shift adapting approaches are not applicable for mining frequent patterns. And, none of the existing frequent pattern mining methods can predict the future frequent patterns based on the given knowledge under dataset shift. Therefore, in this paper, we propose a new approach called reframing in frequent pattern mining in order to solve this important and challenging task.

III. Our Proposed Approach

In this section, we present our proposed approach for reframing in frequent pattern mining. Consider a real-life scenario, where we have the transactions of a retail store in City-1 and City-2 for June and July, respectively. Fig. 1 (a)-(b) represents this example with the frequent patterns for these two cities for a minimum support threshold of 33% (3 out of 9 transactions). Note that dataset shift exists between these two datasets such as patterns b and ab are frequent in City-1, but infrequent in City-2. On the other hand, patterns c and ac are frequent in City-2, but infrequent in City-1. Now consider that we are in the beginning of July and want to predict the complete set of frequent patterns for City-2. Let us assume that we have one-third transactions (3 for this example) of the full dataset and obviously the complete set of frequent patterns of June from City-1. Before explaining our proposed reframing solution, we also propose two preliminary solutions called base and retraining. Here we report precision, recall and F-measure to show the performance of a method.

A. Base Model

Applying the base model means directly using the source model to test the deployment data. Here, if we directly predict frequent patterns of City-1 to be the frequent patterns of City-2, it can be stated that we are using the base model. Applying the base model does not require any knowledge from the deployment. In this example, the results of the base model is shown in Fig. 1 (c). Note that the true positives (TP), false positives (FP) and false negatives (FN) represent the frequent patterns common in two cities, frequent in City-1 but not in City-2 and frequent in City-2 but not in City-1,



City-1 (June)

TID	Transactions
1	a, b, d
2	a, f
3	b, e
4	a, c, f
5	a, b, d, f
6	x, y
7	a, b, d, f
8	a, f, g
9	a, f

Frequent Patterns of City-1 for min-sup=33% (3)

a: 7	ab: 3	abd: 3
b: 4	ad: 3	
d: 3	bd: 3	
f: 6	af: 6	

(a)

City-2 (July)

TID	Transactions
1	a, c, d, g
2	x, y
3	a, c
4	b, c, e
5	a, c, d
6	a, f
7	a, c
8	a, f
9	a, d, f

Frequent Patterns City-2 for min-sup=33% (3)

a: 7	ac: 4
c: 5	ad: 3
d: 3	af: 3
f: 3	

(b)

TP (5): a, d, f, ad, af
FP (4): b, ab, bd, abd
FN (2): c, ac

Precision = 5/9 = 55.55%
Recall = 5/7 = 71.43%
F-measure = 62.5%

(c)

City-2 (July):
first one-third transactions

TID	Transactions
1	a, c, d, g
2	x, y
3	a, c

(d)

Frequent Patterns for min-sup=33% (1)

a	ac	acd	acdg
c	ad	acg	
d	ag	adg	
g	cd	cdg`	
x	cg		
y	dg		
	xy		

(e)

TP (5): a, c, d, ac, ad
FP (13): g, x, y, ag, cd, cg, dg, xy
acd, acg, adg, cdg, acdg
FN (2): f, af

Precision = 5/18 = 27.78%
Recall = 5/7 = 71.43%
F-measure = 40%

(f)

Figure 1. Example transactional databases with resultant frequent patterns for a minimum support threshold of 33% (a) City-1 (June) (b) City-2 (July) (c) Performance of the base model of City-1 (June) over City-2 (July). (d)-(f) Performance of retraining when only one-third data of City-2 (July) is available.

