Learning Rules from Incomplete Examples via a Probabilistic Mention Model

Mohammad S. Sorower, Thomas G. Dietterich, Janardhan Rao Doppa, Prasad Tadepalli, and Xiaoli Fern
School of EECS, Oregon State University
Corvallis, OR 97331 USA
{sorower,tgd,doppa,tadepall,xfern}@cs.orst.edu

Abstract

We consider the problem of learning rules from natural language text sources. These sources, such as news articles, journal articles, and web texts, are created by a writer to communicate information to a reader, where the writer and reader share substantial domain knowledge. Consequently, the texts tend to be concise and mention the minimum information necessary for the reader to draw the correct conclusions. We study the problem of learning domain knowledge from such concise texts, which is an instance of the general problem of learning in the presence of missing data. However, unlike standard approaches to missing data, in this setting we know that facts are more likely to be missing from the text in cases where the reader can infer them from the facts that are mentioned combined with the domain knowledge. Hence, we can explicitly model this "missingness" process and invert it via probabilistic inference to learn the underlying domain knowledge. This paper introduces an explicit *probabilistic mention model* that models the probability of facts being mentioned in the text based on what other facts have already been mentioned and domain knowledge in the form of Horn clause rules. Learning must simultaneously search the space of rules and learn the parameters of the mention model. We accomplish this via an application of Expectation Maximization within a Markov Logic framework. An experimental evaluation on synthetic and natural text data shows that the method can successfully learn accurate rules and apply them to new texts to make correct inferences.

1 Introduction

The immense volume of textual information available on the web provides an important opportunity and challenge for AI: Can we develop methods that can learn domain knowledge by reading natural texts such as news articles, journal articles, and web pages. We would like to acquire at least two kinds of domain knowledge: concrete facts and general rules. Concrete facts can be extracted as logical relations or as tuples to populate a data base. Systems such as Whirl [Cohen, 2000], TextRunner [Etzioni *et al.*, 2008], and NELL [Carlson *et al.*, 2010a] learn extraction patterns that can be applied to text to extract instances of relations.

General rules can be acquired in two ways. First, they may be stated explicitly in the text—particularly in tutorial texts. Second, they can be acquired by generalizing from the extracted concrete facts. In this paper, we focus on the latter setting: Given an incomplete data base of literals extracted from natural language texts, we seek to learn a set of probabilistic Horn clauses that capture general rules.

The problem of learning rules from extracted texts has been studied previously [Nahm and Mooney, 2000; Carlson *et al.*, 2010b; Schoenmackers *et al.*, 2010]. These systems rely on finding documents in which all of the facts participating in a rule are mentioned. If enough such documents can be found, then standard rule learning algorithms can be applied. A drawback of this approach is that it is difficult to learn rules unless there are many documents that provide such complete training examples. The central hypothesis of our work is that by explicitly modeling the process by which facts are mentioned, we can learn rules from sets of documents that are smaller and less complete.

To illustrate the challenges, consider the following sentence that discusses a National Football League (NFL) game:

"Given the commanding lead of Kansas city on the road, Denver Broncos' 14-10 victory surprised many"

This mentions that Kansas City is the away team and that the Denver Broncos won the game, but does not mention that Kansas City lost the game or that the Denver Broncos was the home team. Of course these facts can be inferred from domain knowledge rules such as the rule that "if one team is the winner, the other is the loser (and vice versa)" and the rule "if one team is the home team, the other is the away team (and vice versa)".

In our previous work [Doppa *et al.*, 2010], we employed an "implicit mention model" that did not explicitly represent the way in which mentions arise. Instead, it simply modified the way candidate rules were scored. The current paper builds on that work to show how an explicit mention model can be learned and applied.

2 Technical Approach

Consider a writer and a reader who share domain knowledge K. The writer wishes to efficiently communicate the conjunction of two (true) formulas $F \wedge G$.

