The Thirty-Third International
FLAIRS Conference (FLAIRS-33)

Modeling Procedural State Changes over Time with Probabilistic Soft Logic

Michael Mohler, Sean Monahan, Marc Tomlinson
{michael,smonahan,marc}@languagecomputer.com
Language Computer Corporation
Richardson, Texas

Abstract

Robust natural language understanding involves the automatic extraction and representation of entities, events, and states from unstructured text. However, a significant portion of the knowledge required for human-level understanding is implicit in the text and can only be accessed through inference. In this work, we employ Probabilistic Soft Logic (PSL) as a framework for leveraging common-sense knowledge to support natural language understanding over procedural texts. Under this framework, we combine logical consistency constraints with succinct representations of common-sense knowledge to probabilistically model entity-centric stative information over time. We demonstrate the feasibility of using PSL to represent procedural stative knowledge through a scalability assessment over an in-house, multi-domain, synthetic dataset.

Introduction

The goal of natural language understanding is to convert textual information into a machine-readable knowledge representation in a manner analogous to human reading and comprehension. Derived knowledge can then be used in support of automatic question answering (Ostermann et al. 2018), automated planning (McCluskey, Vaquero, and Vallati 2017), and other artificial intelligence tasks. While there have been significant advances in information extraction (IE) and knowledge base population (KBP) over the past few decades, explicit information directly extracted from text is insufficient for driving artificial intelligence tasks on its own.

For example, in an automated planning (or plan recognition) scenario, it may be necessary to identify which entities are capable of traveling long distances. In the example sentence "Alice gave Bob a car; then Bob left", extraction components will identify and populate a knowledge base (KB) with three entities ("Alice", "Bob", and "a car") and two events ("gave" and "left") but no explicit stative information. The extracted information cannot be used on its own to answer the underlying question (i.e., who can travel long distances) and cannot represent the complex facts that both individuals have possessed the car, but not at the same time.

In order to overcome this limitation, we apply logical inference as a secondary step to expand the content of the knowledge base. In particular, we seek to model the evolution of state information through time by combining a time-aware knowledge representation with probabilistic inference rules which correspond to common-sense knowledge. For this example, it is possible to identify two dimensions of state associated with the text (i.e., "colocation" and "possession") and track changes among those states, for each entity, throughout the temporal scope of the events. The resulting state information is shown in Table 1.

Time	States
Before "gave"	Possess(Alice, car) ¬Possess(Bob, car) Co-located(Alice, Bob, car)
Between "gave" and "left"	Possess(Bob, car) ¬Possess(Alice, car) Co-located(Alice, Bob, car)
After "left"	Possess(Bob, car) ¬Co-located(Alice, Bob)

Table 1: Evolution of state information for the sentence "Alice gave Bob a car; then Bob left."

Our objectives in this work are three-fold. **First**, we will describe our representation of common-sense knowledge and its applicability to modeling entity state across time. **Second**, we will show how the inference engine automatically derives probabilistic logic rules from this knowledge and exploits global consistency to perform entity-focused state modeling over sequences of events. **Finally**, we will address the practical limitations of PSL-based inference over increasing quantities of realistic data through a two-part scalability assessment.

Related Work

Amassing and representing common-sense knowledge for use in artificial intelligence applications has long been regarded as a critical path in achieving human-level reasoning. WordNet,[1] the most widely used source of knowledge in natural language processing, was constructed manually beginning in 1985 with the goal of identifying distinct senses of nouns, verbs, and adjectives and encoding the hierarchical (IS-A) relationships between them. The Cyc KB,[2] which was also developed by hand over several decades, was constructed to formalize common-sense knowledge in a logical

[1] https://wordnet.princeton.edu/download
[2] https://www.cyc.com/

framework. More recently, the ConceptNet toolkit (Liu and Singh 2004) combined crowdsourced knowledge acquisition with automatic extraction to encode such common-sense relationships as EffectOf, DesireOf, and CapabilityOf.

