A Prototype for Maritime Event Forecasting

Elias Alevizos[1] and Alexander Artikis[2,1]

[1] Institute of Informatics & Telecommunications, NCSR Demokritos
[2] Department of Maritime Studies, University of Piraeus

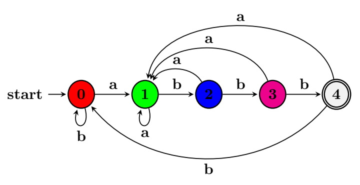
1 Introduction

We have built a prototype for complex event forecasting and applied it to the maritime domain [1]. The problem may be stated as follows: given a stream of input events and a pattern defining relations between such events, in the form of a regular expression, the goal is to estimate at each new event arrival the number of future events that we will need to wait for until the expression is satisfied, and therefore a match be detected.

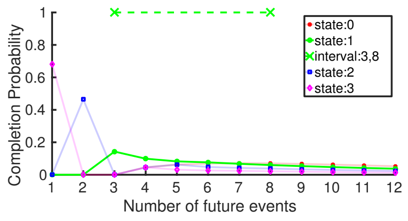
2 Approach

Event patterns are first converted to deterministic finite automata (DFA) through standard conversion algorithms. As an example, see Fig. 1a, which depicts the DFA for the pattern $R = a \cdot b \cdot b \cdot b$, i.e., an occurrence of a must be followed by three occurrences of b. Next, we derive a Markov chain that will be able to provide a probabilistic description of the DFA's run-time behavior. If the input events are independent and identically distributed (i.i.d.), then there is a direct mapping of the states of the DFA to states of a Markov chain and the transitions of the DFA to transitions of the Markov chain. The transition probabilities of the Markov chain are the occurrence probabilities of the various event types. If the input events are dependent on some of the previous events seen in the stream, i.e., the stream is generated by an m^{th} order Markov process, we perform a more complex transformation. The transition probabilities are then conditional probabilities on the event types. We call such a derived Markov chain a Pattern Markov Chain (PMC) of order m and denote it by PMC_R^m, where R is the initial pattern and m the assumed order of the Markov process. After constructing a PMC, we can use it in order to calculate the so-called *waiting-time* distributions. Given a specific state of the PMC, a *waiting-time* distribution gives us the probability of detecting a full match of the original regular expression in k events from now. Forecasts are in the form of intervals, like $I = (start, end)$. The meaning is that the DFA is expected to reach a final state sometime in the future between $start$ and end with probability at least some constant threshold θ_{fc} (provided by the user). These intervals are estimated by a single-pass algorithm that scans a waiting-time distribution and finds the smallest (in terms of length) interval that exceeds this threshold. See, e.g., Fig. 1b, which shows distributions for the states of the DFA of Fig. 1a when $m = 0$. The dashed green line is the forecast interval produced when the DFA is in state 1, with $\theta_{fc} = 50\%$, i.e., this is the smallest interval whose probability is above 50%.

We implemented a forecasting system, Wayeb, based on Pattern Markov Chains. Algorithm 1 presents in pseudo-code the steps taken for recognition and forecasting.



(a) DFA.



(b) Waiting-time distributions.

Fig. 1: DFA and waiting-time distributions for $R = a \cdot b \cdot b \cdot b$, alphabet $\Sigma = \{a, b\}$, $m = 0$.

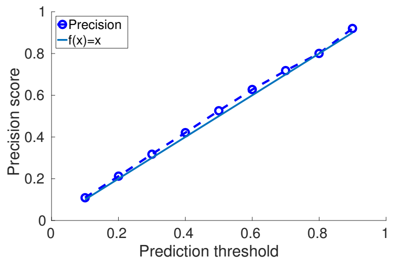
ALGORITHM 1: Forecasting algorithm

```
Input: Stream $S$, pattern $R$, order $m$, maximum spread $ms$, forecasting threshold $P_{fc}$
Output: For each event $e \in S$, a forecast $I = (start, end)$
1  $DFA_{\Sigma^* \cdot R}$ = BuildDFA($R$, $m$);
2  $PMC_R^m$ = WarmUp($S$, $DFA_{\Sigma^* \cdot R}$);
3  $F_{table}$ = BuildForecastsTable($PMC_R^m$, $P_{fc}$, $ms$);
4  $CurrentState$ = 0;
5  $RunningForecasts$ = $\varnothing$;
6  repeat
7      $e$ = RetrieveNextEvent($S$);
8      $CurrentState$ = UpdateDFA($DFA_{\Sigma^* \cdot R}$, $e$);
9      if $CurrenState$ not final then
10         $I$ = $F_{table}(CurrentState)$;
11         $RunningForecasts = I \cup RunningForecasts$
12     else
13         UpdateStats($RunningForecasts$);
14         $RunningForecasts$ = $\varnothing$;
15     end
16 until $true$;
```

