Towards *Neural Situation Evolution Modeling:* Learning a Distributed Representation for Predicting Complex Event Sequences

Andrea Salfinger
Department of Cooperative Information Systems
Johannes Kepler University Linz
Linz, Austria
andrea.salfinger@cis.jku.at

Lauro Snidaro
Department of Mathematics, Computer Science and Physics
University of Udine
Udine, Italy
lauro.snidaro@uniud.it

***Abstract*—In real-world monitoring tasks, a *situation* can be understood as a sequence of causally related events of interest. In road traffic control, such a situation could be a rear-end collision at the end of a traffic jam, which worsens congestion and requires clearing operations and potentially rerouting. Whereas conventional event sequence prediction focuses on sequences of individual events $\langle e_1, \ldots, e_n \rangle$, evolving situations thus can be conceived as sequences of states composed of multiple concurrent events, i.e., *complex events*: $\langle \{e_1, \ldots, e_m\}, \ldots, \{e_l, \ldots, e_n\} \rangle$. *Situation (evolution) prediction* thus requires learning a transition model for these complex events to provide the expectations for potential successor event types. In previous work, this was represented by a Markov Chain defined on the observed complex events. However, using the entire event composite as "atomic" situation state representation does not allow capturing patterns between its individual events (e.g., events of type "accident" share similar successor event types across different event composites), nor generalizing behaviors between similar event types or incorporating additional features. Hence, we propose a *neural modeling* approach to learn a *distributed representation* of a given situation dataset. By encoding the input states as conjunction of their individual comprised events, the devised model can learn associations (i.e., enable an "information flow") between individual event types, allowing to capture similar behaviors across different situations. We test our approach on both synthetic and real-world datasets.**

I. Introduction

Motivation. In many control center domains, such as road traffic control, human operators need to track and forecast the development of situations composed of an evolving set of related events. Human operators typically rely on their experience for forecasting a situation's evolution as a basis for their decision making and planning of appropriate response and mitigation actions. Modern-day control centers' data recording infrastructures, however, would enable complementing their human intuition with empirically grounded forecasting models, developed from previously recorded situations. Hence, this would allow deriving rich models factoring in a variety of information (such as subtle location- and time-specific patterns) to support *predictive situation management* [1].

Challenges. Whereas the recent advances in machine learning have led to elaborate forecasting models in a variety of application domains, these have mainly focused on predicting the evolution of individual *objects and events* from a fixed set of input features, e.g. measurements, such as [2], [3]. Conversely, *situations* are defined as *sets of interrelated entities* [4]. Their development over time thus corresponds to the *joint evolution* of the set of comprised entities (see Fig. 1). Previous work [5] formulated situation evolution prediction as a *sequence prediction problem* defined on sequences of *sets* of *concurrent events*, i.e., sequences of *complex events* (CEs).

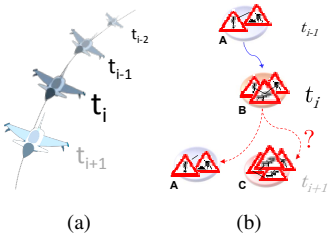

Figure 1: *Object tracking* (a) vs. *situation tracking* (b).

Open Issues. However, this approach encoded CEs with "atomic" tokens: each CE is represented by a *single symbol* from the *alphabet* for encoding the observed situations[1]. This alphabet hence can be conceived as the state space of an underlying Markov Chain (MC), which allows estimating the transition probability between the different symbols, i.e., *the situations' states*, from the transition frequencies observed in the database. Consequently, this entails the following shortcomings:

▷ **Problem 1** (*sparsity*): Since the situations' states are encoded by single tokens (each representing a particular joint event type composite), this triggers combinatorial explosion. We may observe many slightly different combinations of event types $e_1, \ldots, e_n$ (both during model learning and operation), which thus yield different symbolic representations. This leads to *sparsity* in the sequence and MC representation of the dataset, as we may have only few observations for each token (in particular for tokens corresponding to large event com-

[1]For example, the four potential situation states shown in Fig. 1b correspond to three distinct event type constellations (indicated by the differently colored ellipses), thus the prediction problem could be encoded by three different symbols $\{A, B, C\}$.

posites or rare event types), thus resulting in rather unreliable probability estimates.

▷ **Problem 2** (*out-of-vocabulary words*): Event type composites encountered at operation time but not during model learning conform to "*out-of-vocabulary words*". As the MC has not "tokenized" this event type composite before, i.e., does not have a corresponding state, no predictions can be made.

▷ **Problem 3** (*"shared" event type associations*): Furthermore, the MC cannot generalize from similar observations. The model's representation using the entire *joint* event type composite as basic token or symbolic representation does not allow to generalize event-level patterns between similar event type composites, i.e., does not allow to exploit similar, "shared" event type associations. For instance, situations involving the event type "accident" may require similar clearing works and potentially trigger the formation of traffic jams, thus likely will share some event types across their successor states.

▷ **Problem 4** (*additional features*): Finally, the MC does not inherently support incorporating additional non-symbolic features, such as location or time. However, such properties may indeed influence a situation's evolution (e.g., accidents during rush hour may lead to more severe traffic conditions), thus, likely would impact the transition probability distribution.

Goals. We argue that the root cause of ▷ **Problems 1 – 4** lies in the coarse state representation adopted by the MC, representing a situation's state by a single symbol denoting the entire CE: This only allows capturing patterns between *entire* states, but does not allow to leverage the associations among and between the states' *individual elements*.

