Framing Situation Prediction as a Sequence Prediction Problem: A Situation Evolution Model Based on Continuous-Time Markov Chains

Andrea Salfinger
Department of Cooperative Information Systems
Johannes Kepler University Linz
Altenberger Strasse 69
4040 Linz, Austria
andrea.salfinger@cis.jku.at

***Abstract*—Following the acknowledged JDL data fusion model, a situation can be characterized as a set of objects in relations. Considering that this object-relational composition may change over time, as the monitored objects may alter their states (such as changing their event type and position), we can summarize a situation's evolution in a high-level fashion by the *sequence of object-relational states* it has evolved through, i.e., the sequence of its *situation states*. Thus, we propose to discretize the continuous evolution of the monitored real-world objects into a sequence of their different joint relational states (e.g., defined by the topological or qualitative distance relations holding between those objects), representing our *alphabet*, and consequently treat the problem of predicting a monitored situation's further qualitative evolution as a *sequence prediction problem*. We examine this approach on real-world data from the domain of road traffic incident management, in which situations are characterized by a changing aggregate of different event types, denoting the distinct phases of the monitored road traffic incidents. For the problem of predicting the monitored situation's next discrete state, we propose a predictive situation evolution model based on a first-order Continuous-Time Markov Chain. Our proposed structured situation prediction approach is applicable to all problem domains in which situations can be formulated as sequences of discretized states.**

I. Introduction

Motivation. According to the widely adopted JDL data fusion model [1], a situation is defined as a set of objects in relations, representing *high-level* information summarizing the relational pattern of the underlying *low-level* object observations. Various mature situation assessment techniques have been developed, which detect the currently on-going situation(s) by fusing the object observations sensed from the monitored environment, for instance [2]–[4]. *Situation (evolution) prediction*, however, i.e., predicting a monitored situation's expected future evolution, represents a less-studied issue up to now. As opposed to low-level fusion (JDL level 1), for which tracking and predicting an object's evolution represents a well-defined objective, typically focusing on predicting spatial evolution (i.e., movement) researched in the field of target/object tracking (e.g., using Bayesian tracking approaches, such as Kalman or Particle Filtering), the problem of situation prediction is hampered by the fact that the prediction goals for this high-level, aggregate information are not as inherently clear.

Reformulating the Problem. Therefore, we aim to approach this problem from a novel perspective, by formulating situation prediction as a *sequence prediction problem*, drawing inspirations from Natural Language Processing (NLP) and event sequence prediction problems: Considering that a situation's object-relational composition may change over time, as the monitored objects may alter their states (such as changing their event type and position), we can summarize a situation's evolution in a high-level fashion by the *sequence of object-relational states* it has evolved through, i.e., the sequence of its *situation states*, as proposed in our previous work on situation evolution modeling and tracking [5], [6]. Thus, we propose to discretize the continuous evolution of the monitored real-world objects into the sequence of their different joint relational states (e.g., defined by the topological or qualitative distance relations holding between those objects). In the present work, we aim to study how this sequence-based formulation supports situation prediction: We propose to regard the set of distinct object-relational states identified in our situation database as our *alphabet*, and consequently treat the problem of predicting a monitored situation's further qualitative evolution as a *sequence prediction problem*.

Contributions. For tackling this sequence prediction problem, we propose an explicit (i.e., white-box) model based on representing evolving situations as time-homogeneous first-order Continuous-Time Markov Chains (CTMCs). We contribute an approach for learning such situation evolution prediction models from recorded situation datasets, thereby extending our previous work on inductive situation evolution modeling [6], and demonstrate its feasibility based on a case study from the domain of road traffic incident management. Our proposed approach is applicable to all problem domains in which situations can be formulated as sequences of discretized states.

Structure of the Paper. In the next section, we motivate the problem by providing the relevant background on the conducted case study. Section III introduces the formal notation required for the latter sections, recapping our previous work on situation evolution modeling and tracking. Section IV presents the developed predictive model, which is practically examined in a real-world case study in Section V. Section VI compares related work on situation prediction and road traffic incident prediction. Finally, Section VII draws conclusions and presents ideas for future work.

II. Motivation and Background

In the following, we will introduce the motivational background of our case study in the domain of road traffic incident management. The analyzed road traffic incident records have been obtained from ASFINAG[1], the company that maintains and operates Austria's highway network, and are conceptually based upon the "Data Exchange for Traffic Management and Travel Information" standard, DATEX II[2], developed by the European Committee for Standardization. DATEX II describes road traffic incident data in form of *situations*: The complex type `Situation` groups causally related road traffic objects and events, denoted as `SituationRecords`[3]. Since the described situation may evolve over time, this element is versioned and has unique identifier. Naturally, also the `SituationRecord` element may evolve over time, which is thus also versioned and has unique identifier. In the following, we will use the term `Object` instead of `SituationRecord`, in order to comply with the domain agnostic and thus more widely adopted JDL data fusion terminology, which defines a situation as *a set of objects in relations* [1]. The comprised `Objects` can be of different categories, notably *traffic elements* (events not planned by the traffic operator, which are affecting, or have the potential to affect traffic flow, including events planned by external organizations such as exhibitions, sports events etc.)[4], *operator actions* (actions that a traffic operator can decide to implement to prevent or help correct dangerous or poor driving conditions, including maintenance of the road infrastructure)[5], as well as *non-road event information* (information about an event which is not on the road, but which may influence the behavior of drivers and hence the characteristics of the traffic flow)[6]. Our data set has been compiled from real-time recording of such real-time road traffic `Situation` publications, as ASFINAG provided us with a list of currently observed road traffic `Objects` in roughly three-minute update intervals, which comprised information on the *current states* of observed traffic `Objects`, including their event types, as characterized by DATEX DOB (object code defining the type of event) and PHR codes (phrase code defining the type of event) [7], their begin times, locations on the road network, and the messages shown to approaching motorists on Variable (Traffic) Message Signs (VMS).

But how can we summarize the recorded DATEX II situations' evolutions in a high-level fashion, in order to compare similar situations? First of all, a situation's most defining information are the different types of objects is it composed of. Hence, we will use the DATEX PHR codes as symbols for representing the different object types occurring in a specific situation. Next, we order a situation's comprised object updates by their begin time, and group object states active at the same time interval into sets, representing a particular *situation state*, for instance shown for one of our recorded real-world situations in Fig. 1, which comprises three distinct situation states.