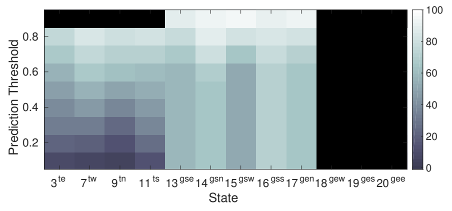
Wayeb reads a given pattern R in the form of a regular expression, transforms this expression into a NFA and subsequently, through standard determinization algorithms, the NFA is transformed into a *m-unambiguous* DFA (line 1 in Algorithm 1). For the task of event recognition, only this DFA is involved. At the arrival of each new event (line 7), the engine consults the transition function of the DFA and updates the current state of the DFA (line 8). Note that this function is simply a look-up-table, providing the next state, given the current state and the type of the new event. Hence, only a memory operation is required.

There are three metrics that we report in order to assess our module's performance and the quality of its forecasts:

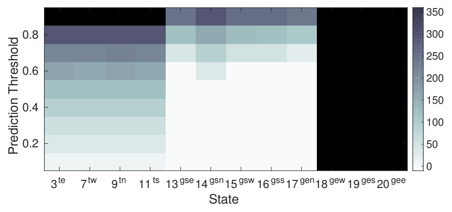
- $Precision = \frac{\#\ of\ correct\ forecasts}{\#\ of\ forecasts}$. At every new event arrival, the new state of the DFA is estimated (line 8 of Algorithm 1). If the new state is not a final state, a new forecast is retrieved from the look-up-table of forecasts (line 10). These forecasts are maintained in memory (line 11) until a full match is detected. Once a full match



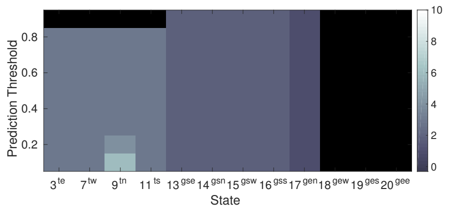
(a) Precision (all states).



(b) Precision (per state).



(c) Spread (per state).



(d) Distance (per state).

Fig. 2: Results for the pattern $Turn \cdot GapStart \cdot GapEnd \cdot Turn$ with $m = 1$.

is detected, we can estimate which of the previously produced forecasts are satisfied, in the sense that the full match happened within the interval of a forecast (line 13). These are the correct forecasts. All forecasts are cleared from memory after a full match (line 14).

– $Spread = end - start$.
– $Distance = start - now$. This metric captures the distance between the time the forecast is made (now) and the earliest expected completion time of the pattern. Note that two intervals might have the same spread (e.g., $(2, 2)$ and $(5, 5)$ both have $Spread$ equal to 0) but different distances (2 and 5, assuming $now = 0$).

$Precision$ should be as high as possible. With respect to $Spread$, the intuition is that, the smaller it is, the more informative the interval. For example, in the extreme case where the interval is a single point, the engine can pinpoint the exact number of events that it will have to wait until a full match. On the other hand, the greater the $Distance$, the earlier a forecast is produced and therefore a wider margin for action is provided. Thus, "good" forecasts are those with high precision (ideally 1.0), low spread (ideally 0) and a distance that is as high as possible (ideal values depend on the pattern). These metrics may be calculated either as aggregates, gathering results from all states (in which case average values for $Spread$ and $Distance$ over all states are reported), or on a per-state basis, i.e., we can estimate the $Precision$, $Spread$ and $Distance$ of the forecasts produced only by a specific state of the DFA.

