Frequent episode mining over the latest window using approximate support counting

Julie Soulas

Laboratory of Artificial Intelligence and Decision Support/ INESC TEC,
Rua Dr. Roberto Frias, Porto 4200-465, Portugal

Abstract. In this paper we propose a streaming approach for the discovery and monitoring of frequent patterns (the episodes) within the recent past of an event stream. This approach is based on the heuristic computation of the estimated support of the episodes of interest, and allows the fast discovery of frequent episodes with limited information storage. In particular, we do not even need to store the window of interest, which is necessary for exact frequent episode discovery. The proposed algorithm, `FEM-ASC` (Frequent Episode Mining, with Approximate Support Counting) is still at an early development stage, but already shows promising results on an activity monitoring task. In particular, `FEM-ASC` yields results very similar to those produced by an exact method, while requiring much fewer resources.

1 Introduction

Frequent pattern mining is a traditional unsupervised data mining task, with applications in different fields, such as sale monitoring, or human activity monitoring. We propose here to mine the frequent episodes in a stream of events. The data used in the experiment comes from a sensor network in a smart home. Instead of considering the whole event history, we propose to focus on the recent past exclusively. The originality of the developed approach lies in its capacity to mine frequent episodes without keeping track of the history. In particular, we do not store the contents of the window of interest. In order to do so, we devise a heuristic strategy for support counting that uses decay strategies.

Contributions. We propose `FEM-ASC`, which stands for Frequent Episode Mining using Approximate Support Counting. This heuristic algorithm estimates the expected support of the episodes, and handles the monitoring of the frequent episodes. `FEM-ASC` complies with the constraints linked to stream mining [3]. Namely, each data point is processed once only and is not stored. Its integration time in the model is small and independent from the history size. Memory usage is also limited and independent from the history size. Finally, the model is capable of adapting to concept drifts. `FEM-ASC` allows the monitoring of changing episode frequency, the discovery of emerging episodes, and the forgetting of formerly frequent episodes if they become rare.

Outline. The remainder of the paper is organised as follows. Section 2 introduces episode mining and relevant related work. Section 3 presents the formalisms and notations used throughout this work, while section 4 details how `FEM-ASC` works. Initial testing and validation results are presented in section 5. Finally, section 6 draws some conclusions and plans for future work.

2 Related work: episode mining

2.1 Events and episodes

Frequent episode mining is a data mining task introduced by Mannila et al. [8], for the discovery of interesting patterns in a sequence of events (definition 1).

Definition 1. *An* event *is a pair* (e, t)*, where:*

- e *is the event* label*. It usually corresponds to a sensor reading, an identifier, an alarm code, etc., and takes values in a finite alphabet* $\mathcal{A}$*;*
- t *is the* timestamp*. The timestamps induce a total order on the events, and allow the computation of distances on the events based on time intervals.*

Definition 2 (Episode). Episodes *are partially ordered collections of event labels. The episodes are called* serial *when the order is total, and* parallel *when there are no order constraints.*

Episode mining. Episode minig is the task of finding relevant episodes (see definition 2). If Mannila et al. [8], as well as subsequent studies propose methods for the discovery of all kinds of episodes, they also acknowledge that mining general episodes is expensive in terms of complexity, and focus either on serial or parallel episodes. Episode mining differs from traditional itemset mining: the temporal order and the distance between the events is the key feature driving the mining.

2.2 Measuring episode support

Definition 3 (Episode occurrence). *An episode* E occurs *in an event sequence* S *if there are events in* S *having the same labels as those of* E *and respecting the order constraints. These events then form an* occurrence *of* E*.*

If it is fairly straightforward to count the support of an itemset in a transactional database, it is however harder to settle on a single definition when it comes to episodes. Several main trends have thus emerged:

Window-based support [7,1,11]. Using a sliding window of fixed size, count the number of windows in which the episode occurs.

Table 1: Characteristics of some prominent episode mining algorithms

Algorithm	Searched episodes	Support measure	Stream
Mannila and Toivonen [6]	General	MO	No
Mannila et al. [7]	General	WF	No
Casas-Garriga [1]	Serial + Parallel	A-priori	No
Zhu et al. [13]	Serial	Non-overlapping MO	No
Zhou et al. [12]	Closed serial	MO	No
Tatti and Cule [11]	Closed general	PG	No
Patnaik et al. [9]	General	Distinct	Yes
Lin et al. [5]	Serial	Distinct MO	Incremental
Gan and Dai [4]	Serial	Custom WF	Yes
Soulas and Lenca [10]	Parallel	MO	Yes
FEM-ASC	Parallel	Non-overlapping MO	Yes

MO: minimal occurrences WF: window frequency

Minimal occurrence count [6,13,12,5]. See definition 4.