Contributions. Therefore, we adopt the idea of a *distributed representation* [6]: We propose representing a situation's state by its individual composing events and fitting a neural network to these decomposed CE sequences to learn the probabilistic evolutionary relationships between the individual situational elements. Inspired by *neural language modeling*, this approach thus learns an *implicit representation* of the evolution patterns underlying the situation dataset. Unlike conventional language modeling, the learned model forms a distributed representation *by design*[2], since the composite states have been decomposed by splitting up ("distributing") the original alphabet from one token for each *set* of events (i.e., CE) into one token for each *individual event*. This enables the network to learn a representation individually capturing the individual elements' impact on a situation's evolution. The learned neural evolution model provides *soft* decision support to the human operators by outputting the individual estimated expectations for observing each event type in the next evolution step. This accounts for the fact that road traffic incidents frequently evolve according to common patterns, however, alternative – and often more critical – courses of events are possible and need to be prepared for. We empirically analyze the learned situation evolution models in experiments on both generated and real-world datasets.

[2]Distributed representations of tokens in language models are only achieved by means of the additional preprocessing step of training embedding layers.

II. Related Work

As defined by the well-known JDL data fusion model [4], *situations* represent high-level information fusing the joint behavior of a set of interrelated elements. By incorporating the interplay of the set of related elements, valuable information can be gleaned (e.g., the estimated duration of a traffic jam may not only depend on the features of the traffic jam itself, but also on associated co-located events, such as roadworks), which motivates the need for predictive models operating on such situation-level input. Interestingly, however, only few approaches have tackled this problem so far. The problem of situation tracking has been formalized in [7], proposing logical reasoning for inferring potential developments of a monitored situation. Such qualitative situation prediction has also been approached with Colored Petri Nets [8]. These approaches to situation (evolution) prediction base on formal logics, thus infer potential future developments by determining which courses of events could be reached from the currently observed state, but do not provide any any estimates of their probabilities derived from empirical data. Therefore, extensions of classical logics have been proposed to support probabilistic reasoning based on empirical evidence and uncertainty, such a Markov Logic Networks (MLNs), which have been adopted for the fusion of uncertain sensory and contextual information for maritime situational awareness [9]. The MLN framework has been employed to both encode a priori and contextual knowledge [10] and to fuse evidence from multiple sources, possibly reasoning over incomplete data. Knowledge is expressed by formulas in first-order logic with the possibility of associating to each of them a level of uncertainty encoded by a weight factor. In [11], empirical data was leveraged for parametrizing situation models represented by Hidden Markov Models, thereby also supporting evolution prediction, which have been employed for tracking overtaking situations in a driver-assistance system. However, prediction accuracy strongly depends on the suitable discretization of the state space (number of states etc.), which has to be parametrized manually, and the states' actual semantics needs to be determined by manual analysis. To enable an automated mining of a such a discretized situation state space from structured datasets, dedicated representations and approaches have been developed in [12], based on which [5] formalized *situation evolution prediction* as a sequence prediction problem. We will briefly recap these conceptualizations in the following, laying the preliminaries for the present work.

III. Background

Situation Modeling. Fig. 2 shows a road traffic situation as recorded by human control center operators following the European Union's "Data Exchange for Traffic Management and Travel Information" standard, DATEX II [13]. From the high-level perspective of a human control center operator, such situations can be conceived as sets of causally related events of different types (e.g., traffic jams, accidents, roadworks etc.) that evolve over time, i.e., receive updates on their current state (which might change their *event type* or other properties

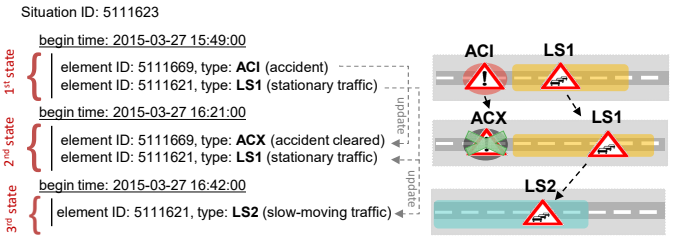

Figure 2: The evolution of a road traffic situation through three different states.

such as location etc.). The individual situational elements' evolution thus can be characterized by a sequence of their different states (for example, the element with ID 5111669 can be characterized by the sequence ⟨ACI, ACX⟩ to describe its evolution from the accident to the cleared accident site). Consequently, the entire situation's evolution corresponds to the *joint development* of its comprised events. For example, the course of events shown in Fig. 2 could be characterized by the following sequence of sets of concurrent events[3]: ⟨{ACI,LS1}, {ACX,LS1}, {LS2}⟩, i.e., a sequence of the three "event type composites" (i.e., states) it passes through. Similarly, a situation starting with slowly moving traffic (LS2), which caused an accident (ACI) and worsened the traffic jam to stationary traffic (LS1) that still persisted when the accident site was already cleared (ACX), could be expressed as ⟨{LS2}, {ACI,LS1}, {ACX,LS1}⟩. Predicting a currently observed situation's further evolution thus translates to the following sequence prediction problem: ⟨{LS2}, {ACI,LS1},?⟩.

To tackle this problem, previous work [5] thus denoted each such set of concurrent events (and potentially their interrelations) by a CE [9], [14] describing this state, termed *Situation State Type* ($\mathcal{SS}^{\mathcal{T}}$). Hence, these $\mathcal{SS}^{\mathcal{T}}s$ can serve as "token representation" (i.e., *alphabet*) for encoding a situation $\mathcal{S}^{\mathcal{I}}$ in a high-level fashion by the sequences of $\mathcal{SS}^{\mathcal{T}}s$ it has evolved through, i.e., $\mathcal{S}^{\mathcal{I}} = \langle \mathcal{SS}^{\mathcal{T}}_1, \ldots, \mathcal{SS}^{\mathcal{T}}_n \rangle$. Consequently, the probability distribution of potential successor event states, given the already observed sequence, can be mathematically expressed in terms of the following posterior distribution:

$$P(\mathcal{SS}^{\mathcal{T}}_{k+1} | \langle \mathcal{SS}^{\mathcal{T}}_1 \ldots, \mathcal{SS}^{\mathcal{T}}_k \rangle) \tag{1}$$

This sequence prediction problem was approached by fitting a first-order MC model to the situation dataset (assuming "memorylessness", i.e., that the Markov property holds), taking the set of $\mathcal{SS}^{\mathcal{T}}s$ encountered in the dataset as the set of possible situation states and thus the *state space* of the MC [5].

