MTLearn: Extracting Temporal Rules Using Datalog Rule Learners

Dingmin Wang , Przemysław A. Wałęga , Bernardo Cuenca Grau
Department of Computer Science, University of Oxford
{dingmin.wang, przemyslaw.walega, bernardo.grau}@cs.ox.ac.uk

Abstract

We propose a framework for temporal rule learning from datasets, which capitalises on the availability of increasingly mature Datalog rule learners. Our approach is based on the idea of splitting a temporal dataset into windows, extracting static rules from each window with an off-the-shelf Datalog rule learner, and then combining the obtained static rules into temporal rules corresponding to the whole dataset. Temporal rules generated by our approach are expressed in DatalogMTL and are assigned time-sensitive confidence scores. We have implemented our approach in a system MTLearn compatible with any Datalog rule learner, as well as with a range of strategies for scoring the output temporal rules. Results on the task of temporal link prediction show that our proposed approach is highly competitive, achieve performance comparable to that of state-of-the-art machine learning models for both the extrapolation and the interpolation settings, while at the same time providing interpretable results.

1 Introduction

Temporal data management is crucial in dynamic domains, such as online payments, healthcare, weather forecasting, and sensor networks (Chen et al. 2020; Zeroual et al. 2020; Thennakoon et al. 2019; Hewage et al. 2021). Yet, supervised machine learning models for temporal data (Dasgupta, Ray, and Talukdar 2018; Pareja et al. 2020; Jin et al. 2020; Messner, Abboud, and Ceylan 2022; Park et al. 2022) suffer from limited interpretability, hindering their use in some applications. In contrast, Knowledge Representation (KR) enables the formalisation of expert knowledge as logical rules and the deduction of new information via logical entailment (Van Harmelen, Lifschitz, and Porter 2008). Expert knowledge written as rules not only enhances the understanding of inferred outcomes, but also promotes trust in system predictions, ensures compliance with norms, and facilitates the verification of fairness standards. This is key in applications where accuracy and interpretability are essential. Thus, rules equipped with temporal operators are increasingly being used as the foundation for dependable and explainable AI systems (Thadeshwar et al. 2020; Vitelli et al. 2022).

Obtaining high-quality rules presents practical hurdles due to the scarcity of expert knowledge and the substantial human effort required. Research on automated rule extraction from data encompasses areas such as inductive logic programming (Muggleton and De Raedt 1994; Cropper and Dumančić 2022), neural logic programming (Yang, Yang, and Cohen 2017), and rule learning (Fürnkranz, Gamberger, and Lavrač 2012; Fürnkranz and Kliegr 2015). Additionally, a number of ready-to-use systems such as AnyBURL (Meilicke et al. 2019), AMIE+ (Galárraga et al. 2013), RLvLR (Omran, Wang, and Wang 2018), RARL (Pirrò 2020), and Popper (Cropper and Morel 2021), have been developed. These systems, however, were not designed to handle temporal data and the rules they generate do not incorporate appropriate temporal constructs.

Recently, there has been a growing interest in rule generation from temporal knowledge graphs (tKGs) including StreamLearner (Omran, Wang, and Wang 2019), TLogic (Liu et al. 2022), TILP (Xiong et al. 2024a), and TEILP (Xiong et al. 2024b). StreamLearner is particularly relevant to our research as it also relies on established Datalog rule learners (unlike TLogic, TILP, and TEILP). In particular, StreamLearner computes temporal Datalog rules from a stream of KGs. To this end, it considers a fixed-length initial fragment of the stream to generate non-temporal rules, referred to as "structure rules". These rules are then extended with a temporal component so that the resulting temporal rules hold in the stream. StreamLearner, however, comes with some limitations. First, output rules are constrained to temporal closed path rules, where all body atoms refer to the same time point; this limits the system's ability to capture dependencies across different time points. Second, the initial time points dictate the structure of all generated rules across all time points; however, as time progresses beyond the initial time points, a distinct rule structure may be necessary to effectively capture patterns emerging in the data. In contrast, TLogic, TILP, and TEILP extract temporal rules by taking into account the entire dataset. However, the form of extracted rules is limited; generated rules can propagate information only to the future (they are "forward-propagating") and restrict the allowed form of joins between body atoms (they are "temporal chain rules").

Our aim is to tackle the aforementioned limitations and offer a flexible and scalable solution for temporal rule learning. Similarly to StreamLearner, our framework capitalises on the availability of increasingly mature systems for learning Datalog rules and leverages a sliding window algorithm. Our approach, however, does not impose additional restrictions on the structure of temporal rules and applies the Datalog rule learner for each window of the dataset, rather than just the initial fragment of the timeline as in StreamLearner. The latter allows us to uncover regularities in arbitrary segments of the timeline, enabling a more comprehensive analysis of temporal patterns.

Furthermore, our approach extracts rule in a much more expressive temporal language than those considered by StreamLearner and TLogic. Indeed, rules extracted by our approach are written in a fragment of DatalogMTL (Brandt et al. 2018), a powerful extension of Datalog with operators from metric temporal logic (MTL) which can represent dependencies over intervals that span multiple adjacent time points. For example, it allows us to extract rules such as

$$\begin{aligned} &\mathit{ProvideHumanitarianAid}(x,z) \\ &\quad\leftarrow \boxminus_{[1,3]}\mathit{DemobilizeArmedForces}(x,y) \land \\ &\qquad \boxminus_{[1,2]}\ \mathit{UseUnconventionalViolence}(y,z), \quad (1) \end{aligned}$$

which predicts that a country x will provide humanitarian aid to country z if x was demobilising armed forces of y in the last three days (MTL operator $\boxminus_{[1,3]}$ expresses "continuously between 1 and 3 days in the past") and y was using unconventional violence against country z last two day ($\boxminus_{[1,2]}$ expresses "continuously between 1 and 2 days in the past").[1] Importantly, our approach is compatible with different strategies for computing a confidence score for the generated temporal rules, thus providing users with adequate means for assessing their reliability.

We have implemented this approach in a new system MTLearn[2] and evaluated its performance on link prediction tasks over tKGs (Trivedi et al. 2017; Leblay and Chekol 2018; Dasgupta, Ray, and Talukdar 2018; Xu et al. 2021; Lacroix, Obozinski, and Usunier 2020; Park et al. 2022). Our experiments show that our approach achieves comparable performance to that of state-of-the-art approaches, while offering the advantage of interpretable predictions justified by the application of the extracted rules to the given data.

2 Background

In this section, we recapitulate the basics of DatalogMTL, rule learning, and temporal link prediction.

Rule Learners Datalog is a standard rule-based language used in KR (Van Harmelen, Lifschitz, and Porter 2008), Data Management (Garcia-Molina, Ullman, and Widom 2009; Abiteboul, Hull, and Vianu 1995), and Logic Programming (Dantsin et al. 2001). Datalog *rules* are of the form $A' \leftarrow A_1 \land \cdots \land A_n$, where $n \geq 1$; each expression A is a relational atom $P(\mathbf{s})$ consisting of a predicate P and a tuple $\mathbf{s}$ of terms (i.e., constants and variables) matching the arity of P. The conjunction $A_1 \land \cdots \land A_n$ is the rule *body*, whereas A' is the rule *head*. A Datalog *program* is a finite set of rules and a *dataset* is a finite set of *facts* (i.e., variable-free atoms). Datalog rules are interpreted as universally quantified function-free Horn formulas in First Order Logic (Abiteboul, Hull, and Vianu 1995; Dantsin et al. 2001).

Datalog rule learners are systems designed to extract Datalog rules from a dataset, together with a confidence value for each rule. Existing systems exhibit significant heterogeneity due to variations in the algorithms they are based on, their underlying assumptions, and the syntactic form of the rules they generate. AnyBURL (Meilicke et al. 2019) and AMIE+ (Galárraga et al. 2015) were developed to efficiently generate rules from large-scale KGs. Both systems accept a set of facts involving binary predicates as input and compute Datalog rules satisfying certain syntactic restrictions. AnyBURL uses a bottom-up approach to generate rules. Initially, the system identifies a collection of paths from the input KG; subsequently, these identified paths are generalised into rules by substituting constants with variables. Rules obtained by AnyBURL conform to the following forms:

$$\begin{aligned} A'(c',x) &\leftarrow A_1(x,z_2) \land \cdots \land A_n(z_n,c), \\ A'(c',x) &\leftarrow A_1(x,z_2) \land \cdots \land A_n(z_n,z_{n+1}), \\ A'(y,x) &\leftarrow A_1(x,z_2) \land \cdots \land A_n(z_n,y), \end{aligned}$$

with c, c' constants, x, y variables occurring in both the body and the head, and each z_i a variable occurring only in the body. Rules generated by AMIE+ are connected (i.e., the graph obtained from the rule body by interpreting each term as a node and each atom as an undirected edge is connected), thus precluding rules with unrelated atoms; furthermore, each variable in a rule is required to appear at least twice in different atoms. Other systems following similar approaches include RLvLR (Omran, Wang, and Wang 2018) and RARL (Pirrò 2020). Popper (Cropper and Morel 2021) is an inductive logic programming system which provides flexibility in specifying the syntactic rule structure according to the user's requirements and preferences.

