Markov Logic Networks for Natural Language Question Answering

Tushar Khot
Allen Institute for AI
Seattle, WA 98103

Niranjan Balasubramanian
Dept. of Computer Science
Stony Brook University
Stony Brook, NY 11794

Eric Gribkoff
Computer Sci. and Engr.
University of Washington
Seattle, WA 98195

Ashish Sabharwal
Peter Clark
Oren Etzioni
Allen Institute for AI
Seattle, WA 98103

Abstract

Our goal is to answer elementary-level science questions using knowledge extracted automatically from science textbooks, expressed in a subset of first-order logic. Given the incomplete and noisy nature of these automatically extracted rules, Markov Logic Networks (MLNs) seem a natural model to use, but the exact way of leveraging MLNs is by no means obvious. We investigate three ways of applying MLNs to our task. In the first, we simply use the extracted science rules directly as MLN clauses. Unlike typical MLN applications, our domain has long and complex rules, leading to an unmanageable number of groundings. We exploit the structure present in hard constraints to improve tractability, but the formulation remains ineffective. In the second approach, we instead interpret science rules as describing prototypical entities, thus mapping rules directly to grounded MLN assertions, whose constants are then clustered using existing entity resolution methods. This drastically simplifies the network, but still suffers from brittleness. Finally, our third approach, called Praline, uses MLNs to align the lexical elements as well as define and control how inference should be performed in this task. Our experiments, demonstrating a 15% accuracy boost and a 10x reduction in runtime, suggest that the flexibility and different inference semantics of Praline are a better fit for the natural language question answering task.

Introduction

Many question answering or QA tasks require the ability to reason with knowledge extracted from text. We consider the problem of answering questions in standardized science exams (Clark, Harrison, and Balasubramanian 2013). In particular, we focus on a subset that tests students' understanding of various kinds of general rules and principles (e.g., *gravity pulls objects towards the Earth*) and their ability to apply these rules to reason about specific situations or scenarios (e.g., *which force is responsible for a ball to drop?*).

This task can be viewed as a natural first-order reasoning problem specified over general truths expressed over classes of events or entities. However, this knowledge is automatically derived from appropriate science texts.

In order to effectively reason over knowledge derived from text, a QA system must handle incomplete and potentially noisy knowledge, and reason under uncertainty. Markov Logic Network (MLN) is a formal probabilistic inference framework that allows for robust inference using rules expressed in probabilistic first-order logic (Richardson and Domingos 2006). MLNs have been widely adopted for many tasks (Singla and Domingos 2006a; Kok and Domingos 2008; Poon and Domingos 2009). Recently, Beltagy et al. (2013; 2014) have shown that MLNs can be used to reason with rules derived from natural language.

While MLNs appear a natural fit, it is *a priori* unclear how to effectively formulate the QA task. Moreover, the unique characteristics of this domain pose new challenges in grounding and ability to control inference under incomplete evidence. We investigate two standard formulations, uncover efficiency and brittleness issues, and propose an enhanced formulation more suitable for this domain. This enhanced formulation, called Praline, significantly outperforms our other MLN formulations, reducing runtime by 10x and improving accuracy by 15%.

Setup: Question Answering Task

Following Clark et al. (2014), we formulate QA as a reasoning task over knowledge derived from textual sources. Specifically, a multiple choice question with k answer options is turned into k true-false questions, each of which asserts some known facts, referred to as the $setup$, and posits a $query$. The reasoning task is to determine whether the $query$ is true given the $setup$ and the input knowledge.

The input knowledge is derived from 4th-grade level science texts and augmented with a web search for terms appearing in the texts. Much of this knowledge is in terms of generalities, expressed naturally as IF-THEN rules. We use the representation and extraction procedures of Clark et al. (2014), recapitulated briefly here for completeness.

Rule Representation: The generalities in text convey information about classes of entities and events. Following the neo-davidsonian reified representation (Curran, Clark, and Bos 2007), we encode information about events (e.g, falling, dropping etc.) and entities (e.g., ball, stone etc.) using variables. Predicates such as $agent, cause, function, towards, in$ etc., define semantic relationships between the variables. Rather than committing to a type ontology, the variables are associated with their original string representation using an isa predicate.