Definition 4 (Minimal occurrence - MO). *Let $E = \{e_1, ...e_n\}$ be an episode, and o an occurrence of E starting at timestamp t_1 and finishing at timestamp t_n. We refer to the interval $[t_1, t_n]$ as the* span *of E. o is a* minimal occurrence *if there is no other, shorter occurrence whose span is included in $[t_1, t_n]$. That is to say, there cannot be an occurrence o' of E starting in t'_1 and finishing in t'_n such that $t_1 \leq t'_1$, $t'_n \leq t_n$ and $t'_n - t'_1 < t_n - t_1$.*

Maximal count of distinct occurrences. With both previous methods, some events participate in several occurrences of an episode. This lead researchers to look for the maximal number of distinct [9] (occurrences do not share events), or non-overlapping occurrences (the first occurrence finishes before the second one starts).

Hybrid measures. In order to benefit from the properties of different support families, the measures may be combined. For example, minimal occurrences are sometimes counted only if they are distinct from previous minimal occurrences [5], non-overlapping [13], or if they last for less than a maximal duration bound.

2.3 Problem statement

Table 1 summarizes the characteristics of some prominent episode mining algorithms, and positions our current contribution. While most combinations of episode types and support measures have been explored, most approaches do not adapt well to data streams.

With `FEM-ASC`, we search for *parallel* episodes, that is to say unordered sets of labels, which we subsequently simply refer to as episodes. Since the events occurring in the vicinity of each other are more likely related than distant events,

a threshold T_{ep} constraints the *maximal occurrence duration* (longer occurrences are discarded). We define a hybrid support as the count of the *non-overlapping minimal occurrences* within the lastest *window of interest* $[t - T_W, t]$ (duration T_W). Occurrences satisfying these constraints are refered to as T_{ep}-*NOMO*.

The support verifies the downward closure property [13]. That means that for E an episode and E' any of its sub-episodes ($E' \subset E$), the support of E' is greater or equal to this of E.

3 Definitions and Formalisms

3.1 Support and approximate support

Computing the exact support of an episode in a given time window requires to store information on each T_{ep}-NOMO, as in the formalism used in [10]. We propose here to estimate the support, knowing exclusively (i) the date t_0 when the first occurrence started, (ii) the date t_{last} at which finished the last T_{ep}-NOMO and (iii) the support $\hat{S}_{last}$ estimated at that date. We use a decay-factor inspired strategy: assuming that the occurrences of the episode are evenly spread throughout the window of interest, the support is expected to decrease linearly with time if no new occurrence is observed, until it reaches 0 when T_W time has passed.

Definition 5 (Expected support). *At each time t, we define the expected support $\hat{S}(t)$ of the episode as:*

$$\hat{S}(t) = \begin{cases} \hat{S}_{last} & t_0 > t - T_W \\ 0 & t - t_{last} \geq T_W \\ \hat{S}_{last} \cdot \left(1 - \frac{t - t_{last}}{T_W}\right) & otherwise \end{cases} \quad (1)$$

The first case considers the initial monitoring of the support: all occurrences are too recent to be outdated. The second case is the situation when all occurrences are outdated. Finally, the third emuates the expected support decrease: the term $\frac{t-t_{last}}{T_W}$ corresponds to the proportion of the occurrences that we expect occurred between $t_{last} - T_W$ and $t - T_W$ and are thus now outdated.

If a new non-overlapping occurrence is detected (see the conditions in section 3.2) at time t, the approximate support is updated: $\hat{S}_{last} \leftarrow \hat{S}(t) + 1$, as is $t_{last} \leftarrow t$. Figure 1 presents the evolution of the real and expected supports for episode $\{\texttt{A}\}$ on a toy stream with a window of size $T_W = 6$. The expected support does not follow exactly the real support, and unfortunately, the divergence between the real and expected supports may be important, for example if the temporal distribution of the occurrences follows burst patterns. But if we consider a fairly homogeneous occurrence distribution over the window of interest, the discovered trends are very similar to the real support evolution.