DatalogMTL DatalogMTL (Brandt et al. 2018; Wałęga et al. 2019) is an extension of Datalog with operators from metric temporal logic (Koymans 1990). These operators build upon the standard linear temporal logic (LTL) operators, such as $\diamondminus$ standing for "sometime in the past", $\boxminus$ for "always in the past", and $\mathcal{S}$ for "since", as well as their future counterparts $\diamondplus$ for "sometime in the future", $\boxplus$ for "always in the future", and $\mathcal{U}$ for "until". In MTL, however, these LTL operators are annotated with intervals; for instance, the expression $\diamondminus_{[1,2]}\mathit{LiveIn}(x,y)$ is true at time t if entity x lived in location y sometime between times $t-1$ and $t-2$. Similarly, $\boxminus_{[1,2]}\mathit{LiveIn}(x,y)$ holds at time t if x continuously lived in y throughout the aforementioned time interval. In this paper, we consider DatalogMTL interpreted over the integer timeline (Wałęga et al. 2020) and restrict ourselves to a fragment in which metric atoms are generated by the following grammar, where $P(\mathbf{s})$ is a relational atom and ϱ an interval including only non-negative numbers:

$$M ::= P(\mathbf{s}) \mid \boxminus_{\varrho} M \mid \boxplus_{\varrho} M.$$

To simplify notation we will represent punctual intervals (e.g., in subscripts of MTL operators) of the form $[t,t]$ with

[1] This rule was extracted by the application of our approach to the ICEWS18 dataset.

[2] Code of our system together with the benchmarks are available at https://github.com/wdimmy/MTLearn.

t. A rule in this fragment is an expression of the form

$$P(\mathbf{s}) \leftarrow M_1 \wedge \cdots \wedge M_n, \qquad \text{for } n \geq 1, \tag{2}$$

where the body atoms $M_1, \ldots, M_n$ are metric atoms and the head atom $P(\mathbf{s})$ is relational. A program is a finite set of rules. Temporal dataset is a finite set of temporal facts $P(\mathbf{c})@t$ with $P(\mathbf{c})$ a ground relational atom (i.e., with no variables) and $t \in \mathbb{Z}$.

An interpretation $\mathfrak{I}$ is a function assigning truth values to ground relational atoms $P(\mathbf{c})$ and time points $t \in \mathbb{Z}$. It determines if $P(\mathbf{c})$ is satisfied at t, denoted as $\mathfrak{I}, t \models P(\mathbf{c})$, or not, denoted as $\mathfrak{I}, t \not\models P(\mathbf{c})$. This notion of truth assignment extends to other ground metric atoms in the considered fragment as follows:

$$\mathfrak{I}, t \models \boxminus_\varrho M \quad \text{iff} \quad \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t - t' \in \varrho,$$
$$\mathfrak{I}, t \models \boxplus_\varrho M \quad \text{iff} \quad \mathfrak{I}, t' \models M \text{ for all } t' \text{ with } t' - t \in \varrho.$$

For example, an interpretation making atom $\mathit{LiveIn}(Ann, Paris)$ true everywhere within $[10, 30]$ and false elsewhere makes $\boxminus_{[1,2]}\mathit{LiveIn}(Ann, Paris)$ true at the time point 31, but false at 32. An interpretation can be alternatively seen as the (possibly infinite) set of facts that it satisfies, which yields a natural meaning to containment and minimality of interpretations. Interpretation $\mathfrak{I}$ is a model of a temporal fact $P(\mathbf{c})@t$, if $\mathfrak{I}, t \models P(\mathbf{c})$ holds. Then, $\mathfrak{I}$ is a model of a rule of the form (2) if, for each substitution (of constants to variables) ν grounding the rule and for each $t \in \mathbb{Z}$, the condition $\mathfrak{I}, t \models P(\mathbf{s})\nu$ is satisfied whenever $\mathfrak{I}, t \models M_i\nu$ for all $i \in \{1, \ldots, n\}$. We say that $\mathfrak{I}$ as a model of a dataset $\mathcal{D}$ (or program Π), if $\mathfrak{I}$ is a model of all facts in $\mathcal{D}$ (or rules in Π). A program Π and a temporal dataset $\mathcal{D}$ entail a fact $P(\mathbf{c})@t$, written $(\Pi, \mathcal{D}) \models P(\mathbf{c})@t$, if each model of Π and $\mathcal{D}$ satisfies $P(\mathbf{c})@t$. Program Π entails a rule r if every model of Π is also a model of r. Let $\mathfrak{I}_\mathcal{D}$ be the minimal model of $\mathcal{D}$; then $T_\Pi(\mathcal{D})$ is the minimal interpretation containing $\mathfrak{I}_\mathcal{D}$ and satisfying the following property for each ground instance r of a rule in Π: whenever $\mathfrak{I}_\mathcal{D}$ satisfies each body atom of r at a time point t, then $T_\Pi(\mathcal{D})$ satisfies the head of r at t. Materialisation-based reasoning algorithms syntactically apply rules to datasets in order to mimic the semantics of the immediate consequence operator. In particular, they compute a set $r[\mathcal{D}]$ of temporal facts derived by rule r from $\mathcal{D}$; more formally, $r[\mathcal{D}]$, for a rule r of the form (2), consists of all temporal facts $P(\mathbf{s})\nu@t$ such that ν is a substitution grounding r and $\mathcal{D} \models M_i\nu@t$ for each $i \in \{1, \ldots, n\}$—that is, $M_i\nu$ holds at t in $\mathfrak{I}_\mathcal{D}$.

Temporal Link Prediction We identify a *temporal knowledge graph* (tKG) with a temporal dataset $\mathcal{D}$ consisting of temporal facts of the form $R(c_1, c_2)@t$. In the context of link prediction, it is assumed that an incomplete tKG $\mathcal{D}$ is given and the aim is to predict which temporal facts hold in the (unknown) completion $\mathcal{D}^*$ of $\mathcal{D}$. We focus on answering prediction queries of the form $R(x, b)@t$ or $R(a, x)@t$ where x is a variable, whereas a, b and t are fixed constants and a time point, respectively. Hence, given $\mathcal{D}$ and a prediction query q, we focus on the task of finding variable assignments making the resulting temporal fact true in $\mathcal{D}^*$.

The most common temporal link prediction variants are referred to as *interpolation* and *extrapolation* (Jin et al. 2020; Sun et al. 2021; Chen and Wang 2022; Jia et al. 2023; Ma et al. 2023; Wang et al. 2023). Let 0 and $t_{\max}$ be the least and largest time points mentioned in the input tKG $\mathcal{D}$; for simplicity of presentation we will assume that 0 is the minimal time point in all datasets. In the interpolation setting, a time point t in the prediction query satisfies $0 \leq t \leq t_{\max}$. In contrast, in the extrapolation (or forecasting) setting, it holds that $t > t_{\max}$. A common approach to temporal link prediction tasks is to extend neural architectures and embedding techniques with a temporal dimension. Prominent models include RE-Net (Jin et al. 2020), TTransE (Leblay and Chekol 2018), TA-DisMult (Garcia-Duran, Dumančić, and Niepert 2018), CyGNet (Zhu et al. 2021), TeLM (Xu et al. 2021), TIMEPLEX (Jain et al. 2020), TComplEx (Lacroix, Obozinski, and Usunier 2020), and LCGE (Niu and Li 2023).

3 Our Method

At a high level, the application of our approach to a temporal dataset $\mathcal{D}$ consists of the following steps. First, we compute a sequence of datasets $\mathcal{F}_t$, each containing the facts in $\mathcal{D}$ holding at t. Using a sliding window of size w (w is a configurable parameter), the algorithm applies to each window a rule learner to extract a Datalog program, with each generated rule accompanied by a confidence score. For every window we rewrite the extracted Datalog rules into DatalogMTL rules and then combine rules from various windows (using one of the scoring strategies which we will introduce later on) into a single DatalogMTL program with confidence scores assigned to rules.