Language Modeling. Problems of the form in (1), i.e., determining the probabilities of the next symbol(s) conditioned on the already observed sequence, can be conceived as a sequence-to-sequence prediction problem familiar from Natural Language Processing (NLP). In particular, statistical language modeling seeks to determine the probability of the next word w_t given its context (such as the preceding $n-1$ words) from a given corpus: $P(w_t | w_{t-n+1}^{t-1})$. Recently, neural language modeling [15] has become the predominant technique to learning these distributions from large-scale corpora [16]. Neural sequence-to-sequence prediction has also been applied to event sequence prediction problems [17], focusing on sequences of individual successive events. Thus, language modeling and event sequence prediction essentially model sequences of individual "tokens" and events, respectively. Conventional event sequence prediction corresponds to predicting the individual situation elements' evolution (i.e., in terms of our previous examples, sequences ⟨ACI, ACX⟩ representing the accident's development and ⟨LS1, LS1, LS2⟩ representing the traffic jam's development), but does not consider jointly evolving *sets* of events.

[3]Notation: Sets are denoted by {...} and sequences by ⟨...⟩.

IV. A Model for Learning a Distributed Situation Representation

Core Idea. Concluding, existing approaches to situation evolution modeling, language modeling and event sequence prediction operate on sequences of *individual, atomic tokens*, which are either individual events, or tokens representing the *entire* CE describing the situation's state. However, by employing a symbol notation which uses the entire event type composite, i.e., $\mathcal{SS}^{\mathcal{T}}$, as "atomic" symbol, models learned on these representations are incapable of dealing with the problems outlined in Sec. I, which would require a more fine-grained modeling of these states in terms of the individual elements making up the CEs. Thus, we need to find a representation capable of "tearing apart" this atomic situation state representation. Therefore, we next develop a *distributed* representation based on a more fine-grained modeling of these states: Instead of using an alphabet defined on the observed CEs, we represent each state by a *composite token* (which thus represents the major difference to conventional language modeling) defined on the alphabet of possible event types, and learn the transition probabilities between succeeding states by a neural network modeling the situation dataset, obtained in an unsupervised manner analogously to *neural language modeling*. Note that our focus here is exclusively on learning the *predictive model*, assuming that the detected states are given (i.e., the set of interrelated event types). This is motivated by the fact that *situation state detection* and *situation tracking* can effectively be handled by dedicated *Complex Event Processing* systems, as proposed in previous work [9], [10], [18].

Stateless Approach. For learning a transition model analogous to a first-order MC (which only considers the *current* state as relevant for the probabilities of its successor event types, but not its history), our goal could be phrased as learning a (stochastic) mapping function for the current situation state's descriptor s_t, which maps s_t to some probability distribution on its potential successor states:

$$f(s_t) = P(s_{t+1}), \tag{2}$$

whereby, as opposed to the MC-based model, these states s are represented in a distributed fashion. Thus, instead of being simply expressed by one particular $\mathcal{SS}^{\mathcal{T}}$, a situation state s

at time t, composed of m event types, would be expressed in terms of its individual constituents:

$$s_t = \{e_{t,1}, \ldots, e_{t,m}\} \tag{3}$$

Hence, our goal would be learning the posterior event distribution after observing the present state s_t:

$$p(s_{t+1}|s_t) = p\left(\{e_{t+1,1}, \ldots, e_{t+1,n}\}|\{e_{t,1}, \ldots, e_{t,m}\}\right) \tag{4}$$

Since we may expect one to multiple successor events $e_{t+1,i}$ to occur, our problem can be conceived as a *multi-label classification* problem. Consequently, we formulate our prediction goal as learning n individual Bernoulli distributions $p(e_{t+1,i})$ for a domain encompassing n potential event types[4], each expressing the individual likelihood of observing event type e_i in the next evolution step $t+1$:

$$p(s_{t+1}|s_t) \sim p(e_{t+1,1}), \ldots, p(e_{t+1,n}) \tag{5}$$

These quantities should be computed by the sought-after stochastic mapping function, which should output the expectations of observing each of the n individual event types e in the next situation state s_{t+1}:

$$f\left(\{e_{t,1}, \ldots, e_{t,m}\}\right) = p(e_{t+1,1}), \ldots, p(e_{t+1,n}) \tag{6}$$

