Exploring Markov Logic Networks for Question Answering

Tushar Khot[*], **Niranjan Balasubramanian**[+], **Eric Gribkoff**[$], **Ashish Sabharwal**[*], **Peter Clark**[*], **Oren Etzioni**[*]
[*]Allen Institute for AI, [+]Stony Brook University, [$]University of Washington
{tushark,ashishs,peterc,orene}@allenai.org, niranjan@cs.stonybrook.edu, eagribko@cs.washington.edu

Abstract

Elementary-level science exams pose significant knowledge acquisition and reasoning challenges for automatic question answering. We develop a system that reasons with knowledge derived from textbooks, represented in a subset of first-order logic. Automatic extraction, while scalable, often results in knowledge that is incomplete and noisy, motivating use of reasoning mechanisms that handle uncertainty.

Markov Logic Networks (MLNs) seem a natural model for expressing such knowledge, but the exact way of leveraging MLNs is by no means obvious. We investigate three ways of applying MLNs to our task. First, we simply use the extracted science rules directly as MLN clauses and exploit the structure present in hard constraints to improve tractability. Second, we interpret science rules as describing prototypical entities, resulting in a drastically simplified but brittle network. Our third approach, called Praline, uses MLNs to align lexical elements as well as define and control how inference should be performed in this task. Praline demonstrates a 15% accuracy boost and a 10x reduction in runtime as compared to other MLN-based methods, and comparable accuracy to word-based baseline approaches.

1 Introduction

We consider the problem of answering questions in standardized science exams (Clark et al., 2013), which are used as a benchmark for developing knowledge acquisition and reasoning capabilities. The 4th grade science exam dataset from Clark et al. (2013) tests for a wide variety of knowledge and its application to specific scenarios. In particular we focus on a subset that test 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?*).

Answering these questions can be naturally formulated as a reasoning task given the appropriate form of knowledge. Prior work on reasoning based approaches has largely relied on manually input knowledge (Lenat, 1995). We present an investigation of a reasoning approach that operates over knowledge automatically extracted from text.

In order to effectively reason over knowledge derived from text, a QA system must handle incomplete and potentially noisy knowledge, and allow for reasoning under uncertainty. We cast QA as a reasoning problem in weighted-first order logic. While many probabilistic formalisms exist, we use Markov Logic Networks for the ease of specification of weighted rules. MLNs have been adopted for many NLP tasks (Singla and Domingos, 2006a; Kok and Domingos, 2008; Poon and Domingos, 2009). Recently, Beltagy et al. (2013) and Beltagy and Mooney (2014) have shown that MLNs can be used to reason with rules derived from natural language. While MLNs appear to be a natural fit, it is a priori unclear how to effectively formulate the QA task as an MLN problem. We find that unique characteristics of this domain pose new challenges in efficient inference. Moreover, it is unclear how MLNs might perform on automatically extracted noisy rules and how they would fare against simpler baselines that do not rely as much on structured logical representations.

Our goal is to build a high accuracy reasoning-based QA system that can answer a question in near real time. To this end, we investigate three MLN-based formulations: (1) A natural formulation that is intuitive but suffers from inefficient inference (e.g., over 10 minutes on 31% of the questions); (2) an extension that improves efficiency by using prototypical constants, but is brittle to variation in structure; and (3) a formulation with improved flexibility to handle variation in text and structure that is 15% more accurate and 10x faster than the other approaches.

Despite significant improvements over two natural MLN formulations, the best reasoning-based configuration still does not outperform a simpler word-based baseline. We surmise that without effective salience models on text-derived rules, reasoning is unable to leverage the systematic advantages of the MLN-based models. The improved flexibility in the MLN-based models essentially appears to approximate word-based approaches due to the noisy and incomplete nature of the input knowledge. Nevertheless, the reasoning based method shows improved performance when adding multiple rules, which provides a principled way to inject additional knowledge and control inference for further improvements.