Unlike other approaches (Dasgupta, Ray, and Talukdar 2018; Jin et al. 2020; Zhu et al. 2021; Han et al. 2021; Park et al. 2022; Liu et al. 2022), our method is not restricted to tKGs as we do not impose any restrictions on the arity of predicates occurring in $\mathcal{D}$. Furthermore, our approach does not introduce constraints on the form of the extracted Datalog rules beyond those imposed by the Datalog rule learner.

3.1 The Rule Extraction Algorithm

Our rule extraction approach is specified in Algorithm 1 and Figure 1 illustrates its execution on a concrete example. We assume, for convenience of presentation, that the earliest time point in any input dataset is 0.

Algorithm 1 takes as input a temporal dataset $\mathcal{D}$ and is parameterised by a window size $w \in \mathbb{N}$, an off-the-shelf Datalog rule learner L, and a strategy for scoring temporal rules (we describe details of scoring strategies in Section 3.2). In Line 1, the algorithm sets $t_{\max}$ to the maximal time point in $\mathcal{D}$ and in Line 2, for each $t \in \{0, \ldots, t_{\max}\}$, it computes a dataset $\mathcal{F}_t$. In particular, we rewrite each temporal fact $P(\mathbf{c})@t$ in the input temporal dataset $\mathcal{D}$ as a fact $P_t(\mathbf{c})$, where P_t is a fresh predicate uniquely associated to time point t. Then, $\mathcal{F}_t$ consists of all facts associated to t. We illustrate the construction of datasets $\mathcal{F}_t$ from $\mathcal{D}$ on the left-hand side of Figure 1. Here, the input temporal dataset $\mathcal{D}$ is transformed into datasets $\mathcal{F}_0$, $\mathcal{F}_1$, $\mathcal{F}_2$, and $\mathcal{F}_3$ through

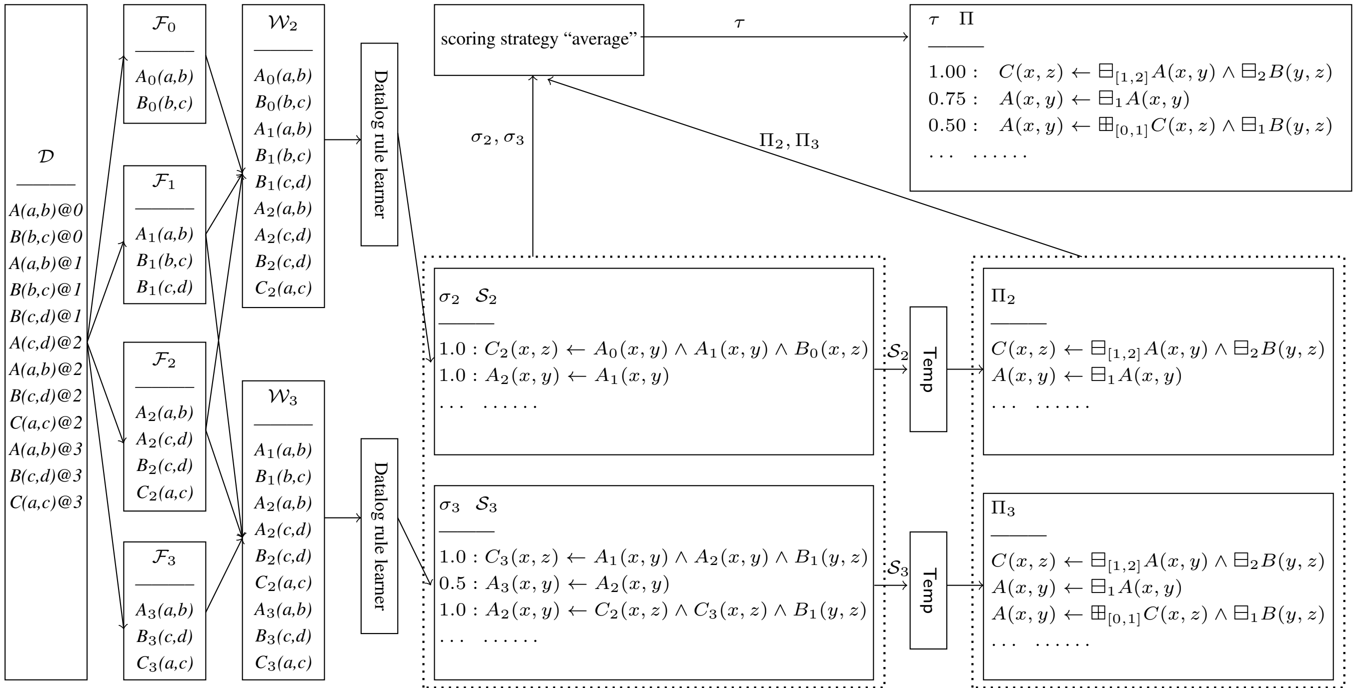

Figure 1: Execution of Algorithm 1 on the running example, with window size $w = 3$ and a scoring strategy "average"; the input temporal dataset is depicted on the left-hand-side and the output set of DatalogMTL rules together with their confidence scores are in the top-right

Algorithm 1: Temporal rule extraction

Input: A temporal dataset $\mathcal{D}$
Parameters: A window size $w \in \mathbb{N}$, a Datalog rule learner L, and a scoring strategy
Output: A set of temporal rules and their scores

1 $t_{\max} \coloneqq$ the maximal time point in $\mathcal{D}$;
2 $\mathcal{F}_t \coloneqq \{(P_t(\mathbf{c}) \mid P(\mathbf{c})@t \in \mathcal{D}\}$, for each $0 \leq t \leq t_{\max}$;
3 **for** *each* $t \in \{w-1, \ldots, t_{\max}\}$ **do**
4 $\quad \mathcal{W}_t \coloneqq \mathcal{F}_{t-w+1} \cup \cdots \cup \mathcal{F}_t$;
5 $\quad (\mathcal{S}_t, \sigma_t) \coloneqq \mathsf{L}(\mathcal{W}_t)$;
6 $\quad \Pi_t \coloneqq \{\mathsf{Temp}(r) \mid r \in \mathcal{S}_t\ \}$;
7 $\Pi = \Pi_w \cup \cdots \cup \Pi_{t_{\max}}$;
8 Construct $\tau : \Pi \mapsto [0,1]$ using the scoring strategy;
9 **return** (Π, τ);

the introduction of fresh binary predicates A_i, B_i, and C_i for each time point $0 \leq i \leq 3$.

In Lines 3–6, the algorithm constructs *windows* from datasets $\mathcal{F}_t$, applies the Datalog rule learner to each window, and transforms the constructed rules into DatalogMTL rules. In particular, each window $\mathcal{W}_t$ is the union of $w+1$ consecutive datasets $\mathcal{F}_{t-w+1}, \ldots, \mathcal{F}_t$ (Line 4). In our running example from Figure 1, we let $w = 3$, so the algorithm constructs two windows $\mathcal{W}_2 = \mathcal{F}_0 \cup \mathcal{F}_1 \cup \mathcal{F}_2$ and $\mathcal{W}_3 = \mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$. Note that each $\mathcal{W}_t$ is a (non-temporal) dataset. For each such $\mathcal{W}_t$, Algorithm 1 applies an off-the-shelf Datalog rule learner L to compute a set $\mathcal{S}_t$ of Datalog rules together with a scoring function σ_t (Line 5) which assigns to each constructed rule a value from the interval $[0,1]$. In our example, the rule learner extracts $\mathcal{S}_2$ and σ_2 from $\mathcal{W}_2$, as well as $\mathcal{S}_3$ and σ_3 from $\mathcal{W}_3$.