Such a stochastic mapping function f can be obtained by training a feedforward neural network with a suitable output layer activation and loss function on our situation dataset, which corresponds to finding a parametrization θ for this function f that minimizes the error between our model's predictions $f(s_t;\theta)$ and the actually observed next state s_{t+1}. To represent the varying sets of input events, our input $\boldsymbol{i}$ at each time step t is encoded as a binary vector of size $|E| = n$, E being the set of event types possible in our domain, i.e., $\boldsymbol{i} \in \{0,1\}^n$. The input signature for a particular s_t is a vector having 1 at the index positions of the event types observed in s_t, and 0 elsewhere. Without loss of generality, we may allocate several "slots" per event type, if it is possible that multiple events of the same type may occur in a domain: The first event of type e_q would be allocated at position q, the second on $q+1$ etc., whereby the number of provided slots represents the offset to the index of the next event type. We may incorporate additional (also non-symbolic) features by extending the input vector $\boldsymbol{i}$ accordingly (e.g., by also including the event's location, timing, or spatial extent features). Analogously to $\boldsymbol{i}$, the network's output layer $\boldsymbol{o}$ represents an index over the set of event types possible in this domain, each of its positions o_e conforming to the probability of observing an event of type e in the next evolution step, i.e., $\boldsymbol{o} \in [0,1]^n$. Thus, will use the logistic function as activation function on the output layer to transform the network's final activations into the range $[0,1]$, representing $p(e_{t+1,i})$. To obtain these estimates from training to predict our data set, we train our network based on *Maximum-Likelihood Estimation*, i.e., by minimizing *cross-entropy loss* [20]. During training, stochastic gradient descent (SGD) optimizes the network's parameters to maximize its fit to the given dataset, by minimizing the cross-entropy loss $\mathcal{L}$ based on the input-output pairs (s_t, s_{t+1}) given from our situation evolution sequences. Since we have decomposed the problem into predicting the individual event probabilities $p(e_{t+1,i})$, we aim to optimize each output event's probability independently, thus employ the *binary* cross-entropy loss [20] as follows:

$$\mathcal{L}\left(f\left(s_t;\theta\right), s_{t+1}\right) = -\sum_{e=1}^{n} y_e \; log(o_e) + (1-y_e) \; log(1-o_e), \tag{7}$$

where y_e represents the true label for event type e in the successor state s_{t+1} (1 if it is observed, otherwise 0), and o_e corresponds to its predicted probability $p(e_{t+1,i})$.

Stateful Approach. Whereas training a feedforward neural network allows learning a memoryless – or fixed-size memory – stochastic mapping function, we can overcome the limitations of the MC and the feedforward network (in terms of an a-priori fixed history horizon determined by the order of the MC, respectively the fixed context-size encoded in the input vector) by employing a stateful model that autonomously learns how much of the observed situation's history should be incorporated in its predictions, by learning the mapping:

$$f\left(\langle s_1, \ldots, s_t \rangle\right) = P(s_{t+1}) \tag{8}$$

This can be achieved by modeling situation evolution prediction as a *sequence-to-sequence prediction* problem using a Recurrent Neural Network (RNN). We can simply change our previous architecture to an RNN by replacing its hidden layer of feedforward neurons with a hidden layer of recurrent units, such as LSTM cells [21], which learn to store relevant aspects of a situation's *history* in their hidden cell state h. Hence, the resulting RNN essentially learns the following mapping of the situation's currently observed state s_t and its history accumulated in h:

$$f(\underbrace{\langle \underbrace{\{e_{1,1}, \ldots, e_{1,l}\}}_{s_1}, \underbrace{\ldots\ldots\ldots}_{s_2,\ldots,s_{t-1}}, \underbrace{\{e_{t,1}, \ldots, e_{t,m}\}}_{s_t} \rangle}_{h}) = f(s_t, h) = p(e_{t+1,1}), \ldots, p(e_{t+1,n}) \tag{9}$$

We feed in the entire evolution sequence observed up to the current state s_t to predict the expectations of the next tokens in s_{t+1}. Note that we understand t here merely as a sequential index, which does *not* imply equidistant time steps, but the time point where situation composition changes (or the "jump" to the next state in terms of the MC), i.e., its qualitative change.

Fig. 3 visually summarizes the architecture of the proposed model. By using a distributed representation based on the individual composing event types, information can flow from the individual event types, exploiting both *co-occurrence* as well as *sequential correlations* between the different event types. The co-occurrence of event types may increase the likelihood of specific successor event types (e.g., the co-occurrence of an accident and roadworks may increase the

[4]Thus, we do not explicitly account for potential cross-correlations between sets of successor events [19], which represents a direction for future work.

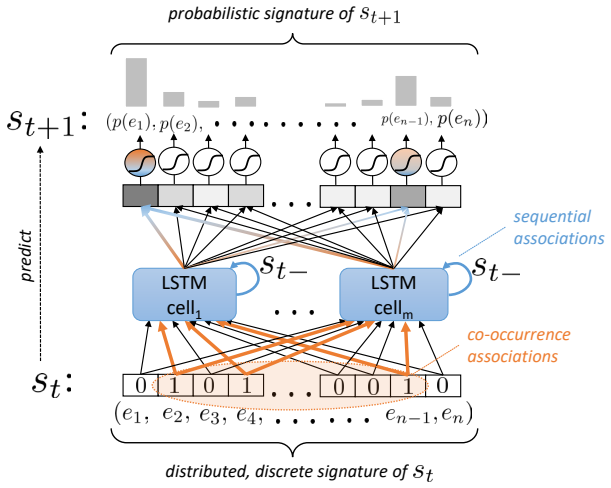

Figure 3: Architecture of the proposed model. Different model configurations have been tested yielding similar results on the datasets.

likelihood of observing a traffic jam as follow-up event), whereas in terms of sequential correlations, for example the occurrence of event type "accident" makes it likely to observe an event type "accident cleared" in the future. This contrasts with neural languages models, which, implied by the inherent structure of language operating on words as atomic units (i.e., tokens), solely capture sequential correlations. Conversely, our situation evolution model operating on event sets can also factor in the co-occurrence associations between simultaneously observed events. Note that our *distributed* representation has been achieved "by design", i.e., is directly encoded by our problem formulation, and does not refer to the mapping of a discrete representation to a continuous, lower-dimensional space, termed *embedding*, as common in NLP, like word2vec [22]. Since the cardinality of event type sets in situation monitoring domains typically is not comparable to the huge size of vocabularies common in NLP, we have decided to keep our initial architecture on the purely symbolic, i.e., discrete, level. However, the use of an embedding layer might be investigated in future work, in order to examine whether such an additional distributed representation may aid the model's generalization capabilities by capturing commonalities between event types (e.g., by allocating different types of accidents to the same vector subspace).

