Probabilistic Rule Induction from Event Sequences with Logical Summary Markov Models

Debarun Bhattacharjya, **Oktie Hassanzadeh**, **Ronny Luss** and **Keerthiram Murugesan**
IBM Research
{debarunb, hassanzadeh, rluss}@us.ibm.com, keerthiram.murugesan@ibm.com

Abstract

Event sequences are widely available across application domains and there is a long history of models for representing and analyzing such datasets. Summary Markov models are a recent addition to the literature that help identify the subset of event types that influence event types of interest to a user. In this paper, we introduce logical summary Markov models, which are a family of models for event sequences that enable interpretable predictions through logical rules that relate historical predicates to the probability of observing an event type at any arbitrary position in the sequence. We illustrate their connection to prior parametric summary Markov models as well as probabilistic logic programs, and propose new models from this family along with efficient greedy search algorithms for learning them from data. The proposed models outperform relevant baselines on most datasets in an empirical investigation on a probabilistic prediction task. We also compare the number of influencers that various logical summary Markov models learn on real-world datasets, and conduct a brief exploratory qualitative study to gauge the promise of such symbolic models around guiding large language models for predicting societal events.

1 Introduction & Related Work

Applications concerning sequences of events are ubiquitous in today's world where data is increasingly accessible, e.g., primary care doctors often now enter information from a patient's visit in real-time directly into their computer records rather than through using many file cabinets to hold records maintained by other staff. The prevalence of such event sequences give possibility to new modeling methods that can offer new insights. For example, selecting the best treatment for diabetes patients is often done by trial and error based on physical reactions with no general consensus among a wide array of medicine options [Grant *et al.*, 2007]. Data-driven rules from event sequence data could help guide doctors to better outcomes or even identify reasons for poor outcomes.

In this work, we are primarily concerned with learning from datasets involving sequences of various types of events without meaningful timestamps, i.e. either the time is unavailable or too noisy to be useful. Such datasets arise in many areas, such as medicine, advertising, product recommendations, social networks, or possibly curated from unstructured textual sources through natural language processing. For example, a diabetic patient may undergo a sequence of events involving insulin dosage, exercise, meal ingestion, blood glucose level changes and negative side effects such as rash occurrences. We are interested in learning probabilistic logical rules that: 1) identify a subset of event types that influence user-specified event types of interest, and 2) relate their historical summaries to the occurrence of the event types of interest. For instance, one may learn in this example that only insulin dosage events affect rash occurrences, and that observing a rash is quite likely after a patient's first injection but much less likely after a subsequent injection.

There is a wealth of literature on event sequence analysis, particularly on data mining approaches such as frequent episode mining, sequence mining and association rule mining [Mannila *et al.*, 1997; Weiss and Hirsh, 1998; Fournier-Viger *et al.*, 2011; Rudin *et al.*, 2012; Letham *et al.*, 2013]. Much of this research has centered around the challenge of efficiently identifying application-specific patterns from large input sequences, which are then used to derive rules over the identified patterns. There is also a long history of work on Markov models for sequences [Raftery, 1985; Rabiner, 1989; Begleiter *et al.*, 2004; Bhattacharjya *et al.*, 2022] as well as more application-driven data mining related work that is concerned with learning rules from sequences, including specification mining in software engineering [Lemieux *et al.*, 2015] and declarative process mining in information systems [Di Ciccio *et al.*, 2018]. Temporal rule induction is also of interest in communities where datasets involve events that have time-stamps, both for interval-based events [Tran and Davis, 2008; Brendel *et al.*, 2011] and point events modeled by temporal point processes [Li *et al.*, 2021].

A key concern with event models in general is around explainability; rather than simply offering a distribution over events that may occur next, it may be important in many applications to justify or explain the prediction. As noted in [Miller, 2019], one is often interested in a small subset of the possible explanations for an event. Recent work explores this idea in the context of event sequences in the form of summary Markov models [Bhattacharjya *et al.*, 2022], which seek to find *influencing sets*, defined as a subset of event types that explain predictions for event types of interest. Here we introduce **logical summary Markov models**, which provide probabilistic logical rules that depend on predicates involving the influencing set. We propose new predicates for summarizing history and more expressive models which bring along the additional benefits of a logical representation while retaining the key feature of influencing sets. While it is clearly not always the case that simple explanations offer the best predictions, we propose greedy algorithms that are designed to search for explanations that grow in complexity if required.

Our work is closely related to the area of probabilistic inductive logic programming (PILP) where probabilistic logical rules are learned from data [Raedt *et al.*, 2008; Riguzzi, 2018]. While probabilistic logic programs (PLPs) were defined long ago, e.g. [Ng and Subrahmanian, 1992; Poole, 1993], unlike inductive logic programming (ILP), probabilistic versions of ILP have received less attention until recently, as noted in [Riguzzi *et al.*, 2014] and [Cropper *et al.*, 2020]. From the lens of PILP, our learning approach can be viewed as incorporating both *structure learning* – identifying the influencing set and the associated rules of the logic program, as well as *parameter learning* – estimating an occurrence distribution over event types of interest in the head of the logical rules. This is inspired more broadly by learning algorithms for probabilistic graphical models [Pearl, 1988; Chickering, 2002]. We show connections to logic programs with annotated disjunctions (LPADs) [Vennekens *et al.*, 2004] which also utilize probabilistic rules, albeit with more restrictive structure. Importantly, as our work learns rules from event sequences, we consider sequence-specific predicates and a broader class of logical operations than typically considered in PILP frameworks. Thus our approach can be viewed as one that adopts greedy search for probabilistic rule learning, e.g. [De Raedt *et al.*, 2015], in a setting involving complex dynamics, e.g. [Thon *et al.*, 2011]. Note that logical rules learned via ILP/PILP systems are of great interest to reasoning tasks [Yang and Song, 2020; Jedwabny *et al.*, 2021] where one can leverage (probabilistic) rules in accordance with a knowledge base to answer queries.

Contributions. In this paper:

- We introduce a family of interpretable models for event sequences where history is summarized by temporal logical rules – logical summary Markov models – and illustrate their connection to prior parametric summary Markov models as well as probabilistic logic programs.
- We propose new predicates for practical problems, resulting in novel logical summary Markov models, i.e. with new forms of temporal logical rules. These include a model that leverages historical counts and one that allows different historical orders to share parameters.
- We describe efficient greedy search algorithms for learning that make tackling the inherent combinatorial optimization problem practically feasible.
- We empirically demonstrate the benefits of the new proposed models over related prior work on a probabilistic prediction task involving real-world datasets, and compare influencing set sizes on these datasets.
- We conduct an initial investigation around guiding the output of large language models for predicting societal events using logical summary Markov models.