In Line 6, each Datalog rule r in $\mathcal{S}_t$ is rewritten into a temporal DatalogMTL rule using function Temp, which implements the following computations. First, the subscript t of the head predicate of r is removed; for example if the head of r is $C_t(x,y)$, then it is replaced with $C(x,y)$. Second, every (maximal) sequence of body atoms $P_m(\mathbf{s}), P_{m+1}(\mathbf{s}), \ldots, P_n(\mathbf{s})$ in r with consecutive subscripts and $m \leq t$ is replaced with $\boxminus_{[t-n,t-m]}P(\mathbf{s})$. Similarly, every (maximal) sequence of body atoms $P_m(\mathbf{s}), P_{m+1}(\mathbf{s}), \ldots, P_n(\mathbf{s})$ in r with consecutive subscripts and $m \geq t$ is replaced with $\boxplus_{[m-t,n-t]}P(\mathbf{s})$. Finally, we simplify presentation of the obtained temporal rules by replacing atoms with punctual intervals of the form $\boxminus_{[k,k]}P(\mathbf{s})$ with $\boxminus_k P(\mathbf{s})$ and atoms of the form $\boxminus_{[0,0]}P(\mathbf{s})$ with $P(\mathbf{s})$; We perform an analogous simplification for atoms mentioning $\boxplus$. For example, a Datalog rule

$$A_2(x,y) \leftarrow C_2(x,z) \wedge C_3(x,z) \wedge B_1(y,z)$$

extracted from window $\mathcal{W}_3$ in our running example is rewritten as the DatalogMTL rule

$$A(x,y) \leftarrow \boxplus_{[0,1]}C(x,z) \wedge \boxminus_{[1,1]}B(y,z).$$

Next, in Line 7, the algorithm computes the union Π of all generated DatalogMTL rules from all windows. Finally, in Line 8, the algorithm computes a scoring function τ which assigns a value ranging from 0 to 1 to each rule in Π. The construction of τ depends on the scoring strategy, which is a parameter of the algorithm. We consider several types of scoring strategies, as described in the following subsection, and in Section 4 we will experimentally evaluate the impact of choosing a particular scoring strategy.

3.2 Scoring Strategies

In this section, we describe different strategies implemented in MTLearn for assigning scores to temporal rules.

The first type of strategy involves the computation of scores $\tau_t : \Pi_t \mapsto [0,1]$ for each time point t individually; these are then combined to derive the overall scoring function τ. For each time point t and rule $r \in \Pi_t$, we assign $\tau_t(r) = \sigma_t(r')$, where $r' \in \mathcal{S}_t$ represents the unique Datalog rule such that $\mathsf{Temp}(r') = r$ (uniqueness is guaranteed by the definition of Temp) and $\sigma_t(r')$ is the score assigned by the Datalog rule learner to r' in window $\mathcal{W}_t$. The computation of $\tau(r)$ for a generated DatalogMTL rule r is based on the list Val_r containing the defined values among $\tau_w(r), \ldots, \tau_{t_{\max}}(r)$, where for some $t \in \{w, \ldots, t_{\max}\}$ we may have $r \notin \Pi_t$ and hence $\tau_t(r)$ may be undefined. We propose four alternative ways of computing $\tau(r)$:

$$
\begin{aligned}
\text{maximum:} \quad & \max(Val_r), \\
\text{average:} \quad & \operatorname{avg}(Val_r), \\
\text{weighted maximum:} \quad & \max(Val_r) \cdot \frac{|Val_r|}{t_{\max} - w + 1}, \\
\text{weighted average:} \quad & \operatorname{avg}(Val_r) \cdot \frac{|Val_r|}{t_{\max} - w + 1}.
\end{aligned}
$$

Example In our example from Figure 1, rule $A(x,y) \leftarrow \boxminus_1 A(x,y)$ is obtained in Π_2 and Π_3 with scores 1.0 and 0.5, respectively. We use "average" as the scoring strategy for $\Pi = \Pi_2 \cup \Pi_3$, and so τ assigning a score of 0.75 to the DatalogMTL rule $A(x,y) \leftarrow \boxminus_1 A(x,y)$.

Next, we define a scoring strategy which disregards the scores computed by the off-the-shelf Datalog rule learner and relies instead on temporalised versions SC_T and HC_T of the standard confidence and head coverage metrics in the field of rule learning. In particular, we set $\tau(r)$ to

$$\beta \cdot SC_T(r) + (1-\beta) \cdot HC_T(r), \tag{3}$$

for $\beta \in [0,1]$ a parameter controlling the influence of SC_T and HC_T on the function τ. While confidence (SC) and head coverage (HC) metrics are widely used in the context of rule learning (Chen, Wang, and Goldberg 2016; Meilicke et al. 2019; Galárraga et al. 2015), the definition of their temporal counterparts varies depending on the form of temporal rules, and different approaches adopt different definitions (Liu et al. 2022; Omran, Wang, and Wang 2019). As a starting point, we will first discuss the definition of SC and HC in the Datalog setting and then, we will show how to adapt them to the setting of DatalogMTL.

In the non-temporal case, for a Datalog rule r and a fixed dataset $\mathcal{D}$, SC and HC are usually given by the fractions

$$SC(r) = \frac{\operatorname{supp}(r)}{\operatorname{body-supp}(r)}, \quad HC(r) = \frac{\operatorname{supp}(r)}{\operatorname{head-supp}(r)},$$

where the *rule support* $\operatorname{supp}(r)$, *body support* $\operatorname{body-supp}(r)$, and the *head support* $\operatorname{head-supp}(r)$ are defined as follows, using the set G of all ground instances of r. The rule support, $\operatorname{supp}(r)$, is the number of distinct ground head atoms H such that there exists in G a rule with head H and body B, both of which hold in $\mathcal{D}$. The definitions of $\operatorname{body-supp}(r)$ and $\operatorname{head-supp}(r)$ are analogous, but they require only the body or only the head, respectively, to hold in $\mathcal{D}$. Formally:

$$
\begin{aligned}
\operatorname{supp}(r) &= |\{H \mid (H \leftarrow B) \in G, \mathcal{D} \models H, \mathcal{D} \models B\}|, \\
\operatorname{body-supp}(r) &= |\{H \mid (H \leftarrow B) \in G, \mathcal{D} \models B\}|, \\
\operatorname{head-supp}(r) &= |\{H \mid (H \leftarrow B) \in G, \mathcal{D} \models H\}|.
\end{aligned}
$$

The notions of support for temporal rules should also take into account the time points in which they hold. These can be obtained, for example, by grounding the temporal variables in a rule, as in TLogic (Liu et al. 2022), or by defining *dynamic* versions of SC and HC recursively through time, as in StreamLearner (Omran, Wang, and Wang 2019). Our definition is close to that in TLogic, except that in DatalogMTL rules do not mention temporal variables (DatalogMTL rules are meant to hold in all points of the timeline), so their temporal component cannot be grounded. Instead, we define *temporalised variants* of SC and HC as follows:

$$SC_T(r) = \frac{\operatorname{supp}_T(r)}{\operatorname{body-supp}_T(r)}, \quad HC_T(r) = \frac{\operatorname{supp}_T(r)}{\operatorname{head-supp}_T(r)},$$

where the temporalised variants of supports count the number of facts $H@t$, with t ranging over all time points:

$$
\begin{aligned}
\operatorname{supp}_T(r) &= |\{H@t \mid (H \leftarrow B) \in G, \mathcal{D} \models H@t, \mathcal{D} \models B@t\}|, \\
\operatorname{body-supp}_T(r) &= |\{H@t \mid (H \leftarrow B) \in G, \mathcal{D} \models B@t\}|, \\
\operatorname{head-supp}_T(r) &= |\{H@t \mid (H \leftarrow B) \in G, \mathcal{D} \models H@t\}|,
\end{aligned}
$$

where $\mathcal{D} \models B@t$ is an abbreviation for stating that $\mathcal{D} \models M@t$, for each atom M in the conjunction B.

Note that in our setting, SC_T and HC_T can be equivalently defined using the result $r[\mathcal{D}]$ of applying r to $\mathcal{D}$ and $\mathcal{D}_r$, which we define as the set of all facts $H@t \in \mathcal{D}$ such that H is a grounding of the head of r, namely we obtain the following equalities:

$$SC_T(r) = \frac{|r[\mathcal{D}] \cap \mathcal{D}|}{|r[\mathcal{D}]|}, \qquad HC_T(r) = \frac{|r[\mathcal{D}] \cap \mathcal{D}|}{|\mathcal{D}_r|}.$$

Indeed, observe first that $|\mathcal{D}_r| = \operatorname{head-supp}(r)$, by the definition. Moreover, $|r[\mathcal{D}]|$ is the number of facts which can be derived from $\mathcal{D}$ by an application of r, that is, by grounding r and checking if the obtained body holds in $\mathcal{D}$. Hence, $|r[\mathcal{D}]| = \operatorname{body-supp}_T$. Finally, $|r[\mathcal{D}] \cap \mathcal{D}|$ is the number of facts in $r[\mathcal{D}]$ which additionally hold in $\mathcal{D}$, and so, $|r[\mathcal{D}] \cap \mathcal{D}| = \operatorname{supp}_T(r)$. Thus, SC_T and HC_T can be determined using a reasoner which can compute $r[\mathcal{D}]$; in particular, we obtain it using the DatalogMTL reasoner MeTeoR (Wang et al. 2022).