Let $\text{READERWILLINFER}(G, F, K)$ denote the fact that the reader, when told F, will infer that G is true by applying the domain knowledge K. Then it suffices for the writer to mention only F and not G. We can express this as the general rule schema:

$$\text{MENTION}(F) \wedge \text{READERWILLINFER}(G, F, K) \wedge G \Rightarrow \neg\text{MENTION}(G).$$

Similarly, if the writer wishes to communicate the formula $F \wedge \neg G$, which is an exception to the general rule that $F \Rightarrow G$, then the writer must explicitly mention that G is not true:

$$\text{MENTION}(F) \wedge \text{READERWILLINFER}(G, F, K) \wedge \neg G \Rightarrow \text{MENTION}(\neg G).$$

In this paper, we consider only the first case. In addition, rather than explicitly modeling the reader's inference process, we instantiate the rule schema for each domain knowledge rule under the assumption that the reader's reasoning process can perform simple Horn clause chaining. Hence, given the rule $P \Rightarrow Q$, we define the mention rule

$$\text{MENTION}(P) \Rightarrow \neg\text{MENTION}(Q).$$

To represent and reason with the domain knowledge, the mention rules, and the observed mentions, we employ Markov Logic. There are four kinds of rules in our knowledge base (see Figure 1):

- **Fact-to-Fact Rules**. These are the unknown domain knowledge rules that we seek to learn. We prefix each literal with FACT_ and write the rules in the form

  $$\text{FACT\_HOMETEAM}(g, t2) \wedge \text{FACT\_TEAMINGAME}(g, t1) \wedge \text{FACT\_TEAMINGAME}(g, t2) \Rightarrow \text{FACT\_AWAYTEAM}(g, t1).$$

  $\text{TEAMINGAME}(g, t)$ states that team t was one of the teams playing in game g. $\text{HOMETEAM}(g, t)$ states that team t was the home team in game g. Similarly, $\text{AWAYTEAM}(g, t)$ indicates that t was the away team in game g. (We apply the unique names constraint that variables with unique names must refer to distinct objects; hence $t1 \neq t2$ is assumed implicitly.)

- **Fact-to-Mention Rules**. These express the belief that if something is true, it is likely to be mentioned. These rules are probabilistic, and during learning they are assigned weights that reflect the baseline probability that any particular literal will be mentioned.

  $$\text{FACT\_HOMETEAM}(g, t) \Rightarrow \text{MENTION\_HOMETEAM}(g, t).$$

- **Mention-to-Fact Rules**. These express the belief that the writer is not lying, so anything that is mentioned is true:

  $$\text{MENTION\_HOMETEAM}(g, t) \Rightarrow \text{FACT\_HOMETEAM}(g, t).$$

  Weights for these rules are not learned. Instead, the rules are assumed to hold with probability 1 (i.e., have infinite weight).

- **Mention-to-Mention Rules**. These rules instantiate the general mention rule schemas described above.

  $$\text{MENTION\_HOMETEAM}(g, t2) \wedge \text{MENTION\_TEAMINGAME}(g, t1) \wedge \text{MENTION\_TEAMINGAME}(g, t2) \Rightarrow \neg\,\text{MENTION\_AWAYTEAM}(g, t1).$$