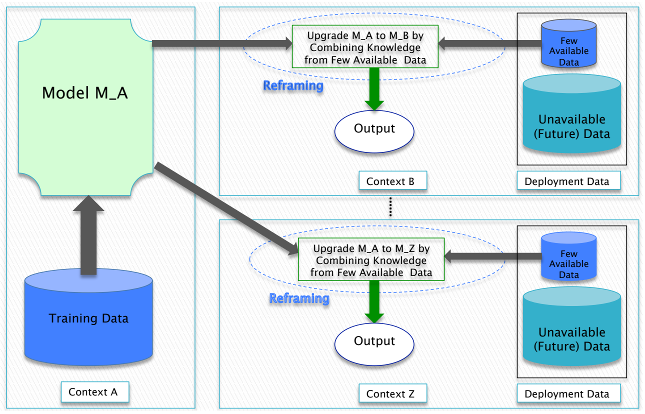

Model M_A
Training Data
Context A
Upgrade M_A to M_B by Combining Knowledge from Few Available Data
Reframing
Output
Few Available Data
Unavailable (Future) Data
Context B
Deployment Data
Upgrade M_A to M_Z by Combining Knowledge from Few Available Data
Reframing
Output
Few Available Data
Unavailable (Future) Data
Context Z
Deployment Data

Figure 2. The proposed framework for reframing in frequent pattern mining.

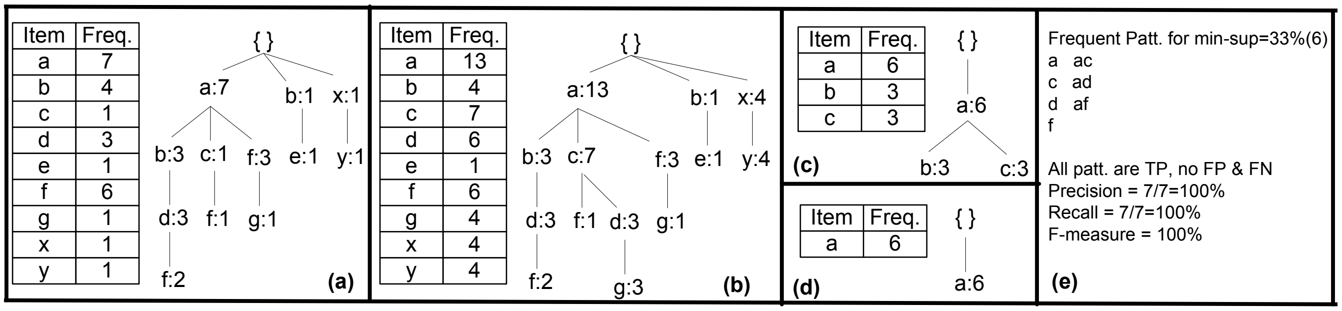
respectively. Therefore, the base model indicates the level of dataset shift between source and deployment.

B. Retraining

When we have few transactions in a deployment, another straightforward way is to predict the whole set of frequent patterns based on that knowledge. For example, here we have one-third transactions of City-2 for the month of July and we might predict the complete set of frequent patterns of July based on those available transactions. We refer this process as retraining in frequent pattern mining.

Note that retraining does not care about the knowledge from other sources. Accordingly, it goes for the mining operation with the small amount of transactions available with the given threshold. Moreover, it does not try to learn anything for the missing transactions. Performance of retraining for this example is shown in Fig. 1 (d)-(f). It is obvious in retraining method that too many frequent patterns are generated for the same minimum support threshold as the total number of transactions is limited here. The outcome of this example reveals that retraining predicts most of the patterns occurring in that small part as frequent. Hence, it misses a frequent pattern that does not appear in that part or has a very low frequency value. By doing so, it naturally achieves a reasonable recall value. On the other hand, its precision becomes very poor as it generates too many false positives. For this reason, its F-measure is also not good in most of the cases. In this example, even though retraining achieves the same recall (71.43%) of the base model, its precision is very poor (27.78%) as well as the F-measure (40%).

