A Time-Window Approach to Recommending Emerging and On-the-rise Items

Tubagus Mohammad Akhriza
IEEE Member
Dept. of Information System
Pradnya Paramita School of Informaticcs Management and Computer, Kampus IT STIMATA, Malang, East Java, Indonesia
akhriza@stimata.ac.id (corresponding author)

Indah Dwi Mumpuni
Dept. of DIII – Information System
Pradnya Paramita School of Informaticcs Management and Computer, Kampus IT STIMATA
Malang, East Java, Indonesia
indah@stimata.ac.id

***Abstract*— The recommendation system (RS) filters large sales transaction data, to promote item Y as an alternative or pair to item X that the application user is looking for. Items that are no longer in season usually decrease in transactions, even to zero. Constantly recommending these items is irrelevant, while other items that are more relevant and profitable are not recommended, such as the emerging and on-the-rise items. The proposed solution to this problem is the time window approach, which is a block of data of a certain size and recorded at a certain time stamp. Item combinations (itemsets) are mined from each window with an association rule approach, and changes in the number of transactions containing these items are evaluated from window to window. From the number of transactions that contain the itemsets, the system distinguishes status items into four types: risky, normal, emerging, and on-the-rise. Items that are currently suffering from zero sales risk the same fate in the next window, so they are called risky items. The system will notify admins to review this item, and look for other emerging or on-the-rise items to promote along with item X. Experiments were carried out on two datasets, and it was found that 42.7% and 59.4% items from each dataset are risky.**

Keywords—association rule, emerging patterns, recommendation system, time-windows

I. INTRODUCTION

Along with the development of electronic-based transactions applications, such as e-commerce, e-library and e-learning, recommendation systems (RS) have also been developed and implemented extensively. RS can filter large-scale transaction data, then generate items with features that app users are expected to buy or like. If a user is browsing or has purchased item X, the RS also promotes item Y. Here, X and Y have a certain relationship and can be developed using collaborative filtering (CF) [1], [2], content-based filtering (CBF) [3], or association rule-based filtering (ARBF) techniques [4], [5].

Time-aware recommendation systems have been proposed in the literature, in which changes to user features and items over time are considered, such as in [6], [7]. However, while methods that focus on removing old features and retaining new ones are widely developed; on the other hand, methods that focus on items that have the potential to appear and disappear in a season are still underdeveloped.

In fact, there are products that only sell well in certain seasons, where seasons can be created due to changes in the weather, the season of religious or local customs celebrations, the season of celebrating holidays in a country, and so on. Items that run the risk of not selling because they are out of season should not be recommended; as well as because the item is not being produced; instead, stores can focus on items that have sold well in recent months. The existence of items that will appear or disappear in transactions cannot be discovered by traditional time-aware RS. As a solution, a time-window based recommendation method was developed in our study.

A time-window is a specific-sized block of data created at a certain timestamp. The proposed method combines the concept of ARBF with the concept of emerging patterns (EP) mining. In general, the rule $R: X \rightarrow Y$ is said to be significant if a) support for itemset (X, Y), written sup(X, Y), meets the minimum support (*minsup*) threshold, and b) confidence or conf(X, Y) meets the minimum confidence (*minconf*) threshold, where conf(X, Y) represents the probability that Y will appear, if X appears. The rule becomes a reference for recommending Y as a pair of X, the item the user is looking for. Recommended item Y for X, in this article is written as $Rec: Y \rightarrow X$. Combining this concept with the EP concept, we can recommend a pair (X, Y) if sup(X, Y) and conf(X, Y) increases significantly in the current time window, compared to the previous window. So that time-aware recommendations are obtained.

The contribution of our proposed recommendation method is as follows. First, the emerging rate is not defined on the pair (X, Y) all at once, but only on Y. The reason is, when X is being searched by the user, sup(X) is not important; but if X is very popular, while Y is not yet popular, according to the monotonic nature of frequent itemset [8], then (X, Y) must also be unpopular and not mined as an EP; consequently, Y would also not be recommended. Second, the value of conf(X, Y) may still be low in the current time window, and this is also not a problem in the proposed approach. However, as sup(Y) and sup(X, Y) increase in the next window, so does conf(X, Y) as well. In addition, we propose four types of situations that occur in item Y, with respect to changes in sup(Y): risky, normal, emerging and on-the-rise.

A risky situation arises when Y has zero support in the current time window, and it is this situation that most shop owners should anticipate because it could appear again in the next window. This item can be seasonal, so notifications can be sent systemically to shop owners about this, while the system can recommend emerging or on-the-rise items rather than recommending risky items. From an experiment using two real sales data sets, it was found that in each dataset there were 42.7% and 59.4% of the rules containing risky items Y at multiple timestamps; and using the proposed approach, the system can make alternative recommendations for all these rules.

II. RELATED WORKS

A. Recommendation System

The recommendation system is one of the applications of artificial intelligence in our daily lives. It serves to filter large-scale transaction data and promote items that an e-commerce

visitor might like and buy, apart from the item he or she is searching for. Transactions here are not limited to those made through e-commerce. Transactions on borrowing books at the library can also be processed to recommend books to library visitors [9].