2 Background

2.1 Basic Notation

An **event sequence dataset** is a multi-set of sequences of different types of events, $\mathbf{D} = \{\mathbf{D_k}\}_{k=1}^{K}$, where $\mathbf{D_k} = [l_i]_{i=1}^{N_k}$ and event label (or type) l_i at index i in the sequence is from a known label set $\mathscr{L}$ with cardinality M. There are K sequences of events in the dataset with a total of $N = \sum_{k=1}^{K} N_k$ instances of events.

For modeling sequence dynamics, we consider the potential effect of prior events as determined by the **history** at position i in an event sequence, $h_i = \{(j, l_j)\}_{j=1}^{i-1}$. The **restricted history** with respect to some label set $\mathbf{Z} \subset \mathscr{L}$ at position i only includes prior occurrences of labels from $\mathbf{Z}$, i.e. $h_i^{\mathbf{Z}} = \{(j, l_j) : j < i, l_j \in \mathbf{Z}\}$. Throughout the paper, we will remove subscript i when referring to a generic position.

In the next section, we describe several predicates pertaining to any (restricted) history. Since a history may be arbitrarily long and complex, some simplifications may be appropriate in practice. For some of the predicates, we use a **look-back** period parameter κ to restrict the history h_i to labels in up to κ positions before sequence position i. Also, whenever we are concerned with the historical order of a set of event labels, we use a **masking function** $\phi(\cdot)$ which takes an event sequence $s = \{(j, l_j)\}$ as input and returns a subsequence where no label is repeated, $s' = \{(k, l_k) \in s : l_k \neq l_m \text{ for } k \neq m\}$ [Bhattacharjya *et al.*, 2020]. In the spirit of Markov models, in this paper we consider a $\phi(\cdot)$ which only retains the most recent (i.e. last) occurrence of an event label, but we note that other masking approaches may also be applicable in practice.

Example 1. Figure 1 presents an illustrative event sequence dataset over 5 event labels for a diabetic patient. The history at position 8 is h_8 ={(1, RD), (2, LM), (3, RD), (4, RE), (5, LM), (6, HM), (7, RD)}. The history at this position restricted to label set Z ={RD, LM} is h_8^Z={(1, RD), (2, LM), (3, RD), (5, LM), (7, RD)} (HM and RE are excluded). If we restrict the history even further at this position to within a look-back period $\kappa = 5$, then the relevant restricted history is $h_8^Z(\kappa = 5)$ ={(3, RD), (5, LM), (7, RD)}. Applying masking function $\phi(\cdot)$ where only the most recent occurrence of each event type is maintained to h_8^Z (without the look-back restriction) results in the following sub-sequence: {(5, LM), (7, RD)} yielding the order [LM, RD]. The same order is obtained on applying this masking function to $h_8^Z(\kappa = 5)$. □

2.2 Summary Markov Models

We review some basic terminology pertaining to summary Markov models [Bhattacharjya *et al.*, 2022], which capture event sequence dynamics for a user-specified subset of the event labels $\mathbf{X} \subseteq \mathscr{L}$. As an example, although an IT system's log may register many events in a sequence, the user may be particularly interested in failure events. A random variable denoted $\mathscr{X}$ is used for modeling the occurrence of label $X \in$

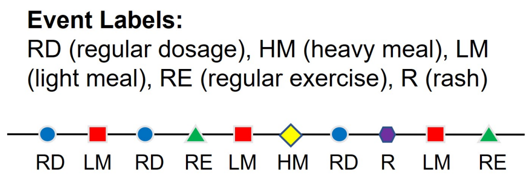

Event Labels:
RD (regular dosage), HM (heavy meal), LM (light meal), RE (regular exercise), R (rash)
RD LM RD RE LM HM RD R LM RE

Figure 1: Illustrative event sequence dataset for a single diabetic patient ($K = 1$) with $N = 10$ events over $M = 5$ labels.

$\mathbf{X}$ at any arbitrary position in the sequence. $\mathscr{X}$ has a state for each label in $\mathbf{X}$ and a single state for when the label belongs to $\mathscr{L} \setminus \mathbf{X}$, if the set $\mathscr{L} \setminus \mathbf{X}$ is not empty. The states of $\mathscr{X}$ are denoted x.

The global dynamics of the sequence is assumed to be governed by a conditionally homogeneous sequential process over labels in $\mathscr{L}$, parameterized using probabilities $\Theta = \{\Theta_X\}$, $\Theta_X = \{\theta_{x|h}\}$ s.t. $\sum_{X \in \mathscr{L}} \theta_{x|h} = 1$ for all possible histories h, where $\theta_{x|h}$ is the probability of event label X occurring at any position in the sequence given history h. $\tilde{\Theta}_{\mathbf{X}}$ is the sum of probabilities over the label set of interest $\mathbf{X} \subseteq \mathscr{L}$, i.e. $\tilde{\Theta}_{\mathbf{X}} = \{\tilde{\theta}_{x|h}\}$ where $\tilde{\theta}_{x|h} = \sum_{X \in \mathbf{X}} \theta_{x|h}$.

As a practical matter, particularly from the perspective of learning with limited data, it is not possible for a sequence model to have unique parameters for all possible histories; thus a mechanism for summarizing history is necessary.

Definition 1. *A **summary function** $s(\cdot)$ for some label set $\mathbf{Z}$ maps any restricted history $h^{\mathbf{Z}}$ at any position to some **summary state** $s_{\mathbf{Z}}$ in a range $\Sigma_{\mathbf{Z}}$. h is said to be **consistent** with state $s_{\mathbf{Z}}$ if the summary function applied to h restricted to $\mathbf{Z}$ results in $s_{\mathbf{Z}}$, i.e. $s(h^{\mathbf{Z}}) = s_{\mathbf{Z}}$.*

A summary function summarizes history into a smaller number of states. The key idea behind summary Markov models is that a summary of only a subset of all event labels $\mathscr{L}$ is sufficient to predict random variable $\mathscr{X}$. Formally, a label set $\mathbf{U}$ is an **influencing set** for event labels of interest $\mathbf{X}$ under summary function $s(\cdot)$ if for all $s_{\mathbf{U}} \in \Sigma_{\mathbf{U}}$, $\tilde{\theta}_{x|h} = \tilde{\theta}_{x|h'}$ for all h, h' consistent with $s_{\mathbf{U}}$. $\mathbf{U}$ is minimal if the condition cannot be satisfied after removing any label in $\mathbf{U}$.