Example Let r be the DatalogMTL rule $A(x,y) \leftarrow \boxminus_1 A(x,y)$ which was extracted in Figure 1 from both windows (so r belongs to both Π_2 and Π_3). The set $r[\mathcal{D}]$, of facts derived by an application of r to $\mathcal{D}$, consists of five temporal facts $A(a,b)@1$, $A(a,b)@2$, $A(c,d)@3$, $A(a,b)@3$, and $A(a,b)@4$. Three of these, namely $A(a,b)@1$, $A(a,b)@2$, and $A(a,b)@3$ belong to $\mathcal{D}$, so $|r[\mathcal{D}] \cap \mathcal{D}| = 3$. Hence, $SC_T(r) = \frac{3}{5} = 0.6$. In this example $\mathcal{D}_r$ coincides with $r[\mathcal{D}]$, so $|\mathcal{D}_r| = 5$, and so $HC_T(r) = \frac{3}{5} = 0.6$. Hence, in this particular example $\tau(r) = 0.6$ no matter what the value of the parameter β is.

4 Experiments

We have tested our approach on temporal link prediction (see Section 2), which is commonly used for evaluating temporal rule learners (Omran, Wang, and Wang 2019; Liu et al. 2022; Xiong et al. 2024a). We conducted experiments across standard benchmarks (ICEWS14, ICEWS18, ICEWS0515) as well as our newly introduced synthetic benchmarks (tLUBM and iTEMP) and considered both the extrapolation and interpolation settings.

4.1 Experimental Setup and Metrics

We have adopted the standard protocol with temporal filtering (Han et al. 2021). Each dataset (tKG) is split into training, validation, and testing sets according to the restrictions on time points imposed by the extrapolation and interpolation settings, where appropriate. We used the training and validation datasets to extract a DatalogMTL program Π and a function $\tau : \Pi \mapsto [0,1]$ assigning scores to rules of Π.

To test performance of (Π, τ) on temporal link prediction we proceeded as follows. For each fact $R(a,b)@t$ in the test dataset, we construct a query $R(x,b)@t$, for which a is assumed to be the correct answer, and a query $R(a,x)@t$, for which b is the correct answer. Assume that the query is of the form $R(x,b)@t$ with the correct answer a. In the interpolation setting we let $\mathcal{D}$ be the union of the training and validation sets, whereas in the extrapolation setting $\mathcal{D}$ contains also all facts from the test set with time points smaller than t, which corresponds to the so-called *single-step* setting (Gastinger et al. 2022; Liu et al. 2022). Then, we compute the set of constants c such that $R(c,b)@t$ holds in the interpretation $T_\Pi(\mathcal{D})$ obtained by applying Π to $\mathcal{D}$. We let the score of each such answer c be the maximum amongst $\tau(r)$ for each $r \in \Pi$ deriving $R(c,b)@t$ from $\mathcal{D}$. Formally, the score of c is $\max_{\tau(r)}\{r \in \Pi \mid R(c,b)@t \in r[\mathcal{D}]\}$. We sort answers in descending order of scores (ties are broken by considering the scores of the next highest-score rules producing the answer, and if this does not differentiate the answers we use the alphabetic ordering). We perform time-aware filtering (Han et al. 2021; Sun et al. 2021; Li et al. 2022; Liu et al. 2022), where we delete from the list of answers all constants $c \neq a$ such that $R(c,b)@t$ holds in the training, validation, or test dataset. The rank of an answer is given by its position on the list. The process is analogous for queries of the form $R(a,x)@t$.

We use mean reciprocal rank (MRR) and Hits@k, for $k \in \{1,3,10\}$, as standard metrics. MRR is defined as the mean of the reciprocals $\frac{1}{rank_i}$ over all queries q_i, where $rank_i$ is the rank of the correct answer in the list of answers to a query q_i (if the correct answer is not on the list, we let $\frac{1}{rank_i} = 0$). Hits@k is the number of queries q_i, for which $rank_i \leq k$, divided by the number of all queries. Both metrics yield values within $[0,1]$ with higher values indicating better performance. We report these values as percentages.

Some of our benchmarks are equipped with a set of gold-standard temporal rules, which are used for generating the validation and test sets. For such benchmarks we also introduce the *rule quality* (RQ) metric. Given a program Π extracted by MTLearn, a gold-standard program Π' from the benchmark, a union of training, validation, and test sets $\mathcal{D}$, and a test set $\mathcal{D}_T$, the value of RQ is the percentage of rules r in Π' such that $T_{\{r\}}(\mathcal{D}) \cap \mathcal{D}_T \subseteq T_{\Pi'}(\mathcal{D})$. Hence RQ indicates the percentage of rules in Π' which derive facts in $\mathcal{D}_T$ that are also derived by Π.

Our approach takes as hyperparameters a window size, a Datalog rule learner, and a scoring strategy. We have determined the following hyperparameters which maximise MRR scores on the validation sets: window size 10 for all ICEWS benchmarks and 6 for the remaining benchmarks, AnyBurl as a rule learner, and the scoring strategy from Equation (3) with $\beta = 0.5$. Whenever the hyperparameters are not specified in some experiment, we use the above values as default.

4.2 Extrapolation

Our main experiments consider the extrapolation setting, where answers to queries are computed based on the facts which did hold in the past.

Benchmarks We used several standard benchmarks constructed from the Integrated Crisis Early Warning System[3] (ICEWS) dataset. These include CEWS14, ICEWS18, and ICEWS0515, which contain data about years 2014, 2018, and from 2005 to 2015, respectively. We use their splits into training, validation, and test sets as provided by Han et al. (2021) and adopted also by Lin et al. (2023). We also generated six synthetic benchmarks tLUBM and iTEMP_1–iTEMP_5, each consisting of a training and validation datasets, together with a set of gold-standard DatalogMTL rules. Testing data consists of all facts which can be derived by one round of application of the gold-standard rules to the union of the training and validation sets, and such that the time points in these facts are greater than time points in the training and validation sets (the latter condition is required by the extrapolation setting). tLUBM is a temporalised version of LUBM (Guo, Pan, and Heflin 2005), whereas iTEMP_1–iTEMP_5 are designed using the iTemporal[4] (Bellomarini, Nissl, and Sallinger 2022) generator. Table 1 provides key statistics of these benchmarks, namely the number of entities, predicates, time points, as well as the number of temporal facts in the training, validation, and test sets, respectively. The table also provides the number of gold-standard temporal rules in the synthetic benchmarks.

Baseline Models We used the following models as baselines. TTransE (Jiang et al. 2016) which is a temporal extension of the KG embedding model TransE (Bordes

[3]https://dataverse.harvard.edu/dataverse/icews
[4]https://github.com/kglab-tuwien/iTemporal.git

	Entities	Predicates	Points	Train	Valid	Test	Rules
ICEWS14	7,128	230	365	63,685	13,823	13,222	–
ICEWS18	23,033	256	304	373,018	45,995	49,945	–
ICEWS0515	10,488	251	4017	322,958	69,224	69,147	–
tLUBM	29,265	28	145	389,498	50,523	61,026	46
iTEMP$_1$	1,000	10	100	40,000	5,000	5,000	10
iTEMP$_2$	1,000	14	100	40,000	5,000	5,000	12
iTEMP$_3$	1,000	18	100	40,000	5,000	5,000	14
iTEMP$_4$	1,000	20	100	40,000	5,000	5,000	16
iTEMP$_5$	1,000	22	100	40,000	5,000	5,000	20

Table 1: Statistics of extrapolation benchmarks

et al. 2011), TA-DistMult (Garcia-Duran, Dumančić, and Niepert 2018) which relies on recurrent neural networks to learn time-aware representations, TNTComplEx (Lacroix, Obozinski, and Usunier 2020) which learns complex-valued temporal-aware embeddings, RE-NET (Jin et al. 2020) which combines an RNN-based event encoder and a neighborhood aggregator to capture both time and KG structure, DE-SimplE (Goel et al. 2020) which proposes a diachronic entity embedding with a static segment and a time-varying segment, xERTE (Han et al. 2021) which is based on a subgraph extraction, TLogic (Liu et al. 2022) which extracts temporal rules using random walks, and TECHS (Lin et al. 2023) which exploits a graph convolutional network to embed topological structures and temporal dynamics. We could not obtain a usable version of StreamLearner.

Results The results for the ICEWS14, ICEWS18, and ICEWS0515 benchmarks are summarised in Table 2, whereas the results for the synthetic benchmarks tLUBM and iTEMP$_1$–iTEMP$_5$ are presented in Table 3. The results for baseline models on the standard benchmarks CEWS14, ICEWS18, and ICEWS0515 are as reported by Lin et al. (2023), whereas the results on our synthetic benchmarks tLUBM and iTEMP$_1$–iTEMP$_5$ have been obtained by exploiting default hyperparameters and using the validation set to perform early stop of the training. Recall that MTLearn exploits Datalog rule learners. Such rule learners are often non-deterministic, for example due to reliance on random sampling, and so, distinct runs of MTLearn are also non-deterministic in the sense that they can lead to extraction of different temporal rules. Thus, for each experiment we performed five independent runs of MTLearn and reported mean results.