This system analyzes the history of items that the user has purchased or liked in the past, and predicts the items that the user might like today. In practice, when a user is browsing an item X, the system promotes other item Y. The relationship between X and Y can be determined collaboratively, by studying people who have rated the same items as respective user has rated. This recommendation method is known as collaborative-based filtering–CF[1], [2]. If collaborative rating data is not available, the system learns the history of items that individual users have purchased. Item Y and X are paired if both have similar features. This technique is known as content-based filtering–CBF [3].

In AR-based filtering–ARBF, the rules are the material for generating recommendations. This rule is basically an (X, Y) combination generated by applying several interestingness metrics, such as support, confidence, and lift. One of the characteristics of the ARBF is that X and Y are paired not because of the similarity of features as in the CBF, but they are paired because they meet those metrics with a certain threshold. Rules $R: X \rightarrow Y$ that meet this metric are called strong and interesting rules [4], [5]. The itemset X is called the antecedent, and the itemset Y is called the consequent of R. Formally, given a transaction dataset T of size N, and the set I contains all the items in T. Each $t_i \in T, i = 1..m$ is a transaction. Given itemsets X, Y, and XY $\subseteq I$. Support, confidence and lift for itemset (X, Y) are calculated using (1), (2) and (3).

$$\mathrm{Sup}(XY) = \frac{|t_i \in T; XY \subset t|}{N} \quad (1)$$

$$conf(X \rightarrow Y) = conf(XY) = \frac{\sup(XY)}{\sup(X)} \quad (2)$$

$$lift(XY) = \frac{\mathrm{conf}(XY)}{\sup(Y)} \quad (3)$$

B. Time-aware Recommendation System

Time-aware RS is proposed to overcome traditional recommendation methods that are not aware of changes that occur in item features or user features over time. Huang and Stakhiyevich proposes a time-aware group recommendation system by combining neural CF and CBF to solve the issue of changing user profiles [6]. Jia et al. investigates group recommendation methods with a focus on the impact of member activity levels on recommendation outcomes, where sliding windows are used to filter the current group activity [7]. Changes to item features are investigated by Rabiu et al. in [10], and proposed a matrix factorization technique in the temporal dataset as a solution. Yang, Nie and Yang proposed a hybrid method combining CF with temporal AR that is able to capture changes in user interest and recommend time contexts in conducting transactions [11]. Harshvardhan et al. uses the Boltzmann machine to reveal hidden features of users' preferences through their historical data [12].

Our review of these literatures shows that the focus of these time-aware recommendation methods is to maintain features that are still actively used, while discarding obsolete features. Meanwhile, a method for finding features which is starting to emerge or to disappear in popularity, has not been seen to be proposed in the literature.

C. Mining Emerging Patterns

EP is an itemset whose support grew significantly in the current time-window compared to the previous window; or from one data group to another [13], [14]. Formally, given an ordered pair of datasets D_1 and D_2, set I contains all items in both datasets and $X \subseteq I$ is an itemset. Let $sup(X, D_i)$denote the support of X in D_i. The support growth rate of X from D_1 to D_2, denoted as $GR(X, D_{1,2})$, is defined as (4):

$$GR(X, D_{1,2}) = \begin{cases} 0, \text{if } \sup(X, D_i) = 0; i = 1,2 \\ \infty, \text{if only } \sup(X, D_1) = 0 \\ \frac{\sup(X, D_2)}{\sup(X, D_1)}, \text{if otherwise} \end{cases} \quad (4)$$

Given that $\rho > 1$ is a minimum support growth threshold, a itemset X is said to be a ρ-emerging pattern (or simplify ρ-EP or EP) from D_1 to D_2 if $GR(X) \geq \rho$. The EP mining problem is, for a given ρ, to find all ρ-EPs. In a special case when $\sup(X, D_1) = 0$ and $GR(X) \geq \rho$, X is called a jumping EP.

D. Time-windows Model

The concept of time-windows appears in the discussion of mining patterns in dynamic data, such as temporal datasets or data streams. The goal is to make it easier for data scientists to make observations on items and their features which it is impossible to do it on all data at once, because transaction data come continuously. The time-window is basically a block of transaction data of size n, recorded at a specific timestamp. The most recent window is denoted by W_M, and the oldest window by W_1; so, W_{N-1} means one current window before W_N, and so on.

According to Akhriza, Ma and Li [15], the time-windows model is divided into four models: landmark, damped, sliding and tilted-time models. The landmark model considers the support found in all windows to be equally important, but this is not the case in the damped and sliding window models. In damped windows a pattern has a lifetime and can expire after a certain period of time, in sliding windows a pattern is considered important if it is found only in recent windows[16]. The tilted-time model is a landmark model but the support in each window is stored in an array over the long term. This model combines supports in old windows into an array, while supports in newer windows are not merged [17].

III. Proposed Methods

A. Framework of the Proposed time-aware RS

The framework of the proposed time-aware RS is described in Fig. 1.

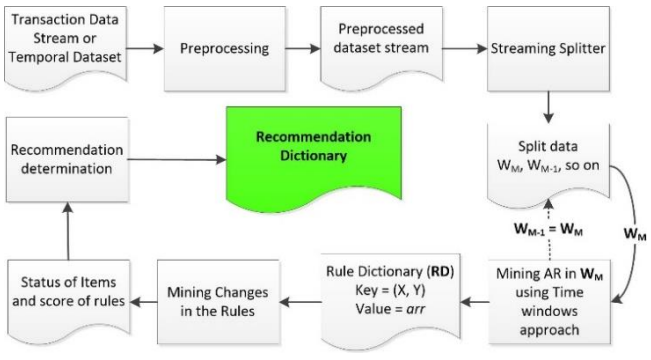

Fig. 1. Framework of the proposed time-aware RS

The first part is the preprocessing stage, including feature extraction, feature normalization, and transaction transformation to bag of item-ID. After that, the stream is split into time-windows with the size per window determined by the RS admin. This activity generates several time-windows. The second part is mining the rules from each window, and recording the rules to a rule dictionary. The third part is mining changes in the rules that produces item status and rule score, which are then used to determine the recommendation.