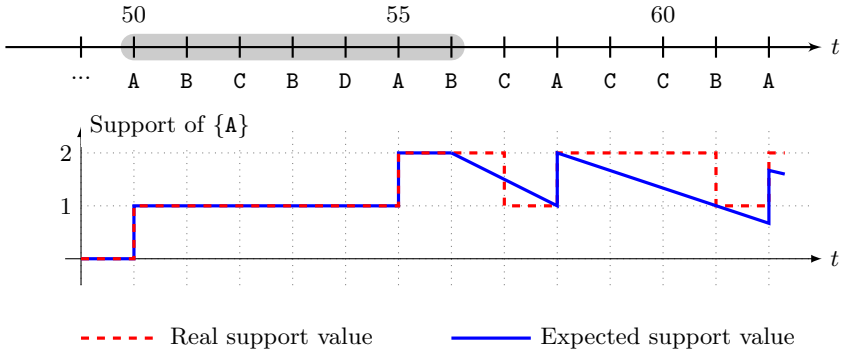


Fig. 1: Real VS Expected support for episode {A}, with a window length $T_W = 6$

3.2 New occurrence detection

When a new event (e, t) is recorded, we can detect whether it forms a new occurrence of an episode $E' = E \cup \{e\}$, where E is an episode using exclusively:

- The span of the last seen minimal occurrence of E: Δt^E_{last}
- The span of the last seen minimal occurrence of E': $\Delta t^{E'}_{last}$
- The end time of the last seen NOMO of E': $t^{E'}_{last}$
- The new event (e, t)

Indeed, minimal occurrences have convenient properties, as mentioned for instance in [6,5,10]. The most important in our context are:

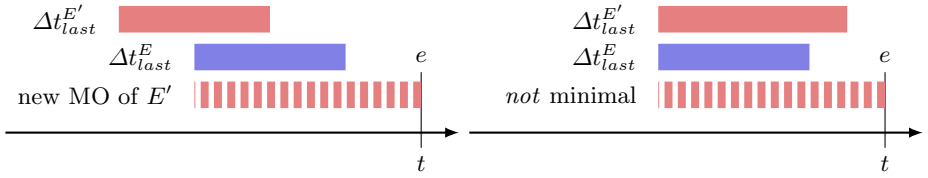
- If Δt^E_{last} started less than T_{ep} prior to the arrival event (e, t), then there is a new occurrence of E' spanning $[\Delta t^E_{last}.start, t]$
- Moreover, this occurrence is *minimal* if Δt^E_{last} starts strictly after $\Delta t^{E'}_{last}$. Then, we can update $\Delta t^{E'}_{last}$ as $[\Delta t^E_{last}.start, t]$.
- Finally, if this new $\Delta t^{E'}_{last}$ starts after $t^{E'}_{last}$, this occurrence does not overlap previous T_{ep}-NOMOs: it counts for the support, $\hat{S}_{last}$ and $t^{E'}_{last}$ are updated.

We refer to the process of building the next minimal occurrence of E' from the last occurrence of E and a new event (e, t) as *augmenting E with (e, t)*. This gives a particular importance to the minimal occurrences that started less than T_{ep} before the event currently under consideration. The episodes whose last minimal occurrence started less than T_{ep} ago are said to be *recently observed*. Figure 2 illustrates some of the possible configurations (minimal and non-minimal new occurrences).

4 Frequent episode mining with FEM-ASC

4.1 General framework

The event stream is processed one event at the time, chronologically. The arriving event (e, t) is susceptible of augmenting any recently observed episode. The



(a) New minimal occurrence (b) Not a new minimal occurrence

Fig. 2: Discovering whether a new event (e, t) helps form a new minimal occurrence of an episode $E' = E \cup \{e\}$

Table 2: Parameters allowing the tuning of `FEM-ASC`

Name	Description	Range
Window length T_W	Size of the period of interest. Older events are outdated, and should not appear in the support.	$0 < T_{ep} < T_W$
Maximal episode duration T_{ep}	Occurrences longer than T_{ep} are not counted	
Frequency threshold S_{min}	Support threshold for an episode to be considered as frequent	$0 < S_{sub} < S_{min}$
Subfrequency threshold S_{sub}	Support threshold for an episode to be considered as subfrequent, and thus maintained in the lattice	