As shown in Table 2, TECHS, TLogic, and MTLearn obtained the highest scores on ICEWS14, ICEWS18, and ICEWS0515, with TECHS being usually slightly better than the other two models. It is worth to observe, however, that MTLearn and TLogic are able to also provide interpretable temporal rules—a key advantage—while providing results comparable to those obtained by TECHS. On ICEWS18 MTLearn was able to slightly outperform TECHS for H@1.

Test data in the synthetic benchmarks tLUBM and iTEMP$_1$–iTEMP$_5$ is generated by temporal rules, which may be a more appropriate setting for temporal rule learning models. As Table 3 shows, in this case MTLearn obtains very high scores and outperforms all the other approaches, with TLogic being the second-best model. Unfortunately, we could not gain access to TECHS's code and hence we could not test it on these benchmarks. All models typically obtained better results in synthetic benchmarks than on ICEWS-based benchmarks, likely due to their regular and structured nature. In addition, Table 4 suggests that MTLearn was able to achieve very high RQ scores; interestingly, on iTEMP$_2$ we obtained a score of 100%. These results suggest a strong alignment between the benchmark gold-standard rules and the rules generated by our system.

Choice of Hyperparameters Furthermore, we have analysed the impact of MTLearn parameters, namely the choice of Datalog rule learner, size of a window, and scoring strategy, on the performance of our approach. For these experiments we used the ICEWS18 and tLUBM benchmarks.

Rule Learners We have tested MTLearn using AnyBURL, AMIE+, and Popper as Datalog rule learners. It turns out that using AnyBURL leads to the highest scores of MTLearn, which seems to be correlated with high performance of AnyBURL on Datalog rule learning task (Meilicke et al. 2019). Indeed, when using AnyBURL (with window size 5 and the scoring strategy from Equation (3) with $\beta = 0.5$), we observed the following improvements compared to using AMIE+: *(i)* for ICEWS18, an increase of 3.8% for MRR, of 4.3% for H@1, and of 3.1% for H@10; *(ii)* for tLUBM, an increase of 2.2% for MRR, of 3.3% for H@1, and of 2.6% for H@10. We obtained similar performance gaps when comparing MTLearn using AnyBURL and Popper: *(iii)* for ICEWS18 using AnyBURL leads to an increase of 5.3% for MRR, 4.6% for H@1, and 4.7% for H@10; *(iv)* for tLUBM using AnyBURL leads to an increase of 3.4% for MRR, of 3.5% for H@1, and of 3.7% for H@10.

Window Size We conducted experiments with window sizes 2–6, 8, and 10 using AnyBURL as a default rule learner and the scoring strategy from Equation (3) with $\beta = 0.5$. As shown in Figure 2, the performance of MTLearn improves overall as the window size increases; indeed, larger window sizes may allow for the discovery of rules capturing dependencies which take into account larger fragments of the timeline. For larger window sizes, however, performance levels off, indicating that further increasing the window size does not lead to significant gains.

Scoring Strategies We compared the impact using different scoring strategies: maximum ($\max$), average (avg), weighted maximum (w-$\max$), weighted average (w-avg), and the one from Equation (3) with varying values of β. For this experiment we used AnyBURL as a Datalog rule learner and we set the window size to 10 and 5 for ICEWS18 and tLUBM benchmarks, respectively. As shown in Table 5, w-$\max$ and w-avg scoring strategies outperform their non-weighted counterparts, but the best results were achieved using the strategy from Equation (3) with $\beta = 0.5$. It is worth observing, that although this strategy leads to the best performance, it requires most complex computations (it requires running a DatalogMTL reasoner). Therefore, whenever computational resources are limited, it maybe preferable to exploit other scoring strategies.

	ICEWS14				ICEWS18				ICEWS0515			
	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
TTransE	13.4	3.1	17.3	34.5	8.3	1.9	8.6	21.9	15.7	5.0	19.7	38.0
TA-DistMult	26.5	17.1	30.2	45.4	16.8	8.6	18.4	33.6	24.3	14.6	27.9	44.2
DE-SimplE	32.7	24.4	35.7	49.1	19.3	11.5	21.9	34.8	35.0	25.9	39.0	52.8
TNTComplEx	32.1	23.4	36.0	49.1	21.2	13.3	24.0	36.9	27.5	19.5	30.8	42.9
CyGNet	32.7	23.7	36.3	50.7	24.9	15.9	28.3	42.6	35.0	25.7	39.1	52.9
RE-Net	38.3	28.7	41.3	54.5	28.8	19.1	32.4	47.5	43.0	31.3	46.9	63.5
xERTE	40.8	32.7	45.7	57.3	29.3	21.0	33.5	46.5	46.6	37.8	52.3	63.9
TLogic	43.0	33.6	48.3	61.2	29.8	20.5	34.0	48.5	47.0	36.2	53.1	67.4
TECHS	**43.9**	**34.6**	**49.4**	**62.0**	**30.9**	21.8	**35.4**	**49.8**	**48.4**	**38.3**	**54.7**	**68.9**
MTLearn	42.8	33.9	48.3	60.4	28.8	**22.0**	34.8	46.7	47.5	35.6	53.4	67.1

Table 2: Results for the extrapolation setting with the highest scores written in bold and the second highest underlined; baseline results are provided by Lin et al. (2023)

	tLUBM		iTEMP_1		iTEMP_2		iTEMP_3		iTEMP_4		iTEMP_5	
	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@3	MRR	H@10	MRR	H@10
TTransE	38.9	49.3	50.2	61.4	52.1	63.0	54.2	64.3	50.1	63.2	56.3	67.2
CyGNet	42.7	58.2	68.8	76.2	70.1	79.8	69.4	78.8	72.4	80.4	65.1	74.9
RE-Net	49.8	59.1	72.5	80.1	73.2	82.1	72.7	81.8	72.4	82.4	68.1	78.8
xERTE	59.4	70.1	78.3	84.2	89.2	93.4	73.4	83.2	76.4	84.8	84.3	90.8
TLogic	64.1	72.4	85.3	91.1	90.2	94.1	85.2	92.9	79.2	88.1	90.1	94.9
MTLearn	**65.2**	**75.1**	**87.3**	**94.2**	**93.4**	**96.7**	**89.4**	**95.2**	**81.4**	**92.8**	**92.3**	**96.2**

Table 3: Results for the extrapolation setting on the synthetic benchmarks; the highest scores are in bold and the second highest are underlined

tLUBM	iTEMP_1	iTEMP_2	iTEMP_3	iTEMP_4	iTEMP_5
84.1	90.1	100.0	86.1	88.3	85.2

Table 4: RQ scores for MTLearn on synthetic benchmarks

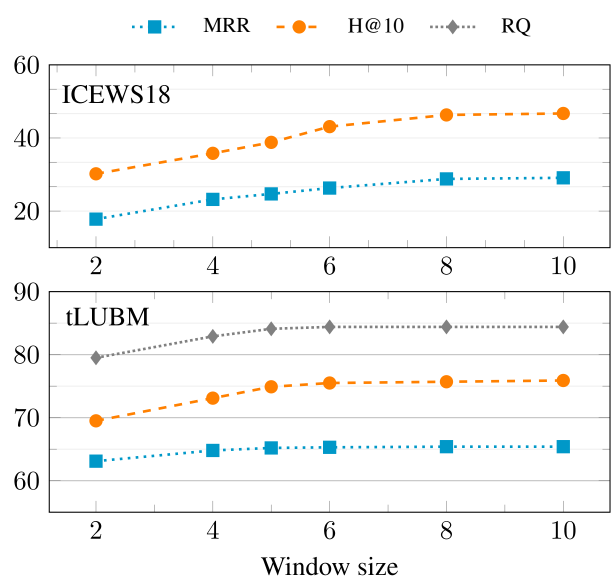

Figure 2: Impact of increasing window size on MTLearn

4.3 Interpolation

MTLearn is also applicable to the interpolation setting, and in this section we discuss the results we obtained.