V. Experiments

We empirically examine our approach on a real-world situation dataset and devise controlled experiments for a systematic analysis of its representation capabilities. Our experiments are specifically designed to study whether our model is able of overcoming ▷ **Problems 1 – 4** characterized in Sec. I. Note that ▷ **Problem 2** (*out-of-vocabulary words*) is dealt with *by design*: As our input representation is now directly built on the domain's set of distinct event types (which is fixed a-priori), also event combinations not seen during model learning can be input which the model should generalize from similar situations, as we will examine.

A. Real-world Dataset

We apply our proposed approach on a road traffic dataset consisting of 12,082 situation and situation fragment records[5] logged by human control center operators in the form shown in Fig. 2, comprising 185 different event types. Only 21% of these are actually evolving situations that consist of more than one state, whereas the majority ends without having received any situation update[6]. Thus, we are dealing with a small[7] and highly imbalanced dataset (as common in many real-world domains), which also applies to the observed types of situational elements – whereas some of them occur rather frequently (such as traffic jams), others (like wrong-way drivers or vehicles on fire) naturally have extremely rare occurrences. Hence, this represents a challenging evaluation setting, and is particularly suited for investigating whether our model is capable of addressing ▷ **Problem 1** (*sparsity*). We preprocess our data by postfixing each sequence with a dedicated "stop" token (.), denoting that a situation has ended. Since the longest observed evolution sequence consists of ten different situation states, we thus train our model with sequences of length ten, padding shorter sequences with zeros. To assess the generalization capabilities of our model, we split our dataset into two disjunct periods: We train our models on a training split involving 10,195 situations (84% of the data) recorded between April and December, 2014. We then examine how our model performs on future situations, by testing its predictions on 1,887 situations (16% of the data) recorded between December 2014 and April 2015.

Model Configurations. We empirically verified that different architecture configurations produce *consistent* results, i.e., converge to the same overall results irrespective of initialization and network configuration: We used standard *keras* [23] hyperparameter settings and analyzed the standard deviation (σ) of the prediction results on our dataset produced by networks with one hidden layer comprising 1 – 7 LSTM cells. After training for 500 epochs, the average σ of the predicted probability per event type across these seven experiments is ≤ 0.0012. If we only consider predictions with a mean $p \geq 0.05$ (across the different models), i.e., the actual predictions (to rule out the large fraction of basically zero probability event types), we obtain an average σ of 0.042 between the different models' predictions. We also verified that different types of recurrent units (i.e., LSTMs, GRUs and simple RNNs) produce similar results (results omitted due to space constraints).

Evaluation. Fig. 4 shows the resulting predictions for various types of situations. As expected, the probability for ending in the current state (.) is always highest (since the vast majority of situations are non-evolving), but indeed varies for different

[5]Note that this dataset also comprises "situations" solely consisting of a single element, as each appearing traffic element is automatically assigned to a situation (since each appearing traffic element could potentially cause a more complex chain of events later on).

[6]This may be due to the sampling frequency of data collection, or incompletely recorded situations.

[7]Note, however, that we are dealing with a rather small number of discrete features only, which thus simplifies the learning problem.

input sequences. From an intuitive viewpoint, the resulting predictions appear feasible, i.e., events are predicted as we would typically expect based on our domain knowledge.

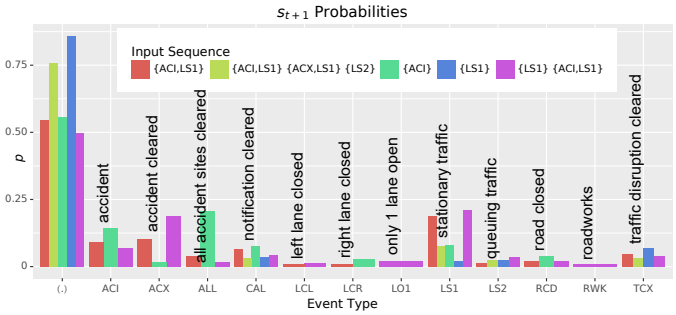

Figure 4: Event type distributions for various ACI situations [best viewed in color; only event types with $p > 0.05$ shown]. We observe how the distributions change with respect to the situation's composition and progress, despite the similar composing event types. Traffic jams ($\{\texttt{LS1}\}$, *blue*) are more likely to end without incurring additional events or updates than traffic jam situations which have already involved an accident (ACI, *red, violet*), as those more likely trigger additional situation updates. Similarly, a situation which has already progressed towards its clearance phase ($\langle\{\texttt{ACI,LS1}\}, \{\texttt{ACX,LS1}\}, \{\texttt{LS2}\}\rangle$, *yellow-green*), i.e., the accident site has been cleared (ACX) and the traffic jam has been dissolving from stationary traffic (LS1) to slowly moving traffic (LS2), is more likely to end than situations still in their accident phase (ACI, *red, green, violet*).