Within the more constrained space of automated planning with artificial agents, common-sense knowledge is defined as "whatever the agent needs to know about how its environment works in order to achieve its task" and is typically created by hand. The standard for representing such knowledge is the STRIPS action language (Fikes and Nilsson 1971) which is based on propositional logic and defines (a) the set of actions available to the agent, (b) the preconditions for each action (i.e., what must be true about the environment before the action can occur), and (c) the effects of each action (i.e., how it will change the environment). Our knowledge representation described in this work is modeled off of STRIPS with hierarchical and logical components inspired by WordNet and Cyc.

Research in the representation and use of dynamic knowledge bases, which track changes in entity state across time, was pioneered in the development of an Event Calculus (Shanahan 1999), which uses a logical formalism to represent the relationships between EVENTS (actions) and STATES (fluents) over time by encoding the initiating and terminating effects of known events and ensuring logical consistency. Within the planning domain, state modeling proceeds one step at a time, where planners heuristically model the progress of an agent's state towards a goal as a function of an initial state permuted by an action which results in a new state based on the effects of the action (Frances and Geffner 2015). In contrast, contemporary work in process modeling (Mishra et al. 2018) combines knowledge of precondition/effect relations with forward- and backward-propagation through time to result in a "Participant Grid" showing state for each participant at each point in time. In this work, we apply PSL to model state evolution in a logical framework that ensures global consistency subject to small amounts of common-sense knowledge.

Probabilistic Soft Logic (PSL) has recently been used in a variety of reasoning tasks due to (a) its approachability being derived from first-order logic, (b) its defeasibility, (c) its ability to model uncertainty, and (d) its scalability compared to other logical frameworks. PSL has been applied to tasks ranging from knowledge base completion (Yang, Yang, and Cohen 2017) to textual similarity identification(Beltagy, Erk, and Mooney 2014) and causal inference (Sridhar and Getoor 2016). With respect to scalability, Chekol et al. (2017) have applied both Markov Logic Networks (MLNs) and PSL to the task of time-aware knowledge base completion. Using only a small number of hand-crafted rules in a given domain, they report that MLNs are not capable of scaling up to the size of a real-world knowledge base but that PSL has better scalability potential. Our goal in this work is to assess the scalability of PSL on datasets of increasing size and to identify efficiency bottlenecks moving forward.

Knowledge Representation

Before introducing our inference engine, we first define three types of primitives which are used in the representation of both extracted and common-sense knowledge – entities, events, and states. An *entity* is a particular object, thing, or person in the real world with distinct and independent existence. An *event* is defined as an occurrence which results in a change in the state of the world. A *state* is the condition at some moment in time of an entity, an event, or a collection thereof. We define the relationship between the three primitives such that (a) all events, by definition, lead to some new state, and (b) each state or event is grounded by entities (or other states/events) to fill zero or more type-constrained semantic roles. In this way, we represent knowledge in a way analogous to semantic frames, where each frame (i.e., state or event) is grounded with a time parameter. As a part of this work, we have defined a shallow entity type ontology (e.g. THING→LIVING_THING→PERSON), which is used to constrain the arguments of events and states.

In order to model the evolution of state over time, we define common-sense knowledge in the context of a single dimension of state at a time. For instance, the most foundational stative dimension is "existence" – i.e., something either "exists" or does not exist, with "before existence" and "after existence" being distinguished. Each of these three states has two arguments – a single entity argument (i.e., a THING which exists or not) and a time argument. There are three principal events related to the dimension of existence – "creation", "destruction", and "transformation" – which deterministically result in transitions from one state in this dimension to another. Each of these two events have, in addition to a time of occurrence, a pair of arguments – a cause (an ACTOR/PROCESS) and an object of type THING.

Within the context of a single stative dimension, we represent four types of inferential knowledge. The first of these types are *preconditions*, which are states that are known (or expected) to be true immediately before an event occurs. The state "before existence" is a precondition for a "creation" event. Second, we represent *effects* as states which are known (or expected) to be true immediately after an event has taken place. The state "existence" is an effect of "creation". Taken together, preconditions and effects can be organized into a graphical structure where states within the dimension are represented as nodes and events are represented as directed edges linking those nodes. An outlink from a state encodes a precondition while an inlink encodes an effect. Third, we define *sustaining conditions* which are states which must be in effect at a given time for some other state (in a different dimension) to simultaneously be in effect. The states "alive" and "dead" are not relevant for entities which do not exist, so "existence" is a sustaining condition for "life". Finally, we define an *entailment* as an event which occurs simultaneously with another event (e.g., "running" and "moving"). A "transformation" event may simultaneously entail both a "creation" and a "destruction" event. A graphical structure corresponding to the dimension of "marital status" is shown in Figure 1 with "ALIVE" indicated as a sustaining condition.