In real-life datasets, frequent patterns may occur regularly or irregularly. For example, frequent pattern {milk, bread} may appear regularly throughout the year, but frequent pattern {trousers, T-shirt} may appear in some particular time slots. In our example of Fig. 1 (a)-(b), patterns a, b and ab are regular in City-1 and patterns a, c and ac are regular in City-2 (their occurrences and frequencies are almost regular in all the three parts of a month). On the other hand, patterns f and af are frequent in both City-1 and City-2, but they are not regular. In City-1, observe that they appear only once in



Item	Freq.
a	7
b	4
c	1
d	3
e	1
f	6
g	1
x	1
y	1

Item	Freq.
a	13
b	4
c	7
d	6
e	1
f	6
g	4
x	4
y	4

Item	Freq.
a	6
b	3
c	3

Item	Freq.
a	6

Frequent Patt. for min-sup=33%(6)
a ac
c ad
d af
f

All patt. are TP, no FP & FN
Precision = 7/7=100%
Recall = 7/7=100%
F-measure = 100%

Figure 3. (a) Constructed tree for the City-1(June) dataset of the RfmFP algorithm (b) Constructed tree of RfmFP algorithm after combining the one-third available deployment transactions (each transaction is considered three times) of City-2 (July) (c) Prefix-tree of item d (d) Conditional-tree of item d (e) Mining performance.

part-1 (one-third transactions), and twice and thrice in part-2 and part-3, respectively. In City-2, this type of behavior continues for these patterns. They (f and af) do not occur in part-1, once in part-2 and twice in part-3. Retraining cannot detect these frequent patterns in City-2 as they did not appear in part-1. This indicates the inability of retraining to discover irregular frequent patterns even though it generates too many false positives.

C. Reframing

In most of the real-life scenarios we have some source of knowledge to build a model and several deployment scenarios where we might have little knowledge. If we rely only on the source knowledge and do not care about the little knowledge given in a deployment, we will not be able to detect a possible dataset shift in that deployment. This is the case of the base model described above. On the other hand, if we only consider the small knowledge of that deployment, we might overestimate some patterns and miss some general behavior of the complete data which can easily be guided from the rich knowledge of the neighbours/other similar places. The outcome of retraining described above illustrates this phenomenon. In order to combine the both, we define reframing to reuse the source model at each deployment with a possible adaptation for any dataset shift using the small given knowledge in each deployment.

The framework of our approach is presented in Fig. 2. Note that the original source model M_A is reused in several deployment contexts after upgrading with the few available deployment data for that context. In the running example of this paper, we can build a frequent pattern mining model for City-1 and upgrade it with the few available transactions in City-2. In this way, we can reuse the rich knowledge (learnt over the full June month in another city) of City-1 as well as incorporate the small knowledge given in City-2 (or in other cities) for tackling a possible dataset shift.

Our proposed algorithm for reframing in frequent pattern mining is called Reframing Frequent Patterns (RfmFP). We have used a prefix-tree structure [2], [6], [7] for capturing the transactions of a dataset in order to mine frequent patterns. At first, the tree structure is created from the source dataset of City-1 (shown in Fig. 3 (a)). Adjacent links are maintained like FP-tree here but not shown in the figure for simplicity. We have sorted the transactions in lexicographic order before inserting them into the prefix-tree in order to facilitate the incremental insertion of transactions. For the convenience of the presentation, we have presented our datasets (Fig. 1 (a)-(b)) here in lexicographic order.

When we get some deployment data (one-third transactions of City-2 for this example), we can insert these new transactions in the source tree easily. It is a research issue that how much importance should be given to the new transactions. In this example, we present the upgraded tree by giving equal importance to both source and deployment transactions. Note that City-1 has 9 transactions and we have 3 transactions of City-2 up to the current time. If we consider each available transactions in City-2 three times, then it will be equivalent to consider total 9 transactions in City-2. Now, if we insert these 9 transactions (actually we insert each available 3 transactions with a frequency value of 3) into the tree, it will contain total 18 transactions considering 9 transactions from each city. Here a pattern has to occur 6 times to be frequent for a 33% minimum support threshold.