B. Preprocessing

Preprocessing is carried out on the transaction dataset with the aim of extracting only the features needed by the data scientist, and preparing these features to be effectively and efficiently processed by the AR miner program. Features can be prepared in the form of a bag of item-ID (one record contains many item-IDs), or feature vectors (depends on the miner program requirement). Because we are dealing with dynamic data, we must split the data into several time-windows with a certain size, for example *n* transactions. If the data is a stream, then the time allotted to process the data is also usually not much; but if it is temporal, then we can store the data for a period, maybe a day, a week, a month, and so on, before processing it. The output of this stage is the most recent time-window, i.e., W_M, to be processed. When another window is processed, the W_M becomes W_{M-1}, like that repeatedly so that a series of time-windows W_{M-2}, W_{M-3}, and so on, are created.

C. Mining AR using Time-windows

The AR miner program used in our study is APRIORI[5], which is modified for dynamic data environments, so that the resulting rules can be stored in the long term. The following conditions apply to mining rules in each time-window:

1. Two support thresholds: a) *minsup* which is set to two transactions per time-window, used to mine all rules with a maximum length of two; b) moderate support threshold (*modsup*) ≤ *minsup* is the minimum threshold for an itemset to be considered emerging per time-window; In particular, any itemset x is called a frequent itemset if $minsup \geq \text{sup(X)} \geq \text{modsup}$. Itemsets with support under *modsup* is called infrequent.
2. The minimum confidence (*minconf*) threshold is set to low as per RS admin requirements, for example 1% or 5%, so that more rules can be formed. Conf(X, Y) will increase as sup(X, Y) increases, so we don't give it a high lower bound. *Minconf* is applied in a time-window.
3. The lift is also unrestricted, even if it is worth less than 1; This is because the lift value also changes as the value of sup(X) and sup(Y) changes. Lift(X, Y) will even decrease if sup(Y) increases and sup(X) tends to stagnate. Lift is also applied in a time-window.
4. Only short rules are stored in array arr i.e., $R: X \rightarrow Y$ with $|X| = 1$, and $|Y| = 1$. This is because long consequent ($|Y| > 1$) can be observed from short rule.

After mining, the rules and their interesting metric values are stored in the rule dictionary RD, which has the general form **RD**(*key*, *value*). As key is (X, Y), and value is array arr of tuple $(sup(X), sup(Y), sup(XY), conf(XY))$. Searching for rules in each window will generate a number of rules, be it new or old rules. We designed a mechanism so that when a new rule (X, Y) is found in a W_M, whereas in the previous window(s) the rule does not exist, all values of support are set to zero but still are stored in the array according to respective index. **Example**: suppose that currently being processed is W_4, then key is (X, Y), while $arr[0, 0, (50, 40, 10, 0.2), 0]$ is the value; means that the rule does not yet exist in W_1, W_2, then it is in W_3 with tuple interestingness metric values (50, 40, 10, 0.2). In W_4 this rule is also not found. **Example**: $\text{sup}(Y)$ is extracted from arr and returns an array of $\text{sup}(Y)$, written with $\boldsymbol{arr\_Y}$ notation.

D. Mining Changes in the Rules

Mining changes in the rules is an extension of the EP concept, which traditionally only compares sup(Y) in two windows, and considers only increments of sup(Y) only. After RD is generated, then the change of sup(Y) can be evaluated from the arr in each rule. Given $arr\_Y[i]$ and $arr\_Y[i-1]$ that contains the sup(Y) obtained from W_M and W_{M-1} respectively. We define four possible situations for Y, as follows:

1. Special attention when $arr\_Y[i] = 0$ and $arr\_Y[i-1] \geq 0$, thus Y is in a **risky** situation, because $sup(Y)$ could continue to be zero in the future.
2. If $arr\_Y[i] > k * arr\_Y[i-1]$ and $arr\_Y[i] \geq modsup$ then Y is called an **emerging item** for X. Here, $k > 1$ is a multiplier that filters out the presence of emerging items Y.
3. If $arr\_Y[i] \geq arr\_Y[i-1] \geq arr\_Y[i-2]$ and $arr\_Y[i] \geq modsup$ then Y is called as **on-the-rise** item for X. In other words, recommendations "on-the-rise" are given to items whose support increased in two consecutive time-windows.
4. Otherwise, if both $arr\_Y[i] \geq minsup$ and $arr\_Y[i-1] \geq minsup$, then the situation is **normal**. Here, $sup(Y)$ in $arr\_Y[i]$ could be stagnant, or slightly up/down compared to $sup(Y)$ in $arr\_Y[i-1]$

After that, the rule is scored using the formula in (5)

$$score = sw * \sum_{j=i-2\ to\ i} arr\_Y[j] \qquad (5)$$

The status weights (*sw*) are given to the rule according to the status of item Y, namely: "on-the-rise" has a weight of 4, "emerging" has a weight of 3, then "normal" has a weight of 2. The "risky" status has a weight of 1. The sigma section on (5) explains that the score will be higher if Y is on-the-rise with a weight of 4, where sup(Y) has increased from the last two windows to the current window, thus indicating that Y is trending. The rule scoring mechanism is described in the following procedure.