Based on this temporal grouping, we can represent these DATEX II situations as sequences of the *object state sets* (represented by their DATEX PHR codes) we encountered in each of these situation states, as in the following examples:

- `{ACI,LS1}` → `{ACX,LS1}` → `{LS2}`

This sequence of object state sets would summarize the situation evolution outlined in Fig. 1. The situation starts in state `{ACI,LS1}`, i.e., we are observing a situation comprising an object state of type `ACI` ("accident") and an object state of type `LS1` ("stationary traffic") co-occurring on the specified road segment. It then evolves to state

[1] `www.asfinag.at`

[2] `www.datex2.eu`

[3] `http://d2docs.ndwcloud.nu/level2user/level2PayloadPublication.html`

[4] different types of abnormal traffic, activity, accident, conditions, equipment or system fault, obstruction

[5] different types of roadworks, sign setting, network management, roadside assistance

[6] different types of transit information, information about other transport means, e.g., cancellation, or delay, on a tram, train, plane, etc. journey; service disruption (rest area closed, no diesel etc.), road service disruption (no patrol, emergency call out of order, etc.)

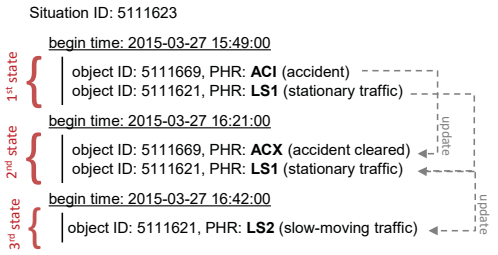

Figure 1: Situation evolution outlined.

`{ACX,LS1}`: Upon checking the object state's `Object` IDs, we see that `ACX` ("accident site cleared") is an update to our object previously in state `ACI`, i.e., the accident site has meanwhile been cleared. After the next update, leading to state `{LS2}` ("queuing traffic"), we see that the traffic jam is slowly resolving now that the road has been cleared. Thus, we note that a situation is defined upon the joint evolution of its comprised objects – whereas its objects evolve individually (i.e., `LS1` → `LS1` → `LS2` and `ACI` → `ACX`), the situation summarizes the joint evolution of the sets of related objects, i.e., is a sequence of sets of object states.

- `{LS2}` → `{ACI,LS1}` → `{ACX,LS1}`

This situation, also taken from our real-world dataset, starts in state `{LS2}`, i.e., only comprising an object state of type `LS2` ("queuing traffic"). It then evolves to state `{ACI,LS1}`, i.e., another object state has appeared - `ACI` ("accident"), indicating that an accident has happened in the jam area, and the traffic jam has worsened. After the next update, the accident has been cleared, which is now in state `ACX` ("accident cleared"), whereas the traffic jam still persists. This situation shares events and states with the previous one, but defines a different course of events, i.e., situation evolution. Thus, we see that this representation allows us to capture commonalities in the evolutions of different situations.

As we observe, this *sequence-based*, *symbolic* situation evolution representation allows us to focus the high-level event evolution patterns in terms of the associated event sequence patterns. For real-time situation monitoring, given a situation which is currently in a particular state (e.g., `{ACI,LS1}`), to allow for predictive Situation Management (SM) [8], we would require a realistic prediction on the *next state* the situation might evolve to, and *when* this transition will happen, thus turning situation prediction into a sequence prediction problem, for instance:

`{LS2}` → `{ACI,LS1}` → `?`

Consequently, the question arises whether we can learn typical evolution patterns from recorded situations represented in this sequence-based fashion. Thus, in the following sections, we will try to answer the following question: Can we learn predictive situation evolution models from recorded situations, which allow us to project how a currently monitored situation will evolve?

III. Problem Formulation

After having illustrated the problem on our motivational examples, we will abstract from the peculiarities of our case study to provide a *domain-agnostic problem formulation*. As a prerequisite for the subsequent sections, we will recap and extend the formal notation for representing evolving situations in a sequence-based fashion, as developed in our previous work [5], [6].

A. Prerequisites

We assume we are given a dataset of situation instances $\mathcal{S}^{\mathcal{I}}$, i.e., each recorded situation has been labeled with a unique ID ($\mathcal{I}$). Each situation comprises one or multiple objects. Situation instances might have been recorded by human operators

responsible for their situation life-cycle management, who handled the corresponding case. Note that in the following, we will use super-scripts to notationally distinguish instance-level ($^{\mathcal{I}}$) from type-level ($^{\mathcal{T}}$) information.

We further require that also each encompassed object record is identified by a unique ID, and of a particular object type ($\mathcal{O}^{\mathcal{T}}$). Objects might evolve, i.e., we receive updates on this object - in this case, we observe a new object state record with the same ID but different begin time, i.e., we can reconstruct this object's evolution as a sequence of its object states, which are temporally ordered by their begin times. An object's type $\mathcal{O}^{\mathcal{T}}$ might change throughout its evolution, for instance, we might observe the following change of $\mathcal{O}^{\mathcal{T}}s$ between two object states: `LS1` $\rightarrow$ `LS2`, meaning that an object starting in an object state of type LS1 ("stationary traffic") evolved to LS2 ("queuing traffic"), or `ACI` $\rightarrow$ `ACX`, meaning that an object starting as type "accident" evolved to a state of type "accident cleared".

B. Representing Evolving Situations by Sequences

Formalizing these requirements, we denote the evolution of an *object* $\mathcal{O}^{\mathcal{I}}$ by a sequence of observations[7] of its different object states $o^{\mathcal{I}}_t$:

$$\mathcal{O}^{\mathcal{I}} := < o^{\mathcal{I}}_{[t_1,t_2[}, o^{\mathcal{I}}_{[t_2,t_3[} \ \cdots, \ o^{\mathcal{I}}_{[t_n,t_{n+1}]} >, \tag{1}$$

where n denotes the number of observed object states (starting at time instants $t = t_1$ to t_n), i.e., updates. The total lifetime of the monitored object can be computed from the interval $[t_1, t_{n+1}]$. An object may spend variable amounts of time in its states $o^{\mathcal{I}}_{[t_j,t_{j+1}[}$, whereby the duration d_j spent in a state can be computed from the interval $[t_j, t_{j+1}[$. We note index variable j on the update time stamps t_j only specifies their sequential position in the sequence of update time stamps $< t_1, \ldots, t_n, t_{n+1} >$, hence should not be understood as a fixed time step-size.