The reframed tree structure is shown in Fig. 3 (b). We can now perform pattern growth mining operation [2] from this tree. We have to create prefix and conditional trees for each frequent item. For illustration, prefix and condition trees of item d are shown in Fig. 3(c) and Fig. 3(d), respectively. To create the prefix-tree of item d, all branches prefixing d are taken with its frequency value. Subsequently, a conditional-tree of d is created from the prefix-tree by eliminating the nodes containing infrequent items with d. Here items a, b and c appear in the prefix-tree, but items b and c have frequency value less than min-sup (6). So, items b and c are eliminated in the conditional-tree of d. We get frequent patterns ad and d from here. As this conditional tree has only one item, it will not be divided recursively into any further prefix and conditional trees. Other frequent patterns can also be found by creating the prefix and conditional trees from the other frequent items in the same way. The resultant frequent patterns of RfmFP are shown in Fig. 3(e) with mining performance. It is noticeable that RfmFP can discover all actual frequent patterns for this example. It detects dataset shift in City-2 and discover patterns c and ac by weighting the deployment transactions (missed by the base model). It also discovers frequent patterns f and af by taking the knowledge from City-1 (missed by the retraining). As patterns f and af are highly frequent (have frequency value of 6) in City-1, they appear in the final result even though they are totally absent in the 3 given transactions in City-2. On the other hand, RfmFP successfully removes false positive patterns related to item b (generated by the base model) and others (generated by retraining) as they have low frequency values in the reframed tree. As a result, RfmFP achieves a complete balance performance to achieve a very good precision and recall at the same time. So, its F-measure is also very good.

In the described reframing example shown in Fig. 3, half of the transactions (9 out of 18) originate from training and another half from deployment, achieved by making multiple copies of each transaction in deployment. We denote by SR the split ratio between deployment and training, this is the ratio of the numbers of transactions in deployment and training. We also denote the number of copies of each transaction for deployment by MF (multiplicative factor). In this example, SR=1/3 and MF=3. We denote the weight value of deployment transactions by W, which is actually the proportion of transactions used from deployment in the combined dataset of source and deployment data. In order to achieve more expressivity, we represent W within the range from 0 to 1. Hence, the proportion for training data is 1-W. Accordingly, W=0 is equivalent to the base model, W=1 is equivalent to retraining and W=0.5 is equivalent for the given example in Fig. 3.

However, it is possible to consider other proportions (W values) of deployment and training data. The question is, what should be the value of MF to get the desired W? Our desired ratio between deployment and training is W/(1-W) and the original ratio is SR. Therefore,, we have to multiply SR by MF to make it equal to the desired ratio. Equation 1 represents the value of MF in terms of W and SR. Note that in our experiments we use such values of W for which MF can be integer as the number of transactions should be an integer value. Equation 2 can be derived from Equation 1 and it will calculate the value of W between 0 to 1. For a given SR, the more we increase MF, W will be more closer to 1. We assume W=1 when MF is infinity.

$$MF = \frac{W}{SR \times (1 - W)} \tag{1}$$

$$W = \frac{MF \times SR}{1 + (MF \times SR)} \tag{2}$$

IV. Experimental Results

We have performed several experiments on two real-life datasets to show the efficiency and effectiveness of our approach. As mentioned earlier, none of the existing frequent pattern mining methods investigate the dataset shift problem. Therefore, it is also an important task to show the existence of dataset shift in real-life datasets. Accordingly, we report the existence of dataset shift in these two real-life datasets at first. Subsequently, we show the performance analysis of the RfmFP algorithm with respect to the base model and retraining.

A. Occurrences of Dataset Shift in Real-Life Transactional Datasets

We present the occurrences of dataset shift in the following two real-life transactional datasets.

- **BMS-WebView-1:** The BMS-WebView-1 dataset contains several months worth of click stream data from one e-commerce web site. Each transaction in this dataset is a web session consisting of all the product detail pages viewed in that session. That is, each product detail view is an item. The goal of this dataset is to find associations among products viewed by visitors in a single visit to the website. It contains 59,602 transactions with 497 distinct items. The average transaction size is 2.5. The dataset [5] is available in SPMF [18]: an open source data mining library[1].
- **Retail:** The dataset Retail is provided by Tom Brijs, and contains the retail market basket data from an anonymous Belgian retail store [4]. It contains 88,162 transactions and 16,470 distinct items. Its mean transaction size is 10.3. The dataset is available in Frequent Itemset Mining Dataset Repository[2].