Benchmarks We used the versions of ICEWS14 and ICEWS0515 provided by Dasgupta, Ray, and Talukdar (2018), where the split into training, validation, and test sets does not involve any restriction on ordering of time points and hence is well-suited for interpolation settings. We will use ICEWS14^I and ICEWS0515^I when referring to these benchmarks in order to emphasise the difference between them and their counterparts used for the extrapolation setting. It is worth noting that ICEWS14^I and ICEWS0515^I are commonly used in interpolation settings (Lacroix, Obozinski, and Usunier 2020; Xu et al. 2021). The key statistics of these benchmarks are summarised in Table 6.

Baseline Models As a baseline we considered five models applicable to the interpolation setting. All of them are embedding-based approaches: HyTE (Dasgupta, Ray, and Talukdar 2018) which uses a temporally aware KG embedding method explicitly incorporating time in the entity-relation space, ATiSE (Xu et al. 2019) which incorporates time information into entity/relation representations by using additive time series decomposition, TeRo (Xu et al. 2020) which introduces a temporal evolution of entity em-

	ICEWS18		tLUBM	
	MRR	H@10	MRR	H@10
max	25.8	43.5	62.5	72.2
avg	26.4	44.4	62.1	71.4
w-max	27.7	45.2	63.1	73.1
w-avg	27.2	44.7	62.9	72.2
$\beta = 0.3$	<u>28.5</u>	<u>46.6</u>	<u>64.3</u>	**75.7**
$\beta = 0.5$	**28.8**	**46.7**	**65.1**	<u>75.1</u>
$\beta = 0.7$	27.9	46.1	63.9	74.6

Table 5: Impact of scoring strategies on MTLearn; the highest scores are written in bold and the second highest are underlined

	Entities	Predicates	Points	Train	Valid	Test
ICEWS14^I	7,128	230	365	72,826	8,941	8,963
ICEWS0515^I	10,488	251	4,017	368,962	46,275	46,092

Table 6: Statistics of interpolation benchmarks

bedding as a rotation in the complex vector space, TeLM (Xu et al. 2021) which performs a 4th-order tensor factorization of a tKG, and LCGE (Niu and Li 2023) which adopts a rule-guided predicate embedding regularisation strategy for learning the causality among events.

Results Table 7 summarises the evaluation results, where all results for the baseline methods are as provided by Niu and Li (2023). We have compared these results with two versions of our method, namely MTLearn, which extracts rules as described in Section 3 and MTLearn^p, which restricts the form of extracted rules so that future temporal operators are not allowed. Thus, such rules represent only dependencies between past and future facts. Notice that in the extrapolation setting the distinction between MTLearn and MTLearn^p is not meaningful, as all queries are about future time points, and so, rules with future operators in rule bodies do not allow us to derive answers to such queries. In the interpolation setting, in contrast, the distinction between MTLearn and MTLearn^p allows us to verify the impact of allowing future operators in rules on the performance of the method. As shown in Table 7, MTLearn offers a competitive performance with respect to embedding-based methods designed for the interpolation setting. Although LCGE outperforms all other methods, MTLearn is the second best approach and comes with a significant added advantage of being able to provide interpretable rules. It is worth observing also that MTLearn significantly outperforms MTLearn^p, which suggests the importance of considering in the interpolation setting temporal rules containing both past and future operators in rule bodies.

4.4 Rule Application to Other Benchmarks

Since MTLearn enables the extraction of expressive temporal rules, another potential advantage of our method is that (unlike many other approaches) once trained on some benchmark, it could be also used for other benchmarks. To verify this hypothesis, we have used MTLearn to extract rules from

	ICEWS14^I			ICEWS0515^I		
	MRR	H@1	H@10	MRR	H@1	H@10
HyTE	29.7	10.8	65.5	31.6	11.6	68.1
ATiSE	55.0	43.6	75.0	51.9	37.8	79.4
TeRo	56.2	46.8	73.2	58.6	46.9	79.5
TeLM	62.5	54.5	77.4	67.8	59.9	82.3
LCGE	**92.5**	**91.6**	**93.7**	**91.2**	**90.3**	**92.5**
MTLearn^p	60.4	52.3	68.6	56.7	47.2	63.6
MTLearn	<u>73.4</u>	<u>62.5</u>	<u>80.1</u>	<u>74.9</u>	<u>66.4</u>	<u>84.3</u>

Table 7: Results for the interpolation setting with the highest scores written in bold and the second highest underlined; baseline results are provided by Niu and Li (2023)

ICEWS14 (in the extrapolation setting) and evaluated the performance of the obtained rules on temporal link prediction on ICEWS18 (i.e., we have applied extracted rules to the union of the training and validation sets from ICEWS18 and checked performance of this method in answering queries corresponding to ICEWS18). The results are presented in the second row of Table 8 and compared to the scores from the first row, where MTLearn extracts rules from ICEWS18. As expected, scores in the second row all lower, but importantly the difference is not large. Moreover, the scores from the second row are higher than the scores in Table 2 of several baseline models trained in ICEWS18. This suggests that rules extracted by MTLearn encode temporal dependencies transferable to other benchmarks. It is worth observing that although extracted rules can be applied to a benchmark mentioning any entities, it should mention similar relations to those present in the benchmark used for rule extraction. For example, ICEWS18 mentions $23,033$ entities and ICEWS14 only $7,128$, but they mention very similar predicates.

Extract from	Evaluate on	MRR	H@1	H@3	H@10
ICEWS18	ICEWS18	28.8	22.0	34.8	46.7
ICEWS14	ICEWS18	26.4	20.2	31.9	43.1

Table 8: Evaluation of rules extracted from ICEWS14 on ICEWS18

5 Conclusion and Future Work

In this paper, we have introduced a framework named MTLearn for learning DatalogMTL rules from temporal datasets, using off-the-shelf Datalog rule learners. Our approach achieved comparable performance on temporal link prediction (in both extrapolation and interpolation settings) to state-of-the-art baselines, while at the same time providing interpretable rules in an expressive temporal logic programming language DatalogMTL. We expect the performance of MTLearn to improve as increasingly mature Datalog rule learners become available in the future. As future work, we aim to generate DatalogMTL rules using a wider range of operators, which can capture a richer class of temporal dependencies.

Acknowledgments

This research was supported by the EPSRC projects OASIS (EP/S032347/1), ConCuR (EP/V050869/1) and UK FIRES (EP/S019111/1), as well as SIRIUS Centre for Scalable Data Access and Samsung Research UK. For the purpose of Open Access, the authors have applied a CC BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of databases*, volume 8. Addison-Wesley Reading.

Bellomarini, L.; Nissl, M.; and Sallinger, E. 2022. iTemporal: an extensible generator of temporal benchmarks. In *IEEE 38th International Conference on Data Engineering*, 2021–2033.

Bordes, A.; Weston, J.; Collobert, R.; and Bengio, Y. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 301–306.

Brandt, S.; Kalaycı, E. G.; Ryzhikov, V.; Xiao, G.; and Zakharyaschev, M. 2018. Querying log data with metric temporal logic. *Journal of Artificial Intelligence Research* 829–877.

Chen, S., and Wang, J. 2022. A survey on temporal knowledge graphs-extrapolation and interpolation tasks. In *The International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery*, 1002–1014.

Chen, Y.-C.; Lu, P.-E.; Chang, C.-S.; and Liu, T.-H. 2020. A time-dependent SIR model for COVID-19 with undetectable infected persons. *IEEE Transactions on Network Science and Engineering* 3279–3294.

Chen, Y.; Wang, D. Z.; and Goldberg, S. 2016. ScaLeKB: scalable learning and inference over large knowledge bases. *The VLDB Journal* 893–918.

Cropper, A., and Dumančić, S. 2022. Inductive logic programming at 30: a new introduction. *Journal of Artificial Intelligence Research* 765–850.

Cropper, A., and Morel, R. 2021. Learning programs by learning from failures. *Machine Learning* 801–856.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys (CSUR)* 374–425.

Dasgupta, S. S.; Ray, S. N.; and Talukdar, P. P. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2001–2011.

Fürnkranz, J., and Kliegr, T. 2015. A brief overview of rule learning. In *Rule Technologies: Foundations, Tools, and Applications: 9th International Symposium, RuleML*, 54–69.

Fürnkranz, J.; Gamberger, D.; and Lavrač, N. 2012. *Foundations of rule learning*. Springer Science & Business Media.

Galárraga, L. A.; Teflioudi, C.; Hose, K.; and Suchanek, F. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the International Conference on World Wide Web*, 413–422.

Galárraga, L.; Teflioudi, C.; Hose, K.; and Suchanek, F. M. 2015. Fast rule mining in ontological knowledge bases with AMIE+. *The VLDB Journal* 707–730.

Garcia-Duran, A.; Dumančić, S.; and Niepert, M. 2018. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 4816–4821.