The "if" or $antecedent$ part of the rule is semantically interpreted as being universally quantified (omitted below for conciseness) whereas every entity or event mentioned only in the "then" or $consequent$ part of the rule is treated as existentially quantified. Both $antecedent$ and $consequent$ are interpreted as conjunctions. E.g., "Growing thicker fur

in winter helps some animals to stay warm" translates into:

$$isa(g, \text{grow}), isa(a, \text{some\_animals}), isa(f, \text{thicker\_fur}), isa(w, \text{the\_winter}), agent(g, a), object(g, f), in(g, w) \Rightarrow \exists s, r : isa(s, \text{stays}), isa(r, \text{warm}), enables(g, s), agent(s, a), object(s, r) \quad (1)$$

Question Representation: The question representation is computed similarly except that we use fixed constants (represented as block letters) rather than variables. E.g., consider the question: "A fox grows thick fur as the season changes. This helps the fox to (a) hide from danger (b) attract a mate (c) find food (d) keep warm?" The T/F question corresponding to option (d) translates into:

$$setup : isa(F, \text{fox}), isa(G, \text{grows}), isa(T, \text{thick\_fur}), agent(G, F), object(G, T)$$
$$query : isa(K, \text{keep\_warm}), enables(G, K), agent(K, F)$$

Lexical Reasoning: Since entities and event variables hold textual values, reasoning must accommodate the lexical variability and textual entailment. For example, the surface forms "thick_fur" in the question and "thicker_fur" are semantically equivalent. Also, the string "fox" entails "some_animal". We use a lexical reasoning component based on textual entailment to establish lexical equivalence or entailment between variables.

Most Likely Answer as Inference: Given the input KB rules and the question, we formulate a probabilistic reasoning problem by adding lexical reasoning probabilities and incorporating uncertainties in derived rules. Specifically, given setup facts S and k answer options Q_i, we seek the most likely answer option: $\arg\max_{i \in \{1,\ldots,k\}} \Pr[Q_i \mid S, KB]$. This is a Partial MAP computation, in general #P-hard (Park 2002). Hence methods such as Integer Linear Programming are not directly applicable.

Challenges

Reasoning with text-derived knowledge presents, in addition to lexical uncertainty, challenges that expose the *brittleness* and *rigidity* inherent in pure logic-based frameworks. In particular, text-derived rules are incomplete and include lexical items as logical elements, making rule application in a pure logical setting extremely brittle: Many relevant rules cannot be applied because their pre-conditions are not fully satisfied due to poor alignment. For example, naive application of rule (1) on *setup* would not conclude *query* as the rule requires "in the winter" to be true. A robust inference mechanism must allow for rule application with partial evidence.

Further, a single text-derived rule may be insufficient to answer a question. E.g., "Animals grow thick fur in winter" and "Thick fur helps keep warm" may need to be chained.

Probabilistic Formulations

Statistical Relational Learning (SRL) models (Getoor and Taskar 2007) are a natural fit for QA. They provide probabilistic reasoning over knowledge represented in first-order logic, thereby handling uncertainty in lexical reasoning and incomplete matching. While there are many SRL formalisms including Stochastic Logic Programs (SLPs) (Muggleton 1996), ProbLog (Raedt, Kimmig, and Toivonen 2007), PRISM (Sato and Kameya 2001), etc., we use Markov Logic Networks (MLNs) for their ease of specification and their ability to naturally handle potentially cyclic rules. We explore three formulations:

a) First-order MLN: Given a question and relevant first-order KB rules, we convert them into an MLN program and let MLN inference automatically handle rule chaining. While a natural first-order formulation of the QA task, this struggles with long conjunctions and existentials in rules, as well as relatively few atoms and little to no symmetries. This results in massive grounding sizes, not remedied easily by existing solutions such as lazy, lifted, or structured inference. We exploit the structure imposed by hard constraints to vastly simplify groundings and bring them to the realm of feasibility, but performance remains poor.

b) Entity Resolution MLN: Instead of reasoning with rules that express generalities over classes of individuals, we replace the variables in the previous formulation with prototypical constants. This reduces the number of groundings, while retaining the crux of the reasoning problem defined over generalities. Combining this idea with existing entity resolution approaches substantially improves scalability. However, this turns out to be too brittle in handling lexical mismatches, e.g., different sentence parse structures.

c) Praline MLN: Both of the above MLNs rely on exactly matching the relations in the KB and question representation, making them too brittle for this task. In response, PRobabilistic ALignment and INferencE (Praline) performs inference using primarily the string constants but guided by the edge structure. We relax the rigidity in rule application by explicitly modeling the desired QA inference behavior.

(A) First-Order MLN Formulation

For a set R of first-order KB rules, arguably the most natural way to represent the QA task of computing $\Pr[Q_i \mid S, R]$ as an MLN program M is to simply add R essentially verbatim as first-order rules in M. Quantified variables of M are those occurring in R. Constants of M are the string representations (e.g., "fox", "thicker_fur") in Q_i, S, and R, as well as the constants in the Q_i and S (e.g., F, T). In addition, for all existentially quantified variables, we introduce a new domain constant. Predicates of M are those in R, along with a binary $entails$ predicate representing the lexical entailment blackbox, which allows $M$to probabilistically connect lexically related constants such as "thick_fur" and "thicker_fur" or "fox" and "some_animals". $entails$ is defined to be closed-world and is not necessarily transitive.