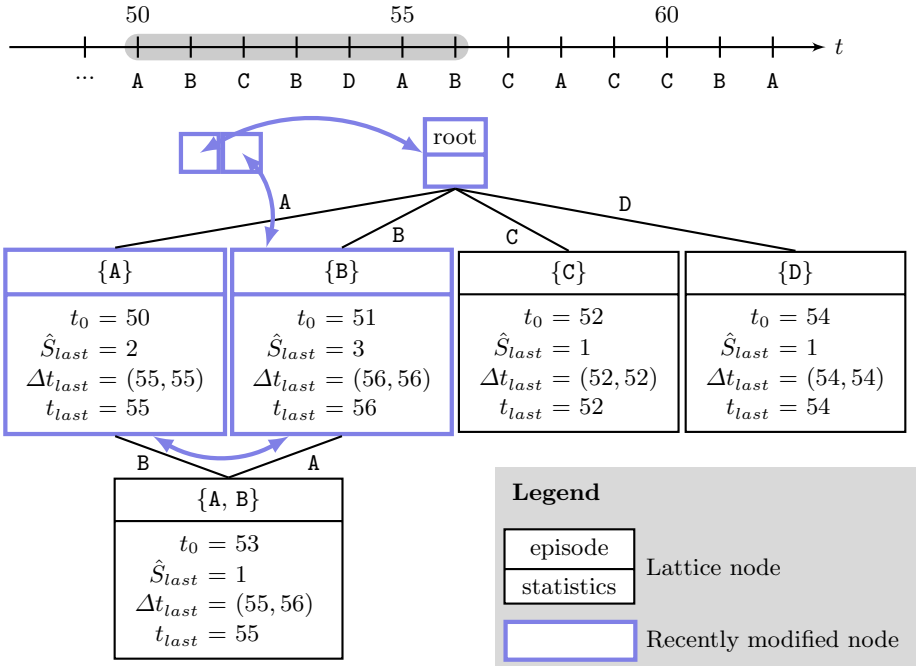
recently observed episodes are thus traversed in order to check if they can be augmented. Any new minimal occurrence discovered leads to the update of the corresponding episode, thus making it a candidate for future augmentations.

The key to efficient update is the use of a lattice structure for the storage of the episodes. The lattice is described in section 4.2, and the lists for the retrieval of the recently observed episodes in section 4.3. The update procedures are detailed in section 4.4. We use two support thresholds, which allows us to make a compromise between the size of the search space and the reactivity to concept drifts: a frequency threshold S_{min}, and a subfrequency threshold S_{sub}, with $S_{sub} \leq S_{min}$. Table 2 summarizes the parameters used to tune `FEM-ASC`.

4.2 Episode Lattice

The relevant episodes are stored in a frequent episode lattice (FEL). Each node in the lattice corresponds to an episode E, and contains the information required for the approximate support computation and update. That is to say:

- The date t_0 at which started the first recorded occurrence for E
- The span of its last minimal occurrence Δt^E_{last}
- The end of the last non-overlapping minimal occurrence t^E_{last}
- The support approximated at time t^E_{last}: $\hat{S}_{last}$



$S_{sub} = 2$, $S_{min} = 3$, $T_{ep} = 3$, and $T_W = 6$

Fig. 3: Example lattice when (B, 56) is the last seen event

However, not every episode is stored in the lattice, it would otherwise grow impractical. We choose to maintain a node in the lattice if: (i) it is at least subfrequent: it may then be a candidate for augmentation, or (ii) it was created less than T_W ago: we do not know yet if it is going to be frequent. This strategy choice means that an episode needs to be monitored for a while, until it gets at least subfrequent before we start considering longer episodes containing it. Formerly frequent episodes also remain in the lattice until they fall under the subfrequency threshold. This reduces the repetitive addition / deletion of nodes, and allows faster adaptation to drifts in the data distribution.

The parents of a node (located at depth d) correspond to its sub-episodes of length $d-1$, and its children to its super-episodes of length $d+1$. The edges linking two episodes are indexed on the only event label that is present in the child but not in the parent. In spite of its possibly big edge count, the lattice structure was preferred to the standard prefix tree: it allows faster episode retrieval.