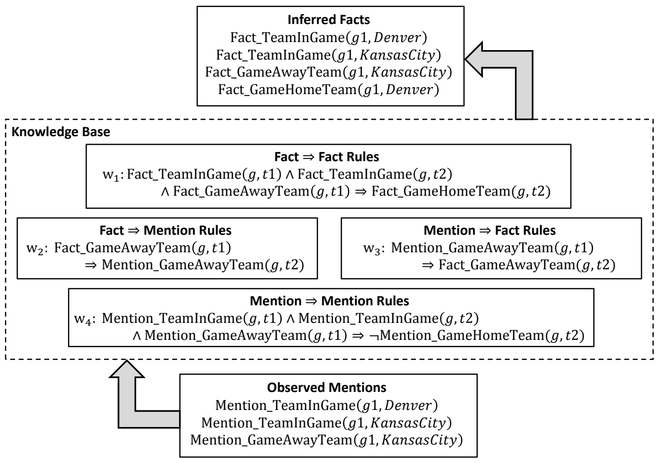
Inferred Facts
Fact_TeamInGame($g1$, $Denver$)
Fact_TeamInGame($g1$, $KansasCity$)
Fact_GameAwayTeam($g1$, $KansasCity$)
Fact_GameHomeTeam($g1$, $Denver$)
Knowledge Base
Fact ⇒ Fact Rules
w_1: Fact_TeamInGame(g, $t1$) ∧ Fact_TeamInGame(g, $t2$) ∧ Fact_GameAwayTeam(g, $t1$) ⇒ Fact_GameHomeTeam(g, $t2$)
Fact ⇒ Mention Rules
w_2: Fact_GameAwayTeam(g, $t1$) ⇒ Mention_GameAwayTeam(g, $t2$)
Mention ⇒ Fact Rules
w_3: Mention_GameAwayTeam(g, $t1$) ⇒ Fact_GameAwayTeam(g, $t2$)
Mention ⇒ Mention Rules
w_4: Mention_TeamInGame(g, $t1$) ∧ Mention_TeamInGame(g, $t2$) ∧ Mention_GameAwayTeam(g, $t1$) ⇒ ¬Mention_GameHomeTeam(g, $t2$)
Observed Mentions
Mention_TeamInGame($g1$, $Denver$)
Mention_TeamInGame($g1$, $KansasCity$)
Mention_GameAwayTeam($g1$, $KansasCity$)

Figure 1: Markov Logic Model

In Markov Logic, each of these rules is assigned a (learned) weight which can be viewed as a cost of violating the rule. The probability of a world ω is proportional to

$$\exp\left(\sum_j w_j I[\text{Rule } j \text{ is satisfied by } \omega]\right),$$

where j iterates over all groundings of the Markov logic rules in world ω and $I[\phi]$ is 1 if ϕ is true and 0 otherwise.

An advantage of Markov Logic is that it allows us to define a probabilistic model even when there are contradictions and cycles in the logical rules. Hence, we can include both a rule that says "if the home team is mentioned, then the away team is not mentioned" and a rule that says "if the away team is mentioned, then the home team is not mentioned". We can also include rules that say "the home team is mentioned" and "the away team is mentioned". Obviously a possible world ω cannot satisfy all of these rules. The relative weights on these rules determine the probability that particular literals are actually mentioned.

Learning. We seek to learn both the rules and their weights. We proceed by first learning the fact-to-fact rules and then automatically generating the other rules (especially the mention-to-mention rules) from the general rule schema described above. Then we apply EM to learn the weights on all of the rules. This can have the effect of removing unnecessary rules by driving their weights to zero.

Learning Fact-to-Fact Rules. For each predicate, we generate a set of candidate Horn clauses with that predicate as the head. We consider all conjunctions of literals involving other predicates (i.e., we do not allow recursive rules) up to a fixed maximum length. Each candidate rule is scored on the training documents for *support* (number of training examples that satisfy the body) and *confidence* (the conditional probability that the head is true given that the body is satisfied). We discard all rules that do not achieve minimum support σ and then keep the top τ most confident rules. The values of σ and τ are determined via cross-validation within the training set. The selected rules are then entered into the knowledge base.

From each rule, we also derive a mention-to-mention rule that says if all of the literals in the body have been mentioned, then the head will not be mentioned. For each predicate, we also generate fact-to-mention and mention-to-fact rules.