Procedure: Rule scoring
input: RD # rule dictionary
return: RecD #recommendation dictionary
1. ***for*** key, value in RD.items():
2. ***for*** X, Y in key:
3. ***calculate*** $arr\_Y[i]$, $arr\_Y[i-1]$ and $arr\_Y[i-2]$ (if exists), then set *weight* according to Y's status as follows:
   if status == 'risky': *sw* = 1; if status == 'normal': *sw* = 2; if status == 'emerging': *sw* = 3;
   if status == 'on-the-rise': *sw* = 4
4. ***Compute*** $score = sw * \{arr\_Y[i] + arr\_Y[i-1] + arr\_Y[i-2]\}$

5. key = (X, Y); val = (status, score)
   RecD[key] = val
6. ***return*** RecD

E. Recommendation Determination

In ARBF, X and Y are associated because they meet the interestingness metric at a certain threshold, even though they perhaps have nothing in common in the category or product description. However, in our study, this similarity was included as the reason Y was recommended for X. The purpose of this similarity was to address the need to provide accurate or novel recommendations [18]. This similarity is denoted in sim(X, Y) which can be calculated using well-known distance formulas, such as the Jaccard coefficient or Euclidean distance[19]. Given X_{desc} and Y_{desc} that represent a set of term in the description of X and Y respectively. We used Jaccard coefficient (*Jac*) to count Sim(X, Y) that show number of common terms in X_{desc} and Y_{desc}, as shown in (6).

$$sim(X,Y) = Jac(X,Y) = {|X_{desc} \cap Y_{desc}|}/{|X_{desc} \cup Y_{desc}|} \quad (6)$$

Item Y is recommended for X if it satisfies two main conditions: **it has a high score**, and **a certain desired similarity value**. If high accuracy is required, select sim(X, Y) $\geq \tau > 0$ where τ is minimum similarity threshold defined by the admin, otherwise, if high novelty is required, select $0 \leq$ sim(X, Y) $< \tau$. Novelty of recommendations determined by a formulation and/or mechanism in the system is known as system-wise novelty[18].

In fact, some items are only available in certain seasons, such as in America and Europe which have four seasons. In the off-season, sales of these seasonal items usually drop to zero. For example, hot water bottles are only sought in winter. If it is seasonal, the system should notify the admin to review the current situation of item Y; otherwise, the system will look for the other item, e.g., Y_1 to be recommended with X, where Y_1 has the highest score and is not the risky item. Labels for seasonal items can be pre-assigned by admin. The procedure for determining recommendations is described as follows:.

Procedure: Recommendation Determination
input:
RecD #Recommendation dictionary
τ #Minimal similarity threshold
return: **RecSet** #a set of high score recommendation

1. A user is searching for item *x*
2. ***for*** key, value in RecD.items():
3. X, Y = key #extract X and Y from key
4. if (X == x) :
5. if Y is risky:
6. if Y is a seasonal item:
7. ***notify*** the admin to review the situation of item Y
8. else:
   select $\boldsymbol{Rec}$: $\boldsymbol{Y_1 \rightarrow x}$ with highest score **and** status != risky
   compute sim(x, Y)
   if sim >= τ: *Q* = 'accurate'; else: *Q* = 'novel'
   add (Rec, sim, Q) to *RecSet*
9. if Y is not risky:
   select $\boldsymbol{Rec}$: $\boldsymbol{Y \rightarrow x}$ with highest score
   compute sim(x, Y)
   if sim >= τ: *Q* = 'accurate'; else: *Q* = 'novel'
10. ***add (Rec, sim, Q)*** to *RecSet*
11. ***return*** RecSet

IV. EXPERIMENTAL WORKS

A. Business problem understanding

In the experiment, the proposed RS is tested on two real transaction datasets: the UCI and UQI datasets with the aim of, firstly, finding rules, and differentiating the items in the rules based on their four kinds of status. Secondly, the system looks for rules without risky items to recommend based on scores and similarity between items X and Y.

UCI dataset is a public online retail dataset downloaded from the UCI dataset repository. The dataset was obtained from a London-UK based SME, which was founded in 1981 [20]. The shop mainly sells unique gifts for all occasions. Located in Europe, sales at this store are season-related, so some items may look great in one season but not sell at all in another. UCI dataset has 13 time-windows (year-month), namely from 2010-12, 2011-01 to 2011-12.

UQI dataset was obtained from a small-scale SME, located in Malang, East Java, Indonesia. The company was founded in 2011, and sells leather products. Located in Indonesia, the sale of products in this store is not related to the season, since Indonesia only has two: rain and dry seasons. The sales dataset obtained from this company is called the UQI dataset, which has 11 time-windows, namely from 2019-02 to 2019-12.

Feature specifications for each dataset are described in Table I. The UCI dataset contains 500K records, but after preprocessing based on different invoice numbers, the number of records becomes 22,106 transactions. Similarly, the size of the UQI dataset shrunk from over 91K to 1576 transactions. In addition, while UCI dataset has more than 4K items, the UQI dataset has 45 items.