3 Demo for Maritime Event Forecasting.

Wayeb was tested against a real-world dataset that came from the field of maritime monitoring. When sailing at sea, (most) vessels emit messages relaying information about their position, heading, speed, etc.: the so-called AIS (automatic identification system) messages. AIS messages may be processed in order to produce a compressed trajectory, consisting of critical points, i.e., important points that are only a summary of the initial trajectory, but allow for an accurate reconstruction [2]. The critical points of interest for our experiments are the following:

– Turn: when a vessel executes a turn.
– $\mathit{GapStart}$: when a vessel turns off its AIS equipment and stops transmitting its position.
– GapEnd: when a vessel turns on its AIS equipment back again (a $\mathit{GapStart}$ must have preceded).

We used a dataset consisting of a stream of such critical points from ≈ 6.500 vessels, covering a 3 month period and spanning the Greek seas. Each critical point was enriched with information about whether it is headed towards the northern, eastern, southern or western direction. For example, each Turn event was converted to one of $\mathit{TurnNorth}$, $\mathit{TurnEast}$, $\mathit{TurnSouth}$ or $\mathit{TurnWest}$ events. We show results from a single vessel, with ≈ 50.000 events.

Figure 2 shows results for the pattern

$$\mathit{Turn} \cdot \mathit{GapStart} \cdot \mathit{GapEnd} \cdot \mathit{Turn} \tag{1}$$

where Turn is shorthand notation for

$$(\mathit{TurnNorth} + \mathit{TurnEast} + \mathit{TurnSouth} + \mathit{TurnWest})$$

with $+$ denoting the OR operator. Similarly for $\mathit{GapStart}$ and GapEnd. With this pattern, we would like to detect a sequence of movements in which a vessel first turns (regardless of heading), then turns off its AIS equipment and subsequently re-appears by turning again. Communication gaps are important for maritime analysts because they often indicate an intention of hiding (e.g., in cases of illegal fishing in a protected area).

The aggregate precision score (Figure 2a) is very close to the baseline performance. This precision score is calculated by combining the forecasts produced by all states of the PMC. In order to better understand Wayeb's behavior, a look at the behavior of individual states could be more useful. Figures 2b – 2d depict image plots for various metrics against both the forecast threshold and the state of the PMC. The metrics shown are those of precision (on the recognized matches), spread and distance. In each such image plot the y axis corresponds to the various values of P_{fc}. The x axis corresponds to the states of the PMC. The x axis shows how advanced we are in the recognition process, when moving from one state to the next. The black areas in these plots are "dead zones", meaning that, for the corresponding combinations of P_{fc} and state, Wayeb fails to produce forecasts (i.e., it cannot guarantee, according to the learned transition probabilities, that the forecast intervals will have at least P_{fc} probability of being satisfied). On the contrary, areas with light colors are "optimal", in the sense that they have high precision, low spread (the colorbar is inverted in the spread plots) and high distance in their respective plots. A look at the per-state plots reveals something interesting (Figures 2b, 2c, 2d). Note that, in order to avoid cluttering, we have removed duplicate states

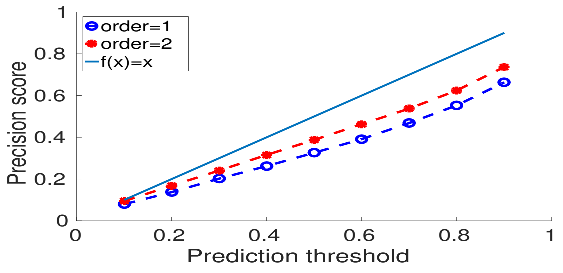


Fig. 3: Results for the pattern $TurnNorth \cdot (TurnNorth + TurnEast)^{*} \cdot TurnSouth$.