Learning Weights. The goal of weight learning is to maximize the likelihood of the observed mentions (in the training set) by adjusting the weights of the rules. Because our training data only consists of mentions and no facts, the facts are latent (hidden variables), and we must apply the EM algorithm to learn the weights. Each game can be summarized by four literals: $\textsc{homeTeam}(g, t1)$, $\textsc{awayTeam}(g, t2)$, $\textsc{gameWinner}(g, t3)$, and $\textsc{gameLoser}(g, t4)$. For purposes of weight learning, we must consider these four literals together, so we introduce an auxiliary predicate $\text{GT}\textsc{uple}(g, t1, t2, t3, t4)$ and introduce the hard rule

$$\begin{aligned}&\text{GT}\textsc{uple}(g, t1, t2, t3, t4) \equiv \\ &\quad \textsc{homeTeam}(g, t1) \wedge \textsc{awayTeam}(g, t2) \wedge \\ &\quad \textsc{gameWinner}(g, t3) \wedge \textsc{gameLoser}(g, t4).\end{aligned}$$

Given two teams, there are 16 possible instantiations of this literal in each game. Let us index them from 0 to 15 corresponding to $\text{GT}\textsc{uple}(g, A, A, A, A)$ through $\text{GT}\textsc{uple}(g, B, B, B, B)$.

We employ the Markov Logic system Alchemy [Kok *et al.*, 2007] for learning and inference. In the E step, we apply the lazy MCSAT inference algorithm to estimate the marginal probability of each of the 16 possible instantiations of $\text{GT}\textsc{uple}$. In the M step, we then treat these possible instantiations as weighted evidence and apply generative learning to find the weight values that maximize the pseudo-likelihood of the weighted evidence. To introduce weighted evidence into Alchemy, we apply the following "dummy evidence trick". Specifically, suppose we wish to introduce the evidence $\text{GT}\textsc{uple}(g1, A, B, A, B)$ (configuration 5) with probability p_5, for teams A and B. We define an evidence literal $\text{GE}\textsc{vidence}(g1, A, B, A, B)$ in Alchemy's database file, and then add to the "mln" file a rule $\text{GE}\textsc{vidence}(g1, A, B, A, B) \equiv \text{GT}\textsc{uple}(g1, A, B, A, B)$ with a weight of

$$w_5 = \log p_5 - \frac{1}{K}\sum_{k=1}^{K} \log p_k,$$

where p_k is the marginal probability of configuration k. To avoid an explosion of instantiations during inference in Alchemy, we arbitrarily rename the teams in all of the games to be the constants A and B. This is a hand-coded form of lifted inference.

EM is iterated to convergence, which only requires a few iterations. Algorithm 1 summarizes the pseudo-code of the algorithm.

Treating Missing Mentions as Missing At Random: An alternative to the explicit mention model described above is to assume that the writer choose which facts to mention (or omit) at random according to some unknown probability. When data are missing-at-random (MAR), it is possible to obtain unbiased estimates of the true distribution via imputation using EM. We implemented this approach as follows. we apply the same method of learning rules (requiring minimum support σ and then taking the τ most confident rules).

Algorithm 1 Learn Explicit Mention Model

Input: $\mathcal{D}_I$ =Incomplete training examples
τ = number of rules per head
σ = minimum support per rule
Output: $\mathcal{M}$ = Explicit mention model

```
 1: LEARN MENTION MODEL:
 2: exhaustively learn rules for each head
 3: discard rules with less than $\sigma$ support
 4: select the $\tau$ most confident rules $\mathcal{R}$ for each head
 5: $\mathcal{R}' := \mathcal{R}$
 6: for each rule $(factP => factQ) \in \mathcal{R}$ do
 7:     $\mathcal{R}' := \mathcal{R}' \cup \{mentionP \Rightarrow \neg mentionQ\}$
 8: end for
 9: for every $factP \in \mathcal{R}$ do
10:     add $factP \Rightarrow mentionP$ to $\mathcal{R}'$
11:     add $mentionP \Rightarrow factP$ to $\mathcal{R}'$
12: end for
13: repeat
14:     E-Step: apply inference to predict weighted facts $\mathcal{F}$
15:     define complete weighted data $\mathcal{D}_C := \mathcal{D}_I \cup \mathcal{F}$
16:     M-Step: learn weights for rules in $\mathcal{R}'$ using data $\mathcal{D}_C$
17: until convergence
18: return the set of weighted rules $\mathcal{R}'$
```