Scalable Inference

Our state-focused inference engine has been designed around Probabilistic Soft Logic (PSL) (Bach et al. 2015),

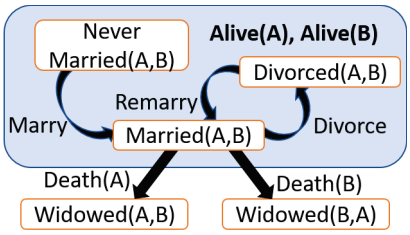

Figure 1: At all times, a pair of living PERSON entities are in exactly one of the marital states – (1) NEVER MARRIED, (2) MARRIED, or (3) DIVORCED – and the life status of both persons is a sustaining condition for all three.

which employs the syntax of first-order logic to scalably and probabilistically drive inference over continuous truth values. We define three steps in the process of using PSL for probabilistic reasoning. **First**, we use the entity ontology, event and state definitions, and the precondition/effect graph structure, sustaining conditions, and other entailments that have been defined for each stative dimension to automatically generate PSL rules. **Second**, we ingest relevant events and states from an existing KB[3] to populate the internal PSL database with input facts and time-grounded candidate inferences. **Third**, we employ joint inference with PSL to compute truth values for all candidate inferences in the database, while satisfying hard and soft constraints defined by the rules. Those stative candidates inferred to be true can then be output to indicate the evolution of state for individual entities throughout the temporal range of the input facts.

Rule Generation

Our first step in modeling state evolution through time with PSL is to derive rules using the four types of common-sense, inferential knowledge encoded for all stative dimensions.[4] Before defining rules, however, we must define the categories of facts which will be available to the rule generation and reasoning components. Each fact (or atom) consists of a semantic predicate (e.g., an event or state) and a predetermined number of terms, where each term is defined by reference to a variable name or to an identifier for a known entity or for another fact.

All fact predicates derived from events and states are categorized in one of five ways. Three of these categories correspond to event or state types from our common-sense stative knowledge and enable the system to distinguish and reason about what is happening now (CURRENT), what has happened before (PERFECT), and what cannot happen in the future (IMPOSSIBLE). Another category (BETWEEN) supports reasoning about events and states bounded by known times but which cannot be fixed at a specific point in time. For example, if a group is known to be ATTENDING an event at time T1, but not at some later time T2, then it can be inferred that some DEPART event must have occurred between T1 and T2. Those states and events which were in the KB before inference are flagged as (EXTRACTED) and serve as an input to inference.

In addition to those predicates derived from events and states, we define two additional categories. The first is an entity type predicate – e.g. PERSON(X) – which defines that entity X is of type "person". Importantly, all distinct times known to the system can be bound in this way, e.g., TIME(T). The second class of predicate enables us to constrain reasoning to focus on more localized problems by modeling temporal adjacency. In particular, we have defined two predicates — NEXT(T,TN) and PREV(T,TP) – which define respectively the next (TN) and previous time (TP) associated with time T. These predicate classes serve as canopies (McCallum, Nigam, and Ungar 2000) and allow us to significantly constrain the number of rules to be satisfied by only applying rules with relevant times and compatible entity typing.[5]

A PSL rule is a set of atoms joined together using a constrained form of the first-order logic syntax, such that the head of the rule must consist of a set of atom conjuncts and the body of the rule must consist of a set of atom disjuncts.[6] Each rule is defined with an associated weight indicating the relative cost of violating that rule during joint inference. Infinite weights indicate a hard logical constraint which cannot be violated without producing a logical contradiction. For instance, we may define an infinite weighted rule (i.e., preconditions for a marriage event) as shown in Figure 2.

Altogether, we define fourteen classes of PSL rules expressing logical consistency and common-sense knowledge, with examples and explanations shown in Table 2. These are derived from the graph structure defined by our stative dimension knowledge resources, but are independent of the particular events and states in any dimension. The first of these groups, labeled "Logical Axioms", represent the logical system itself across all dimensions. Rules from this group have infinite weight, meaning that they cannot be violated without resulting in a logical contradiction. They define the excluded middle principle, mutual exclusion among states in a group, relationships among predicate categories, and the inference of existential quantification facts (i.e., there exists argument X, such that the fact holds).