2 Background: QA Task

Following Clark et al. (2014), we formulate QA as a reasoning task over knowledge derived from textual sources. A multiple choice question with k answer options is turned into k true-false questions, each of which asserts some known facts (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 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 et al., 2007), we encode information about events (e.g, falling) and entities (e.g., ball or stone) using variables. Predicates such as *agent*, *cause*, *function*, *towards*, and *in* define semantic relationships between 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. For example, "Growing thicker fur in winter helps some animals to stay warm" translates into:

$$\begin{aligned}
&isa(g, \text{grow}), isa(a, \text{some animals}),\\
&isa(f, \text{thicker fur}), isa(w, \text{the winter}),\\
&agent(g, a), object(g, f), in(g, w)\\
&\Rightarrow \exists s, r : isa(s, \text{stays}), isa(r, \text{warm}),\\
&\quad enables(g, s), agent(s, a), object(s, r) \qquad (1)
\end{aligned}$$

Question Representation: The question representation is computed similarly except that we use fixed constants (represented as block letters) rather than variables. For example, 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:

$$\begin{aligned}
setup :& isa(F, \text{fox}), isa(G, \text{grows}), isa(T,\\
& \text{thick fur}), agent(G, F), object(G, T)\\
query :& isa(K, \text{keep warm}), enables(G, K),\\
& agent(K, F)
\end{aligned}$$

Lexical Reasoning: Since entity and event variables hold textual values, reasoning must accommodate lexical variability and textual entailment. For example, the surface forms "thick fur" 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 KB rules and the question as input, we formulate a probabilistic reasoning problem by adding lexical reasoning probabilities and incorporating uncertainties in derived rules. 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 which is known to be #P-hard (Park, 2002). Hence methods such as Integer Linear Programming are not directly applicable.

2.1 Challenges

Reasoning with text-derived knowledge presents challenges that expose the *brittleness* and *rigidity* inherent in pure logic-based frameworks. 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 preconditions are not fully satisfied due to poor alignment. For example, naive matching of rule (1) with the facts in the $setup$ would not conclude the $query$ since 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. For example, "Animals grow thick fur in winter" and "Thick fur helps keep warm" may need to be chained.

3 Probabilistic Formulations

Statistical Relational Learning (SRL) models (Getoor and Taskar, 2007) are a natural fit for QA reasoning. They provide probabilistic semantics over knowledge 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 et al., 2007), and PRISM (Sato and Kameya, 2001), we use Markov Logic Networks (MLNs) for their ease of specification and ability to naturally handle potentially cyclic rules.

Markov Logic Networks (MLNs) are relational models represented using weighted first-order logic rules. The rules provide a template for generating a Markov network by grounding the variables to all the constants in the rules. Each rule f_i forms a clique in the ground network and its weight w_i determines the potential for the clique. Since all cliques generated by grounding the same clause have the same weight, the probability of a given assignment is calculated as:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\mathbf{x})\right)$$

where $n_i(\mathbf{x})$ is the number of times the ith formula is satisfied by the world $\mathbf{x}$ and Z is a normalization constant. Intuitively, a rule with a positive weight is more likely to be true than false.Higher the weight of a rule, more likely it is to be true.

We explore three MLN 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, especially in the presence of differences in 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 sensitive to syntactic differences. In response, PRobabilistic ALignment and INferencE (Praline) performs inference using primarily the string constants but guided by the edge or relational structure. We relax the rigidity in rule application by explicitly modeling the desired QA inference behavior instead of relying on MLN's semantics.

3.1 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. 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.

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

Query: The query atom in M is $\textit{result}()$, a new zero-arity predicate $\textit{result}()$ 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 $\Pr[\textit{result}() = \text{true}]$.

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, $\textit{cause}(x, y) \rightarrow \textit{effect}(y, x)$, 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 $\textit{antecedent}_r$ of each KB rule r, and how much does $\textit{consequent}_r$ entail *query*. These are then added as two additional MLN rules per KB rule.

Our rules have a specific 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 naively translated into a standard form without incurring an exponential blowup. We introduce a new "existential" predicate $E_j^\alpha(x_1, \ldots, x_k, x_{k+j})$ for each existential variable x_{k+j} in each such rule α. This predicate becomes the consequent of α, and hard MLN rules make it equivant to the original consequent.

3.1.1 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 example, 34 out of 108 runs timed out during MLN grounding phase after 6 minutes. On average, the ground networks in these runs were of the order of 1.4×10^6 ground clauses. Such behavior has also been observed, perhaps to a lesser degree, in related NLP tasks such as RTE (Beltagy and Mooney, 2014) and STS (Beltagy et al., 2014).

Existing techniques address large grounding size by focusing on relevant atoms (Singla and Domingos, 2006b; Shavlik and Natarajan, 2009) or grouping atoms into large classes of interchangeable atoms (de Salvo Braz et al., 2005; Gogate and Domingos, 2011; Venugopal and Gogate, 2012). Our QA encoding has very few atoms (often under 500) but very long clauses and highly asymmetric structure. This makes existing methods ineffective. For example, lazy inference in Alchemy-1[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.