Each learned rule has the general form $\text{M}\textsc{ention\_A} \Rightarrow \text{M}\textsc{ention\_B}$. The collection of rules is treated as a model of the joint distribution over the mentions (or, more precisely, over the possible groundings of $\text{GT}\textsc{uple}$). Generative weight learning in then applied to learn the weights on these rules. The marginal probabilities of $\text{GT}\textsc{uple}$ are computed using the rule weights.

3 Experimental Evaluation

We evaluated our mention model approach on a synthetic data set to understand its behavior as we varied the amount of missing data. Then we compared its performance to the MAR approach on actual extractions from news stories about NFL football games.

Synthetic Data Experiment. The goal of this experiment was to evaluate the ability of our method to learn accurate rules from data that match the assumptions of the algorithm. We also sought to understand how performance varied as a function of the amount of information omitted from the text.

The synthetic data were generated using a database of NFL games (from 1998 and 2000-2005) downloaded from `www.databasefootball.com`. These data were then encoded using the following predicates: $\textsc{teamInGame}(Game, Team)$, $\textsc{gameWinner}(Game, Team)$, $\textsc{gameLoser}(Game, Team)$, $\textsc{homeTeam}(Game, Team)$, $\textsc{awayTeam}(Game, Team)$, and $\textsc{teamGameScore}(Game, Team, Score)$, and treated as the ground truth. Note that these predicates can be divided into two correlated sets: $WL = \{\textsc{gameWinner}, \textsc{gameLoser}, \textsc{teamGameScore}\}$ and $HA = \{\textsc{homeTeam}, \textsc{awayTeam}\}$.

From this ground truth, we generate a set of mentions for

Table 1: Synthetic Data Properties

q	**0.17**	**0.33**	**0.50**	**0.67**	**0.83**	**0.97**
Mentioned literals(%)	91.38	80.74	68.72	63.51	51.70	42.13
Complete records(%)	61.70	30.64	8.51	5.53	0.43	0.00

Table 2: Probabilistic Mention Model Performance on Synthetic Data. Each cell indicates the fraction of complete records inferred.

	Test q					
Training q	**0.17**	**0.33**	**0.50**	**0.67**	**0.83**	**0.97**
0.17	1.00	1.00	1.00	1.00	1.00	1.00
0.33	1.00	0.99	0.97	0.96	0.90	0.85
0.50	1.00	0.99	0.98	0.97	0.93	0.87
0.67	1.00	0.98	0.92	0.92	0.81	0.66
0.83	0.99	0.98	0.72	0.71	0.61	0.54
0.97	0.91	0.81	0.72	0.68	0.56	0.41

each game as follows. One literal is chosen uniformly at random from each of WL and HA and mentioned. Then each of the remaining literals is mentioned with probability $1 - q$, where q is a parameter that we varied in the experiments. Table 1 shows the average percentage of literals mentioned in each generated "news story" and the percentage of generated "news stories" that mentioned all literals. Note that the stories generated in this way reflect the actual statistics of game scores and outcomes.

Synthetic Experiments. For each q, we generated 5 different datasets, each containing 235 games. For each value of q, we ran the algorithm five times. In each iteration, one dataset was used for training, another for validation, and the remaining 3 for testing. The training and validation datasets shared the same value of q. The resulting learned rules were evaluated on the test sets for all of the different values of q. The validation set is employed to determine the thresholds τ and σ during rule learning and to decide when to terminate EM.