As the above real-life datasets are collected over time, dataset shift might naturally occur if we split the datasets sequentially. In order to check the assumption, we are going to compare a sequential split to a random split (shuffled transactions). We split the BMS-WebView-1 dataset into two equal random partitions denoted by R-1 and R-2, where R-1 is used as training and R-2 is used as test. We report the average results of 10 runs for these random partitions using different minimum support thresholds in Table I. Note that the base model of R-1 can achieve a good performance in R-2, for example, it achieves around 95% precision, recall and F-measure with min-sup=0.5%.

Subsequently, we split the BMS-WebView-1 dataset into two equal sequential partitions. The first (P-1) and second (P-2) partitions are used as training and test, respectively. Table II shows the performance of the base model of P-1 over P-2 with the same minimum support thresholds used in the random partitions. The performance presented here is quite

[1]http://www.philippe-fournier-viger.com/spmf/index.php
[2]http://fimi.ua.ac.be/data/

Table I
PERFORMANCE (AVERAGE OF 10 RUNS) OF THE BASE MODEL OF R-1 OVER R-2 IN THE BMS-WEBVIEW-1 DATASET WHERE THESE PARTITIONS (EQUAL) HAVE BEEN CREATED RANDOMLY.

min-sup (%)	TP	FP	FN	Precision (%)	Recall (%)	F-measure (%)
0.6	157.2	7.2	7.5	95.638	95.473	95.533
0.5	193.3	11.5	11.2	94.417	94.58	94.456
0.4	262.6	21.8	23.1	92.41	91.982	92.126

Table II
PRESENCE OF DATASET SHIFT IN THE BMS-WEBVIEW-1 DATASET. IT IS DIVIDED INTO TWO EQUAL PARTITIONS (SEQUENTIAL) AND PERFORMANCE OF THE BASE MODEL OF P-1 OVER P-2 HAS BEEN SHOWN.

min-sup (%)	TP	FP	FN	Precision (%)	Recall (%)	F-measure (%)
0.6	124	36	60	77.5	67.391	72.093
0.5	151	52	89	74.384	62.917	68.172
0.4	206	67	120	75.458	63.19	68.781

Table III
PERFORMANCE (AVERAGE OF 10 RUNS) OF THE BASE MODEL OF R-1 OVER R-2 IN THE RETAIL DATASET WHERE THESE PARTITIONS HAVE BEEN CREATED RANDOMLY.

min-sup (%)	TP	FP	FN	Precision (%)	Recall (%)	F-measure (%)
0.6	382	40.1	35.5	90.531	91.508	90.996
0.5	527.9	55.5	55	90.504	90.567	90.526
0.4	765.6	84.1	80.6	90.11	90.476	90.286

different than the performance of the random partitions. For example, if we consider 0.5% min-sup, we get only 74.384% precision, 62.917% recall and 68.172% F-measure. These results clearly show the presence of remarkable dataset shift between these two sequential partitions.

In the next experiment, we also divide the Retail dataset randomly and sequentially. The results of random and sequential splits have been presented in Table III and Table IV, respectively. Note that we take random partitions (shuffled transactions) of the same size as the sequential partitions. These results also indicate that remarkable dataset shift can only be found in the sequential partitions.

The results presented in this subsection clearly show that dataset shift exists in these two real-life transactional datasets. Moreover, it is revealed that large dataset shift can only be observed in these datasets according to sequential partitions which indicates the natural change of customer behavior over time.