We utilize the combinatorial structure imposed by the set H of hard constraints (e.g., semantic rules, definition style rules, some science rules) present in the MLN, and use it to simplify the grounding of *both* hard and soft constraints. Importantly, this does not alter the first-order MLN semantics. The approach thus embraces hard clauses rather than relaxing them, as is often done in probabilistic inference techniques, especially when avoiding infinite energy barriers in MCMC based methods. Assuming an arbitrary constraint ordering 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 control the intermediate grounding size as a propositional variable is no longer generated for a frozen atom. 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. Our approach may also be seen as an extension of a proposal by Papai et al. (2011).

[1]http://alchemy.cs.washington.edu

3.2 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. Despite the drastically reduced number of groundings by our inference approach, the first-order MLN formulation still timed out on 31% of the questions. Hence we 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. 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 define rules 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. For example,

$$agent(Grow, Animals), object(Grow, Fur) \Rightarrow \\ enables(Grow, StayWarm)$$

What was a first-order rule in FO-MLN is now already fully grounded! Entities/events in the question are also similarly represented by constants. Note that the efficiency boost using hard constraints (Section 3.1.1) is orthogonal to using prototypical constants and can be applied here as well.

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. We use a probabilistic $sameAs$ predicate which is reflexive, symmetric, and transitive, and interacts with the rest of the MLN as follows:

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

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. This differs from FO-MLN which uses an external alignment system to find parts of the $antecedent$ mentioned in the $setup$, L' and creates one rule $L' \Rightarrow R$.

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 (e.g. two distinct plants in a question would map to the same entity). Further, it fails to apply valid rules in the presence of syntactic differences such as $agent(\text{Fall}, \text{Things})$ generated by "things fall due to gravity" and $object(\text{Dropped}, \text{Ball})$ 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.

3.3 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) ER-MLN 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.

(2) As entities bound to lexically equivalent strings are forced to "behave" identically, ER-

MLN fails on questions that involve two different entities that are bound to equivalent string representations. Consider the question: "A student puts two identical plants in the same type and amount of soil. She puts one of these plants near a sunny window and the other in a dark room. This experiment tests how the plants respond to (A) light (B) air (C) water (D) soil." The entities corresponding to the two plants will be bound to equivalent string representations and hence will be treated as the same entity. To avoid this, we do not force the entailment-based clusters of constants to behave similarly. Instead, as discussed below, we use the clusters to guide inference in a softer manner.

To introduce such flexibility, we 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. For example, 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 et al., 2007) and PRISM (Sato and Kameya, 2001).

While acyclic inference and false unless proven can be handled by setting high negative priors in MLNs, inference behavior is susceptible to variations in these weights. By using hard rules to control the direction of inference, we can explicitly enforce these constraints.

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. With n string constants, this reduces the number of unobserved groundings from $O(n^2)$ edges in the ER-MLN to $O(n)$ existence predicates. 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 question as a labeled graph G (Figure 1), alignment between string constants corresponds to node alignment 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.

Evidence: 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:

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

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 $holds$:

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

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

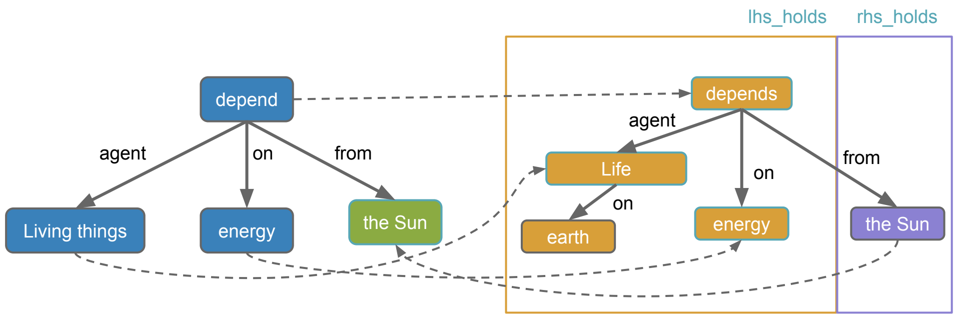

Figure 1: KB rule and question as a graph where blue:$setup$; green:$query$; orange:$antecedent$; purple:$consequent$; dotted lines: alignments. $lhsHolds$ combines individual probabilities of $antecedent$ nodes and $rhsHolds$ captures the probability of the $consequent$.