Besides qualitative inspection of results, a major challenge is how to systematically evaluate our model's learned distributions. Our model does not make hard decisions, but provides *soft decision support* by estimating each event type's expected probability of occurring in the next situation state, i.e., $p(e_{t+1,i}|\langle s_1,\ldots,s_t\rangle)$. Since our output of interest is the entire set of probability distributions (not just the most likely next event type, which would be always the "stop" token due to the imbalanced dataset), we would like to validate whether the estimated distributions conform to the empirical distribution observed in the data. Established metrics for comparing the related language models, such as the information theoretic-measure *perplexity* [16], which base on the idea of comparing models' *entropy* (the lower the entropy, the better the model characterizes the real-world data), are not applicable in our case. For comparing two different models' perplexity, both need to employ the *same vocabulary* (i.e., alphabet), whereas the $\mathcal{SS}^{\mathcal{T}}$-based model [5] and the distributed representation naturally use different alphabets, and we have changed the problem from a *multi-class classification problem* for the $\mathcal{SS}^{\mathcal{T}}$-based model to a *multi-label classification problem*.

To validate whether the learned probability distributions for the individual event types, $p(e_{t+1,i})$, reflect the empirical distributions observed in our test dataset $\hat{p}(e_{t+1,i})$, we compare the distributions learned by model with "ground truth" estimates $\hat{p}(e_{t+1,i})$, i.e., n-gram statistics explicitly computed from the evolution sequences observed in the test dataset. We do not a-priori fix the n-gram size, but use a situation's entire history as context for predicting a state s_{k+1} (i.e., employ n-grams of size k). Hence, we split each unique evolution sequence in our test dataset into its k individual subsequences from length 1 to k, k ranging from 1 to the index of the last state (excluding the stop token). For each of these k evolution (sub)sequences $\langle s_1,\ldots,s_l\rangle, l \in \{1,\ldots,k\}$, we let the model learned from the training dataset predict its successor event distributions for s_{l+1}, i.e., $p(e_{l+1,1}),\ldots,p(e_{l+1,n})$. We compare these quantities to the empirical ground truth distribution approximated from our test dataset[8]: For all g situation evolution sequences in our test dataset starting with the current prefix $\langle s_1,\ldots,s_l\rangle$, i.e., $g = |\{\langle s_1,\ldots,s_l\rangle\}|$, we determine their g observed successor states s_{l+1}. For each event type $e_{l+1,i}$ occurring in at least one of these successor states, we compute the fraction of states it occurs, giving our "ground truth" estimate of the underlying probability distribution:

$$\hat{p}(e_{l+1,i}) = \frac{|\{e_{l+1,i} \in s_{l+1} \mid \langle s_1,\ldots,s_l,s_{l+1}\rangle\}|}{|\{\langle s_1,\ldots,s_l\rangle\}|} \quad (10)$$

We compute these individual l-gram statistics for all k subsequences of each unique evolution sequence and analyze the difference between these "ground truth" estimates and the predicted probabilities, i.e., $\delta = p(e_{l+1,i}) - \hat{p}(e_{l+1,i})$, which, across all sequence prefixes and successor event types, results in a mean $\bar{\delta} < 0.0002$, and variance $var(\delta) < 0.0023$. To account for the fact that the vast majority of event types will have a predicted probability around zero (hence, $\bar{\delta}$ will be driven by this), we also report the error statistics only for those predictions > 0.1, for which we obtain $\bar{\delta} < 0.03$ and $var(\delta) < 0.12$. Thus, our learned probabilistic evolution patterns roughly match the empirical distributions observed in the data, i.e., we conclude our resulting model approximates the situation dataset's characteristics.

B. Controlled Experiments on Synthetic Data

We next discuss experiments on generated data, to compare the model's learned distributions to known ground truth.

Recovering the Markov Model's transitions. We start with a simplistic scenario, which we subsequently extend to more complex questions. We first focus on sequences consisting of sets of a single event type only, to analyze the network's capability of recovering the underlying MC's transition probabilities. Inspired by our real-world data, we consider the following scenario: With probability p, we get a clearance message (ACX) for an observed accident (ACI). Otherwise, the accident situation ends without observing the clearance notification (i.e., p can be interpreted as our fraction of evolving situations). We generate a dataset with a proportion of p% the sequence $\langle\{\texttt{ACI}\}, \{\texttt{ACX}\}, .\rangle$, and $1-p$% the sequence $\langle\{\texttt{ACI}\}, .\rangle$. As the underlying MC in Fig. 5 shows, we only have one probabilistic transition, whereas the others are deterministic. However, the self-transition in the end-state (.) has never been observed in

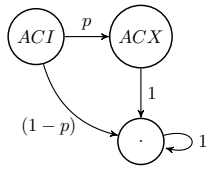

Figure 5: MC 1.

[8]Notation: $|\{.\}|$ denotes the *cardinality* or size of the set $\{.\}$.

the training data. Fig. 6 shows five different neural model's predictions[9] for the sequence $\langle\{\texttt{ACI}\}\rangle$, trained on datasets with increasing p.

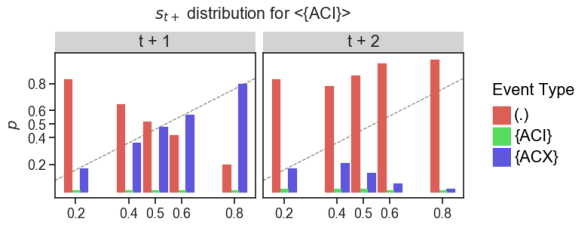

Figure 6: Predictions for the successor events for the next evolution steps for situation $\langle\{\texttt{ACI}\}\rangle$, on 5 different datasets with varying fractions of evolving situations, by varying p in $[0.2, 0.8]$ (x-axis). For s_{t+1} (left sub-figure), the probability of observing update $\{\texttt{ACX}\}$ (*blue*) exactly conforms to p, i.e., the fraction of evolving situations in the underlying data, as emphasized by the identity line.