We simplify notation by only referring to the states' begin times:

$$\mathcal{O}^{\mathcal{I}} := < o^{\mathcal{I}}_{t_1}, \ \ldots, \ o^{\mathcal{I}}_{t_n} > \tag{2}$$

Analogously to formulating object evolution, a situation's evolution over its entire situation life-cycle would be formalized as

$$\mathcal{S}^{\mathcal{I}} := < s^{\mathcal{I}}_{[t_1,t_2[}, \ \ldots, \ s^{\mathcal{I}}_{[t_n,t_{n+1}]} >, \tag{3}$$

whereby $\mathcal{S}^{\mathcal{I}}$ denotes a particular *situation instance*, which can be expressed as a sequence of n situation state updates, in simplified notation written as

$$\mathcal{S}^{\mathcal{I}} := < s^{\mathcal{I}}_{t_1}, \ \ldots, \ s^{\mathcal{I}}_{t_n} > \tag{4}$$

Each *situation state instance* $s^{\mathcal{I}}_t$ simply represents a container for the set of k object states composing the situation state:

$$s^{\mathcal{I}}_{[t_j,t_{j+1}[} := \{o^i_{[t_j,t_{j+1}[}\}, \ i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\} \tag{5}$$

The *cardinality* of a situation state instance $s^{\mathcal{I}}_t$ thus is the size of its object state set, i.e., $\|s^{\mathcal{I}}_t\| := \|\{o^i_t\}\| = k$, which may vary across different situation state instances $s^{\mathcal{I}}_t$, if objects join or leave the situation. Concluding, a situation instance simply represents a container for a changing set of related object states. Updates of a situation (i.e., a change of its situation state) correspond to updates of its contained objects. Note that an object update does not necessarily lead to a new situation state, if this update do not change the object-relational configuration of its situation state. For instance, assume we are observing a situation state instance $s^1_{10:05}$ comprising two objects O^5 and O^9 with histories $O^5 =< o^5_{09:25}, o^5_{09:56} >$ and $O^9 =< o^9_{10:05} >$, being in the relation "spatially close" under a given definition for spatial closeness stating a corresponding lower and upper bound for this relation to hold. Since o^9_1 appears at time point 10:05, the situation state instance s^1 starts at time point 10:05, thus comprising the states of O^5 and O^9 active in the interval [10:05, [, i.e., their time slices $O^5_{[10:05,[}$ and $O^9_{[10:05,[}$. Their states active at time point 10:05 are o^5_2 (which has not changed since 09:56) and o^9_1. Assume we are now receiving another position update on O^5 at 10:10, leading to the object evolution sequence $< o^5_{09:25}, o^5_{09:56}, o^5_{10:10} >$. This update, however, does not change the spatial closeness relation, as both objects still remain close to each other. Thus, $s^1_{10:05}$ would now stretch across $< o^5_2, o^5_3 >$ and o^9_1. The situation changes with the next position update on O^5 at time point 10:17, when the objects now become very close: Thus situation state instance $s^1_{[10:05,10:17[}$ can now be closed, comprising $< o^5_2, o^5_3 >$ and o^9_1, and a new situation state instance $s^1_{[10:17,[}$ is instantiated, comprising o^5_4 and o^9_1. Thus, we see that situation states span their comprised objects' states according to the holding object-relational configurations, i.e., comprise time slices of their objects' evolution histories. Hence, these may fuse multiple individual object state updates as long as the overall object-relational configuration remains unchanged, thereby, summarizing the joint behavior of the monitored object set of interest in a concise manner.

C. Prior Work – Constructing the State Space from Observations

In Def. (3), we have represented a situation as a sequence of the different "object-relational" configurations or "states" it has evolved through. However, these individual *situation states*, as for example shown in Fig. 1, are specific for each observed situation. Analogously to the abstract evolution sequences developed for our motivational examples in Section II, we would thus require an "abstract description" of the observed situation states, representing the *alphabet* for our evolution sequences, so that we are able to compare different situation evolutions on a common basis. Therefore, in our previous work, we have developed a dedicated *situation mining algorithm* [6], which takes a set of human-labeled situation instances as input, and derives *abstract, type-level* representations of their situation states, which we term *Situation State Types*, short $\mathcal{SS}^{\mathcal{T}}s$: Thus, *situation mining* essentially constructs an *alphabet* or *state space* for comparatively describing our observed situation instances. $\mathcal{SS}^{\mathcal{T}}s$ are formed by analyzing which *object types* ($\mathcal{O}^{\mathcal{T}}s$) have been observed in which relation types ($\mathcal{R}^{\mathcal{T}}s$) in the analyzed situation states (e.g., an object of type *accident* and an object of type *traffic jam* being in the relation "spatially co-located"), thus representing an abstract description of the observed object-

[7]Notation: We denote sets by $\{\ldots\}$ and sequences by $< \ldots >$.

relational configuration. Hence, $\mathcal{SS}^{\mathcal{T}}s$ can be sought of as being a descriptive rule defining the situation state, and thus can be used for automated situation detection. Formalizing this, $\mathcal{SS}^{\mathcal{T}}$s are defined as:

$$\mathcal{SS}^{\mathcal{T}} := (\Omega, \rho), \tag{6}$$

whereby Ω is a finite, non-empty set of *object references* ($\mathcal{O}^{\mathcal{R}}$). An object reference corresponds to an *object type* ($\mathcal{O}^{\mathcal{T}}$) referenced by an *alias*, such as *Wrong-way Driver* w or *Tunnel* t. The alias is required to distinguish between different objects of the same type process (e.g., different traffic jams j^1 and j^2), hence an $\mathcal{O}^{\mathcal{R}}$ can be considered as a variable (in the reasoning process for automated situation assessment, matching object states are bound to these variables). ρ is a set of n-ary relation types $\mathcal{R}^{\mathcal{T}}$s that need to hold between these $\mathcal{O}^{\mathcal{R}}$s, i.e., $\rho : \Omega^n \rightarrow \{true, false\}$ for the situation state to be true. Hence, an $\mathcal{SS}^{\mathcal{T}}$ is defined by a set of $\mathcal{O}^{\mathcal{R}}$s in specific $\mathcal{R}^{\mathcal{T}}$s, thereby characterizing a situation's different object-relational states.

Consequently, each individual *situation state instance* $s_t^{\mathcal{I}}$ can now be linked to its abstract type, notably its particular $\mathcal{SS}^{\mathcal{T}}$. Hence, we can represent an evolving situation $\mathcal{S}^{\mathcal{I}}$ by the sequence of $\mathcal{SS}^{\mathcal{T}}s$ it has evolved throughout its development.

D. Goal

Thus, upon transforming our situation dataset to the *common symbolic sequence representation* introduced in the previous section via *situation mining*, we can now formulate our prediction goal as follows: We are given a situation instance represented by the sequence of $\mathcal{SS}^{\mathcal{T}}s$ it has evolved through so far, as well as the *durations* d spent in each of those $\mathcal{SS}^{\mathcal{T}}s$. We denote a situation instance which is currently in its k^{th} state by[8]:

$$\mathcal{S}^{\mathcal{I}} := < (s_1^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}x, d_1), \quad \ldots, \quad , (s_k^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}y, d_k) > \tag{7}$$

or, in short[9]:

$$\mathcal{S}^{\mathcal{I}} := < (\mathcal{SS}^{\mathcal{T}}{}_1, d_1), \ldots, (\mathcal{SS}^{\mathcal{T}}{}_k, d_k) > \tag{8}$$

We would like to predict the next $\mathcal{SS}^{\mathcal{T}}$ it will evolve into:

$$P(s_{k+1}^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}z \mid < (s_1^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}x, d_1), \ldots, (s_k^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}y, d_k) >) \tag{9}$$

or, in short:

$$P(\mathcal{SS}^{\mathcal{T}}{}_{k+1} \mid < (\mathcal{SS}^{\mathcal{T}}{}_1, d_1), \ldots, (\mathcal{SS}^{\mathcal{T}}{}_k, d_k) >) \tag{10}$$

Hence, given a situation instance expressed by the sequence of $\mathcal{SS}^{\mathcal{T}}s$ it has evolved through, and the respective durations it has lasted in each of those states, we aim to predict the expected duration of the present state, and the next $\mathcal{SS}^{\mathcal{T}}$ of the sequence we expect to observe.