B. Performance Analysis

We compare the performance of RfmFP algorithm with the base model and retraining on the real-life BMS-WebView-1 and Retail datasets. According to the results of Section IV-A, we use sequential partitions of these datasets. The first partition P-1 is used as source and other remaining partitions are used as deployment. First, we show the comparison results on the BMS-WebView-1 dataset in Fig. 4 with different weight values and split ratios. We use F-measure (Y-axis) to present this comparison in order to show the combined effect of precision and recall. The X-axis represents W values between 0 to 1. Several values of W including 0.5 (equal weights to both source and deployment data), 0.67 (2/3 weight to deployment data and 1/3 weight to source data) and 0.33 (1/3 weight to deployment data and 2/3 weight to source data) have been reported. Three different figures have been shown to present the results of three different SR values. Note that performance of the base model remains the same when we vary W or SR, as it does not use any knowledge from the deployment. For a particular SR, the performance of retraining remains the same when we vary W as retraining does not use any knowledge from the source. However, its performance increases when SR increases as the availability of deployment data is also increasing.

The performance of RfmFP is affected by both W and SR. When W=0, it acts like a base model as no importance is given to the deployment data. On the other hand, RfmFP acts like retraining when W=1 as this time no importance is given to the source data. Fig 4 shows that the performance

Table IV
PRESENCE OF DATASET SHIFT IN THE RETAIL DATASET. IT IS DIVIDED INTO FOUR EQUAL PARTITIONS (SEQUENTIAL). PERFORMANCE OF THE BASE MODEL OF P-1 OVER P-2, P-3 AND P-4 HAVE BEEN SHOWN.

min-sup	Partitions	TP	FP	FN	Precision (%)	Recall (%)	F-measure (%)
0.6	P-2	347	148	135	70.101	71.992	71.034
	P-3	265	230	128	53.535	67.43	59.685
	P-4	323	172	267	65.253	54.746	59.539
0.5	P-2	483	176	181	73.293	72.741	73.016
	P-3	373	286	177	56.601	67.818	61.704
	P-4	440	219	363	66.768	54.795	60.192
0.4	P-2	669	315	302	67.988	68.898	68.44
	P-3	522	462	290	53.049	64.286	58.129
	P-4	637	347	510	64.736	55.536	59.784

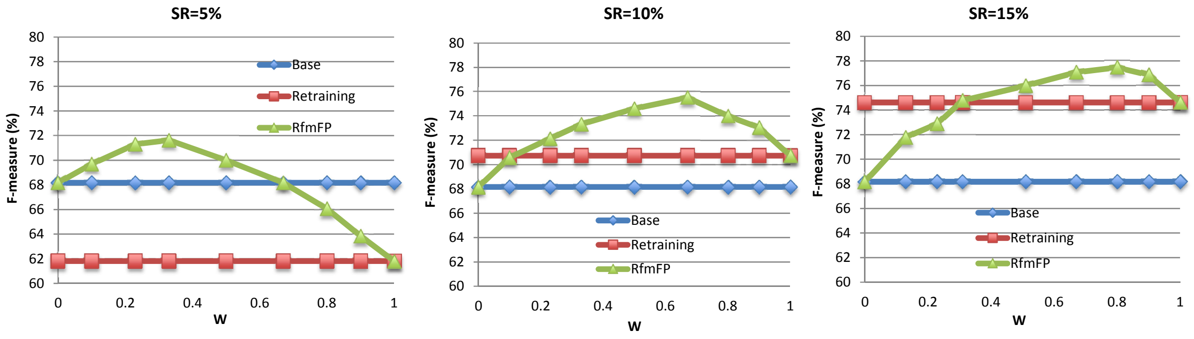


Figure 4. Performance comparison (min-sup = 0.5%) on the BMS-WebView-1 dataset with different weights (W) and split ratios (SR).