As to be expected, the predicted probabilities for an update (i.e., transition to $\{\texttt{ACX}\}$), given situation $\langle\{\texttt{ACI}\}\rangle$, converge towards the observed distributions in the training dataset, i.e., recover the underlying MC. SGD drives our model's output to match the proportions of positive samples (per class) of the training set, which yields the minimum cross-entropy loss we have been optimizing for.

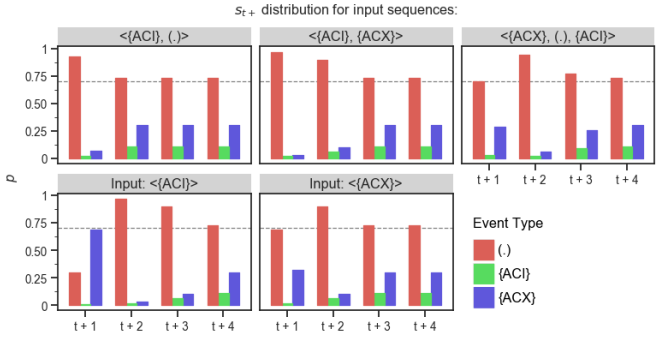

Figure 7: Predictions for different input sequences (panel heading in light-grey) across the next 4 evolution steps (x-axis).

Fig. 7 shows the predictions of our model trained on a dataset with $p = 0.7$ for the next four states given different input sequences. It also examines the "generalization" capabilities of the network, i.e., its output to sequences not observed during training, such as the (semantically) "nonsense" input $\langle\{\texttt{ACX}\}, .., \{\texttt{ACI}\}\rangle$, or starting in state $\langle\{\texttt{ACX}\}\rangle$. For the latter, predicting the next state would be an easy inference task for humans, since we only have observed transitions from ACX to the end state in the training data. Indeed, the network overall gets the prediction right (in terms of the largest probability), but seems to be sensitive to the sequential position it expects the event types – even if the presented sequence represents a sub-sequence of the observed training data, the confidence of this - actually deterministic transition - is comparably lower than for the actually observed sequence $\langle\{\texttt{ACI}\}, \{\texttt{ACX}\}\rangle$.

[9]Note that the results are averaged across 5 different experiment runs, as for all following experiments, unless explicitly mentioned. Since standard deviations are negligible (< 0.05), we do not show error bars to avoid clutter.

▷ **Problem 4** (*additional features*). Next, we simulate the effect of incorporating additional features, which we claimed to be an advantage of our neural over the MC-based model. We assume that evolution characteristics are influenced by the situation's location (e.g., accident situations at some urban intersection may produce severe traffic jams, whereas accident situations on rural roads can be typically cleared without severe follow-up effects). Conceptually, this would mean we would not have observed data produced from a single MC anymore, but actually different MCs based on the location characteristics. Whereas this problem cannot be easily modeled with MCs anymore, which would require determining the different location characteristics before constructing the MCs, we examine whether the neural network is capable of autonomously learning the locations' effects. Technically, this requires our simple neural architecture to discriminate a mixture of distributions, conditioned on individual input features. In our example, we consider three different distributions generated by the MC shown in Fig. 5, assuming that situations at different locations exhibit different evolution tendencies: We generate three different distributions for accidents at three different locations (i.e., feature $loc \in \{1, 2, 3\}$, locations binary encoded), whereby probability p depends on the feature loc of the preceding state ACI, i.e., we determine the conditional probability $p(.\,|\langle(\texttt{ACI}, loc)\rangle)$. Fig. 8 shows that the neural model has correspondingly learned this feature relation.

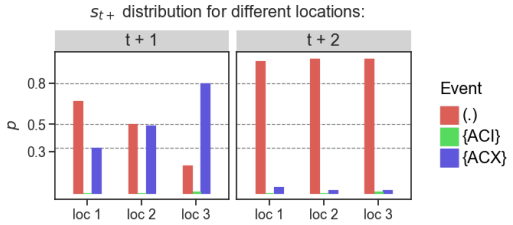

Figure 8: Predictions for different locations across 2 time steps, with $p(ACX|(ACI, loc = l)) = p_l$ for $l \in \{1, 2, 3\}$ and $p_l \in \{0.33, 0.5, 0.8\}$ (dashed lines), and prior probabilities for the locations 1, 2, 3 of 0.7, 0.1 and 0.2, resp. Despite different priors, the network perfectly disambiguates the 3 distributions based on the location feature (since our scenario is noise-free), whereas the predictions for the next evolution step do not differ.

▷ **Problem 3** (*"shared" event type associations*). We finally examine whether the model conforms according to intuitive expectations with respect to *unseen event constellations*. We construct the test scenario generated by the MC shown in Fig. 9, using five different tokens and arbitrarily fixing $p_1 = 0.2, p_2 = 0.3$ and $p_3 = 0.55$ to generate a dataset according to these proportions. In this dataset, ACI, LS1 and LCR (right-lane closure) may be starting states,

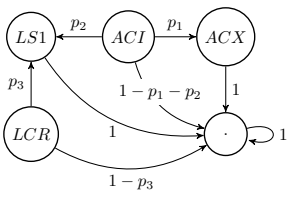

Figure 9: MC 3.

whereas ACX can only be a consequence state of ACI. After training our model on the generated data, we feed in the previously unseen event combination {ACI,LCR} (i.e., co-occurrence of an accident and a right-lane closure – note that this state is not modeled in the MC). The model is capable of intrapolating from its previous observations, by increasing the expectation of observing a traffic jam in s_{t+1} to 0.56 (*mean* across 100 training runs, $\sigma = 0.1$), thus approximating intuitive expectations (as opposed to 0.55 for only observing the right-lane closure and 0.3 for only observing the accident).

Concluding, our experiments have empirically validated that the learned models encode the empirical distributions of the underlying situation dataset, conditioned on feature values, and generalize event combinations not seen during training, thus are capable of overcoming ▷ **Problems 1 – 4**.