[8]read: $s_{t_1}^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}x$ as "$s_{t_1}^{\mathcal{I}}$ is of situation state type x", notation borrowed from Description logics

[9]We will omit the type name, when generically referring to any arbitrary $\mathcal{SS}^{\mathcal{T}}$, and instead index $\mathcal{SS}^{\mathcal{T}}s$ with respect to their position in the sequence.

IV. Situation Prediction Approach

For addressing such sequence prediction problems, Markov Models have found wide-spread adoption for determining the transition probabilities between different symbols or states [9]. In the following, we thus introduce a situation prediction approach based on the ideas of *Markov Chains* (MCs). In general, a k-order MC or Markov process represents a sequence of random variables for which the probability of the next state depends only on its previous k states. In terms of our situation prediction problem, a situation instance $\mathcal{S}^{\mathcal{I}}$ could be considered as a Markov process, characterized by the sequence of $\mathcal{SS}^{\mathcal{T}}s$ it has evolved through. If we assume the Markov property holds, i.e., for a *memoryless* approach, the probability of the next state reduces to the following:

$$P(\mathcal{SS}^{\mathcal{T}}{}_{k+1} \mid < \mathcal{SS}^{\mathcal{T}}{}_1, \ldots, \mathcal{SS}^{\mathcal{T}}{}_k >) = P(\mathcal{SS}^{\mathcal{T}}{}_{k+1} \mid \mathcal{SS}^{\mathcal{T}}{}_k), \tag{11}$$

i.e., only the current state of the situation is considered as relevant for making the prediction, corresponding to a first-order MC.

What distinguishes our situations $\mathcal{S}^{\mathcal{I}}$ from Markov processes in the conventional notion is that Markov processes are considered to be infinitely running processes, whereas our situations may potentially end in any $\mathcal{SS}^{\mathcal{T}}$. We can circumvent this by introducing an artificial end state, which we append to all observed sequences. In terms of the MC framework, this end state represents an *absorbing* state (i.e., it does not have outgoing transitions). Thus, whenever a situation instance "ends" in a particular state (i.e., its last state is "closed" and we do not receive any further updates), we implicitly regard this as making a transition to this absorbing state, which can be reached from any $\mathcal{SS}^{\mathcal{T}}$, meaning that this situation instance now will remain infinitely in that "ended" state.

In the following, we will start by formulating our Situation Evolution Model as a first-order *Discrete-Time Markov Chain* (DTMC). DTMCs essentially assume a fixed observation and prediction step-size, i.e., that our observations are taken at *equidistant time steps* of length Δt, and thus are applicable in domains with fixed update frequencies. Since this is generally not the case for our situation instances, which may comprise varying time lags between subsequent $\mathcal{SS}^{\mathcal{T}}s$, we will need to correspondingly preprocess our situation evolution sequences by splitting them into evenly sized time slices, in order to apply the framework of DTMCs to our situation prediction problem. In the subsequent section, we will then extend our model to *Continuous-Time Markov Chains* (CTMCs), which allow to directly model the durations spent in each state.

A. Discrete-Time Markov Chains

Since DTMCs assume a fixed observation time step-size, we need to preprocess our sequences to satisfy this requirement. Hence, we choose a fixed step-size k suitable for the domain at hand (e.g., in our road traffic incident management scenario, we may set $k = 3$ minutes, corresponding to our observation update interval). Then, we convert our situation instances $\mathcal{S}^{\mathcal{I}}$ to "unfolded" situation instances $\mathcal{S}^{\mathcal{I}'}$, by repeating each

$\mathcal{SS}^{\mathcal{T}}{}_i$ $\lfloor d_i/k \rfloor$ times, to transform our situation instances to an observation sequence at equidistant time steps:

$$expand(\mathcal{S}^{\mathcal{I}}, k) := f(\mathcal{S}^{\mathcal{I}}, k) = \left((\mathcal{SS}^{\mathcal{T}}{}_i)^{\lfloor d_i/k \rfloor}\right)_{i=1}^{|\mathcal{S}^{\mathcal{I}}|}, \quad (12)$$

whereby $|\mathcal{S}^{\mathcal{I}}|$ denotes the length of situation instance $\mathcal{S}^{\mathcal{I}}$, $(\mathcal{SS}^{\mathcal{T}}{}_i)^x$ means symbol $\mathcal{SS}^{\mathcal{T}}{}_i$ is repeated x times, and $(.)_{i=1}^{|\mathcal{S}^{\mathcal{I}}|}$ denotes iterating over all positions in sequence . with index variable i.

In order to exploit the framework of MCs for our situation prediction task, we next need to learn a DTMC characterizing our observed situation dataset $\mathcal{S}^{\mathcal{I}'}$, by determining the following quantities:

1) the *state space* $\mathbb{S}$ of the random variables our variable of interest $\mathcal{S}^{\mathcal{I}'}$ can take on, which represents the *alphabet* of our observed sequences
2) the *transition probabilities* between those states

Our countable state space $\mathbb{S}$ is given by the set of different $\mathcal{SS}^{\mathcal{T}}s$ our *situation mining algorithm* [6] has identified, in other words, the different $\mathcal{SS}^{\mathcal{T}}s$ in our situation sequences.