Definition 2. *A **summary Markov model** (SuMM) for event label set $\mathbf{X} \subseteq \mathscr{L}$ (and corresponding random variable $\mathscr{X}$) includes a summary function $s(\cdot)$, a set of influencing labels $\mathbf{U}$ and probability parameters $\theta_{x|s_{\mathbf{U}}}$ for each state of $\mathscr{X}$ and each summary state $s_{\mathbf{U}} \in \Sigma_{\mathbf{U}}$.*

Example 2. Suppose $\mathbf{X} = \{\text{R}\}$ (rash) and $\mathbf{U} = \{\text{RD}\}$ (regular dosage) from the event sequence in Figure 1. For a binary summary function $s(\cdot)$ that determines whether regular dosage has been taken at least once within a look-back of $\kappa = 5$ positions, there are two summary states, denoted rd (True) and $\overline{rd}$ (False). The corresponding SuMM would have two free parameters: $\theta_{x|rd}$ and $\theta_{x|\overline{rd}}$. In this example sequence, note that the summary state at position 8 is True because $h_8^{RD}(\kappa = 5)$ ={(3, RD), (7, RD)}. □

3 Logical Summary Markov Models

We introduce a family of models for event sequences where the dynamics are governed by probabilistic logical rules involving **historical predicates** that summarize the history restricted to influencing sets. The body of a logical clause/rule includes predicate terms that are related via logical operators $(\wedge, \vee, \neg)$ whereas the head includes predicates for whether an event label will occur at a position in the sequence. In this work, all rules are grounded by specifying the position index i for the prediction.

3.1 Predicates

A *positional occurrence* predicate $p_P(X, i)$ specifies whether an event of label X occurs at position i in an event sequence. We model how such a predicate (used in the head of our logical rules) entails from those that summarize history, i.e. events before i. Examples of historical predicates follow:

- *Look-back occurrence* $p_L(i, X, \kappa)$: Whether an event of label X occurs at least once in up to κ positions before position i in an event sequence.
- *Order* $p_O(i, [X_1, \ldots, X_I], \kappa, \phi)$: Whether distinct event labels X_1 through X_I appear in that order within up to κ positions before position i in an event sequence after applying masking function ϕ to the history.
- *Order position* $p_{OP}(i, X, \mathbf{Z}, j, \kappa, \phi)$: Whether an event of label X appears in the j^{th} position in the order resulting from applying masking function ϕ over labels $\mathbf{Z}$ to up to κ positions before position i.
- *Unique* $p_U(i, \mathbf{Z}, c, r \in \{=, <, >\}, \kappa)$: Whether the number of unique occurrences of labels in $\mathbf{Z}$ within up to κ positions before position i in an event sequence satisfy (in)equality relation r w.r.t constant c.
- *Count* $p_C(i, X, c, r \in \{=, <, >, \leq, \geq\}, \kappa)$: Whether the number of occurrences of label X within up to κ positions before position i in an event sequence satisfy (in)equality relation r w.r.t constant c.

To simplify the notation, we omit position index i and occasionally also parameters such as the look-back κ whenever they are clear from context.

We note that the temporal information in these predicates is implicit but the rules could also be expressed using explicit temporal relations. For instance, the order predicate $p_O(i, [X_1, X_2])$ is equivalent to $\exists j, k$ s.t. $p_P(j, X_1) \wedge p_P(k, X_2) \wedge \text{BEFORE}(j, k) \wedge \text{BEFORE}(k, i)$, where BEFORE is a temporal relation between two positions. Also, some predicates are related to others, e.g., the count predicate p_C is a generalization of look-back predicate p_L. While the aforementioned predicates are by no means exhaustive, they are sufficiently comprehensive for many applications as demonstrated in our experiments.

Example 3. Consider the rash occurrence event (R) at position 8 in the sequence in Figure 1. The reader can confirm that the following predicates are true: $p_L(i = 8, \text{RD}, \kappa = 5)$, $p_O(i = 8, [\text{LM}, \text{RD}], \kappa = 5, \phi)$, $p_C(i = 8, \text{RD}, c = 2, '=', \kappa = 5)$ and $p_U(i = 8, \{\text{RD}, \text{LM}\}, c = 2, '=', \kappa = 5)$. □

3.2 LSuMM Family

A probabilistic logic program is a finite set of logical rules comprising atoms, conjunctions and disjunctions that are assigned probabilities [Ng and Subrahmanian, 1992]. We define logical summary Markov models as probabilistic logic programs specifically intended to model event sequences.

Definition 3. *A **logical summary Markov model** (LSuMM) for event label set $\mathbf{X} \subseteq \mathscr{L}$ (and corresponding random variable $\mathcal{X}$) with $|\mathbf{X}| = J$ is a probabilistic logic program with a set of rules, each of the form:*

$$(p_P(X_1) : \theta_1) \vee \cdots \vee (p_P(X_J) : \theta_J) \leftarrow f(p_1, p_2, \cdots, p_n),$$

where the body is a logical formula $f(\cdot)$ (involving conjunctions, disjunctions and negations) of predicates p_i that are either historical predicates or temporal relations that only involve labels from an influencing set $\mathbf{U}$. The head is a disjunction over positional occurrence predicates $p_P(X_j)$ indicating whether event label $X_j \in \mathbf{X}$ occurs at any arbitrary position in an event sequence; these disjunctions are annotated by the probabilities of their occurrence θ_j. The probabilities θ_j in each rule head sum to 1 if $\mathbf{X} = \mathscr{L}$ or sum to ≤ 1 if $\mathbf{X} \subset \mathscr{L}$.