Table 2 reports the proportion of complete game records (i.e., all four literals) that were correctly inferred, averaged over the five runs. Note that any facts mentioned in the generated articles are automatically correctly inferred, so if no inference was performed at all, the results would match the second row of Table 1. Notice that when trained on data with low missingness (e.g. $q = 0.17$), the algorithm was able to learn rules that predict well over articles with much higher levels of missing values. This is because $q = 0.17$ means that only 8.62% of the literals are missing in the training dataset, which results in 61.70% complete records. These are sufficient to allow learning highly-accurate rules. However, as the proportion of missing literals in training data increases, the algorithm starts learning incorrect rules, so performance drops. In particular, when $q = 0.97$, the training documents contain *no* complete records (Table 1). Nonetheless, the learned rules

Table 3: Fraction of Literals Correctly Predicted for $q = 0.97$

	Test q					
Training q	**0.17**	**0.33**	**0.50**	**0.67**	**0.83**	**0.97**
0.97	0.98	0.95	0.93	0.92	0.89	0.85

are still able to completely and correctly reconstruct 41% of the games!

The rules learned under such high levels of missingness are not totally correct. Here is an example of one learned rule (for $q = 0.97$):

$$\textsc{Fact\_homeTeam}(g, t1) \wedge \textsc{Fact\_teamInGame}(g, t1) \Rightarrow \textsc{Fact\_gameWinner}(g, t1).$$

This rule says that the home team always wins. When appropriately weighted in Markov Logic, this is a reasonable rule even though it is not perfectly correct (nor was it a rule that we applied during the synthetic data generation process).

In addition to measuring the fraction of entire games correctly inferred, we can obtain a more fine-grained assessment by measuring the fraction of individual literals correctly inferred. Table 3 shows this for the $q = 0.97$ training scenario. We can see that even when the test articles have $q = 0.97$ (which means only 42.13% of literals are mentioned), the learned rules are able to correctly infer 85% of the literals. By comparison, if the literals had been predicted independently at random, only 6.25% would be correctly predicted.

Experiments with Real Data: BBN provided the output of their extractor applied to two corpora of articles describing NFL games. The first corpus of articles is known as `BBN_training`. We will refer to it as D1. The second corpus is known as `BBN_robustness`. The articles in the second corpus were subjected to more thorough analysis, so the extractions are more complete. However, both D2 and (especially) D1 contain many errors including games involving more than two teams or where one team achieved multiple scores. These errors result primarily from coreference failures.

To address these problems, we decided to take each extracted game and apply a set of integrity constraints. The integrity constraints are learned from 5 complete game records. Examples of the learned constraints include "Every game has exactly two teams" and "Every game has exactly one winner." Each BBN-extracted game is then converted into multiple games by deleting literals in all possible ways until all of the integrity constraints are satisfied. Finally, duplicate records are removed. After this processing, D1 contains 203 games and D2 contains 56 games. Table 4 summarizes the missingness of these game records.

The data from these repaired games is provided as training data to our algorithm. To measure the performance of the learned rules, we constructed a synthetic test set by taking 100 games from the downloaded NFL database and creating games in which various patterns of missingness were exhibited. Specifically, none of the test games mentioned both the home and away team or both the winning and losing team. In 20% of the test games, neither the home nor the away team was mentioned, whereas all test games mentioned exactly one

Table 4: Statistics on BBN-extracted and repaired games. "miss" means "missing".

	Home/Away			Winner/Loser		
	miss both	miss one	miss none	miss both	miss one	miss none
D1	85.7	11.3	3.0	14.8	49.2	36.0
D2	17.9	58.9	23.2	17.9	57.1	25.0
Test	20.0	80.0	0.0	0.0	100.0	0.0

Table 5: Observed percentage of cases where exactly one literal is mentioned and the percentage predicted if the literals were missing at random

	Home/Away		Winner/Loser	
	observed	predicted	observed	predicted
D1	11.3	13.2	49.2	47.1
D2	58.9	49.9	57.1	49.8

of either the winner or the loser (see Table 4).