In order to complete our model, we still require the transition probabilities between those states:

$$p_{i,j} = P(s_{k+1}^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}{}_j \mid s_k^{\mathcal{I}} : \mathcal{SS}^{\mathcal{T}}{}_i), \quad (13)$$

whereby $p_{i,j}$ denotes the probability of evolving from $\mathcal{SS}^{\mathcal{T}}i$ to $\mathcal{SS}^{\mathcal{T}}j$ in the next step. The probabilities for all our $|\mathbb{S}|$ states are stored in the *transition probability matrix* Q of dimension $|\mathbb{S}| \times |\mathbb{S}|$. Per convention, each entry $p_{i,j}$ in this matrix specifies the transition probability from $\mathcal{SS}^{\mathcal{T}}i$ to $\mathcal{SS}^{\mathcal{T}}j$, thus, each row of this stochastic matrix contains the *probability distribution* for the successor states of a particular $\mathcal{SS}^{\mathcal{T}}i$. We need to *estimate* these conditional transition probabilities from our data set $\mathcal{S}^{\mathcal{I}'}$ to obtain our MC-based *Situation Evolution Prediction* Model $\mathcal{SEP}$:

$$\mathcal{SEP} := (\mathbb{S}, Q), \quad (14)$$

In order to estimate these transition probabilities, we need to make the following assumptions: We assume that the next state conditionally only depends on the current state (i.e., that the *Markov property* holds[10]). Furthermore, we assume *time homogeneity*, i.e., that the transition probabilities between two $\mathcal{SS}^{\mathcal{T}}s$ remain stationary over time, irrespective from their index position in the evolution sequence (i.e., these transition probabilities do not change as time goes on). Under these assumptions, we can approximate the entries of Q based on *Maximum Likelihood Estimation* (MLE)[11] from our observed data [12]:

$$\hat{p}_{i,j}^{MLE} = \frac{n_{ij}}{\sum_{u=1}^{|\mathbb{S}|} n_{iu}} \quad (15)$$

n_{ij} conforms to the number of observed transitions from $\mathcal{SS}^{\mathcal{T}}i$ to $\mathcal{SS}^{\mathcal{T}}j$, $\sum_{u=1}^{|\mathbb{S}|} n_{iu}$ represents the total number of 1-step transitions originating from $\mathcal{SS}^{\mathcal{T}}i$. Thus, $\hat{p}_{i,j}^{MLE}$ represents an empirically determined estimate of $p_{i,j}$, taken to be the fraction of all observed transitions from $\mathcal{SS}^{\mathcal{T}}i$ to $\mathcal{SS}^{\mathcal{T}}j$ over all observed transitions originating from $\mathcal{SS}^{\mathcal{T}}i$, the latter representing the normalization to yield a proper probability distribution. To complete our model, we need to include the fact that the situation remains forever in the end state, once it has transitioned into it: Hence, we set the last row of Q fixed with the deterministic self-transition of the "end" state, i.e., append a row vector with $p_{end,end} = 1$ and 0 elsewhere.

To recap our approach for predicting the next discrete state(s) an observed situation instance might evolve into, we have proposed fitting a DTMC to our situation dataset: First, we perform *situation mining* to determine the DTMC's state space as the set of derived $\mathcal{SS}^{\mathcal{T}}s$, and represent the observed situations by the sequences of $\mathcal{SS}^{\mathcal{T}}s$ they have evolved through, using a fixed observation step-size k. For predicting a monitored situation's next-step successor state probability distribution, the corresponding row of its probability transition matrix $Q_{i,:}$ is computed via MLE from the situation database. n-step forward predictions, i.e., prediction across multiple time steps, can be computed as Q^n.

B. Continuous-Time Markov Chains

Instead of discretizing our continuous-time observations $\mathcal{S}^{\mathcal{I}}$ by splitting them into chunks of fixed step-size, which introduces the problem of finding an adequate step size, we may also directly model the variable duration a situation spends in each of its states, before jumping to another state, thus directly allowing to input our $\mathcal{S}^{\mathcal{I}}$ as given in Def. (9). This *jump chain* can be represented by a (stationary) CTMC, which directly models the time spent in a state before a transition to a different state occurs:

$$\mathcal{S}^{\mathcal{I}} = < \mathcal{SS}^{\mathcal{T}}(t) : t \geq 0 > \quad (16)$$

The random variable $\mathcal{SS}^{\mathcal{T}}(t)$ now represents the $\mathcal{SS}^{\mathcal{T}}$ occupied by the CTMC at time t. Time t is considered to be any non-negative real number, corresponding to the time units (e.g., minutes) a situation stays in a specific $\mathcal{SS}^{\mathcal{T}}$. Thus, a situation instance might spend a variable amount of time, the so-called *holding time*, in each of its $\mathcal{SS}^{\mathcal{T}}s$.

If a transition occurs at time t, the situation $\mathcal{S}^{\mathcal{I}}$ evolves to the new state $\mathcal{SS}^{\mathcal{T}}(t)$, whereby $\mathcal{SS}^{\mathcal{T}}(t) \neq \mathcal{SS}^{\mathcal{T}}(t-)$. Thus, as an important distinction to our previously considered DTMC model, which also included self-transitions if the situation instance remained in the same $\mathcal{SS}^{\mathcal{T}}$ in the next step, a transition only occurs if a situation instance jumps to a new state (since remaining in the old state is modeled via the time spent in this state before a transition occurs).

In order to incorporate an $\mathcal{SS}^{\mathcal{T}}$'s exponentially distributed holding time into the model, we need to estimate a *transition rate matrix* or *Generator matrix* Λ providing the exponential holding parameters $\lambda(\mathcal{SS}^{\mathcal{T}})$ for each state [10]–[12]:

$$\hat{\lambda}_{i,j}^{MLE} = \frac{n_{ij}}{\sum_{s:\mathcal{SS}^{\mathcal{T}}i} d(s)} \quad (17)$$

n_{ij} is again the total number of transitions from $\mathcal{SS}^{\mathcal{T}}i$ to $\mathcal{SS}^{\mathcal{T}}j$, where $i \neq j$. The denominator computes the total time the chain spends in state $\mathcal{SS}^{\mathcal{T}}i$ (i.e., computes the total time all our observed situation instances have spent in this state),

[10]It is impossible to determine from a finite number of samples whether the underlying process is actually Markovian [10].

[11]or variations such as Laplace smoothing, which includes a stabilizing parameter, or Maximum A-Posteriori (MAP) [11].

whereby $d(s)$ denotes the duration of situation state instance s. Thus, we obtain an estimate $\hat{\Lambda}$ of the *generator* Λ from our observed situations. Its elements $\hat{\lambda}_{ij}$ are non-negative for all $i \neq j$, and describe the rate of the process transitions from $\mathcal{SS}^{\mathcal{T}}$ i to $\mathcal{SS}^{\mathcal{T}}$ j. Diagonal elements $\hat{\lambda}_{ii}$ have to be chosen such that the row-sums of the transition rate matrix are equal to zero. To model the behavior of the absorbing `end` state, we append a row vector of zeros for $\hat{\lambda}_{end,:}$ (as we will not have observed any transitions from the `end` state in our data). The generator matrix then allows to determine the corresponding transition probability matrix for a future time point of interest as follows (t is taken to be "integer time point", e.g., a prediction for the next t minutes in the future) [10]:

$$\hat{Q}(t) = e^{(\hat{\Lambda}\, t)} \tag{18}$$

Hence, we can compute the custom transition probability distribution after any projection period of interest.

Now that we have defined our model, the question arises on how well it characterizes our observed real-world situations, and whether the CTMC provides additional value over modeling the problem with a DTMC. Thus, in the next section, we will study the usefulness of our proposed situation evolution prediction models on a real-world situation dataset from the domain of road traffic incident management.