Refined Types: For improved semantics and reduced grounding size, M has entities (A), events (E), and strings as three basic types, and predicates of M are appropriately typed (e.g., $agentEA$, $entailsEE$, $entailsAA$). Further, we avoid irrelevant groundings by using *refined types* determined dynamically: if $r(x, y)$ appears only with constants associated with strings T as the second argument, then M contains $r(x, y) \rightarrow !isa(y, s)$ for all strings s with a zero entailment score with all strings in T.

Evidence: Soft evidence for M consists of $entails$ relations between every ordered pair of entity (or event) strings, e.g., $entails(\text{fox}, \text{some\_animals})$. Hard evidence for M comprises grounded atoms in S.

Query: The query atom in M is $result()$, a new zero-arity predicate $result()$ that is made equivalent to the conjunction

of the predicates in Q_i that have not been included in the evidence. We are interested in computing $\Pr[result() = \text{true}]$.

Semantic Rules: In addition to KB science rules, we add *semantic rules* that capture the intended meaning of our predicates, such as every event has a unique agent, $cause(x, y) \rightarrow effect(y, x)$, etc. Semantic predicates also enforce natural restrictions such as non-reflexivity, $!r(x, x)$, and anti-symmetry, $r(x, y) \rightarrow !r(y, x)$.

Finally, to help bridge lexical gaps more, we use a simple external lexical alignment algorithm to estimate how much does the $setup$ entail $antecedent_r$ of each KB rule r, and how much does $consequent_r$ entail $query$. These are then added as two additional MLN rules per KB rule.

These rules have the following first-order logic form:

$$\forall x_1, .., x_k \bigwedge_i R_i(x_{i_1}, x_{i_2}) \rightarrow \exists x_{k+1}, .., x_{k+m} \bigwedge_j R_j(x_{j_1}, x_{j_2})$$

Existentials spanning conjunctions in the consequent of this rule form can neither be directly fed into existing MLN systems nor efficiently expanded naively into a standard conjunctive normal form (CNF) without incurring an exponential blowup during the transformation. To address this, we introduce a new "existential" predicate $E^{\alpha}_j(x_1, \ldots, x_k, x_{k+j})$ for each existential variable x_{k+j} in each such rule α. This predicate becomes the consequent of α, and subsequent hard MLN rules make it equivant to the original consequent.

Boosting Inference Efficiency. A bottleneck in using MLN solvers out-of-the-box for this QA formulation is the prohibitively large grounded network size. For instance, 31% of our runs that timed out during the MLN grounding phase after 6 minutes were dealing, on average, with 1.4×10^6 ground clauses. Such behavior has also been observed, perhaps to a lesser degree, in related NLP tasks (Beltagy and Mooney 2014; Beltagy, Erk, and Mooney 2014).

Large grounding size is, of course, a well-studied problem in the MLN literature. However, a key difference from previously studied MLNs is that our QA encodings have small domain sizes and, therefore, *very few ground atoms* to start with. Existing techniques for addressing the grounding challenge were inspired by the unmanageable number of ground atoms often seen in traditional MLN applications, and work by grouping them into large classes of interchangeable atoms (de Salvo Braz, Amir, and Roth 2005; Gogate and Domingos 2011; Venugopal and Gogate 2012; Domingos and Webb 2012; Niepert and Van den Broeck 2014). Similarly, memory-efficient Lazy Inference (Singla and Domingos 2006b) and FROG (Shavlik and Natarajan 2009) focus only on relevant atoms.

These methods were ineffective on our MLNs. E.g., lazy inference in Alchemy-1 reduced ∼70K ground clauses to ∼56K on a question, while our method, described next, brought it down to only 951 clauses. Further, Lifted Blocked Gibbs and Probabilistic Theorem Proving, as implemented in Alchemy-2, were slower than basic Alchemy-1.

Different from heuristic approximations (e.g., Modified Closed-World Assumption of Beltagy and Mooney (2014)), we propose reducing the grounding size without altering the semantics of the MLN program. We utilize the combinatorial structure imposed by the set H of hard constraints present in the MLN, and use it to simplify the grounding of *both* hard and soft constraints. Lazy inference mentioned above may be viewed as the very first step of our approach.