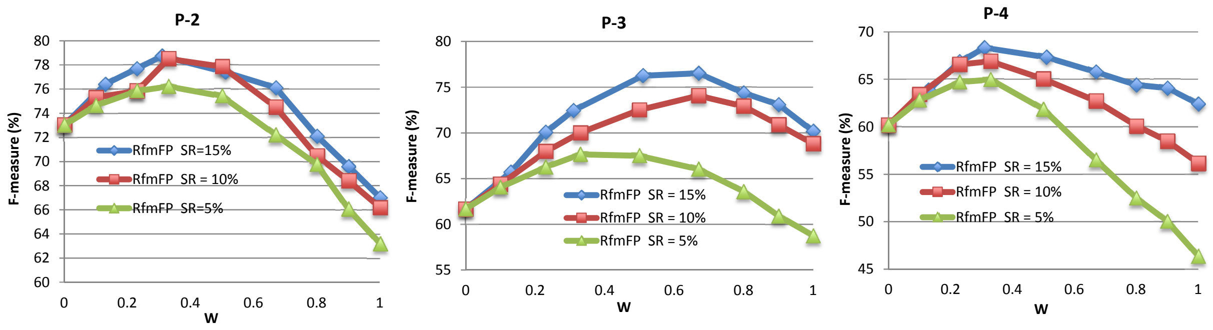


Figure 5. Performance comparison (min-sup = 0.5%) on the Retail dataset with different weights (W) and split ratios (SR).

of RfmFP starts with the base model (W=0) and increases when W increases up to the point of maximum success. Then it decreases down to the performance of retraining (W=1). The same performance comparisons are done on the Retail dataset as shown in Fig 5 with its three deployment partitions P-2, P-3 and P-4. Here we plot only the RfmFP curves for a particular partition with different SR values. Note that we can also observe base and retraining curves from a RfmFP curve as base and retraining curves can be drawn with a straight line from W=0 and W=1 points, respectively. However, these results also show the similar performance behavior of the base, retraining and RfmFP algorithms.

These results indicate that the W value for obtaining the highest performance of RfmFP is not fixed for all the datasets/partitions. As we only have a few data in each deployment, it is not possible to predict the accurate W. However, we notice that the maximum value of F-measure occurs nearby the middle of the X-axis (W=0.5) rather than nearby the two sides (W=0 or W=1). Moreover, it is interesting to observe that W=0.5 always gives better performance than the base model and retraining in all the cases in Fig. 4 and Fig. 5. Hence, we suggest to use W=0.5 (equal importance to both source and deployment data) when a small part of deployment data is available.

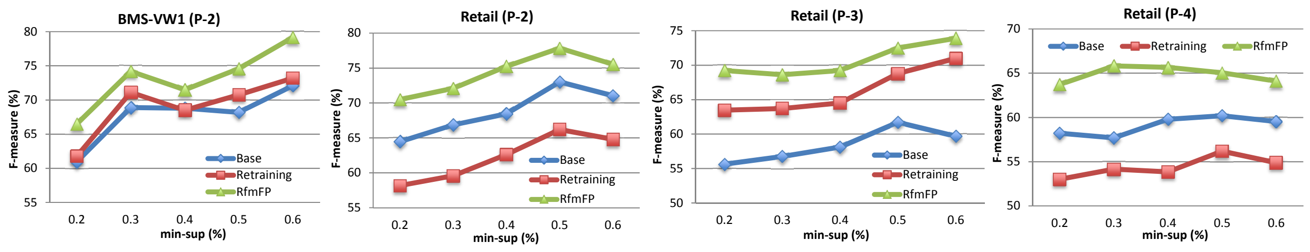


Figure 6. Performance comparison (SR = 10% and W = 0.5) on the BMS-WebView-1 and Retail datasets with different minimum support thresholds.

In order to observe the performance of W=0.5, we now compare the base model, retraining and RfmFP on these deployment partitions with different minimum support thresholds using SR=10%. The results are shown in Fig. 6 which clearly indicates the supremacy of RfmFP in all the cases.