Example 4. Consider the following illustrative LSuMM for predicting social unrest (SU) or peaceful protest (PP) using look-back occurrence predicates p_L on resource shortage (RS) and police violence (PV):

$$(p_P(\text{SU}) : 0.01) \vee (p_P(\text{PP}) : 0.02) \leftarrow \neg p_L(\text{RS}, 3) \wedge \neg p_L(\text{PV}, 5)$$
$$(p_P(\text{SU}) : 0.05) \vee (p_P(\text{PP}) : 0.05) \leftarrow p_L(\text{RS}, 3) \wedge \neg p_L(\text{PV}, 5)$$
$$(p_P(\text{SU}) : 0.07) \vee (p_P(\text{PP}) : 0.03) \leftarrow \neg p_L(\text{RS}, 3) \wedge p_L(\text{PV}, 5)$$
$$(p_P(\text{SU}) : 0.2) \vee (p_P(\text{PP}) : 0.1) \leftarrow p_L(\text{RS}, 3) \wedge p_L(\text{PV}, 5)$$

This program uses predicates with different look-backs (3 and 5) and represents a situation where both RS and PV make SU and PP more likely, particularly their combination. □

The nature of the probabilistic logical rules in an LSuMM are a slight generalization of logic programs with annotated disjunctions (LPADs), which are thus named because the head of each rule is a disjunction that is annotated with probabilities [Vennekens *et al.*, 2004]. In an LSuMM, the rule head is a disjunction over event labels annotated with the probabilities of their occurrence. Note that the body of rules in an LSuMM can involve any logical operator, unlike in an LPAD where they are restricted to Horn clauses, i.e. only involve a conjunction over predicates. The LSuMM in Example 4 happens to be an LPAD.

LSuMMs are motivated by SuMMs in that they represent event sequence dynamics via a historical summary, and the body of each rule is in fact a summary state; we therefore refer to the formulae $f(\cdot)$ as **summary formulae**. Importantly, LSuMMs similarly achieve the objective of identifying the influencing set for event types of interest, since these can be inferred from the chosen predicates in the induced logical rules. This can significantly aid model interpretability and bring practical value as a knowledge acquisition tool.

Prior work on SuMMs has proposed two specific parametric models: binary SuMM (BSuMM) and ordinal SuMM (OSuMM) [Bhattacharjya *et al.*, 2022]. In the former, probability parameters depend on whether or not influencers have been observed within label-specific look-back periods. In the latter, they depend instead on the order in which influencers are observed within a single look-back period. Both models can be defined as LSuMMs.

Definition 4. *A BSuMM for event label $\mathbf{X} \subseteq \mathscr{L}$ and influencing set $\mathbf{U}$ with look-backs $\{\kappa_X : \forall X \in \mathbf{U}\}$ is a logical SuMM with summary formulae of the form $\wedge_{X \in \mathbf{U}} (t_X)$, where term t_X for each event label $X \in \mathbf{U}$ is either $p_L(X, \kappa_X)$ or $\neg p_L(X, \kappa_X)$. There are $2^{|\mathbf{U}|}$ rules in the program, one for each combination of truth values for the predicates, i.e., including either p_L or $\neg p_L$ for each $X \in \mathbf{U}$.*

Example 4 illustrated a BSuMM where resource shortage and police violence were influencers $\mathbf{U}$.

Definition 5. *An OSuMM for event label $\mathbf{X} \subseteq \mathscr{L}$ and influencing set $\mathbf{U}$ with look-back κ and masking function $\phi(\cdot)$ is a logical SuMM with summary formulae of the form $p_O([X_1, \cdots, X_I], \kappa, \phi(\cdot))$, where $X_1, \ldots, X_I$ is a permutation of a subset of $\mathbf{U}$. There are $\sum_{l=0}^{|\mathbf{U}|} \frac{|\mathbf{U}|!}{l!}$ rules in the program, one for each permutation of each subset of $\mathbf{U}$.*

Since an LSuMM captures sequence dynamics, one must formalize additional restrictions on the probabilistic logic program to ensure some sense of model completeness. We do so by ensuring that any possible history can be summarized by the program.

Definition 6. *An LSuMM for event label set $\mathbf{X} \subseteq \mathscr{L}$ (and corresponding random variable $\mathcal{X}$) is **fully determined** if, for any history h, there is exactly one rule in the program where the summary formula in the body is true. This rule governs label occurrence distribution at the next sequence position.*

Remark 7. *BSuMM, OSuMM are fully determined LSuMMs.*

In the next section, we describe a Bayesian approach to handle the case where a program may not be fully determined, such as through missing rules during learning.

A clear advantage that an LSuMM provides is that the induced probabilistic logic program could potentially be leveraged by downstream logical reasoners. The logical interpretation also reveals that prior work on SuMMs uses specific predicates as well as what are essentially *templates* for summary formulae involving those predicates, opening up possibilities for novel predicates and more expressive formulae induced by new methods. We propose two such models next.

3.3 Count SuMM

A count SuMM (CSuMM) leverages count predicates, thereby generalizing BSuMM.

Definition 8. *A CSuMM for event label $\mathbf{X} \subseteq \mathscr{L}$ and influencing set $\mathbf{U}$ with look-backs $\{\kappa_X : \forall X \in \mathbf{U}\}$ and maximum count parameters $\{c_X^* : \forall X \in \mathbf{U}\}$ is a logical SuMM with summary formulae of the form $\wedge_{X \in \mathbf{U}} (p_C(X, c, r, \kappa_X)$ where c is an integer between 0 and c_X^*. The relation r is chosen to be '$=$' for $c \in \{0, \cdots, c_X^* - 1\}$ and '$\geq$' when $c = c_X^*$. There are $\prod_{X \in \mathbf{U}} (c_X^* + 1)$ rules in the program, one for each combination of counts in the model's scope.*

In our version of CSuMM, a rule specifies whether the number of occurrences of an influencing event type X equals a count up to $c_X^* - 1$ or whether it equals or exceeds c_X^*. This restriction forces the LSuMM to be fully determined. Choosing $c_X^* = 1$ for all X results in BSuMM. It is possible to generalize CSuMM further by allowing other relations on counts but we do not pursue such extensions in this work.

3.4 Shared Order SuMM

A major limitation of OSuMM is that there are a super-exponential number of rules and therefore parameters in the LSuMM. This makes it challenging to learn generalizable models, particularly when data is limited. To address this issue, we propose using new predicates and summary formulae in rules that effectively behave like a disjunction of order predicates. The following example and subsequent definitions show how the format for the new formulae enable parameter sharing across orders, in a manner that is conceptually similar to recent work [Bhattacharjya *et al.*, 2021].