Garcia-Molina, H.; Ullman, J. D.; and Widom, J. 2009. *Database systems: the complete book*. Pearson Education.

Gastinger, J.; Sztyler, T.; Sharma, L.; and Schuelke, A. 2022. On the evaluation of methods for temporal knowledge graph forecasting. In *NeurIPS 2022 Temporal Graph Learning Workshop*.

Goel, R.; Kazemi, S. M.; Brubaker, M.; and Poupart, P. 2020. Diachronic embedding for temporal knowledge graph completion. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3988–3995.

Guo, Y.; Pan, Z.; and Heflin, J. 2005. LUBM: a benchmark for OWL knowledge base systems. *Journal of Web Semantics* 158–182.

Han, Z.; Chen, P.; Ma, Y.; and Tresp, V. 2021. Explainable subgraph reasoning for forecasting on temporal knowledge graphs. In *International Conference on Learning Representations*.

Hewage, P.; Trovati, M.; Pereira, E.; and Behera, A. 2021. Deep learning-based effective fine-grained weather forecasting model. *Pattern Analysis and Applications* 343–366.

Jain, P.; Rathi, S.; Chakrabarti, S.; et al. 2020. Temporal knowledge base completion: New algorithms and evaluation protocols. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 3733–3747.

Jia, Y.; Lin, M.; Wang, Y.; Li, J.; Chen, K.; Siebert, J.; Zhang, G. Z.; and Liao, Q. 2023. Extrapolation over temporal knowledge graph via hyperbolic embedding. *CAAI Transactions on Intelligence Technology* 418–429.

Jiang, T.; Liu, T.; Ge, T.; Sha, L.; Chang, B.; Li, S.; and Sui, Z. 2016. Towards time-aware knowledge graph completion. In *International Conference on Computational Linguistics*, 1715–1724.

Jin, W.; Qu, M.; Jin, X.; and Ren, X. 2020. Recurrent event network: Autoregressive structure inferenceover temporal knowledge graphs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 6669–6683.

Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-time Systems* 255–299.

Lacroix, T.; Obozinski, G.; and Usunier, N. 2020. Tensor decompositions for temporal knowledge base completion. *arXiv preprint arXiv:2004.04926*.

Leblay, J., and Chekol, M. W. 2018. Deriving validity time in knowledge graph. In *Companion Proceedings of the Web Conference*, 1771–1776.

Li, Z.; Guan, S.; Jin, X.; Peng, W.; Lyu, Y.; Zhu, Y.; Bai, L.; Li, W.; Guo, J.; and Cheng, X. 2022. Complex evolutional pattern learning for temporal knowledge graph reasoning. In

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics, 290–296.

Lin, Q.; Liu, J.; Mao, R.; Xu, F.; and Cambria, E. 2023. TECHS: Temporal logical graph networks for explainable extrapolation reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 1281–1293.

Liu, Y.; Ma, Y.; Hildebrandt, M.; Joblin, M.; and Tresp, V. 2022. TLogic: Temporal logical rules for explainable link forecasting on temporal knowledge graphs. In *Proceedings of AAAI Conference on Artificial Intelligence*, 4120–4127.

Ma, R.; Mei, B.; Ma, Y.; Zhang, H.; Liu, M.; and Zhao, L. 2023. One-shot relational learning for extrapolation reasoning on temporal knowledge graphs. *Data Mining and Knowledge Discovery* 1591–1608.

Meilicke, C.; Chekol, M. W.; Ruffinelli, D.; and Stuckenschmidt, H. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *Proceedings of the International Joint Conferences on Artificial Intelligence*.

Messner, J.; Abboud, R.; and Ceylan, I. I. 2022. Temporal knowledge graph completion using box embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7779–7787.

Muggleton, S., and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming* 629–679.

Niu, G., and Li, B. 2023. Logic and commonsense-guided temporal knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4569–4577.

Omran, P. G.; Wang, K.; and Wang, Z. 2018. Scalable rule learning via learning representation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2149–2155.

Omran, P. G.; Wang, K.; and Wang, Z. 2019. Learning temporal rules from knowledge graph streams. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

Pareja, A.; Domeniconi, G.; Chen, J.; Ma, T.; Suzumura, T.; Kanezashi, H.; Kaler, T.; Schardl, T.; and Leiserson, C. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5363–5370.

Park, N.; Liu, F.; Mehta, P.; Cristofor, D.; Faloutsos, C.; and Dong, Y. 2022. EvoKG: Jointly modeling event time and network structure for reasoning over temporal knowledge graphs. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 794–803.

Pirrò, G. 2020. Relatedness and TBox-driven rule learning in large knowledge bases. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2975–2982.

Sun, H.; Zhong, J.; Ma, Y.; Han, Z.; and He, K. 2021. Timetraveler: Reinforcement learning for temporal knowledge graph forecasting. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 8306–8319.

Thadeshwar, H.; Shah, V.; Jain, M.; Chaudhari, R.; and Badgujar, V. 2020. Artificial intelligence based self-driving car. In *International Conference on Computer, Communication and Signal Processing*, 1–5.

Thennakoon, A.; Bhagyani, C.; Premadasa, S.; Mihiranga, S.; and Kuruwitaarachchi, N. 2019. Real-time credit card fraud detection using machine learning. In *International Conference on Cloud Computing, Data Science & Engineering*, 488–493.

Trivedi, R.; Dai, H.; Wang, Y.; and Song, L. 2017. Know-evolve: Deep temporal reasoning for dynamic knowledge graphs. In *International Conference on Machine Learning*, 3462–3471.

Van Harmelen, F.; Lifschitz, V.; and Porter, B. 2008. *Handbook of knowledge representation*. Elsevier.

Vitelli, M.; Chang, Y.; Ye, Y.; Ferreira, A.; Wołczyk, M.; Osiński, B.; Niendorf, M.; Grimmett, H.; Huang, Q.; Jain, A.; and Ondruska, P. 2022. SafetyNet: Safe planning for real-world self-driving vehicles using machine-learned policies. In *International Conference on Robotics and Automation*, 897–904.

Wałęga, P. A.; Grau, B. C.; Kaminski, M.; and Kostylev, E. V. 2020. DatalogMTL over the integer timeline. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 768–777.

Wang, D.; Hu, P.; Wałęga, P. A.; and Grau, B. C. 2022. MeTeoR: practical reasoning in datalog with metric temporal operators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 5906–5913.

Wang, J.; Wang, B.; Qiu, M.; Pan, S.; Xiong, B.; Liu, H.; Luo, L.; Liu, T.; Hu, Y.; Yin, B.; et al. 2023. A survey on temporal knowledge graph completion: Taxonomy, progress, and prospects. *arXiv preprint arXiv:2308.02457*.

Wałęga, P. A.; Cuenca Grau, B.; Kaminski, M.; and Kostylev, E. V. 2019. DatalogMTL: Computational complexity and expressive power. In *Proceedings of the International Joint Conferences on Artificial Intelligence*, 1886–1892.

Xiong, S.; Yang, Y.; Fekri, F.; and Kerce, J. C. 2024a. TILP: Differentiable learning of temporal logical rules on knowledge graphs. *arXiv preprint arXiv:2402.12309*.

Xiong, S.; Yang, Y.; Payani, A.; Kerce, J. C.; and Fekri, F. 2024b. TEILP: Time prediction over knowledge graphs via logical reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 16112–16119.

Xu, C.; Nayyeri, M.; Alkhoury, F.; Yazdi, H. S.; and Lehmann, J. 2019. Temporal knowledge graph embedding model based on additive time series decomposition. *arXiv preprint arXiv:1911.07893*.

Xu, C.; Nayyeri, M.; Alkhoury, F.; Yazdi, H. S.; and Lehmann, J. 2020. TeRo: A time-aware knowledge graph embedding via temporal rotation. In *Proceedings of the 28th International Conference on Computational Linguistics*, 1583–1593.

Xu, C.; Chen, Y.-Y.; Nayyeri, M.; and Lehmann, J. 2021. Temporal knowledge graph completion using a linear temporal regularizer and multivector embeddings. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2569–2578.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in Neural Information Processing Systems* 2319–2328.

Zeroual, A.; Harrou, F.; Dairi, A.; and Sun, Y. 2020. Deep learning methods for forecasting COVID-19 time-series data: A comparative study. *Chaos, Solitons & Fractals* 110121.

Zhu, C.; Chen, M.; Fan, C.; Cheng, G.; and Zhang, Y. 2021. Learning from history: Modeling temporal knowledge graphs with sequential copy-generation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 4732–4740.