Assuming an arbitrary ordering of the constraints in H, let F_i denote the first i constraints. Starting with $i = 1$, we generate the propositional grounding G_i of F_i, use a propositional satisfiability (SAT) solver to identify the set B_i of *backbone variables* of G_i (i.e., variables that take a fixed value in all solutions to G_i), freeze values of the corresponding atoms in B_i, increment i, and repeat until $G_{|H|}$ has been processed. Although the end result can be described simply as freezing atoms corresponding to the backbone variables in the grounding of H, the incremental process helps us keep the intermediate grounding size under control as a propositional variable is no longer generated for an atom once its value is frozen. Once the freezing process is complete, the full grounding of H is further simplified by removing frozen variables. Finally, the soft constraints S are grounded much more efficiently by taking frozen atoms into account.

This approach can be seen as an extension of a proposal by Papai, Singla, and Kautz (2011). They used a polynomial-time Generalized Arc Consistency algorithm on H to compute a subset of $B_{|H|}$ that is efficiently identifiable, implemented as a sequence of *join* and *project* database operations

(B) Entity Resolution Based MLN

Representing generalities as quantified rules defined over classes of entities or events appears to be a natural formulation, but is also quite inefficient leading to large grounded networks. We instead consider an alternative formulation that treats generalities as relations expressed over *prototypical* entities and events. This formulation leverages the fact that elementary level science questions can often be answered using relatively simple logical reasoning over exemplar objects and homogeneous classes of objects, if given perfect information as input. The only uncertainty present in our system is what's introduced by lexical variations and extraction errors, which we handle with probabilistic equality.

KB Rules and Question: We create rules defined over prototypical entity/event *constants*, rather than first-order variables. These constants are tied to their respective string representations, with the understanding that two entities behave similarly if they have lexically similar strings. E.g.,

$$\begin{aligned} &isa(G, \text{grow}), isa(A, \text{some\_animals}), isa(F, \text{thicker\_fur}), \\ &isa(W, \text{the\_winter}), agent(G, A), object(G, F), in(G, W) \\ &\Rightarrow isa(S, \text{stays}), isa(R, \text{warm}), \\ &\quad enables(G, S), agent(S, A), object(S, R) \end{aligned}$$

What was a first-order rule in FO-MLN is now already fully grounded! It has no variables. Entities/events mentioned in the question are also similarly represented by constants.

Equivalence or Resolution Rules: Using a simple probabilistic variant of existing Entity/Event Resolution frameworks (Singla and Domingos 2006a; Kok and Domingos 2008), we ensure that (a) two entities/events are considered similar when they are tied to lexically similar strings and (b) similar entities/events participate in similar relations w.r.t. other entities/events. This defines soft *clusters* or equivalence classes of entities/events. To this end, we use a probabilistic $sameAs$ predicate which is reflexive, symmetric, and

transitive, and interacts with the rest of the MLN as follows:

$$w(s, s') : \ entails(s, s')$$
$$isa(x, s), entails(s, s') \rightarrow isa(x, s').$$
$$isa(x, s), isa(y, s) \rightarrow sameAs(x, y).$$
$$w : \ isa(x, s), !isa(y, s) \rightarrow !sameAs(x, y)$$
$$r(x, y), sameAs(y, z) \rightarrow r(x, z).$$

r in the last rule refers to any of the MLN predicates other than $entails$ and isa. The $sameAs$ predicate, as before, is implemented in a typed fashion, separately for entities and events. We will refer to this formulation as ER-MLN.

Partial Match Rules Due to lexical variability, often not all conjuncts in a rule's $antecedent$ are present in the question's $setup$. To handle incomplete matches, for each KB derived MLN rule of the form $(\wedge_{i=1}^{k} L_i) \rightarrow R$, we also add k soft rules of the form $L_i \rightarrow R$. This adds flexibility, by helping "fire" the rule in a soft manner.

Comparison with FO-MLN: Long KB rules and question representation now no longer have quantified variables, only the binary or ternary rules above do. These mention at most 3 variables each and thus have relatively manageable groundings. On the other hand, as discussed in the next section, ER-MLN can fail on questions that have distinct entities with similar string representations. Further, it is brittle to syntactic differences such as $agent$(Fall, Things) generated by "things fall due to gravity" and $object$(Dropped, Ball) for "a student dropped a ball". Although "drop" entails "fall" and "ball" entails "object", ER-MLN cannot reliably bridge the structural difference involving $object$ and $agent$, as these two relationships typically aren't equivalent. Despite these limitations, ER-MLN provides a substantial scalability advantage over FO-MLN on a vast majority of the questions that remain within its scope.