VI. Conclusions and Future Work

Inspired by *neural language modeling*, we have developed an approach towards *neural situation evolution modeling* to learn an *implicit representation* of the evolution patterns in a given situation dataset. Our approach differs from existing work by employing a *distributed representation* defined upon the set of event types that might compose a situation. This allows to capture the individual *sequential* as well as *co-occurrence* correlations between a situation's comprised event types. The datasets and problem studied in this paper have been chosen to contribute a first proof-of-concept and feasibility study demonstrating the utility of a distributed representation for situation evolution modeling. The presented model can be adapted to more complex application domains, by plugging it into a larger, stacked architecture (e.g., for processing video, these may for example involve preceding layers of CNNs for upstream object detection tasks, which may "instantiate" the downstream "symbolic layers", as proposed in [24], [25]). Our approach focuses on *temporal* relations, assuming that all events are characterized by *spatial co-occurrence*, as confirmed in the datasets we analysed. Whereas our present approach has focused on studying the composition of the types of situational elements and modeling their temporal relations, for future work we plan to develop more general relation representations, either by injecting them in terms of symbolic input (e.g., inputting relation tuples), or devising an architecture that autonomously learns these relations from the input features by adopting specialized architectures, such as interaction networks [26] or relation networks [27]. Furthermore, it might be interesting to incorporate duration prediction (i.e., also predicting when the next evolution step is expected to occur), investigate the use of embedding layers, and refine the neural model (e.g., by designing a loss function accounting for mutual exclusiveness between different sets of successor events).

Acknowledgments

This work has been funded by the Austrian Science Fund (FWF) under grant FWF T961-N31. The authors thank ASFINAG (www.asfinag.at) for providing the analyzed data.

References

[1] G. Jakobson *et al.*, "A Framework of Cognitive Situation Modeling and Recognition," in *Military Communications Conference (MILCOM)*. IEEE, 2006.

[2] A. Milan *et al.*, "Online Multi-target Tracking Using Recurrent Neural Networks," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, ser. AAAI'17, 2017.

[3] S. Jung *et al.*, "Sequential Monte Carlo Filtering with Long Short-Term Memory Prediction," in *22nd International Conference on Information Fusion (FUSION)*, 2019.

[4] J. Llinas *et al.*, "Revisiting the JDL Data Fusion Model II," in *Proceedings of the Seventh International Conference on Information Fusion (FUSION)*, 2004.

[5] A. Salfinger, "Framing Situation Prediction as a Sequence Prediction Problem: A Situation Evolution Model Based on Continuous-Time Markov Chains," in *22nd International Conference on Information Fusion (FUSION)*, 2019.

[6] G. E. Hinton *et al.*, "Learning distributed representations of concepts," in *Proceedings of the eighth annual conference of the cognitive science society*, vol. 1, 1986.

[7] M. Kokar *et al.*, "Situation tracking: The Concept and a Scenario," in *IEEE Military Communications Conference (MILCOM)*, 2008.

[8] N. Baumgartner *et al.*, "Situation Prediction Nets," in *Conceptual Modeling – ER 2010*, ser. LNCS. Springer, 2010, vol. 6412.

[9] L. Snidaro *et al.*, "Fusing uncertain knowledge and evidence for maritime situational awareness via markov logic networks," *Information Fusion*, vol. 21, pp. 159–172, January 2015.

[10] ——, "Recent trends in context exploitation for information fusion and AI," *AI Magazine*, vol. 40, pp. 14–27, Fall 2019.

[11] D. Meyer-Delius *et al.*, "Probabilistic situation recognition for vehicular traffic scenarios," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[12] A. Salfinger, "Situation Mining: Event Pattern Mining for Situation Model Induction," in *IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*. IEEE, 2019.

[13] European Committee for Standardization (CEN), CEN Technical Committee 278, Road Transport and Traffic Telematics, "Intelligent transport systems - DATEX II data exchange specifications for traffic management and information," 2011. [Online]. Available: https://datex2.eu

[14] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc, 2001.

[15] Y. Bengio *et al.*, "A Neural Probabilistic Language Model," in *Advances in Neural Information Processing Systems 13*. MIT Press, 2001.

[16] C. Chelba *et al.*, "N-gram Language Modeling using Recurrent Neural Network Estimation," *CoRR*, vol. abs/1703.10724, 2017.

[17] Y. Li *et al.*, "Time-Dependent Representation for Neural Event Sequence Prediction," in *6th International Conference on Learning Representations*, 2018.

[18] A. Salfinger *et al.*, "Staying aware in an evolving world – specifying and tracking evolving situations," in *2014 IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. IEEE, 2014.

[19] K. Dembczyński *et al.*, "On label dependence and loss minimization in multi-label classification," *Machine Learning*, vol. 88, no. 1, 2012.

[20] I. Goodfellow *et al.*, *Deep Learning*. MIT Press, 2016.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, 1997.

[22] T. Mikolov *et al.*, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems 26*, 2013.

[23] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[24] M. Garnelo *et al.*, "Towards Deep Symbolic Reinforcement Learning," 2016. [Online]. Available: http://arxiv.org/pdf/1609.05518v2

[25] M. Shanahan *et al.*, "An Explicitly Relational Neural Network Architecture," 2019. [Online]. Available: http://arxiv.org/pdf/1905.10307v2

[26] P. Battaglia *et al.*, "Interaction Networks for Learning About Objects, Relations and Physics," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16, 2016.

[27] A. Santoro *et al.*, "A simple neural network module for relational reasoning," in *Advances in Neural Information Processing Systems 30*, 2017.