TABLE I. SPECIFICATION OF DATASETS

No	**Dataset features**			**Time-windows features**	
	Name	***Size***	***#items***	***#TS***	***Avg Winsize***
1	UCI	22,106	4,059	13	17K
2	UQI	1,576	45	11	7600

The result of understanding these datasets found that many items with zero sales occurred in several past time-windows, successively. The business problem formulated is how to know when sales drop to zero in a timely manner, so that the system does not recommend products that are out of season continuously?

B. Settings

TABLE II. MINING PARAMETERS

No	*Datasets*	*minsup*	*Modsup (#trans/window)*	*minconf*	*k*
1	UCI	2 / *Winsize*	50	5%	2, 3
2	UQI	2 / *Winsize*	10	1%	2, 3

Table II explains parameters used to mine rules from the datasets. As shown, *minsup* applied is two transactions divided by the size of time-window (*winsize*) which is the lowest, with *modsup* 50 and 10 for UCI and UQI datasets respectively. *Minconf* is also set low, at 5% and 1% for the datasets respectively. With this low setting, the process of

discovering item status and forming recommendations is adaptive to the frequency change of item sales in each timestamp. The number of items that are on-the-rise and emerging are evaluated at k = 2 and 3.

C. *Results and Discussion for the UCI Dataset*

Figs. 2 shows the calculation of support for singleton items in each time-window of the UCI dataset. By setting the parameters as in Table 3, 161 singleton items from UCI were generated which on the chart are represented by lines. Inspection in each timestamp via array *arr* finds that many items have zero support in some time-windows. This explains that the recommendation of item Y to be paired with X is irrelevant if it is delivered in all the time, such as done if we use the traditional AR-based RS.

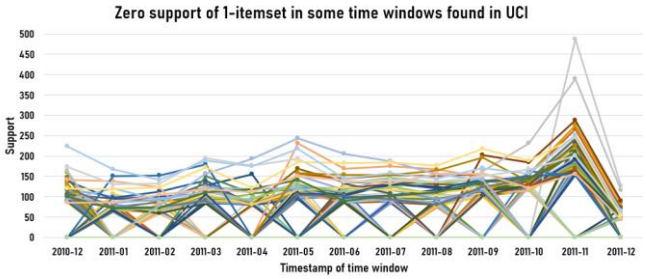

Fig. 2. Zero support of 1-itemset in some time-windows found in UCI

Some items seem to be selling seasonally, for example before and during Christmas season, so these items emerge in sales in the middle to the end of the year. Fig. 3 shows some of the frequently purchased items in this season such as Paper chair kit 50's Christmas and Vintage Christmas bunting. However, the figure also shows that these items are not purchased in other months.

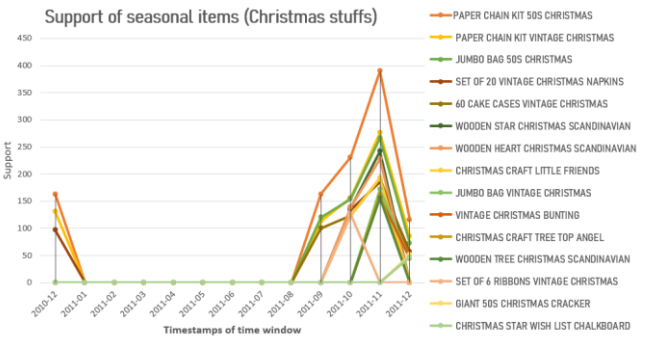

Fig. 3. Support of seasonal Christmas stuffs

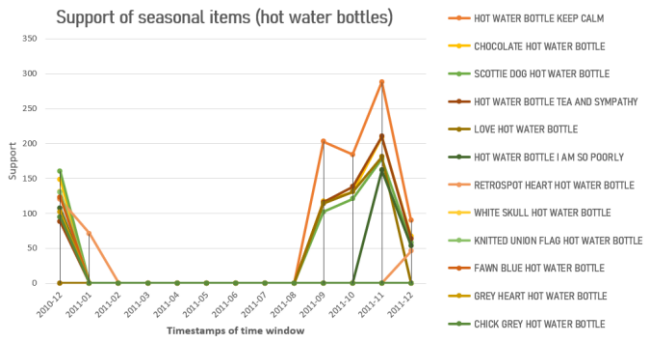

Fig. 4. Support of seasonal hot water bottles

Another interesting item in the spotlight is hot water bottles, such as Chocolate hot water bottle, as shown in Fig. 4. The frequency of purchasing this item increases in the same months as the Christmas season, which is autumn and winter. In contrast, in other seasons, the frequency of sales declines and there are even no sales.

Comparing the purchases of these two kinds of items, we find that individually they are often bought, as shown in Fig 5; But buying the two together shows a fairly small Support, that is, only 50 buys in November at most, as seen in Fig 6. These results indicate that the hot water bottle items (referred to as Y) individually sold quite well with various brands, with the total support almost matching the purchase of the Christmas items (referred to as X) mentioned. The sales side should be able to promote these two types of items to be purchased together more intensively, especially when the summer is gradually over.