V. Conclusions

In this paper, we propose a new approach of reframing in frequent pattern mining and design an efficient algorithm to perform the reframing task by discovering a very good set of approximate frequent patterns even though a few transactions are available in the deployment. Our approach has the capability to tackle different changes in data distributions and make the existing model workable in different deployment environments with the presence of dataset shift. We also define the base model and retraining in frequent pattern mining and clearly indicate how they are different from reframing. We exhibit the existence of dataset shift in two real-life transactional datasets and explicitly show when remarkable dataset shift can be found in these datasets. We demonstrate the capability of our approach to deal with these existing real-life dataset shifts accurately. Moreover, we experimentally analyze how to choose a good weight value for the few available deployment transactions in order to achieve an approximately optimal performance of the RfmFP algorithm. Finally, we show that the proposed RfmFP algorithm can clearly outperform the base model and retraining by using the chosen weight value for the few available deployment transactions.

Acknowledgments

This work was supported by the REFRAME project granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences & Technologies ERA-Net (CHIST-ERA).

References

[1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” in *VLDB*, 1994, pp. 487–499.

[2] J. Han, J. Pei, Y. Yin, and R. Mao, “Mining frequent patterns without candidate generation: A frequent-pattern tree approach,” *Data Min. Knowl. Discov.*, vol. 8, no. 1, pp. 53–87, 2004.

[3] J. G. Moreno-Torres, T. Raeder, R. Alaíz-Rodríguez, N. V. Chawla, and F. Herrera, “A unifying view on dataset shift in classification,” *Pattern Recognition*, vol. 45, no. 1, pp. 521–530, 2012.

[4] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets, “Using association rules for product assortment decisions: A case study,” in *KDD*, 1999, pp. 254–260.

[5] Z. Zheng, R. Kohavi, and L. Mason, “Real world performance of association rule algorithms,” in *KDD*, 2001, pp. 401–406.

[6] C. K. Leung, Q. I. Khan, Z. Li, and T. Hoque, “CanTree: a canonical-order tree for incremental frequent-pattern mining,” *Knowl. Inf. Syst.*, vol. 11, no. 3, pp. 287–311, 2007.

[7] S. K. Tanbeer, C. F. Ahmed, B. Jeong, and Y. Lee, “Sliding window-based frequent pattern mining over data streams,” *Information Sciences*, vol. 179, no. 22, pp. 3843–3865, 2009.

[8] A. Tran, T. C. Truong, and B. Le, “Simultaneous mining of frequent closed itemsets and their generators: Foundation and algorithm,” *Eng. Appl. of AI*, vol. 36, pp. 64–80, 2014.

[9] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, “MAFIA: A maximal frequent itemset algorithm,” *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1490–1504, 2005.

[10] J. Lee and C. W. Clifton, “Top-k frequent itemsets via differentially private FP-trees,” in *KDD*, 2014, pp. 931–940.

[11] X. Yan, H. Cheng, J. Han, and D. Xin, “Summarizing itemset patterns: a profile-based approach,” in *KDD*, 2005, pp. 314–323.

[12] H. Cheng, P. S. Yu, and J. Han, “Approximate frequent itemset mining in the presence of random noise,” in *Soft Computing for Knowledge Discovery and Data Mining*. Springer, 2008, pp. 363–389.

[13] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[14] H. Daumé III and D. Marcu, “Domain adaptation for statistical classifiers,” *J. Artif. Intell. Res. (JAIR)*, vol. 26, pp. 101–126, 2006.

[15] N. Lachiche and P. A. Flach, “Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves,” in *ICML*, 2003, pp. 416–423.

[16] H. Zhao, A. P. Sinha, and G. Bansal, “An extended tuning method for cost-sensitive regression and forecasting,” *Decision Support Systems*, vol. 51, no. 3, pp. 372–383, 2011.

[17] J. G. Moreno-Torres, X. Llorà, D. E. Goldberg, and R. Bhargava, “Repairing fractures between data using genetic programming-based feature extraction: A case study in cancer diagnosis,” *Information Sciences*, vol. 222, pp. 805–823, 2013.

[18] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, “SPMF: A java open-source pattern mining library,” *Journal of Machine Learning Research*, vol. 15, pp. 3389–3393, 2014.