Example 5. Consider the following illustrative LSuMM for predicting failure event C based on influencers $\mathbf{U} = \{A, B\}$:

$$p_P(\mathrm{C}) : 0.01 \leftarrow p_U(\mathbf{U}, c = 0, '=', 5)$$

$$p_P(\mathrm{C}) : 0.02 \leftarrow p_U(\mathbf{U}, c = 1, '=', 5) \wedge (p_O([A], 5) \vee p_O([B], 5))$$

$$p_P(\mathrm{C}) : 0.5 \leftarrow p_U(\mathbf{U}, c = 2, '=', 5) \wedge p_O([A, B], 5)$$

$$p_P(\mathrm{C}) : 0.05 \leftarrow p_U(\mathbf{U}, c = 2, '=', 5) \wedge p_O([B, A], 5)$$

In the situation modeled by this program, failure event type C is most likely to occur when both A and B occur, in that order, within $\kappa = 5$ positions. The unique predicate $p_U(\cdot)$ in the rules controls the number of distinct occurrences of A and B. For example, the second rule conveys that when exactly one of either A or B occurs, C's probability is 2%. Thus the orders $[A]$ and $[B]$ share the same probability parameter. The summary formula in this rule is equivalent to $p_U(\mathbf{U}, c = 1, '=', 5) \wedge True$ since only either A or B can occur. □

Definition 9. *A* ***shared order summary formula*** *corresponding to orders of length l is of the form $P_U(\mathbf{U}, l, '=', \kappa) \wedge_i (t_i)$ where t_i is either True or an order position predicate $p_{OP}(X, \mathbf{U}, j, \kappa, \phi)$ or its negation for some integer j between 1 and l for some event $X \in \mathbf{U}$. It is* ***feasible*** *if compatible with at least one permutation of a subset of* $\mathbf{U}$.

Note that we use a conjunction of order position predicates to capture order sharing by specifying restrictions on positions in the order, instead of disjunctions of order predicates like in Example 5. This is done to be consistent with the learning algorithm as described in the next section.

Definition 10. *A SOSuMM for event label $\mathbf{X} \subseteq \mathscr{L}$ and influencing set $\mathbf{U}$ with look-back κ and masking function $\phi(\cdot)$ is a logical SuMM with feasible shared order summary formulae of lengths from $l = 0$ to $|\mathbf{U}|$. The number of rules depend on the extent of parameter sharing but the summary formulae should be mutually exclusive and collectively exhaustive with respect to possible historical orders.*

4 Learning Algorithms

In this section, we briefly describe greedy search probabilistic rule induction algorithms for the proposed LSuMMs. For simplifying the exposition, we assume that the influencing set $\mathbf{U}$ for labels of interest $\mathbf{X} \subseteq \mathscr{L}$ are known. It is straightforward to embed the proposed algorithms as an inner loop within an outer loop for finding $\mathbf{U}$; an example involving forward-backward search is detailed in Appendix A. First we show how to learn any LSuMM given the rules, and then describe rule induction for CSuMM and SOSuMM.

Algorithm 1 CSuMM Learner (given influencers)

Input: labels of interest $\mathbf{X}$, influencers $\mathbf{U}$, data $\mathbf{D}$
Hyper-parameters: look-back κ, max. count c^*

```
 1: Initialize $c_X = 1\ \forall X \in \mathbf{U}$, explored set $\mathscr{E} = \emptyset$.
 2: Learn BSuMM as initial opt. model $\mathscr{M}^*$.
 3: while opt. score does not increase and $\mathscr{E} \subset \mathbf{U}$ do
 4:     for all labels not in $\mathscr{E}$ do
 5:         Set $c_X = c^*$ for that label (with $c_X$ fixed from $\mathscr{M}^*$
            for all other labels).
 6:         Learn corresponding CSuMM.
 7:     end for
 8:     Set model from this round with max. score as $\mathscr{M}$.
 9:     if $\mathscr{M}$'s score $>$ $\mathscr{M}^*$'s score then
10:         $\mathscr{M}^* = \mathscr{M}$, add corr. label to $\mathscr{E}$, opt. score increases.
11:     end if
12: end while
13: return CSuMM $\mathscr{M}^*$
```

4.1 Learning an LSuMM with Known Rules

Recall that a summary formula involving labels $\mathbf{U}$ summarizes the history and is thus a summary state, which we denote $\mathbf{s}$. We can therefore follow prior work for parameter learning of LSuMMs given rules [Bhattacharjya *et al.*, 2022]. The log likelihood of an LSuMM with known rules on data $\mathbf{D}$ can be written by modeling how summary states affect the random variable $\mathscr{X}$ corresponding to label set $\mathbf{X}$:

$$LL_{\mathscr{X}} = \sum_{x} \sum_{\mathbf{s}} \big(N(x; \mathbf{s}) log(\theta_{x|\mathbf{s}})\big), \tag{1}$$

where $N(x, \mathbf{s})$ counts the number of times in $\mathbf{D}$ where $\mathscr{X}$ is observed to be in state x and summary formula $\mathbf{s}$ is true, based on look-back(s) κ and optionally masking function $\phi(\cdot)$ (which are hidden as a notational simplification).

We take a Bayesian approach by estimating probabilities $\hat{\theta}_{x|\mathbf{s}}$ with a Dirichelet prior following estimation in SuMMs. In case there are missing rules, possibly due to a lack of sufficient data, the Bayesian approach ensures a fully determined LSuMM through default probabilities. We use the Bayesian information criterion (BIC) as a score to prevent over-fitting by penalizing complex models as follows:

$$Score_{\mathscr{X}} = LL^*_{\mathscr{X}} - \gamma |P| \frac{\log{(N)}}{2}, \tag{2}$$

where γ is a penalty complexity parameter in $(0, 1]$, $|P|$ is the number of free model parameters, N is the total number of events in $\mathbf{D}$ and $LL^*_{\mathscr{X}}$ is the log likelihood from equation (1) computed at the probability estimates $\hat{\theta}_{x|\mathbf{s}}$.

4.2 Learning CSuMM

The rule induction pseudo-code for CSuMM is provided in Algorithm 1. The search initializes the program as a BSuMM, and then greedily explores event labels X in the influencing set $\mathbf{U}$ to check if the score improves when count predicates with $c_X = 1$ in the program are modified to $c_X = c^*$, which expands the number of rules and therefore also parameters. c^* is a maximum count hyper-parameter for this LSuMM.