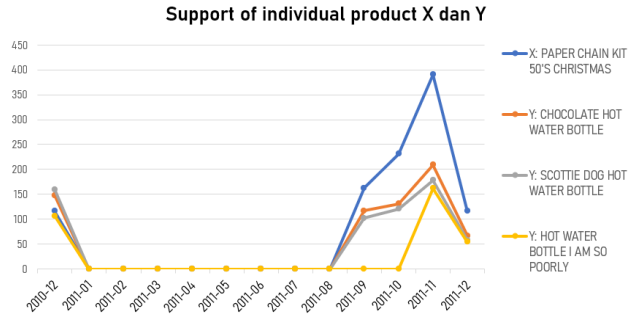

Fig. 5. Support of individual item X and Y

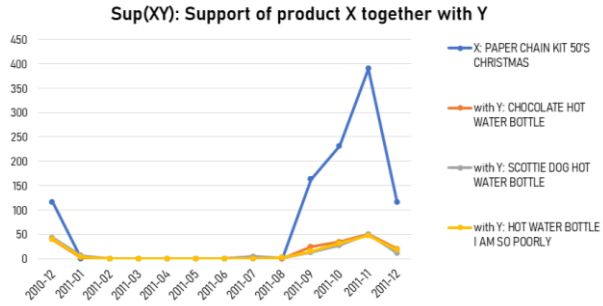

Fig. 6. Support of item X together with Y (Sup(X, Y))

If using the traditional EP approach, then some combinations (X, Y) are not captured as EP, since sup(X, Y) is less than 50 transactions and the growth of sup(X, Y) from last month to the current month is also not so significant, as Fig. 6. However, our approach applies monitoring to items Y separately, regardless of supp(X, Y), so that the recommendation to buy Y with X can be delivered in a timely manner to store customers. The store manager also should not hesitate to postpone the recommendation of these two items together, or individually, if there is no sale; because the store does not need to stock these items for a while. Some items may be damaged if they are piled up in the warehouse for a long time.

Inspecting rules for the categories of risky, normal, emerging and on-the-rise, generates 197,709 rules for all possible time-windows, from some past time-windows to current one ($arr[i]$) such as shown on Fig. 7. A total of 117, 505 rules (59.4%) were found to be risky, while 35,394 items are in normal situation for all ks. A total of 27,392 and 27,235

items were in emerging status, and the remaining 17,418 and 17,575 items were on-the-rise for k=2 and 3, respectively.

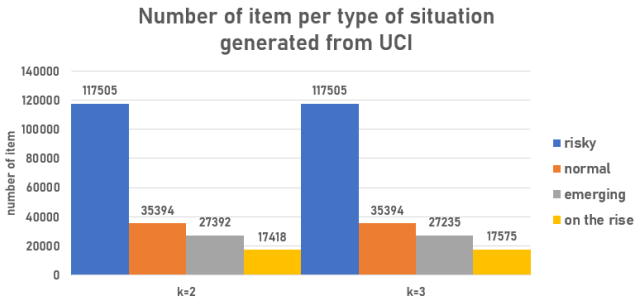

Fig. 7. Number of recommendation for X = 'hot water bottle'

The proposed RS was demonstrated in an experiment using Python, written in the Jupyter Notebook IDE. For example, supposed at W_4, a user searches for a product with a long description X = "victorian glass hanging t-light", then RS found 110 rules in which item Y are risky as shown in Table III. That is, the items in column Y at the W_4 are not selling.

TABLE III. SOME RULES IN UCI WITH Y IN RISKY STATUS

X	Y	status	score	sim
victorian glass hanging t-light	set of 3 cake tins pantry design	**risky**	331	0
	jam making set printed	**risky**	208	0
	recipe box pantry yellow design	**risky**	192	0
	jumbo shopper vintage red paisley	**risky**	177	0
	set/20 red retrospot paper napkins	**risky**	175	0
	wooden heart christmas scandinavian	**risky**	0	0
	wooden star christmas scandinavian	**risky**	0	0

TABLE IV. RECOMMENDATION OF Y WITH HIGH SIM AND SCORE

X	Y	status	score	sim
victorian glass hanging t-light	cream hanging heart t-light holder	**normal**	2976	0.857
	red hanging heart t-light holder	**normal**	532	0.286
	pink hanging heart t-light holder	**emerging**	414	0.286
	hanging heart jar t-light holder	**emerging**	276	0.286

While the system notifies the admin about the existence of Y that is not selling, to prevent the problem of recommending items that do not sell well in the next time-window, the system tries to find rules in $W_{N<4}$, but Y's status is not risky. Four rules are obtained where the score is quite high and with sim > 0. This happens because the descriptions of X and Y are similar (Table IV), which also shows that recommending Y for X is accurate. As seen, both are related to T-light.

TABLE V. RECOMMENDATION OF Y WITH LOW SIM

X	Y	status	score	sim
victorian glass hanging t-light	lunch bag red retrospot	**on-the-rise**	3216	0
	set of 3 cake tins pantry design	**on-the-rise**	3132	0
	set of 6 spice tins pantry design	**on-the-rise**	2248	0
	regency cakestand 3 tier	**on-the-rise**	1832	0
	jumbo bag red retrospot	**on-the-rise**	1656	0

For results where X and Y do not have similar descriptions are given in Table V; here, only Y with on-the-rise status is shown. The pair Y and X may seem new to some, because they seem unrelated; For example, X is a kind of lamp hanger, while Y is a lunch bag. However, that does not mean they are irrelevant because relevance depends on the user's next action, whether he or she will see this recommendation or not. As proof, the status shows that these combinations (X, Y) were on-the-rise in three consecutive time-windows.