4.3 Recently modified nodes

As pointed out in section 3.2, the recently observed episodes may be augmented with a new incoming event to form an occurrence of a longer episode. It is thus necessary to easily retrieve the recently observed episodes that are at least

Algorithm 1 Update of the lattice when a new event is recorded

Input: new event (e, t), FEL, RMN header list

```
for all node n in RMN, where n represents episode E do
    if e ∈ E then
        pass                                                    // e cannot augment E
    else
        if n's last occurrence is not recent enough then
            Remove n from RMN
        else
            if n has a child c on label e then
                if n's last occurrence starts strictly after c's then
                    Add the new minimal occurrence to c
                    if the new occurrence starts after the previous T_ep-NOMO then
                        Update frequency (equation 1)
                    if c is at least subfrequent then
                        Add it to RMN
            else                                    // The child does not exist yet, create it?
                if all of the parents of the child node are at least subfrequent then
                    Create the new node, record its first occurrence
                else
                    pass                            // The child cannot be (sub)frequent (yet)
```

subfrequent. We use doubly linked lists to iterate over the recently observed episodes for each depth level in the lattice. The order of the nodes in each linked list does not influence the occurrence discovery, but all the nodes at a given depth d should be considered for augmentation before the traversal of depth $d+1$. Otherwise, this may lead to inconsistencies in the lattice. Figure 3 presents the lattice on a toy dataset after the first 7 events have been introduced, with the parameters $S_{sub} = 2$, $S_{min} = 3$, $T_{ep} = 3$, and $T_W = 6$.

4.4 Lattice update when a new event arrives

When a new event (e, t) arrives, the lattice update follows algorithm 1: the episode E represented by each of the recently modified node is considered as a candidate for augmentation. Some candidates are pruned: those who already contain e, and those added in the RMN-lists too long ago and thus do not belong there any more. Then, two cases arise: either the node for episode $E \cup \{e\}$ already exists, and we need to figure out whether we have a new minimal occurrence; or the node does not exist, and we must decide whether it should be created.

Removal of outdated information. Regular sweeps over the lattice enable the removal of outdated nodes and branches. An alternative strategy has also been considered: we check whether a node is outdated when it is updated. This alternative strategy misses outdated nodes: those not traversed via the updates with the new events. The alternative strategy has not been evaluated yet.

	Influence of T_W	**Influence of T_{ep}**	**Influence of S_{min}**
T_W	1 min, 1 hour, 1 day, 1 week, 2 weeks, 4 weeks	4 weeks	4 weeks
T_{ep}	min(30 min, T_W)	1 min, 30 min, 1 hour, 12 hours	30 min
S_{min}	median label support over the dataset divided by T_W, with a minimal of 3 (3, 3, 3, 9, 17, 33)	median label support over the dataset divided by T_W (33)	17, 33, 66, 99, 131, 164
S_{sub}	$S_{min}/2$	$S_{min}/2$	$S_{min}/2$

Table 3: Detailed setting used in the experiments

5 Initial experimental results on activity monitoring

This section presents the initial results of `FEM-ASC` on a dataset coming from the activity recognition community: the CASAS Aruba dataset[1] [2]. This dataset contains the sensor readings from a smart home system containing motion detectors, temperature sensors and sensors on the doors, as well as activity annotations, which we use here. The annotation dataset contains 12 954 events, taking 22 different labels (11 activities with *start* and *end* markers).

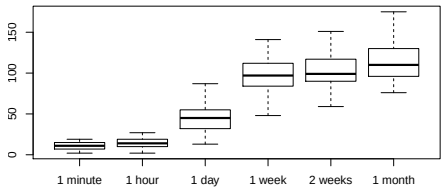
Two aspects are considered: (i) the scalability of `FEM-ASC`, and (ii) its ability to yield good results in spite of its heuristic nature. The scalability is evaluated via the evolutions of the size of the episode lattice and of the length of the RMN linked list during the execution of `FEM-ASC`: it is expected to remain fairly constant. The quality of the results yielded by `FEM-ASC` are compared with those produced by an exact frequent episode miner. We use the one described in [10], which looks exactly for the same patterns, using the same support measure definition and the same control parameters. We compare regularly (each time 100 events are processed) the lists of frequent episodes produced by `FEM-ASC` and the exact method, and compute the precision and recall as follows:

- An episode is considered as a true positive if is discovered by both algorithms. We note TP the number of true positives;
- An episode is said to be a false positive if `FEM-ASC` considers it frequent, but it is actually not. We note FP the number of false positives;
- An episode is said to be a false negative if it is discovered by the exact method but missed by `FEM-ASC`. We note FN the number of false negatives;
- The *precision* p and *recall* r are defined as

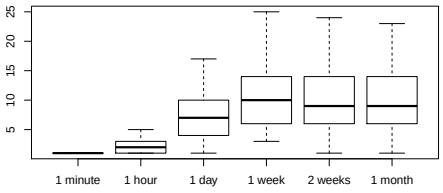
$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN}$$

We evaluate the influence of three parameters on the scalability and quality of `FEM-ASC`: the size of the window of interest T_W (see figure 4), the maximal

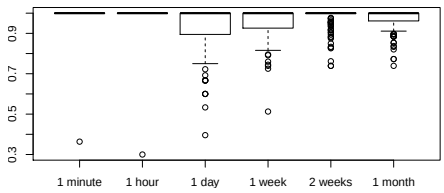
[1] http://ailab.wsu.edu/casas/datasets/aruba.zip



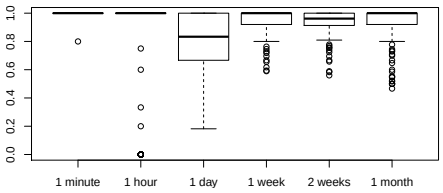
(a) Size of the lattice



(b) Size of RMN

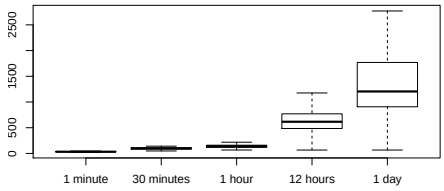


(c) Precision of the list of frequent episodes

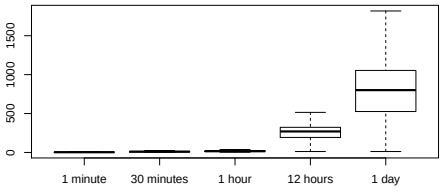


(d) Recall of the list of frequent episodes

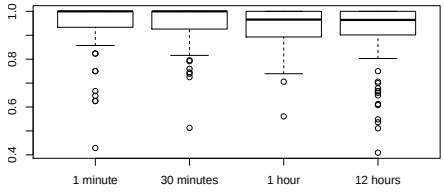
Fig. 4: Influence of the window length T_W