The second group, labeled "Knowledge-Based Rules", are derived from the structure and reachability information of our stative dimension graphical structures. Rules from this group include precondition/effect, entailments, sustaining conditions, inferred events (before a time or between times), and unambiguous state persistence. These rules likewise have infinite weight and cannot be violated without

[3] In this work, the KB is populated manually, but in general, a KB will be populated through traditional information extraction.

[4] Note that this module is not dependent upon the size of any particular input and so is the most efficient from a scalability perspective. However, reducing or increasing the number of rules produced has a significant effect on efficiency as the number of rules being jointly satisfied during inference changes.

[5] A canopy (also known as a block) limits the comparison space between pairs of entities so that only pairs which are within the same canopy are considered together in any rule.

[6] This restriction enables the system of rules to be converted into a set of Horn clauses which can be satisfied efficiently.

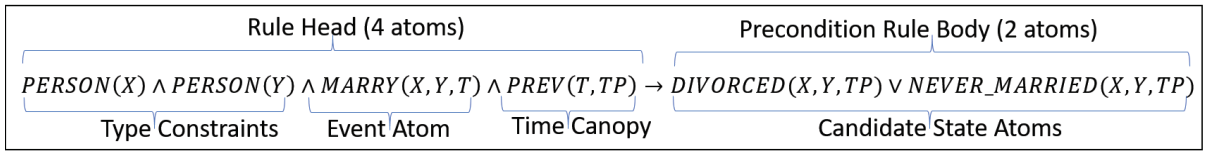

Figure 2: Defines the preconditions for a marriage event with entity variables (X, Y) and time variables (T, TP).

Logical Axioms	
Law of Excluded Middle	A person must be either ALIVE or ¬ALIVE, and one implies that the other is false.
Mutual Exclusion Constraints	A pair of persons must be either NEVER_MARRIED, MARRIED, or DIVORCED (and only one of these) at all times they are alive.
Category Definitions	If a person X is ALIVE_CURRENT at time T, then by definition they are also: ALIVE_PERFECT(X,T); ALIVE_BETWEEN(X,T,T); and ¬ALIVE_IMPOSSIBLE(X,T)
Category Entailments	If a person is ALIVE_PERFECT, they will be ALIVE_PERFECT at all future times. If a person is ¬ALIVE_IMPOSSIBLE, they were also ¬ALIVE_IMPOSSIBLE at all prior times. If a person is ALIVE_BETWEEN in some range, they were also ALIVE_BETWEEN for all ranges containing that time range.
Existential Quantification	If a person X is MARRIED to person Y, then the facts that person X is MARRIED (to someone) and person Y is MARRIED (to someone) are also true.
Knowledge-Based Rules	
Precondition/Effect	When a person experiences a BIRTH event, they enter the ALIVE state Before a person experiences a DEATH event, they must have been in the ALIVE state.
Sustaining Conditions	If a pair is MARRIED, DIVORCED, or NEVER_MARRIED, we can infer they are each ALIVE. If a company is EMPLOYING someone, then the company EXISTS at that time.
Ontological Entailment	If an entity is type PERSON, the entity is also type LIVING_THING.
Inferred Precedents	If a person is ALIVE, they must have once had a BIRTH. If a person is DIVORCED or WIDOWED, they must have once been MARRIED.
Inferred Between	If a person is ALIVE at time T, and DEAD at time T2, there must have been a DEATH event at some time between T and T2.
State Persistence	If a person is DEAD, they will be DEAD at all later times. If a person is ALIVE at time T and ALIVE at time T2, then they are ALIVE at all times between T and T2.
Priors and Assumptions	
Category Priors	Assume a state/event does not hold (CURRENT) Assume states/events have never held (PERFECT=false at final time)
Persistence Assumptions	Assume a state is the same at the next/previous times
Initial Conditions	Assume an initial state holds within a group (e.g., a pair of living persons are NEVER_MARRIED)

Table 2: Fourteen classes of rules which are automatically derived from the graph structure for each stative dimension.

contradiction.