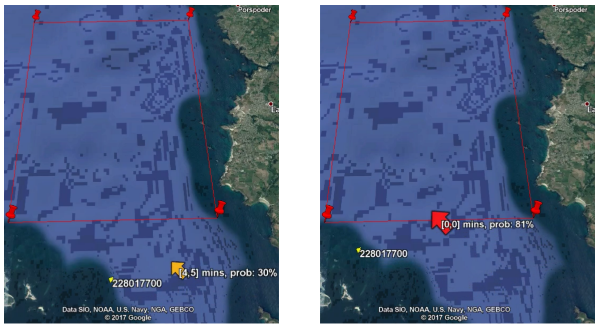
from the per-state plots. In addition, the superscript of each state in the x axis shows the last event seen when in that state. For example, the superscript te corresponds to $TurnEast$, tw to $TurnWest$, tn to $TurnNorth$ and ts to $TurnSouth$ (states 3, 7, 9 and 11 respectively). Similarly for $GapStart$ for which superscripts start with gs (states 13–16) and for $GapEnd$ (ge and states 17–20). These per-state plots show that there is a distinct "cluster" of states (13–17) which exhibit high precision scores for all values of P_{fc} (Figure 2b) and small spread for most values of P_{fc} (Figure 2c). Therefore, these states constitute what might be called "milestones" and a PMC can help in uncovering them. By closer inspection, it is revealed that states 13–16 are visited after the PMC has seen one of the $GapStart$ events (we remind that $GapStart$ is a disjunction of the four directional sub-cases). Moreover, $GapEnd$ events are very likely to appear in the input stream right after a $GapStart$ event, as expected, since during a communication gap (delimited by a $GapStart$ and a $GapEnd$), a vessel does not emit any messages. State 17, which also has a similar behavior, is visited after a $GapEndNorth$ event. Its high precision scores are due to the fact that, after a $GapEnd$ event, a $Turn$ event is very likely to appear. It differs from states 13–16 in its distance, as shown in Figure 2d, which is 1, whereas, for states 13–16, the distance is 2. On the other hand, states 18–20, which correspond to the other 3 $GapEnd$ events, fail to produce any forecasts. The reason is that there are no such $GapEnd$ events in the stream, i.e., whenever this vessel starts transmitting again after a Gap, it is always headed towards the northern direction.

Figure 3 shows results for the pattern

$$TurnNorth \cdot (TurnNorth + TurnEast)^{*} \cdot TurnSouth$$

This pattern is more complex since it involves a *star closure* operation on a nested *union* operation. It attempts to detect a rightward reverse of heading, in which a vessel is initially heading towards the north and subsequently starts a right turn until it ends up heading towards the south. Such patterns can be useful in detecting maneuvers of fishing vessels.

Figure 3 shows that a model with m=1 is unable to approximate well-enough the correct waiting-time distribution. Increasing the order to m=2 improves the precision score, but it still remains under the baseline performance. One could attempt to further increase the value of m, but this would substantially increase the cost of building the PMC. For $m = 1$, the generated PMC has $\approx$ 30 states. For $m = 2$, this number rises to



(a) Same vessel, two routes, early snapshot. (b) Same vessel, two routes, late snapshot.

Fig. 4: *withinArea* event (Google Earth).

≈ 600 and the cost of creating an unambiguous DFA and then its corresponding PMC rises exponentially. When stationarity is assumed (as in our case) and the model does not need to be updated online, an expensive model can be tolerated.

We also demonstrate our method on the so-called *withinArea* event, which reports whether a vessel is located within the boundaries of a designated area, defined as a polygon, e.g., a protected area or a port. Forecasting arrival times at ports can help reduce the operating cost and emissions footprint of ships, as they are typically required to wait outside a port when there is a high traffic volume. Maritime monitoring companies currently rely on manual information in the AIS messages to forecast arrival at ports, which is very often wrong. Thus a forecasting method that does not rely on humans is highly desirable. In order to produce forecasts in terms of time, we sample each trajectory at regular intervals. We finally enrich each message with spatial information about whether the vessel is located within the area, whether it is close to the area and whether its heading points towards a direction that intersects with the area. Fig. 4a and 4b show the behavior of our prototype, where two different routes of the same vessel are shown, at different times, and the red rectangle is the area of interest. The left arrow corresponds to a route that never crosses the area and the right arrow a route that does cross it. The size of the arrow is proportional to the probability of entering the area and its color becomes more red as the $start$ of the forecast interval becomes smaller, i.e., red indicates that the vessel will enter the area very soon. As can be seen, the left route never produces forecasts because the model has learned that this route never leads to a *withinArea* event, whereas, for the right route, as the vessel approaches the area (compare Fig. 4a and 4b) the forecasts become more confident and focused.

References

1. Alevizos, E., Artikis, A., Paliouras, G.: Event forecasting with pattern markov chains. In: Proceedings of DEBS. pp. 146–157. ACM (2017)
2. Patroumpas, K., Alevizos, E., Artikis, A., Vodas, M., Pelekis, N., Theodoridis, Y.: Online event recognition from moving vessel trajectories. GeoInformatica (2016)