Remark 11. *CSuMM $\mathscr{M}^*$ from Algo. 1 is fully determined.*

Algorithm 2 SOSuMM Learner (given influencers)

Input: labels of interest $\mathbf{X}$, influencers $\mathbf{U}$, data $\mathbf{D}$
Hyper-parameters: look-back κ, masking function $\phi(\cdot)$

```
1:  for all integers $l$ from 0 to $|\mathbf{U}|$ do
2:      Initialize this $l$ model with summary formula $F_0 = P_U(\mathbf{U}, l, '=', \kappa)$, unvisited formula set $\mathscr{F} = \{F_0\}$.
3:      Learn SOSuMM with $F_0$ as initial opt. model $\mathscr{M}_l^*$.
4:      while $\mathscr{F}$ is not empty do
5:          Choose any formula $F_c$ from $\mathscr{F}$.
6:          Find all feasible split pairs $(F', F'')$ from $F_c$.
7:          for all feasible split pairs $(F', F'')$ do
8:              Replace formula $F_c$ with $F'$ and $F''$ in program.
9:              Learn corresponding SOSuMM.
10:         end for
11:         Set model from this round with max. score as $\mathscr{M}_l$.
12:         if $\mathscr{M}_l$'s score $>$ $\mathscr{M}_l^*$'s score then
13:             $\mathscr{M}_l^* = \mathscr{M}_l$, add $F'$ and $F''$ to $\mathscr{F}$.
14:         end if
15:         Remove $F_c$ from $\mathscr{F}$.
16:     end while
17: end for
18: return SOSuMM $\mathscr{M}^* = \{\mathscr{M}_l^* : \forall l\}$.
```

4.3 Learning SOSuMM

Algorithm 2 outlines the somewhat more complicated rule induction pseudo-code for SOSuMM, which learns rules for historical orders of all possible lengths l from 0 to $|\mathbf{U}|$. Here the model is initialized through a program with summary formulae involving only unique predicates, i.e. only specifying the length of the order l. Thus the search begins with a program containing $|\mathbf{U}| + 1$ rules. The algorithm greedily searches for potential ways to increase the score by splitting a formula into two formulae. All possible feasible splits for a formula are obtained by including order position predicates through conjunction to apply restrictions to all possible positions in the order. The following example is illustrative.

Example 6. The summary formula in the second rule in Example 5 is $p_U(\mathbf{U}, c = 1, '=', 5) \wedge True$, which specifies that exactly one label occurs in the (masked) order. There is only one feasible split here – either A occurs or B occurs, resulting in the formula pair $p_U(\mathbf{U}, c = 1, '=', 5) \wedge p_O([A], 5)$ and $p_U(\mathbf{U}, c = 1, '=', 5) \wedge p_O([B], 5)$. □

The greedy algorithm will choose to split a formula and increase the number of formulae and therefore parameters if the gain in model fit as deemed by the log likelihood is worth the additional model complexity. Thus the data controls the number of shared order formulae in the induced SOSuMM.

Remark 12. *SOSuMM $\mathscr{M}^*$ from Algo. 2 is fully determined.*

5 Experiments

An empirical investigation involving different tasks is conducted to study the benefits of the newly proposed LSuMMs.

5.1 Prediction

Task. We wish to predict whether a single event label of interest will occur (or not) next in the sequence, given the history. Models are learned using a train set (70%) as well as a dev set (15%) for hyper-parameter tuning, and then evaluated on a test set (15%). Since all models are probabilistic, we use *negative log loss* as the evaluation metric, averaged over labels of interest; this is a popular metric for probabilistic prediction and is equivalent to the logarithmic scoring rule as well as log likelihood in our case [Bishop, 2006].

Datasets & Baselines. We use the same real-world datasets and baselines as recent work on SuMMs [Bhattacharjya *et al.*, 2022]. The first 3 datasets are time-stamped event datasets where time-stamps are omitted, whereas the last 2 datasets are sequences curated from unstructured textual corpora:

- **Diabetes** [Frank and Asuncion, 2010]: Blood glucose level changes, dosage, and other events for diabetic patients.
- **LinkedIn** [Xu *et al.*, 2017]: Events that register jobs or role changes for 1000 LinkedIn users at 10 tech companies.
- **Stack Overflow** [Grant and Betts, 2013]: Events around receipt of badges in a question answering website by 1000 users chosen from [Du *et al.*, 2016].
- **Beige Books**: Event topics in documents published by the Federal Reserve Board on economic conditions in U.S.A.
- **Timelines**: Event-related concepts in Wikidata [Vrandecic and Krötzsch, 2014] extracted from timeline sections of societal event Wikipedia articles [1].

For baselines, we compare with other interpretable probabilistic sequence models: k^{th} order Markov chains (MC) for varying k, logistic regression (LR) with a varying look-back of k positions, and prior SuMMs [Bhattacharjya *et al.*, 2022]. We also consider a simple LSTM [Hochreiter and Schmidhuber, 1997] as a representative neural model with less interpretability. Further details about datasets and experiments (including hyper-parameters) are provided in Appendix B.

Results. Table 1 displays results for this task. The proposed models are competitive, performing best among the interpretable models on 2 datasets each; in particular, CSuMM and SOSuMM improve upon BSuMM and OSuMM respectively. Logistic regression (LR) performs much better than any other model on LinkedIn; we conjecture that since the time-stamped version of the dataset is known to be suitably modeled as a Hawkes process, which can sometimes be reduced to weighted LR [Menon and Lee, 2018], LR is able to adequately fit feature weights based on the recency of events. While LSTM performs best on Timelines, which has more event labels than other dataset, CSuMM is a close second. The results generally show that the proposed LSuMMs improve upon the state-of-the-art in interpretable event sequence prediction models. They retain the benefit of identifying influencers like SuMMs but could potentially also be deployed by reasoners since they induce probabilistic logical rules.