The last group, labeled "Priors and Assumptions", are rules which may be violated without resulting in a contradiction. However, they represent either strong (probabilistic) assumptions or weak priors (with very low weight) on what to expect when no better information is available. By incorporating prior rules into our inference engine, we ensure the defeasibility of facts in a dynamic setting. If newer information results in a previously inferred fact being rejected, the tendency to assume the weak priors will work to remove inferences derived from the rejected fact, thereby ensuring the logical consistency of the system overall.

Fact Population

In the second stage, the set of grounded facts which may be inferred (i.e., the candidate inferences) are ingested into the PSL database. Initially, the database is seeded with only the EXTRACTED facts from the KB. Then, it must be populated with all facts which could conceivably be true to define a closed world within which reasoning takes place. This step significantly impacts the scalability of the system by directly affecting the size of the database and indirectly affecting the number of grounded rules which must be jointly satisfied in the inference stage. In order to provide a practical look at the use of PSL for inference, we describe three alternative methodologies for fact population that were attempted over the course of this work.

Under the first methodology, we only ingested EXTRACTED facts and made use of an option within the PSL framework itself to lazily add facts to the database whenever they were inferred. That is, grounding a rule required only that all facts in the head exist in the database; facts in the body could be automatically added. This mode resulted in an iterative inference stage and incurred a significant penalty to efficiency as the number of iterations before convergence was substantial and each iteration became less and less efficient.

In the second methodology, we populated the database with candidate facts such that each predicate was grounded to all possible entities from the input, with all possible times from the input – e.g., whether all pairs of people were MARRIED/DIVORCED/WIDOWED at every time step in the inputs. This method resulted in a database where the vast majority of candidate facts were never in danger of being inferred, but whose associated rules must still be satisfied during inference.

Ultimately, we worked to automatically identify the most likely candidate facts as follows. First, we employed the entity-type ontology to limit the entity groundings for a given predicate. For instance, the MARRIAGE predicate does not need to be grounded for a "book" entity. Next, we employed two different forms of logical canopies to con-

strain the possibilities. Our first canopy was an input entity-pair canopy which constrained the rules to only consider a pair of entities together if they participated together in ANY input fact. We likewise simplified our temporal space by binding entities to particular times (i.e., all the times that co-occurred with the entity in the input facts, plus the first and the last times over all facts), so that no fact would be defined for an entity if the input made no mention of it at that time.

PSL Inference

Probabilistic Soft Logic (Bach et al. 2015) is an inference framework which employs hinge-loss Markov random fields (HL-MRF) to enable scalable inference over data constrained by grounded rules. A grounded rule is an instantiation of a PSL rule over real facts in the database (i.e., grounding). Within a grounded rule, each atom will have been resolved such that all variables are replaced with the identifier of some entity or fact in the database, under the constraint that all equivalent variables in the rule must be grounded to the same identifier. Unlike other first-order logic frameworks where each atom holds a binary truth value, PSL employs soft logic to enable probabilistic inference and to support significant scalability gains. This requires an alternative definition of both truth values and the Boolean operators. In particular, PSL relaxes truth values to a continuous value in the [0..1] interval and applies the Łukasiewicz t-norm and its dual t-conorm to define the logical conjunction and disjunction operators, respectively. That is, $AND(A,B) = max(0, A+B-1)$ and $OR(A,B) = min(1, A+B)$. Likewise, logical negation is defined as $NOT(A) = 1-A$. With these definitions in mind, each grounded rule in the database, after inference, will be considered satisfied if the conjunctive truth value of the head is less than or equal to the disjunctive truth value of the body and violated otherwise.

At inference time, the PSL reasoning module iteratively updates the truth values of all ground atoms, subject to the rule constraints, so as to induce a joint probability distribution over the set of facts, F, in the database. More formally, the inference module assigns a truth value $T(f)$ to each fact $f \in F$ and computes a distance to satisfaction for each grounded rule $r \in R$ as $\Phi(r) = max(0, T_{head}(r) - T_{body}(r))$, where the truth value of the rule heads and bodies are defined via conjunction or disjunction over the facts which compose them. PSL then defines a probability distribution over R as $P_\Psi(R)$, represented as a weighted combination of the distances to satisfaction over the grounded rules, i.e.:

$P_\Psi(R) = \frac{1}{Z} exp[-\sum_{r \in R} \omega_r \Phi(r)]$

where ω_r is the rule weight.