$antecedent$ holding, and use it to infer $rhsHolds$, the probability of the $consequent$. Similar to ER-MLN, we break the rule into multiple small rules.[2]

$$
\begin{aligned}
w :& holds(x), inLhs(x, r) \Rightarrow lhsHolds(r) \\
w :& !holds(x), inLhs(x, r) \Rightarrow !lhsHolds(r) \\
& lhsHolds(r) \Rightarrow rhsHolds(r). \\
& rhsHolds(r), inRhs(r, x) \Rightarrow holds(x).
\end{aligned}
$$

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

$$
\begin{aligned}
w_p :& proves(x, y), holds(x) \Rightarrow holds(y) \\
w_a :& aligns(x, y) \Rightarrow proves(x, y)
\end{aligned}
$$

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

$$
\begin{aligned}
& proves(x, y), inrhs(x, r1), inlhs(y, r2) \\
& \qquad \Rightarrow ruleProves(r1, r2).
\end{aligned}
$$

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.

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

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.[3]

4 Empirical Evaluation

We used Tuffy 0.4[4] (Niu et al., 2011) as the base MLN solver[5] and extended it to incorporate the hard-constraint based grounding reduction technique discussed earlier, implemented using the SAT solver Glucose 3.0[6] (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 from the same source as a blind test set (Unseen-68)[7].

[3]In this work, we have set the weights manually.
[4]http://i.stanford.edu/hazy/tuffy
[5]Alchemy 1.0 gave similar results.
[6]http://www.labri.fr/perso/lsimon/glucose
[7]http://allenai.org/content/data/Ariscienceexams.txt

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 1: QA performance of various MLN formulations. #MLN-Rules, #GroundClauses, and Runtime per multiple-choice question are averaged 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). Of the 432 Dev MLNs (108×4), #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.

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 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.

4.1 MLN Formulation Comparison

Table 1 compares the effectiveness of our three MLN formulations: FO-MLN, ER-MLN, and Praline. For each question and approach, we generate an MLN program for each answer option 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 around 35 first-order MLN rules. In the remaining MLNs, where our clause reduction technique successfully finished, the ground network size reduced dramatically to 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.

Praline resulted in a 10x speedup over ER-MLN, explained in part by much smaller ground networks with only 219 clauses on average. It boosted exam performance by roughly 15%, pushing it up to 48.8% on Dev-108 and 46.3% on Unseen-68 (statistically significantly better than FO-MLN with p-value < 0.05). This demonstrates the value that the added flexibility and control Praline brings.

4.2 Praline: Improvements and Ablation

We evaluate 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 they 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 runtime increase from 17 seconds per question to 38.

Finally, we evaluate the impact of Praline's additional rules to handle acyclicity (Acyclic) and the false unless proven (FUP) constraint. As Table 2 shows, Praline's accuracy drops upon removing

	One rule		Chain=2	
MLN	Dev-108	Unseen	Dev-108	Unseen
Praline	48.8%	46.3%	**50.3%**	**52.7%**
-Acyclic	44.7%	36.0%	43.6%	30.9%
-FUP	35.0%	30.9%	42.1%	29.4%
-FUP -Acyclic	37.3%	34.2%	36.6%	24.3%

Table 2: QA performance of Praline MLN variations.

	Dev-108	Unseen-68	Dev-170	Unseen-176
Praline	50.3%	**52.7%**	33.2%	36.6%
Word-based	**57.4%**	51.5%	**40.3%**	**43.3%**

Table 3: QA performance: Praline vs. word-based.

either of these constraints, highlighting their importance. Specifically, when using only one KB rule, dropping FUP clauses has a bigger influence that dropping Acyclic constraint clauses. With a single rule, there is still a possibility of cyclic inference within a rule, leading to a small drop in score there as well. When chaining multiple rules, however, the possibility of incorrect cyclic inference is higher and we see a correspondingly larger drop in score when dropping Acyclic constraints.

4.3 Comparison to baseline approaches

Table 3 compares Praline to a baseline word-based method on two question sets. The new set here is from 4th and 5th grade, with 170 Dev and 176 unseen questions. The word-based approach calculates the entailment score, using the same methods as for the soft *entails* evidence earlier, between the words in the T/F question and words in a rule in the KB. It then uses the maximum entailment score from all selected rules as the confidence measure i.e. $\max_{r\in R} \mathit{entailment}(q, r)$. While the scores of the two approaches are statistically similar (p-value > 0.05), the simple word-based approach does have a slight edge over Praline. Automatic extraction of knowledge from text provides additional information (e.g., rule structure) that MLNs are capable of exploiting. However, we found this additional flexibility to not pay off with the current knowledge-base and questions.