5.2 Influencing Set Identification

Task, Datasets & Baselines. We compare the sizes of the influencing sets that are learned for all 4 LSuMMs. All models are trained on the train set using the same hyper-parameters: Dirichlet prior $\alpha = 0.1$, look-back $\kappa = 5$ and penalty complexity $\gamma = 1$. For CSuMM, we choose $c^* = 3$;

[1]https://doi.org/10.5281/zenodo.7964471

Dataset	1-MC	2-MC	3-MC	3-LR	5-LR	BSuMM	OSuMM	CSuMM	SOSuMM	LSTM
Beige Books	-60.91	-40.37	-37.66	-36.85	-36.15	*-36.11*	-38.07	*-36.11*	**-34.01**	-63.65
Diabetes	-513.01	-488.42	-473.96	-506.05	-497.92	-497.90	*-432.89*	-492.48	**-429.57**	-595.57
LinkedIn	-110.58	-112.55	-119.55	**-92.23**	*-93.37*	-114.52	-115.63	-114.46	-112.07	-135.92
Stack Overflow	-1278.96	-1283.66	-1435.12	-1277.84	-1263.54	*-1242.59*	-1246.64	**-1239.74**	-1243.07	-1246.45
Timelines	-154.47	-611.78	-1343.52	-160.84	-184.05	-141.42	-142.18	**-136.38**	*-140.13*	**-135.98**

Table 1: Avg. neg. log loss over labels of interest computed on test sets for 5 datasets. k^{th} order Markov chains (MC) lie in $k = \{1, 2, 3\}$, and logistic regression (LR) is shown for look-back $k = \{3, 5\}$. BSuMM/OSuMM are baselines that are also LSuMMs. **Bold** and *italics* are used for the best and second best performance respectively, among the interpretable models. LSTM acts as a representative neural baseline.

Dataset	BSuMM	OSuMM	CSuMM	SOSuMM
Beige Books	3.53	2.73	3.53	5.60
Diabetes	2.31	2.23	2.08	4.46
LinkedIn	1.80	1.50	1.80	3.30
Stack Overflow	2.40	2.00	2.30	5.00
Timelines	1.67	1.47	1.47	2.93

Table 2: Avg. influencing set size over labels of interest for all LSuMMs on the train sets for 5 datasets.

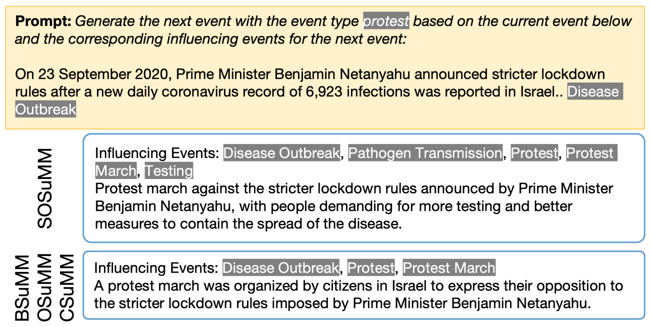
this was seen to be a reasonable empirical choice. We use the same 5 real-world datasets as in the previous experiment.

Results. Table 2 shows that SOSuMM learns around double the number of influencers on average, compared to OSuMM. OSuMM and CSuMM have more parameters than BSuMM for the same influencing set size, therefore typically learn smaller models. While there is no guarantee that SOSuMM identifies the correct influencers, the robust predictive performance from the previous experiment and consistency of the learning approach provides at least some evidence that the induced influencers are not superfluous. Thus SoSuMM could be a useful knowledge acquisition tool in practice.

5.3 Guided Text Generation for Timelines

Task & Setup. We conduct a qualitative investigation to explore the effect of using influencing sets from LSuMMs as "context" in Large Language Models (LLMs). Our goal is to use an LSuMM to propose a potential next event label X (chosen from event-related Wikidata concepts, similar to the Timelines dataset) and then to guide an LLM to generate its textual description by providing additional context using the LSuMM's influencing set $\mathbf{U}$ for X. This is inspired by recent work on context controlled LLMs [Petroni *et al.*, 2020; Guu *et al.*, 2020; Wei *et al.*, 2022]. In our current setup, we choose examples where the current event belongs to $\mathbf{U}$ and explore how other influencers might affect the generated text.

Results. We consider a sample from the raw text of the Timelines dataset around the *COVID-19 Pandemic in Israel* and use GPT3 [Brown *et al.*, 2020] to generate a piece of text about the event type X = *protest*. The influencing set generated by BSuMM, OSuMM and CSuMM (for $c^* = 3$) for event type *protest* is $\mathbf{U} = \{$*disease outbreak, protest, protest march*$\}$. SOSuMM also includes $\{$*pathogen transmission, testing*$\}$ as influencers. We choose *disease outbreak* $\in \mathbf{U}$ as the current event type such that a disease outbreak event leads to a protest event, and



Prompt: *Generate the next event with the event type protest based on the current event below and the corresponding influencing events for the next event:*

On 23 September 2020, Prime Minister Benjamin Netanyahu announced stricter lockdown rules after a new daily coronavirus record of 6,923 infections was reported in Israel.. Disease Outbreak

SOSuMM

Influencing Events: Disease Outbreak, Pathogen Transmission, Protest, Protest March, Testing
Protest march against the stricter lockdown rules announced by Prime Minister Benjamin Netanyahu, with people demanding for more testing and better measures to contain the spread of the disease.

BSuMM OSuMM CSuMM

Influencing Events: Disease Outbreak, Protest, Protest March
A protest march was organized by citizens in Israel to express their opposition to the stricter lockdown rules imposed by Prime Minister Benjamin Netanyahu.

Figure 2: Sample textual output generated by GPT3 based on influencing sets as context from SOSuMM vs. the other 3 LSuMMs.

study the effect of adding other influencing events as context for next event text generation.

Figure 2 shows the sample textual output for the next event generated by GPT3 based on the prompt text (including the current event textual description) and different influencing event types as additional context. We see that a potential advantage of SOSuMM's larger influencing set size is that it may be able to provide more meaningful context for GPT3 to generate richer sample text about *protest*; this additional detail, e.g., demands for more testing, and output variety could be beneficial for an analyst. Further illustrative examples are provided in Appendix C.

6 Conclusion

We introduced logical summary Markov models, a family of event sequence models that enable interpretable predictions in the form of probabilistic logical rules. We proposed two new models that belong to this family: count SuMM and shared order SuMM, along with efficient greedy search algorithms for learning them. Experiments on real-world datasets show the improved performance of these models for event prediction compared with relevant baselines, and potential benefits of larger influencing sets from the shared order SuMM through guiding text generation for event prediction using large language models. Future directions will focus on additional novel models and learning approaches for the LSuMM family, along with neuro-symbolic reasoning systems where the combination of learned logical rules and neural models outperform purely neural or symbolic methods.

References

[Begleiter *et al.*, 2004] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order Markov models. *Journal of Artificial Intelligence Research*, 22:385–421, 2004.

[Bhattacharjya *et al.*, 2020] D. Bhattacharjya, T. Gao, and D. Subramanian. Order-dependent event models for agent interactions. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1977–1983, 2020.