Finding the soft-truth values of every fact in the database, F, so as to minimize the value $P_\Psi(R)$ of the weighted constraint violations across all grounded rules is equivalent to inferring the most probable explanation (MPE) for the input facts. This can be formulated as a convex optimization problem, which is solved under PSL using the Alternating Direction Method of Multipliers (ADMM) (Bach et al. 2012) and scales linearly with the number of grounded rules. Once inference is complete, the database can be queried to identify the truth values for all candidate facts – i.e., all states associated with entities at any point in time – to build up the final model of state evolution. For the purpose of interpretability, the soft-truth values of ground atoms can be seen as *a posteriori* confidences in the truth of each possible fact.

As a side-effect of performing global inference with PSL, it is possible to detect inconsistencies in the input (KB) data. Although, the PSL inference module attempts to assign values to each fact without violating any constraints (rules), it will prefer to violate constraints with lower weight – first priors, but then rules defining the truth of the original KB facts. Therefore, violated input constraints are often indicative of facts in the KB which result in a contradiction. In an operational system, these can be resolved by other downstream components or provided to the user for further analysis.

Experiments

In order to benchmark PSL inference for scalability, we have manually defined predicates and common-sense knowledge associated with 13 stative dimensions including "existence", "life", "marriage", "injury", "damage", "knowledge", "paper publication", "paper topic inclusion", "process stages", "event attendance", "employment", "interpersonal association", and "gender".[7] On average, knowledge engineering for each dimension required 15-25 minutes of manual effort.

Using this knowledge resource, we developed a synthetic KB as follows. First, we defined a set of entities of various types – people, objects, processes, etc. Then, we automatically generated 100 events or states associated with our most foundational domain (i.e., EXISTENCE). These were generated by randomly selecting one of the predicates (events or states) defined in this dimension to produce a fact. We then randomly filled the non-temporal arguments of the fact with an entity of its required type or with an existential quantifier. Finally, we randomly selected a date between 1970 and 2015 to be associated with the fact. From these 100 grounded facts, we then manually selected 15 to 30 in such a way as to ensure the logical consistency of the resultant KB (e.g., an entity cannot EXIST before it was CREATED). This process was then repeated iteratively for each added dimension, resulting in a dataset of 244 atoms covering 60 state types, 61 event types, and 18,235 derived PSL rules — about 150 rules per predicate.

We have used this dataset in two experiments to track the growth in the size of the database and in the time spent in inference as a function of the number of input facts. The first was an iterative experiment where, for each iteration, we append input facts for a single new stative dimension, perform inference, and analyze the size and efficiency characteristics of the iteration. This experiment measures the growth in latency as problem complexity increases and is summarized in Table 3.

For our second scalability experiment, we made use of all 13 stative dimensions but randomly sampled only a subset

[7] These dimensions were selected to be representative of physical, mental, and social constructs that can be used for inference over open-domain data.

Domain	States	Events	Transitions	Entailments	Rules	Entities	Input Atoms	Output Atoms (k)	Grounded Rules (k)	Time
1 dimension	3	2	4	2	572	16	21	6.6	27.6	9s
5 dimensions	18	17	41	10	4,816	38	117	715	2,808	11m36s
9 dimensions	45	41	95	42	11,236	60	212	4,615	16,351	1h27m
13 dimensions	60	61	134	58	18,235	73	244	8,107	28,052	2h46m

Table 3: Scalability results when adding additional complexity.

	Input Atoms (IA)	Entities	Output Atoms (k)	Grounded Rules (k)	Time	Output Atoms/IA	Grounded Rules/IA	Time/IA
10%	26	24	332	1,446	3m16s	12,769/fact	55,615/fact	7.5s/fact
25%	66	43	1,143	4,773	11m44s	17,318/fact	72,318/fact	10.7s/fact
50%	133	63	3,019	11,916	41m6s	22,6997/fact	89,593/fact	18.5s/fact
75%	200	69	5,423	20,537	1h24m	27,115/fact	102,685/fact	25.2s/fact
100%	244	73	8,107	28,052	2h46m	33,225/fact	114,967/fact	40.8s/fact

Table 4: Scalability experiment adding input facts with complexity (i.e., number of stative dimensions) held constant.

of the individual input facts from the synthetic dataset. We sampled the dataset from 10% to 100% of its total size to determine the impact of increasing size (but not complexity) on PSL-based inference. This experiment closely resembles a natural increase in facts over the life span of a knowledge base and is summarized in Table 4.