D. Results and Discussion for the UQI Dataset

Observation of the dataset still found that there were products that were not sold at all in several months, as described in Fig. 8. However, such patterns are not related to the season, but to the company's sales strategy. During the national holiday season, the company even doubled price promotions and product bundling to increase sales. Since the dataset is taken from transaction records in 2019, while the company started its business in 2011, it is assumed that some products are not new products that have just been offered in the middle of the year. The absence of sales of a product is defined in our study as a product not selling well in a certain time-window, so that the proposed recommendation mechanism can be applied.

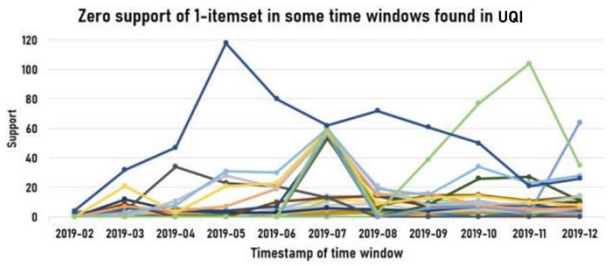

Fig. 8. Zero support of 1-itemset in some time-windows found in UQI

Checking all the existing time-windows, we found the number of products that are risky, continued, emerging and on-the-rise as described in Fig. 9. This result was obtained because many items had zero support at the beginning of the year. Through the chart it is known that 6,773 out of 15,854 (**42.7%**) items in UQI sales data are risky.

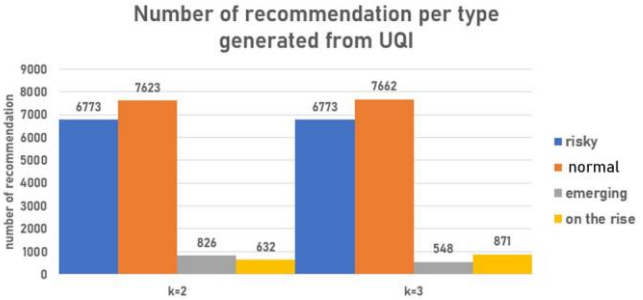

Fig. 9. Zero support of 1-itemset in some time-windows found in UQI

Next is a demonstration of recommendations generation by the proposed system, based on the rules and recommendations generated from UQI. However, checking the status of items at a W_J, is performed on items that have sales at least at one old window $W_{I<J}$. An item that does not have support in $W_{I<J}$ could be because it has not been produced or promoted to the market at that time.

The search results for rules with X = 'multifunctional leather handbag' at W_4. yielded 20 rules with item Y with risky

status; some of these results are described in Table VI. Next, RS looks for rules in the window before W_4, with status Y being not at risk. From such results, system generates several recommendations as described in Table VII. As seen in this table, in the UQI dataset, almost all items have a "leather" description, many pairs (X, Y) are similar. In addition, many products have the same name, and are distinguished only by product code. This also causes many products to have high similarity. However, the sim value may look inappropriate, because some brand codes are omitted since they are trade secrets.

TABLE VI. SOME RULES IN UQI WITH Y IN RISKY STATUS

X	Y	status	score	sim
multifunctional leather handbag (h01)	multifunctional leather handbag (h05)	**risky**	0	0.714
	multifunctional leather handbag (h06)	**risky**	0	0.625
	leather handbag (h07)	**risky**	0	0.625
	leather handbag (h03)	**risky**	0	0.556
	multifunctional leather sling bag (s01)	**risky**	0	0.5
	multifunctional leather pouch bag	**risky**	0	0.444
	genuine leather sling bag	**risky**	0	0.3

TABLE VII. RECOMMENDATION OF Y WITH VARIOUS SCORE AND SIM

X	Y	status	score	sim
multifunctional leather handbag (h01)	16 slot genuine leather wallet	**on-the-rise**	156	0.231
	leather waist bag	**emerging**	222	0.2
	genuine leather belt	**emerging**	126	0.2
	genuine leather waist bag for smartphone	**on-the-rise**	1120	0.182
	pistol sling bag	**Normal**	84	0.000
	premium belt	**Normal**	32	0.000
	waist bag (p03)	**Normal**	26	0.000

Time complexity occurs when looking for rules with the Apriori method. However, the *pyfim* library for Python (available in [21]) used can efficiently find rules with low support. The average search for rules in both datasets only takes less than one second, on a core i7 computer. The complexity of finding recommendations is also $O(m)$, where m is the number of Y for an X that is sought.

The rules generated by Apriori can take up a lot of memory, especially if very small minsup is used. However, with the strategy of only selecting rules with length X and Y of one item, memory can be minimized. The rules of the 11-month UQI dataset are only 116 KB in size, while the rules of the 13-month UCI dataset are 222 KB in size. Both store X, Y, support, confidence, and lift data for each time-window, in CSV format.

V. SUMMARY

The proposed time-aware-based recommendation method can determine the status of items in each time-window, namely: risky, normal, emerging and on-the-rise. From an experiment using two real sales data sets, it was found that in each dataset there were 42.7% and 59.4% of the rules containing risky items at multiple timestamps; and using the proposed approach, the system can make alternative recommendations for all these rules, at that moment. In the future developments, to improve memory efficiency for array support in the long term, tilted-time-window models such as the Fibonacci windows model can be used.