V. Case Study

In the following, we examine whether our MC-based models offer meaningful inference capabilities based on a feasibility study on the real-world road traffic incident data described in Section II. The particular challenge of our collected road traffic incident situations lies in the considerable variance in the durations of situation state instances of different $\mathcal{SS}^{\mathcal{T}}s$. Whereas situation states involving accidents, for instance, are typically on a time scale of minutes to a few hours (in case of severe accidents requiring complicated clearance works), states of other $\mathcal{SS}^{\mathcal{T}}s$ even expose a time scale of multiple days, such as closures (e.g., road closures due to dangerous snow conditions, danger of mudslides in alpine regions). Since we are more interested in predicting unforeseen situations, as opposed to planned and thus known situations such as long-term roadworks, for the present analysis we select those situation instances having a total duration of up to seven days, comprising 1,451 situation instances recorded in the time span between Dec. 27, 2014, to June 14, 2015, which expose a total duration between 1 to 9,159 min.[12].

Due to the wide range of holding times, the sequence expansion technique proposed for DTMCs becomes impractical, because no time step size can be found that suits both short- as well as long-lasting situation states. However, these specifics allow us to highlight the particular strength of the CTMC-based model, which computes the holding parameters individually per $\mathcal{SS}^{\mathcal{T}}$, and thus is capable to take into account the different time scales between different $\mathcal{SS}^{\mathcal{T}}s$. If we chose a small step-size for DTMCs, such as $k = 5$ min., the unrolling of long-lasting situation states would consequently lead to transition probability distributions containing almost all transition probability mass on the self-transition (indicating remaining in that state), while the probabilities of transitioning to other states would become very low. Hence, our model would always predict remaining in that state (unless we roll out simulations, i.e., generate and aggregate sample sequences from the chain), and due to the memoryless property, it does not keep track of how long it has already remained in the currently active state. Conversely, whereas the CTMC is memoryless as well, it actually models the transitions to other states. Hence, it eventually will predict the *jump* to a different $\mathcal{SS}^{\mathcal{T}}$, based on the $\mathcal{SS}^{\mathcal{T}}$'s given exponential holding parameter λ characterizing the MC's tendency to remain in this state. Therefore, since our dataset's characteristics are better captured by CTMCs, we will constrain our case study discussion to the CTMC results.

[12]mean total duration: 642 min., median total duration: 124 min.

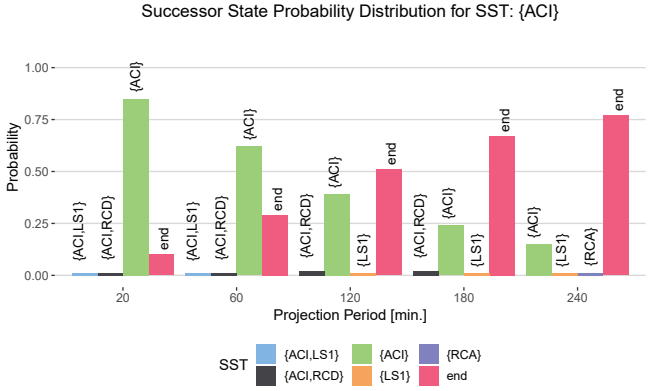

Figure 2: Projections for $\mathcal{SS}^{\mathcal{T}}$ {ACI}. Event codes: ACI = accident, LS1 = stationary traffic, RCD = road closed, RCA = closure

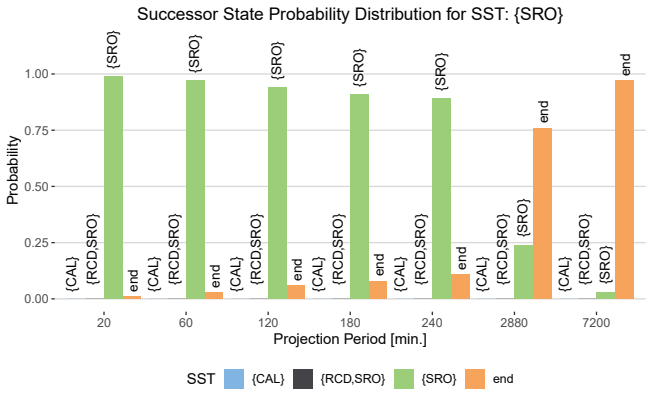

Figure 3: Projections for $\mathcal{SS}^{\mathcal{T}}$ {SRO}. Event codes: SRO = hard-packed snow, RCD = road closed, CAL = message cleared

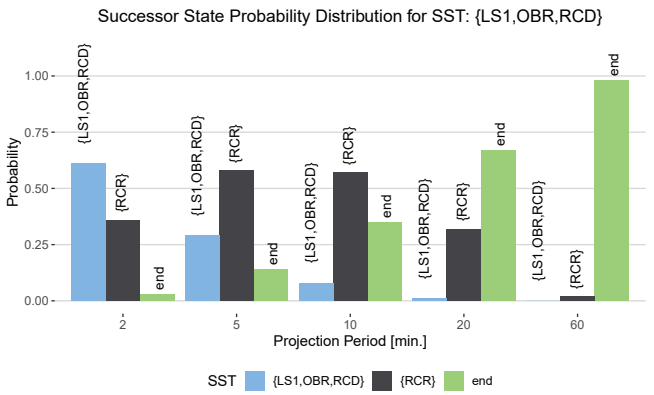

Figure 4: Projections for $\mathcal{SS}^{\mathcal{T}}$ {LS1,OBR,RCD}. Event codes: LS1 = stationary traffic, OBR = objects on the roadway, RCD = road closed, RCR = cleared

We first performed *situation mining* on this situation dataset to reconstruct our situation state space, which delivered 241 different $\mathcal{SS}^{\mathcal{T}}s$, and correspondingly assembled the observed situation evolution sequences. Next, upon performing the required sequence processing steps (e.g., extending each situation evolution sequence with a dedicated "end" state), we fit a CTMC to our processed situation evolution sequences by implementing Eq. (17).

Fig. 2 shows five different forecast horizons for an accident situation computed with the learned CTMC model based on Eq. (18), projecting the successor state probability distribution expected after 20, 60, 120, 180 and 240 minutes, respectively. We observe how the probability distributions of the top four most probable successor $\mathcal{SS}^{\mathcal{T}}s$ change with the projection period – the further we extend the forecast horizon, the less likely it becomes that this situation is still in $\mathcal{SS}^{\mathcal{T}}$ {ACI}, and the more probable it becomes that this situation has already ended. Similar analysis is provided for situations involving bad driving conditions and potential road closures due to hard-packed snow ({SRO}) in Fig. 3, and situations involving obstructions due to objects lying on the roadway ({LS1,OBR,RCD}) in Fig. 4. Whereas the latter situation can be resolved within a few minutes, since the small objects can be quickly removed, we observe that the resolution of the hard-packed snow situations happens on a different time scale, as to be expected. As can be seen, the CTMC-based model adequately takes the different evolution time scales of these different $\mathcal{SS}^{\mathcal{T}}s$ into account. We also observe that our model fully automatically associated interrelated events and operator actions, which would be cumbersome to specify manually given that we are facing 110 different object and event types in this dataset. As this analysis reveals, the benefit of the CTMC-based approach is that the learned model picks up the different time scales of our highly distinct $\mathcal{SS}^{\mathcal{T}}s$ – we could use a single model for learning all of these different situation evolution types.