[Bhattacharjya *et al.*, 2021] D. Bhattacharjya, T. Gao, and D. Subramanian. Ordinal historical dependence in graphical event models with tree representations. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 6759–6767, 2021.

[Bhattacharjya *et al.*, 2022] D. Bhattacharjya, S. Sihag, O. Hassanzadeh, and L. Bialik. Summary Markov models for event sequences. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4836–4842, 2022.

[Bishop, 2006] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[Brendel *et al.*, 2011] W. Brendel, A. Fern, and S. Todorovic. Probabilistic event logic for interval-based event recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3329–3336, 2011.

[Brown *et al.*, 2020] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1877–1901, 2020.

[Chickering, 2002] D. M. Chickering. Optimal structure identification with greedy search. *Journal of Machine Learning Research*, 3:507–554, 2002.

[Cropper *et al.*, 2020] A. Cropper, S. Dumančić, and S. H. Muggleton. Turning 30: New ideas in inductive logic programming. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) Survey Track*, pages 4833–4839, 2020.

[De Raedt *et al.*, 2015] L. De Raedt, A. Dries, I. Thon, G. Van Den Broeck, and M. Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of the International Conference on Artificial Intelligence (IJCAI)*, page 1835–1843, 2015.

[Di Ciccio *et al.*, 2018] C. Di Ciccio, F. M. Maggi, M. Montali, and J. Mendling. On the relevance of a business constraint to an event log. *Information Systems*, 78:144–161, 2018.

[Du *et al.*, 2016] N. Du, H. Dai, R. Trivedi, U. Upadhyay, M. Gomez-Rodriguez, and L. Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1555–1564, 2016.

[Fournier-Viger *et al.*, 2011] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng. RuleGrowth: Mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the ACM Symposium on Applied Computing (SAC)*, pages 956–961. ACM, 2011.

[Frank and Asuncion, 2010] A. Frank and A. Asuncion. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2010.

[Grant and Betts, 2013] S. Grant and B. Betts. Encouraging user behaviour with achievements: An empirical study. In *Proceedings of the IEEE Working Conference on Mining Software Repositories (MSR)*, pages 65–68, 2013.

[Grant *et al.*, 2007] R. W. Grant, D. J. Wexler, A. J. Watson, W. T. Lester, E. Cagliero, E. G. Campbell, and D. M. Nathan. How doctors choose medications to treat type 2 diabetes: A national survey of specialists and academic generalists. *Diabetes Care*, 30(6), 2007.

[Guu *et al.*, 2020] H. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang. Retrieval augmented language model pre-training. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3929–3938, 2020.

[Hochreiter and Schmidhuber, 1997] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Jedwabny *et al.*, 2021] M. Jedwabny, P. Bisquert, and M. Croitoru. Probabilistic rule induction for transparent cbr under uncertainty. In *Artificial Intelligence XXXVIII. SGAI-AI. Lecture Notes in Computer Science*, volume 13101. Springer, 2021.

[Lemieux *et al.*, 2015] C. Lemieux, D. Park, and I. Beschastnikh. General LTL specification mining. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 81–92, 2015.

[Letham *et al.*, 2013] B. Letham, C. Rudin, and D. Madigan. Sequential event prediction. *Machine Learning*, 93(2-3):357–380, 2013.

[Li *et al.*, 2021] S. Li, M. Feng, . Wang, A. Essofi, Y. Cao, J. Yan, and L. Song. Explaining point processes by learning interpretable temporal logic rules. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

[Mannila *et al.*, 1997] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.

[Menon and Lee, 2018] A. Menon and Y. Lee. Proper loss functions for nonlinear Hawkes processes. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 3804–3811, 2018.

[Miller, 2019] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[Ng and Subrahmanian, 1992] R. Ng and V.S. Subrahmanian. Probabilistic logic programming. In *Information and Computation*, volume 101, 1992.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[Petroni *et al.*, 2020] F. Petroni, P. Lewis, A. Piktus, T. Rocktäschel, Y. Wu, A. H. Miller, and S. Riedel. How context affects language models' factual predictions. In *Automated Knowledge Base Construction*, 2020.

[Poole, 1993] D. Poole. Logic programming abduction and probability a top down anytime algorithm for estimating prior and posterior probabilities. In *New Generation Computing*, volume 11, 1993.

[Rabiner, 1989] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Raedt *et al.*, 2008] L. De Raedt, P. Frasconi, K. Kersting, and S. Muggleton. *Probabilistic Inductive Logic Programming - Theory and Applications*. Springer, 2008.

[Raftery, 1985] A. Raftery. A model for high-order Markov chains. *Journal of the Royal Statistical Society, Series B*, 47(3):528–539, 1985.

[Riguzzi *et al.*, 2014] F. Riguzzi, E. Bellodi, and R. Zese. A history of probabilistic inductive logic programming. In *Frontiers in Robotics and AI*, volume 1, 2014.

[Riguzzi, 2018] F. Riguzzi. *Foundations of Probabilistic Logic Programming*. River Publishers, Gistrup, Denmark, 2018.

[Rudin *et al.*, 2012] C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and D. Madigan. Sequential event prediction with association rules. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 615–634, 2012.

[Thon *et al.*, 2011] I. Thon, N. Landwerh, and L. De Raedt. Stochastic relational processes: Efficient inference and applications. *Machine Learning*, 82:239–272, 2011.

[Tran and Davis, 2008] S. D. Tran and L. S. Davis. Event modeling and recognition using Markov logic networks. In *Computer Vision – ECCV 2008*, pages 610–623. Springer Berlin Heidelberg, 2008.

[Vennekens *et al.*, 2004] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. In *Logic Programming*, pages 431–445. Springer Berlin Heidelberg, 2004.

[Vrandecic and Krötzsch, 2014] D. Vrandecic and M. Krötzsch. Wikidata: A free collaborative knowledgebase. *Communications of ACM*, 57(10):78–85, 2014.

[Wei *et al.*, 2022] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[Weiss and Hirsh, 1998] G. M. Weiss and H. Hirsh. Learning to predict rare events in event sequences. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 359–363, 1998.

[Xu *et al.*, 2017] H. Xu, D. Luo, and H. Zha. Learning Hawkes processes from short doubly-censored event sequences. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 3831–3840, 2017.

[Yang and Song, 2020] Y. Yang and L. Song. Learn to explain efficiently via neural logic inductive learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.