Discussion

Altogether, the results of these experiments substantiate our novel approach towards using PSL to jointly model the evolution of state information across time at a small scale. In addition, by scaling up towards real-world dataset sizes and analyzing the atoms and grounded rules in the database, we have identified several bottlenecks for which time does not increase linearly with the number of input facts. First, it was found that BETWEEN predicates (with two time parameters) represent a significant percentage of the total size, and so linear increases in the number of distinct time parameters result in quadratic increases in inference latency. Likewise, it was discovered early on that over-populating the database with all potential facts (e.g. with all combinations of entities) significantly impairs the ability of the system to scale up.

Moving forward, we plan to remove BETWEEN-style fact types from the PSL-based inference components in favor of deriving such knowledge offline after an initial inference stage. In addition to our use of canopies described in this work, we plan to apply additional canopies, such that all facts and only those facts associated with a given entity (or entity pair) will be linked in any sequence of grounded rules, thereby reducing the inter-connectedness of the grounded rules and reducing time spent on highly unlikely inferred facts. Indeed, we propose in future work to explore the possibility of employing an initial "light-weight" inference module which greedily applies precondition, effect, entailment, and other inference across time. We will then only apply PSL when initial inference detects a potential contradiction or ambiguity which needs to be resolved through more complex joint reasoning.

Acknowledgments

This research is based upon work supported by the Defense Threat Reduction Agency under Contract No. HDTRA1-17-C-0056.

References

Bach, S.; Broecheler, M.; Getoor, L.; and O'leary, D. 2012. Scaling MPE inference for constrained continuous markov random fields with consensus optimization. In *NIPS*, 2654–2662.

Bach, S. H.; Broecheler, M.; Huang, B.; and Getoor, L. 2015. Hinge-loss markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Beltagy, I.; Erk, K.; and Mooney, R. J. 2014. Probabilistic soft logic for semantic textual similarity. In *ACL (1)*, 1210–1219.

Chekol, M. W.; Pirrò, G.; Schoenfisch, J.; and Stuckenschmidt, H. 2017. Marrying uncertainty and time in knowledge graphs. In *AAAI*, 88–94.

Fikes, R. E., and Nilsson, N. J. 1971. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence* 2(3-4):189–208.

Frances, G., and Geffner, H. 2015. Modeling and computation in planning: Better heuristics from more expressive languages. In *Int. Conf. on Automated Planning and Scheduling*.

Liu, H., and Singh, P. 2004. ConceptNet—a practical commonsense reasoning tool-kit. *BT technology journal* 22(4):211–226.

McCallum, A.; Nigam, K.; and Ungar, L. H. 2000. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proc. of ACM SIGKDD conference on Knowledge discovery and data mining*, 169–178. ACM.

McCluskey, T. L.; Vaquero, T. S.; and Vallati, M. 2017. Engineering knowledge for automated planning: Towards a notion of quality. In *Proc. of the Knowledge Capture Conference*, 14. ACM.

Mishra, B. D.; Huang, L.; Tandon, N.; Yih, W.-t.; and Clark, P. 2018. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. *arXiv preprint arXiv:1805.06975*.

Ostermann, S.; Roth, M.; Modi, A.; Thater, S.; and Pinkal, M. 2018. Semeval-2018 Task 11: Machine comprehension using commonsense knowledge. In *Proc. of SEMEVAL Workshop*, 747–757.

Shanahan, M. 1999. The event calculus explained. In *Artificial intelligence today*. Springer. 409–430.

Sridhar, D., and Getoor, L. 2016. Joint probabilistic inference of causal structure. In *Proc. of ACM SIGKDD, Workshop on Causal Discovery*.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base completion. *arXiv preprint arXiv:1702.08367*.