Since Figs. 2-4 only have examined the underlying patterns the model has compiled from the situations it has been "trained" with, i.e., its training set, we also investigate its generalization capability, by analyzing how well its forecasts match the yet unseen test situations. Therefore, we randomly shuffle our situation dataset and perform *10-fold cross validation*, i.e., we perform 10 experiments training on 9 folds and using the left out fold as test set, respectively, to obtain an estimate of expected prediction accuracy. Starting from the test situations' initial state, we aim to predict their evolution sequence and the associated durations, and compare these with their actual sequences and durations. We predict the situation's next state by extending the projection horizon until the probability of transitioning to a different $\mathcal{SS}^{\mathcal{T}}$ becomes higher than the probability of remaining in the current state, which we predict as the next $\mathcal{SS}^{\mathcal{T}}$ (hence this projection horizon conforms to our predicted duration of remaining in the current state). For evaluating the resulting predictions, we measure the binary loss based on whether the entire predicted $\mathcal{SS}^{\mathcal{T}}$ sequence was correct, as well as the deviations of the predicted total duration from the actual total duration. In case of $\mathcal{SS}^{\mathcal{T}}s$ only occurring in our test data and not in our training set, which thus are *out of vocabulary* words, i.e., unknown states, for our model, we cannot make a prediction and thus simply count those as incorrect[13]. For predicting the entire evolution sequence correctly, we obtained a mean accuracy of 80% ($\sigma = 0.04$). Our prediction error could be attributed to two error sources, notably the test sequences starting in yet unknown states (on average 7% per test fold, $\sigma = 0.01$), and on average 14% ($\sigma = 0.03$) of true evolution sequences exposing a longer evolution sequence, whereas our model predicted a next transition into the end state. This is since our model exposes a strong bias on predicting a transition to the end state, as on the entire dataset only 15% of our situation sequences actually show evolving behavior (i.e., their sequence consists of more than one $\mathcal{SS}^{\mathcal{T}}$). This is probably owed to the noisy and incomplete nature of our situation dataset, since we might have missed situation updates lying between our sampling intervals[14], and encountered incompletely tracked situations. Hence, this problem is rooted in our specific data set compilation, and would not occur in real-world application settings in which we would have access to the complete history of the recorded situations, without any sampling biases. In general, we note that both types of errors (unknown start state as well as incorrectly predicted sequence) would decrease by training on more comprehensive situation datasets spanning complete time periods.

[13]In real-world applications, we could match those to the semantically most similar $\mathcal{SS}^{\mathcal{T}}$ based on on sequence similarity or the event taxonomy hierarchy provided by DATEX II.

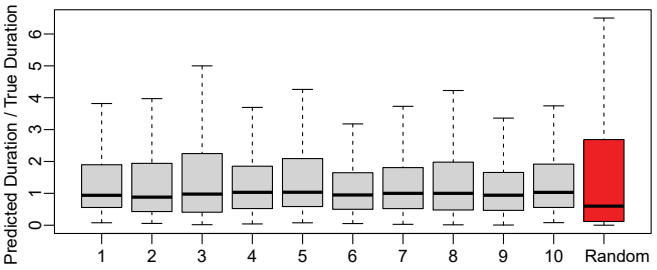

Figure 5: Duration prediction results.

We finally also compare the situations' predicted durations with their actual durations. Since the standard deviation of total durations on our entire situation dataset is as high as 1,267 min., classical regression evaluation measures such as RMSE would not be informative, since the scale of deviations will vary considerably between different $\mathcal{SS}^{\mathcal{T}}s$. To account for this $\mathcal{SS}^{\mathcal{T}}$-specific variance, we thus compute the degree to which the model under- or overestimates the actual duration, i.e., compute $predicted\,duration/true\,duration$. Hence, values close to 1 would indicate accurate predictions. For those on average 93% of test situations for which predictions could be made, we obtain the duration accuracy distributions shown in Fig. 5. Its right-most boxplot shows the accuracy distribution obtained for *random predictions* – i.e., instead of computing the estimated total duration starting from the actual $\mathcal{SS}^{\mathcal{T}}$, we randomly select any $\mathcal{SS}^{\mathcal{T}}$ from the model and continue to predict the corresponding sequence until reaching the end state. These random predictions are significantly different from our model's predictions[15], which demonstrates that our model seems to have indeed learned adequate holding parameters for the different $\mathcal{SS}^{\mathcal{T}}s$. Given that in our situation dataset, only 14% of the $\mathcal{SS}^{\mathcal{T}}s$ were covered by more than 20 corresponding situation state instances, from which the model had to estimate their holding parameters, we consider these results encouraging, which could be improved by training on more extensive datasets.

VI. Related Work

To the best of our knowledge, we are not aware of approaches that address situation (evolution) prediction as a sequence prediction problem modeled with CTMCs. Therefore, we will discuss applications of MCs for related situation monitoring tasks, and contrast the prediction problem examined in our case study with predictive models developed for this application domain. We conclude this discussion by outlining the specifics of our approach compared to conventional event sequence prediction problems.

Alevizos et al. employed Pattern MCs for predicting the estimated number of events (i.e., object states) required to complete the detection of a specific situation state [13]. Hence, in terms of our framework, they consider $\mathcal{SS}^{\mathcal{T}}s$ in isolation only, but do not consider a situation's evolution between different $\mathcal{SS}^{\mathcal{T}}s$.

MCs have previously also been used for situation monitoring in form of *Hidden Markov Models* (HMMs). For instance, Meyer-Delius et al. have used HMMs to model and track vehicular passing situations in a driver-assistance system [14]. HMMs assume that

[14]As described in Section II, we only received snapshots of currently observed states in 3-min. intervals.