ACKNOWLEDGMENT

This research and publication is funded by the 2nd year Applied Research Grant, awarded by the Ministry of Education, Culture, Research and Technology of the Republic of Indonesia, with the contract number: 034/SP2H/PT-L/LL7/2022. We also thank our research partners, for their cooperation in providing the UQI dataset.

REFERENCES

[1] H. Alharthi, D. Inkpen, and S. Szpakowicz, "A survey of book recommender systems," *J. Intell. Inf. Syst.*, vol. 51, no. 1, pp. 139–160, 2017.

[2] R. Burke, "Hybrid Web Recommender Systems," in *The Adaptive Web*, Berlin: Springer, 2007, pp. 377–408.

[3] D. Wang, Y. Liang, D. Xu, X. Feng, and R. Guan, "Knowledge-Based Systems A content-based recommender system for computer science publications," *Knowledge-Based Syst.*, vol. 157, no. May, pp. 1–9, 2018, doi: 10.1016/j.knosys.2018.05.001.

[4] W. Ji, S. Liu, Y. Song, and J. Qi, "Research of Intelligent recommendation system based on the user and association rules mining for books," no. Iccsae 2015, pp. 294–299, 2016, doi: 10.2991/iccsae-15.2016.56.

[5] T. M. Akhriza and I. D. Mumpuni, "Quantitative class association rule-based approach to lecturer career promotion recommendation," *Int. J. Inf. Decis. Sci.*, vol. 13, no. 2, 2021, doi: 10.1504/ijids.2021.116507.

[6] Z. Huang and P. Stakhiyevich, "A Time-Aware Hybrid Approach for Intelligent Recommendation Systems for Individual and Group Users," *Complexity*, vol. 2021, 2021, doi: 10.1155/2021/8826833.

[7] J. Jia, Y. Yao, Z. Lei, and P. Liu, "Dynamic Group Recommendation Algorithm Based on Member Activity Level," *Sci. Program.*, vol. 2021, 2021, doi: 10.1155/2021/1969118.

[8] T. M. Akhriza, Y. Ma, and J. Li, "Revealing the Gap Between Skills of Students and the Evolving Skills Required by the Industry of Information and Communication Technology," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 27, no. 05, pp. 675–698, 2017, doi: 10.1142/s0218194017500255.

[9] B. Xu *et al.*, "Research on Library Knowledge Recommendation Based on Intelligent Discovery System," 2020, doi: 10.1145/3438872.3439076.

[10] I. Rabiu, N. Salim, A. Da'u, A. Osman, and M. Nasser, "Exploiting dynamic changes from latent features to improve recommendation using temporal matrix factorization," *Egypt. Informatics J.*, vol. 22, no. 3, pp. 285–294, Sep. 2021, doi: 10.1016/j.eij.2020.10.003.

[11] D. Yang, Z. T. Nie, and F. Yang, "Time-aware CF and temporal association rule-based personalized hybrid recommender system," *J. Organ. End User Comput.*, vol. 33, no. 3, 2021, doi: 10.4018/JOEUC.20210501.oa2.

[12] G. M. Harshvardhan, M. K. Gourisaria, S. S. Rautaray, and M. Pandey, "UBMTR: Unsupervised Boltzmann machine-based time-aware recommendation system," *J. King Saud Univ. - Comput. Inf. Sci.*, 2021, doi: 10.1016/j.jksuci.2021.01.017.

[13] H. Alhammady and K. Ramamohanarao, "Mining emerging patterns and classification in data streams," in *Proceedings - 2005 IEEE/WIC/ACM InternationalConference on Web Intelligence, WI 2005*, 2005, vol. 2005, doi: 10.1109/WI.2005.96.

[14] G. Dong and J. Li, "Efficient mining of emerging patterns: discovering trends and differences," *Proc. fifth ACM SIGKDD Int. Conf. Knowl. Discov. data Min.*, 1999.

[15] T. M. Akhriza, Y. Ma, and J. Li, "Novel Push-Front Fibonacci Windows Model for Finding Emerging Patterns with Better Completeness and Accuracy:," *ETRI J.*, vol. 40, no. 1, 2018, doi: 10.4218/etrij.18.0117.0175.

[16] J. H. Chang and W. S. Lee, "Finding recent frequent itemsets adaptively over online data streams," 2003, doi: 10.1145/956750.956807.

[17] C. Giannella, J. Han, X. Yan, and P. S. Yu, "Mining Frequent Patterns in Data Streams at Multiple Time Granularities," *Next Gener. data Min.*, 2003.

[18] Z. Zolaktaf, R. Babanezhad, and R. Pottinger, "A generic top-n recommendation framework for trading-off accuracy, novelty, and coverage," *Proc. - IEEE 34th Int. Conf. Data Eng. ICDE 2018*, pp. 149–160, 2018, doi: 10.1109/ICDE.2018.00023.

[19] A. Jain, A. Jain, N. Chauhan, V. Singh, and N. Thakur, "Information Retrieval using Cosine and Jaccard Similarity Measures in Vector Space Model," *Int. J. Comput. Appl.*, 2017, doi: 10.5120/ijca2017913699.

[20] D. Chen, S. L. Sain, and K. Guo, "Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining," *J. Database Mark. Cust. Strateg. Manag.*, vol. 19, no. 3, 2012, doi: 10.1057/dbm.2012.17.

[21] C. Borgelt, "PyFIM - Frequent Item Set Mining for Python." 2022, [Online]. Available: https://borgelt.net/pyfim.html.