It is interesting to ask whether these data are consistent with the explicit mention model versus the missing-at-random model. Let us suppose that under MAR, the probability that a fact will be mentioned is p. Then the probability that both literals in a rule (e.g., home/away or winner/loser) will be mentioned is p^2, the probability that both will be missing is $(1-p)^2$, and the probability that exactly one will be mentioned is $2p(1-p)$. We can fit the best value for p to the observed missingness rates to minimize the KL divergence between the predicted and observed distributions. If the explicit mention model is correct, then the MAR fit will underestimate the fraction of cases where exactly one literal is missing. Table 5 shows the results. For D1, the MAR model gives a fairly good fit to the observed missingness rates. However, for D2, it is clear that the MAR model seriously underestimates the probability that exactly one literal will be mentioned. Qualitatively, we believe that much of the missingness in D1 is due to extraction failures, whereas the missingness in D2 corresponds better to the assumptions of our probabilistic mention model.

We applied both our explicit mention model and the MAR model to these data sets. We measured performance relative to the performance that could be attained by a system that knows the correct rules. The results are summarized in Table 6. When trained on D1, the MAR method performs much better than our method, although both methods perform poorly. The challenge for both methods is that there are very few articles that provide complete training examples of the rules. When trained on D2, our method achieves perfect performance, whereas the MAR method only reconstructs 6% of the reconstructable games. This reflects the extreme difficulty of the test set, where none of the articles mentions all literals involved in any rule.

Table 6: Test set performance of our explicit mention model compared to the missing-at-random approach. Performance is measured as the number of games that are completely and correctly reconstructed by the learned models as a percentage of the number of games that can be completely reconstructed by the correct model.

Training Dataset	Explicit Mention Model	Missing at Random Model
D1	10.0	25.6
D2	100.0	6.0

4 Conclusion

This paper has shown how to apply Markov Logic to represent a probabilistic model of the relationship between the true facts of a situation (such as an NFL game) and the propositions that are mentioned in an article about the situation. Experiments on synthetic data showed that the method is able to correctly reconstruct complete records even when neither the training data nor the test data contain complete records. On real data, the method achieves excellent performance when trained on the D2 dataset, whereas a method that treats facts as mentioned-at-random gives very poor performance. On dataset D1, performance was poor for both methods. This may reflect a fundamental lack of information sufficient to permit rule discovery, or it may reflect a weakness in our rule learning algorithm. It would be interesting to explore a modification of our algorithm in which candidate rules are scored within the Markov Logic mention model rather than relying on a separate rule-learning phase. This might permit the discovery of rules without observing *any* cases where both the head and the body are mentioned in a single document.

Acknowledgments

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-09-C-0179. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the DARPA, the Air Force Research Laboratory (AFRL), or the US government.

References

[Carlson *et al.*, 2010a] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 1306–1313. AAAI Press, 2010.

[Carlson *et al.*, 2010b] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 101–110, New York, NY, USA, 2010. ACM.

[Cohen, 2000] William W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118(1-2):163–196, 2000.

[Doppa *et al.*, 2010] Janardhan Rao Doppa, Mohammad NasrEsfahani, Mohammad S. Sorower, Thomas G. Dietterich, Xiaoli Fern, and Prasad Tadepalli. Towards learning rules from natural texts. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, FAM-LbR '10, pages 70–77, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[Etzioni *et al.*, 2008] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, 2008.

[Kok *et al.*, 2007] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. The Alchemy system for statistical relational AI. *Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA*, 2007.

[Nahm and Mooney, 2000] Un Yong Nahm and Raymond J. Mooney. A mutually beneficial integration of data mining and information extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 627–632. AAAI Press, 2000.

[Schoenmackers *et al.*, 2010] Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. Learning first-order Horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1088–1098, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.