[15]Two-sample Kolmogorov-Smirnov test: p-value < 2.2e-16, whereas for all combinations of actual prediction distributions, the null hypothesis could not be rejected under a p-value of 0.05.

observations are produced by unobserved, thus hidden or *latent* states, and only these latent states are modeled to form a MC [9]. Hence, an observation is only conditionally dependent on its corresponding latent state, which is conditionally dependent on its previous latent state. Thus, HMMs allow to implicitly capture a situation's history via the latent states, and thus are not limited to incorporating only the k most previous observations, as in k-order MCs. However, these latent states now also need to be estimated, which introduces additional complexity for learning these models, and thus are typically employed when the application domain allows to include some a-priori knowledge on the number and structure of those latent states. As in the above mentioned driver-assistance system, HMMs are employed in scenarios where the $\mathcal{SS}^{\mathcal{T}}s$ that actually produced the observations ($s^{\mathcal{I}}$) cannot be deterministically determined (as opposed to our descriptive, i.e., rule-based, situation tracking approach), but are taken to be latent factors that have produced the observed sequence of situation state instances. Hence, their focus is typically more on situation detection and monitoring (i.e., determining the $\mathcal{SS}^{\mathcal{T}}$ the situation is currently in, which cannot be directly observed) than situation prediction.

Whereas several predictive models for road traffic incidents have been developed, these have been focused on conventional, non-relational input data, i.e., predicting the occurrence of a specific event type, instead of considering the complex, joint evolution of multiple incident types on the situation level. As revealed by a recent review on traffic incident prediction [15], most approaches focused on the event type of congestion, i.e., predicting the occurrence and duration of congestion and accidents. Conversely, we have aimed at representing and predicting a diverse range of road traffic incidents on the *situation level*, i.e., exploiting the joint relational composition between multiple incident types, and have proposed a unified, generic model for learning arbitrary types of situations, instead of devising a specific model for a specific object or event type only.

Technically, our approach is rooted in the field of *event sequence prediction*, which bases on the idea of learning event prediction models from observed event sequences. Whereas conventional event sequence prediction only considers sequences of *individual* events, we have proposed a dedicated formulation for representing situations based on evolving complex sets of events, adhering to the situation characterization provided by the JDL data fusion model. Furthermore, we also incorporated the duration of those events in our predictive model, which has only recently been considered in general event prediction tasks [16].

VII. Conclusion and Future Work

In the present work, we have proposed to treat the problem of situation prediction as a sequence prediction problem. We have developed a framework that allows to represent evolving situations by sequences of their traversed situation states, and proposed a situation evolution model based on learning Continuous-Time Markov Chains (CTMCs) from the situation database. We examined the feasibility of our approach on a case study involving a challenging real-world data set from the domain of road traffic incident management, which demonstrated that our proposed approach provides a unified, generic framework for learning highly heterogeneous situation evolution patterns, allowing to incorporate situation evolution types lasting a couple of minutes as well as situations typically spanning multiple days. Our approach automatically learns the CTMCs, as it only requires a dataset of labeled situations.

In addition to *Predictive Situation Management*, the proposed model also supports simulations, as the fitted MCs allow to generate sample situation evolution sequences according to the distributions observed in the underlying real-world data.

For future work, we also plan to investigate on the performance of alternative sequence prediction approaches: In particular on NLP tasks, *neural sequence prediction models* such as Recurrent Neural Networks currently offer the best predictive performance on many problems. However, these represent black-box models that cannot be as easily dissected as MCs. Hence, we plan to compare the performance achievable with neural approaches with our MC baselines, to study the benefits and drawbacks of those different sequence modeling approaches for situation evolution prediction.

Acknowledgments

This work has been funded by the Austrian Science Fund (FWF) under grant FWF T961-N31. The analyzed data has been provided by ASFINAG.

References

[1] J. Llinas, C. Bowman, G. Rogova, A. Steinberg, E. Waltz, and F. White, "Revisiting the JDL Data Fusion Model II," in *Proceedings of the Seventh International Conference on Information Fusion (FUSION 2004)*, 2004, pp. 1218–1230.

[2] C. Matheus, M. Kokar, K. Baclawski, J. Letkowski, C. Call, M. Hinman, J. Salerno, and D. Boulware, "Lessons learned from developing SAWA: a situation awareness assistant," in *Proc. of the 8th International Conference on Information Fusion*, vol. 2, 2005, p. 8 pp.

[3] J. Edlund, M. Grönkvist, A. Lingvall, and E. Sviestins, "Rule-based situation assessment for sea surveillance," in *Proc. SPIE 6242, Multisensor, Multisource Information Fusion: Architectures, Algorithms, and Applications*, vol. 6242, 2006, pp. 624 203–624 203–11.

[4] F. Terroso-Saenz, M. Valdes-Vela, and A. F. Skarmeta-Gomez, "A Complex Event Processing Approach to Detect Abnormal Behaviours in the Marine Environment," *Information Systems Frontiers*, vol. 18, no. 4, pp. 765–780, 2016.

[5] A. Salfinger, W. Retschitzegger, and W. Schwinger, "Staying Aware in an Evolving World — Specifying and Tracking Evolving Situations," in *Proceedings of the 2014 IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. San Antonio, TX, USA: IEEE, 2014, pp. 195–201.

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

[7] E. Arco, A. Ajmar, F. Arneodo, and P. Boccardo, "An operational framework to integrate traffic message channel (TMC)in emergency mapping services (EMS)," *European Journal of Remote Sensing*, vol. 50, no. 1, pp. 478–495, 2017.

[8] G. Jakobson, J. Buford, and L. Lewis, "A Framework of Cognitive Situation Modeling and Recognition," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, 2006, pp. 1–7.

[9] C. M. Bishop, *Pattern recognition and machine learning*, corr. print ed., ser. Information science and statistics. New York, NY: Springer, 2007.

[10] P. Metzner, E. Dittmer, T. Jahnke, and C. Schütte, "Generator estimation of Markov jump processes," *Journal of Computational Physics*, vol. 227, no. 1, pp. 353–375, 2007.

[11] G. Spedicato and M. Signorelli, "The markovchain Package: A Package for Easily Handling Discrete Markov Chains in R," 2014. [Online]. Available: https://cran.r-project.org/web/packages/markovchain/vignettes/an_introduction_to_markovchain_package.pdf

[12] Y. Inamura, "Estimating Continuous Time Transition Matrices From Discretely Observed Data: Bank of Japan Working Paper Series." [Online]. Available: https://ideas.repec.org/p/boj/bojwps/06-e-7.html

[13] E. Alevizos, A. Artikis, and G. Paliouras, "Event Forecasting with Pattern Markov Chains," in *DEBS'17*. New York, New York: The Association for Computing Machinery, 2017, pp. 146–157.

[14] Meyer-Delius *et al.*, "Probabilistic situation recognition for vehicular traffic scenarios," in *IEEE International Conference on Robotics and Automation, 2009. ICRA '09.*, 2009, pp. 459–464.

[15] R. Li, F. C. Pereira, and M. E. Ben-Akiva, "Overview of traffic incident duration analysis and prediction," *European Transport Research Review*, vol. 10, no. 2, p. 109, 2018.

[16] Y. Li, Du Nan, and S. Bengio, "Time-Dependent Representation for Neural Event Sequence Prediction," in *6th International Conference on Learning Representations*, 2018.