(C) PRobabilistic ALignment and INferencE

ER-MLN handles some of the word-level lexical variation via *resolution* and soft *partial match* rules that break long antecedents. However, it is still rigid in two respects:

1. Inference primarily relies on the predicates (also referred to as links or edges) for inference. As a result, even if the words in the $antecedent$ and $setup$ have high entailment scores, the rule will still not "fire" if the edges do not match. To enable effective rule application under such circumstances, we require (at a minimum) some match on the string constants and allow edge matches (if any) to increase the confidence in rule application.
2. As clustering forces entities bound to lexically equivalent strings to "behave" identically, it fails on questions that involve two different entities that are bound to equivalent string representations. To avoid this issue, we do not force the entailment-based clusters of constants to behave similarly. Instead, as we discuss below, we use the clusters to guide inference in a softer manner.

To introduce such flexibility, we convert the problem of uncertainty over links between string constants to the problem of uncertainty over the existence of these constants. To this end, we introduce a unary predicate over string constants to capture what is known to be true (i.e., the $setup$) or can be proven to be true (via inference using the KB rules) in the world. We then define an MLN to directly control how new facts are inferred given the KB rules. The flexibility to control inference helps address two additional QA challenges:

Acyclic inference: While knowledge is extracted from text as a set of directed rules each with an $antecedent$ and a $consequent$, there is no guarantee that the rules taken together are acyclic. E.g., a rule stating "Living things $\rightarrow$ depend on the sun" and "Sun $\rightarrow$ source of energy for living things" may exist side-by-side. Successful inference for QA must avoid feedback loops.

False unless proven: While MLNs assume atoms not mentioned in any rule to be true with probability 0.5, elementary level science reasoning is better reflected in a system that assumes all atoms to be false unless stated in the question or proven through the application of a rule. This is similar to the semantics of Problog (Raedt, Kimmig, and Toivonen 2007) and PRISM (Sato and Kameya 2001).

MLN specification We introduce a unary predicate called $holds$ over string constants to capture the probability of a string constant being true given the $setup$ is true ($\forall x \in setup, holds(x) = true$) and the KB rules hold. Instead of using edges for inference, we use them as factors influencing alignment: similar constants have similar local neighborhoods. This reduces the number of unobserved groundings from $O(n^2)$ edges in the ER-MLN to $O(n)$ existence predicates, where n is the number of string constants. For the example rule (1), Praline can be viewed as using the following rule for inference:

$$holds(Grow), holds(Animals), holds(Fur),$$
$$holds(Winter) \Rightarrow holds(Stays), holds(Warm)$$

If we view KB rules and the question as a labeled graph G shown in Figure 1, alignment between string constants corresponds to alignment between the nodes in G. The nodes and edges of G are the input to the MLN, and the $holds$ predicate on each node captures the probability of it being true given the $setup$. We now use MLNs (as described below) to define the inference procedure for any such input graph G.

Input Predicates: We represent the graph structure of G using predicates $node(nodeid)$ and $edge(nodeid, nodeid, label)$. We use $setup(nodeid)$ and $query(nodeid)$ to represent the question's $setup$ and $query$, resp. Similarly, we use $inLhs(nodeid)$ and $inRhs(nodeid)$ to represent rules' $antecedent$ and $consequent$, resp.

Graph Alignment Rules: Similar to the previous approaches, we use entailment scores between words and short phrases to compute the alignment. In addition, we also expect *aligned* nodes to have similar edge structures:

$$aligns(x, y), edge(x, u, r), edge(y, v, s) \Rightarrow aligns(u, v)$$
$$aligns(u, v), edge(x, u, r), edge(y, v, s) \Rightarrow aligns(x, y)$$

That is, if node x aligns with y then their children/ancestors should also align. We create copies of these rules for edges with the same label, $r = s$, with a higher weight and for edges with different labels, $r \neq s$, with a lower weight.

Inference Rules: We use MLNs to define the inference procedure to prove the $query$ using the alignments from $aligns$. We assume that any node y that $aligns$ with a node x that $holds$, also $holds$:

$$holds(x), aligns(x, y) \Rightarrow holds(y) \qquad (2)$$

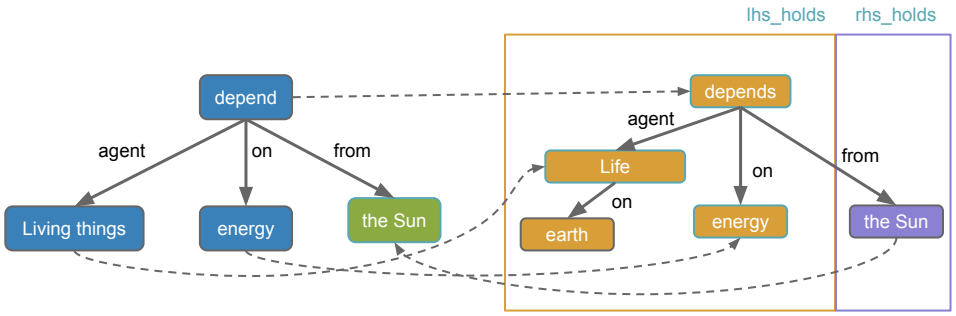

Figure 1: KB rule and question as a graph. *setup* is blue, *query* is green, *antecedent* is orange, and *consequent* is purple. Alignments are shown with dotted lines. *lhsHolds* combines the individual probabilities of *antecedent* nodes.

For example, if the setup mentions "fox", all nodes that entail "fox" also hold. As we also use the edge structure during alignment, we would have a lower probability of "fox" in "fox finds food" to align with "animal" in "animal grows fur" as compared to "animal" in "animal finds food".

We use KB rules to further infer new facts that should hold based on the rule structure. We compute *lhsHolds*, the probability of the rule's *antecedent* holding, and use it to infer *rhsHolds*, the probability of the *consequent*. Similar to ER-MLN, we break the rule into multiple small rules.[1]

$$w : holds(x), inLhs(x, r) \Rightarrow lhsHolds(r)$$
$$w : !holds(x), inLhs(x, r) \Rightarrow !lhsHolds(r)$$
$$lhsHolds(r) \Rightarrow rhsHolds(r).$$
$$rhsHolds(r), inRhs(r, x) \Rightarrow holds(x).$$

Acyclic inference: We use two predicates, $proves(nodeid, nodeid)$ and $ruleProves(rule, rule)$ to capture the inference chain between nodes and rules, resp. We can now ensure acyclicity in inference by introducing transitive clauses over these predicates and disallowing reflexivity, i.e., $!proves(x, x)$. We update rule (2) to:

$$w : proves(x, y), holds(x) \Rightarrow holds(y)$$
$$w : aligns(x, y) \Rightarrow proves(x, y)$$

We capture the direction of inference between rules by checking consequent and antecedent alignments:

$$proves(x, y), inrhs(x, r1), inlhs(y, r2) \Rightarrow ruleProves(r1, r2).$$

False unless proven: To ensure that nodes hold only if they can be proven from *setup*, we add bidirectional implications to our rules. An alternative is to introduce a strong negative prior on *holds* and have a higher positive weight on all other clauses that conclude *holds*. However, the performance of our MLNs was very sensitive to the choice of the weight. We instead model this constraint explicitly. Figure 1 shows a sample inference chain using dotted lines.

Praline defines a meta-inference procedure that is easily modifiable to enforce desired QA inference behavior, e.g. $w : aligns(x, y), setup(x) \Rightarrow !query(y)$ would prevent a term from the *setup* to align with the *query*. Further, by representing the input KB and question as evidence, we can define a single static first-order MLN for all the questions instead of a compiled MLN for every question. This opens up the possibility of learning weights of this static MLN, which would be challenging for the previous two approaches.[2]

[1] An intuitive alternative for the 2nd clause doesn't capture the intending meaning, $-w : !holds(x), inLhs(x, r) \Rightarrow lhsHolds(r)$

[2] In this work, we have set the weights manually.

Empirical Evaluation

We used Tuffy 0.4[3] (Niu et al. 2011) as the base MLN solver[4] and extended it to incorporate the hard-constraint based grounding reduction technique discussed earlier, implemented using the SAT solver Glucose 3.0[5] (Audemard and Simon 2009) exploiting its "solving under assumptions" capability for efficiency. We used a 10 minute timelimit, including a max of 6 minutes for grounding. Marginal inference was performed using MC-SAT (Poon and Domingos 2006), with default parameters and 5000 flips per sample to generate 500 samples for marginal estimation.

We used a 2-core 2.5 GHz Amazon EC2 linux machine with 16 GB RAM. We selected 108 elementary-level science questions (non-diagram, multiple-choice) from 4th grade New York Regents exam as our benchmark (Dev-108) and used another 68 questions as a blind test set (Unseen-68).[6]

The KB, representing roughly 47,000 sentences, was generated in advance by processing the New York Regents 4th grade science exam syllabus, the corresponding Barron's study guide, and documents obtained by querying the Internet for relevant terms. Given a question, we use a simple word-overlap based matching algorithm, referred to as the *rule selector*, to retrieve the top 30 matching sentences to be considered for the question. Textual entailment scores between words and short phrases were computed using WordNet (Miller 1995), and converted to "desired" probabilities for the corresponding soft *entails* evidence. The accuracy reported for each approach is computed as the number of multiple-choice questions it answers correctly, with a partial credit of $1/k$ in case of a k-way tie between the highest scoring options if they include the correct answer.