5 Conclusions

Reasoning with automatically extracted knowledge presents significant challenges. Our investigation of MLN-based formulations for elementary-level science exams highlights two key issues: 1) Natural translations of text derived knowledge into first-order representations are highly inefficient, resulting in large ground networks. 2) When the logical elements in the rules largely mirror the constructs in the source text, reasoning is hampered because of structural variability. In response, we proposed, Praline, an alignment based solution that is both efficient and accurate. Praline reasons with prototypical constants, and provides greater flexibility in how inference is performed and is therefore more robust to structural mismatches.

MLNs provided a flexible, structured framework to define inference for the QA task, while also providing reasoning chains used to arrive at an answer. While models such as MLNs 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 textual feature-based approaches (Beltagy and Mooney, 2014). We conjecture that the increased flexibility of complex relational models results in increased susceptibility to noisy input, and the systematic advantages of MLN models are difficult to exploit with text-derived rules. Automatically learning weights of these models may allow leveraging their flexibility to address these issues, but weight learning remains challenging with only distant supervision.

We hope our datasets, knowledge bases, and MLN models[8] will help push NLP and SRL communities towards designing improved structured reasoning QA systems.

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

Gilles Audemard and Laurent Simon. 2009. Predicting learnt clauses quality in modern SAT solvers. In *21st IJCAI*, pages 399–404, Pasadena, CA, July.

Islam Beltagy and Raymond J Mooney. 2014. Efficient Markov logic inference for natural language semantics. Quebec City, Canada, July.

Islam Beltagy, Cuong Chau, Gemma Boleda, Dan Garrette, Katrin Erk, and Raymond Mooney. 2013. Montague meets Markov: Deep semantics with probabilistic logical form. *2nd Joint Conference on*

[8]Available at http://allenai.org/software.html

Lexical and Computational Semantics: Proceeding of the Main Conference and the Shared Task, Atlanta, pages 11–21.

Islam Beltagy, Katrin Erk, and Raymond J. Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *52nd ACL*, pages 1210–1219, Baltimore, MD, June.

Peter Clark, Phil Harrison, and Niranjan Balasubramanian. 2013. A study of the AKBC/requirements for passing an elementary science test. In *Proc. of the AKBC-WEKEX workshop at CIKM*.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. In *4th Workshop on Automated Knowledge Base Construction (AKBC)*, Montreal, Canada, December.

James Curran, Stephen Clark, and Johan Bos. 2007. Linguistically motivated large-scale NLP with C&C and Boxer. In *45th ACL*, pages 33–36, Prague, Czech Republic, June.

Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. 2005. Lifted first-order probabilistic inference. In *19th IJCAI*, pages 1319–1325.

Lise Getoor and Ben Taskar, editors. 2007. *Introduction to Statistical Relational Learning*. MIT Press.

Vibhav Gogate and Pedro Domingos. 2011. Probabilistic theorem proving. pages 256–265, Barcelona, Spain, July.

Stanley Kok and Pedro Domingos. 2008. Extracting semantic networks from text via relational clustering. In *19th ECML*, pages 624–639, Antwerp, Belgium, September.

Douglas Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications ACM*, 38(11):33–38.

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

Stephen Muggleton. 1996. Stochastic logic programs. In *Advances in Inductive Logic Programming*.

Feng Niu, Christopher Ré, AnHai Doan, and Jude W. Shavlik. 2011. Tuffy: Scaling up statistical inference in Markov Logic Networks using an RDBMS. In *37th VLDB*, pages 373–384.

Tivadar Papai, Parag Singla, and Henry Kautz. 2011. Constraint propagation for efficient inference in Markov logic. In *17th CP*, pages 691–705. Springer, Perugia, Italy.

James Park. 2002. MAP complexity results and approximation methods. pages 388–396, Edmonton, Canada, August.

Hoifung Poon and Pedro Domingos. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *21st AAAI*, pages 458–463, Boston, MA.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *EMNLP*, pages 1–10.

Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. 2007. Problog: A probabilistic Prolog and its application in link discovery. In *International Joint Conference on Artificial Intelligence*.

Taisuke Sato and Yoshitaka Kameya. 2001. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, pages 391–454.

Jude Shavlik and Sriraam Natarajan. 2009. Speeding up inference in Markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *21st IJCAI*, pages 1951–1956, Pasadena, CA.

Parag Singla and Pedro Domingos. 2006a. Entity resolution with Markov logic. In *6th ICDM*, pages 572–582, Hong Kong, China, December.

Parag Singla and Pedro Domingos. 2006b. Memory-efficient inference in relational domains. In *21st AAAI*, pages 488–493, Boston, MA.

Deepak Venugopal and Vibhav Gogate. 2012. On lifting the gibbs sampling algorithm. pages 1664–1672, Lake Tahoe, NV, December.