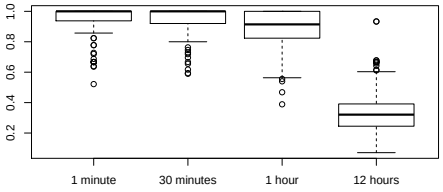
(a) Size of the lattice



(b) Size of RMN

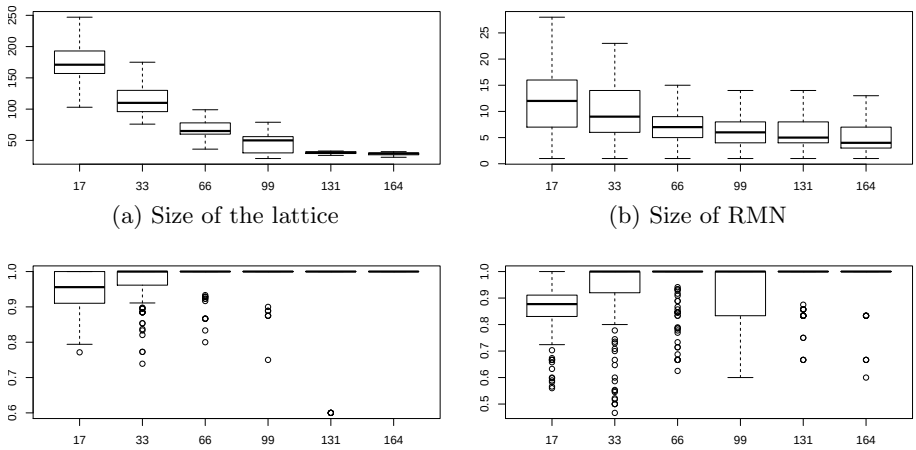


(c) Precision of the list of frequent episodes



(d) Recall of the list of frequent episodes

Fig. 5: Influence of the episode duration T_{ep}



(a) Size of the lattice

(b) Size of RMN

(c) Precision of the list of frequent episodes

(d) Recall of the list of frequent episodes

Fig. 6: Influence of the minimal support S_{min}

duration of the occurrences T_{ep} (figure 5), and the minimal support threshold S_{min} (figure 6). Table 3 details the experimental settings, and we observe the size of the lattice (subfigures (a)), the length of the RMN list (subfigures (b)), the precision (subfigures (c)) and recall (subfigures (d)). For each evaluation metric, we plot the boxplot representing the distribution of the measures throughout the processing of the dataset.

For every setting, the size of the lattice and RMN list remains mostly stable throughout the execution. The variations are linked to the varying event densities, but not to the size of the history. Both precision and recall are usually very close to 100%, with occasional lower values. The performance in terms of precision and recall do not seem significantly influenced by the parameters investigated. However, the resources required for the processing and mining of the dataset are: longer windows, longer episodes durations and lower support thresholds result in a higher number of interesting episodes and thus increase the size of the lattice. The efficiency of the RMN list for the reduction of the search space for the monitoring of new occurrences is directly related to the maximal duration of the considered occurrences, and is thus the main scalability limit.

6 Conclusion

In this paper, we introduce `FEM-ASC`, an algorithm for the discovery and the monitoring of frequent episodes in a stream of events. If its heuristic construction offers no guaranty on the quality of the approximated support, it shows in practice good results on real-life datasets with progressive distribution changes.

This work is however simply at an early development stage, and requires further experimentation. The two strategies for the removal of outdated nodes

need to be thoroughly compared. The performances of `FEM-ASC` should also be assessed in different contexts, and in the presence of different kinds of concept drifts. The influence of $\frac{S_{min}}{S_{sub}}$ should also be assessed on both the scalability, the recall, and the reaction speed to concept drifts.

Acknowledgement. This work is supported by the NanoSTIMA Project: Macro-to-Nano Human Sensing: Towards Integrated Multimodal Health Monitoring and Analytics/NORTE-01-0145-FEDER-000016 which is financed by the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 Partnership Agreement, and through the European Regional Development Fund (ERDF)

References

1. Casas-Garriga, G.: Discovering unbounded episodes in sequential data. In: European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). pp. 83–94 (2003)
2. Cook, D.J.: Learning setting-generalized activity models for smart spaces. IEEE Intelligent Systems 27(1), 32–38 (2012)
3. Gama, J.: A survey on learning from data streams: current and future trends. Progress in Artificial Intelligence 1(1), 45–55 (2012)
4. Gan, M., Dai, H.: Detecting and monitoring abrupt emergences and submergences of episodes over data streams. Information Systems 39, 277–289 (2014)
5. Lin, S., Qiao, J., Wang, Y.: Frequent episode mining within the latest time windows over event streams. Applied Intelligence 40(1), 13–28 (2014)
6. Mannila, H., Toivonen, H.: Discovering generalized episodes using minimal occurrences. In: 2nd International Conference on Knowledge Discovery and Data Mining. pp. 146–151 (1996)
7. Mannila, H., Toivonen, H., Inkeri Verkamo, A.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
8. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovering frequent episodes in sequences. In: International Conference on Knowledge Discovery and Data Mining (KDD). pp. 210–215 (1995)
9. Patnaik, D., Laxman, S., Chandramouli, B., Ramakrishnan, N.: Efficient episode mining of dynamic event streams. In: IEEE International Conference on Data Mining (ICDM). pp. 605–614 (2012)
10. Soulas, J., Lenca, P.: Periodic episode discovery over event streams. In: Progress in Artificial Intelligence - Portuguese Conference on Artificial Intelligence (EPIA). pp. 547–559 (2015)
11. Tatti, N., Cule, B.: Mining closed episodes with simultaneous events. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1172–1180 (2011)
12. Zhou, W., Liu, H., Cheng, H.: Mining closed episodes from event sequences efficiently. In: Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD). pp. 310–318 (2010)
13. Zhu, H., Wang, P., He, X., Li, Y., Wang, W., Shi, B.: Efficient episode mining with minimal and non-overlapping occurrences. In: 10th IEEE International Conference on Data Mining. pp. 1211–1216 (2010)