MLN Formulation Comparison

Table 1 compares the effectiveness of our three MLN formulations, named FO-MLN, ER-MLN, and Praline. For each question and each approach, an MLN program was generated for each of the answer options using the most promising KB rule for that answer option.

In the case of FO-MLN, Tuffy exceeded the 6 minute time limit when generating groundings for 34 of the 108×4 MLNs for the Dev-108 question set, quitting after working with 1.4×10^6 clauses on average, despite starting with only

[3] http://i.stanford.edu/hazy/tuffy

[4] We also tried Alchemy 1.0, which gave similar results.

[5] http://www.labri.fr/perso/lsimon/glucose

[6] Question sets, MLNs, and our modified Tuffy solver are available at `http://allenai.org/software.html`

Table 1: QA performance of various MLN formulations. The number of MLN rules, number of ground clauses, and runtime per multiple-choice question are averages over the corresponding dataset. #Answered column indicates questions where at least one answer option didn't time out (left) and where no answer option timed out (right). #Atoms and #GroundClauses for FO-MLN are averaged over the 398 MLNs where grounding finished; 34 remaining MLNs timed out after processing 1.4M clauses.

Question Set	MLN Formulation	#Answered (some / all)	Exam Score	#MLN Rules	#Atoms	#Ground Clauses	Runtime (all)
Dev-108	FO-MLN	106 / 82	33.6%	35	384*	524*	280 s
	ER-MLN	107 / 107	34.5%	41	284	2,308	188 s
	PRALINE	108	**48.8%**	51	182	219	**17 s**
Unseen-68	FO-MLN	66	33.8%	-	-	-	288 s
	ER-MLN	68	31.3%	-	-	-	226 s
	PRALINE	68	**46.3%**	-	-	-	**17 s**

Table 2: QA performance of Praline MLN variations.

MLN	One rule		Chain=2	
	Dev-108	Unseen	Dev-108	Unseen
Praline	48.8%	46.3%	**50.3%**	**52.7%**
No Acyclic	44.7%	36.0%	43.6%	30.9%
No FUP	35.0%	30.9%	42.1%	29.4%
No FUP, Acyclic	37.3%	34.2%	36.6%	24.3%

around 35 first-order MLN rules. In the remaining MLNs, where our clause reduction technique successfully finished, there is a dramatic reduction in the ground network size: only 524 clauses and 384 atoms on average.

Tuffy finished inference for all 4 answer options for 82 of the 108 questions; for other questions, it chose the most promising answer option among the ones it finished processing. Overall, this resulted in a score of 33.6% with an average of 280 seconds per multiple-choice question on Dev-108, and similar performance on Unseen-68.

ER-MLN, as expected, did not result in any timeouts during grounding. The number of ground clauses here, 2,308 on average, is dominated not by KB rules but by the binary and ternary entity resolution clauses involving the $sameAs$ predicate. ER-MLN was roughly 33% faster than FO-MLN, but overall achieved similar exam scores as FO-MLN.

Praline resulted in a 10x speedup over ER-MLN, explained in part by much smaller ground networks with only 219 clauses on average. Further, it boosted exam performance by roughly 15%, pushing it up to 48.8% on Dev-108 and 46.3% on Unseen-68. This demonstrates the value that the added flexibility and control of Praline bring.

Praline: Improvements and Ablation

We next compare the performance of Praline when using multiple KB rules as a chain or multiple inference paths. Simply using the top two rules for inference turns out to be ineffective as the top two rules provided by the rule selector are often very similar. Instead, we use *incremental inference* where we add one rule, perform inference to determine which additional facts now hold and which $setup$ facts haven't yet been used, and then use this information to select the next best rule. This, as the Chain=2 entries in the first row of Table 2 show, improves Praline's accuracy on both datasets. The improvement comes at the cost of a modest increase in runtime from 17 seconds per question to 38.

Finally, we evaluate the impact Praline's rules for handling acyclicity (Acyclic) and the false-unless-proven (FUP) constraint. Table 2 shows a drop in Praline's accuracy when either of these constraints is removed, which highlights their importance in our QA task. Specifically, when we use only one KB rule, dropping FUP clauses has a larger influence on the score as compared to dropping the Acyclic constraint. Removing Acyclic constraint still causes a small drop even with a single rule due to the possibility of cyclic inference within a rule. When chaining multiple rules, however, cyclic inference becomes more likely and we see a correspondingly larger reduction in score when dropping Acyclic constraints.

Discussion

Our investigation of the potential of MLNs for QA resulted in multiple formulations, the third of which is a flexible model that outperformed other, more natural approaches. We hope our question sets and MLNs will guide further research on improved modeling of the QA task and design of more efficient inference mechanisms for such models.

While SRL methods seem a perfect fit for textual reasoning tasks such as RTE and QA, their performance on these tasks is still not up to par with simple textual feature-based approaches (Beltagy and Mooney 2014). On our datasets too, simple word-overlap based approaches perform quite well, scoring around 55%. We conjecture that the increased flexibility of complex relational models comes at the cost of increased susceptibility to noise in the input. Automatically learning weights of these models may allow leveraging this flexibility in order to handle noise better. Weight learning in these models, however, is challenging as we only observe the correct answer for a question and not intermediate feedback such as ideal alignments and desirable inference chains.

Modeling the QA task with MLNs, an undirected model, gives the flexibility to define a joint model that allows alignment to influence inference and vice versa. At the same time, inference chains themselves need to be acyclic, suggesting that models such as Problog and SLP would be a better fit for this sub-task. Exploring hybrid formulation and designing more efficient and accurate MLNs or other SRL models for the QA task remains an exciting avenue of future research.

Acknowledgments

The authors would like thank Pedro Domingos and Dan Weld for invaluable discussions and the Aristo team at AI2, especially Jesse Kinkead, for help with prototype development and evaluation.

References

Audemard, G., and Simon, L. 2009. Predicting learnt clauses quality in modern SAT solvers. In *21st IJCAI*.

Beltagy, I., and Mooney, R. J. 2014. Efficient Markov logic inference for natural language semantics. In *StarAI*.

Beltagy, I.; Chau, C.; Boleda, G.; Garrette, D.; Erk, K.; and Mooney, R. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. *2nd SEM*.

Beltagy, I.; Erk, K.; and Mooney, R. J. 2014. Probabilistic soft logic for semantic textual similarity. In *52nd ACL*, 1210–1219.

Clark, P.; Balasubramanian, N.; Bhakthavatsalam, S.; Humphreys, K.; Kinkead, J.; Sabharwal, A.; and Tafjord, O. 2014. Automatic construction of inference-supporting knowledge bases. In *4th AKBC*.

Clark, P.; Harrison, P.; and Balasubramanian, N. 2013. A study of the AKBC/requirements for passing an elementary science test. In *AKBC-WEKEX workshop*.

Curran, J.; Clark, S.; and Bos, J. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *45th ACL*, 33–36.

de Salvo Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *19th IJCAI*, 1319–1325.

Domingos, P., and Webb, W. A. 2012. A tractable first-order probabilistic logic. In *26th AAAI*.

Getoor, L., and Taskar, B., eds. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Gogate, V., and Domingos, P. 2011. Probabilistic theorem proving. In *27th UAI*, 256–265.

Kok, S., and Domingos, P. 2008. Extracting semantic networks from text via relational clustering. In *19th ECML*, 624–639.

Miller, G. A. 1995. Wordnet: a lexical database for english. *Communications of the ACM* 38(11):39–41.

Muggleton, S. 1996. Stochastic logic programs. In *ILP*.

Niepert, M., and Van den Broeck, G. 2014. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In *28th AAAI*.

Niu, F.; Ré, C.; Doan, A.; and Shavlik, J. W. 2011. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. In *37th VLDB*, 373–384.

Papai, T.; Singla, P.; and Kautz, H. 2011. Constraint propagation for efficient inference in Markov logic. In *17th CP*. 691–705.

Park, J. D. 2002. MAP complexity results and approximation methods. 388–396.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *21st AAAI*, 458–463.

Poon, H., and Domingos, P. 2009. Unsupervised semantic parsing. In *EMNLP*, 1–10.

Raedt, L. D.; Kimmig, A.; and Toivonen, H. 2007. Problog: A probabilistic Prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence*.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine learning* 62(1–2):107–136.

Sato, T., and Kameya, Y. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *JAIR* 391–454.

Shavlik, J. W., and Natarajan, S. 2009. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *21st IJCAI*, 1951–1956.

Singla, P., and Domingos, P. 2006a. Entity resolution with Markov logic. In *6th ICDM*, 572–582.

Singla, P., and Domingos, P. 2006b. Memory-efficient inference in relational domains. In *21st AAAI*, 488–493.

Venugopal, D., and Gogate, V. 2012. On lifting the gibbs sampling algorithm. In *26th NIPS*, 1664–1672.