

## Abstract

We describe a framework for linking together a structured ontology, deductive logic, and probability, to solve classification problems. We illustrate this framework with the *Whodunit problem*: identifying the perpetrator of a crime. Several experiments show that the use of Cyc's ontology and inference abilities substantially improves classification accuracy, both in decision tree classifiers and with Markov Logic Networks.

---

# Cyc-Enhanced Machine Classification

---

Michael Witbrock, Elizabeth  
Coppock, Robert Kahlert, and  
Benjamin Rode\*

---

Cycorp Technical Report 09-001, August 18, 2009

Copyright ©2009 Cycorp, Inc. All rights reserved.

\*We gratefully acknowledge assistance from Pedro Domingos and Marc Sumner at the University of Washington. This work was partly supported under DARPA contract HR0011-06-C-0025.

## Introduction

In this report, we show how Cyc's ontology and reasoning engine can aid the construction of probabilistic classifiers, and more broadly, how commonsense knowledge and deductive reasoning aids in solving problems that require computing likelihood estimates for probability-ranked possible answers. We have placed particular emphasis on the problem of predicting, for a given terrorism event, who the perpetrator was (the "Whodunit problem"; Halstead and Forbus, 2007). The initial structure of the models used in inference, and the training data used to train these models, are drawn from historical and ongoing terrorism-related content already entered in ResearchCyc and in the existing large-scale Cyc Analysts' Knowledge Base ("AKB", whose terrorism-related content is known as the Cyc Terrorism KB or "TKB"), and which has been extended and augmented most recently under the DARPA IPTO contract #HR001-06-C-0025). In collaboration with the University of Washington, we have formulated problems within the framework of Markov Logic, by constructing Markov Logic Networks that can represent Cyc KB content, can be trained to make statistical generalizations over it, and can be used to perform inference with those generalizations. For the sake of comparison, and to explore other alternatives, we have also developed techniques for exporting KB content in a representation suitable for applying other more standard machine learning techniques, such as J48 decision trees. As a further step, we have worked toward reincorporating the statistical models produced by Markov Logic and J48 decision trees into the KB as probabilistic knowledge.

## Event location

One feature that can be used to identify the perpetrator of an event is the location of the event. The Analyst's Knowledge Base (Deaton et al, 2005) contains location information about terrorism events expressed in the form of `eventOccursAt` assertions, for example:

```
(eventOccursAt TerroristAttack-September-15-1982-Madrid-Spain
  Madrid-AutonomousCommunitySpain)
```

From such an assertion, Cyc can conclude that the event occurred in the country of Spain, although this is not directly asserted. That is, Cyc can infer:

```
(eventOccursAt TerroristAttack-September-15-1982-Madrid-Spain Spain)
```

This conclusion is based on the knowledge that `Madrid-AutonomousCommunitySpain` is geographically subsumed by `Spain`, along with the relationship between location and geographical subsumption:

```
(geographicallySubsumes Madrid-AutonomousCommunitySpain Spain)
```

```
(implies
  (and (eventOccursAt ?E ?L1)
        (geographicallySubsumes ?L1 ?L2))
  (eventOccursAt ?E ?L2))
```

It should be apparent that geographical subsumption reasoning is useful in the construction of a probabilistic model of perpetrator identification. A model that lacks knowledge of

geographical subsumption fails to generalize to new cases when the asserted location is very specific. In many cases, the asserted location is unique to a particular event. On the other hand, using geographical subsumption, it is possible to group events together, and identify patterns that would not otherwise have been possible to identify. With this in mind, we hypothesized that the use of geographical subsumption information can boost the predictive accuracy of a probabilistic model, and as we discuss in the results section, this hypothesis was confirmed. In order to test this hypothesis, it was necessary to lay the groundwork for probabilistic inference using data from Cyc; we describe how this was achieved in the following section.

## Methods

The overall AKB data set has over 4500 terrorist events that belong to over 2000 classes and were performed by over 1000 agents. Due, among other factors, to differences in the level of reporting detail for the events, the representational quality of the data is uneven: some events lack basic information as to when or where precisely they occurred. For the purposes of the natural data experiments, high quality subsets of information from the AKB were selected based on the number of occurrences of terrorist events perpetrated by specific groups. The nefarious agents with more than one hundred represented incidents are:

LebaneseHizballah	350
RevolutionaryArmedForcesOfColombia	246
PalestineIslamicJihad	160
BasqueFatherlandAndLiberty	142
TerroristOrganization-National-Liberation-Front-of-Tripura	140
AlFatah	135
Iraqi-insurgents-Group	110
TerroristOrganization-Al-Aqsa-Martyrs-Brigade	106
NationalLiberationArmyColombia	106

The dataset used for the present experiment was the set of terrorism events such that there is a specific country for which Cyc was able to prove that the event took place in that country, restricted to those perpetrated by the top 12 most frequent perpetrators. In total there were 1564 events satisfying these criteria.

## Feature Representation

A *feature representation* of this data was constructed to enable evaluation of the accuracy of standard machine learning models such as decision trees. A feature representation is a list of instances represented as a set of attribute-value pairs. Features were represented in the ARFF format developed by the University of Waikato for its Weka Machine Learning toolkit (Witten & Frank, 2005), because this allows straightforward use of its classifiers, including J48 decision trees.

In order to represent data in ARFF format, one must first convert the relevant CycL sentences, which are in first-order logic, into an appropriate propositional form. The conversion of first-order logic into propositional logic is a well-understood process referred

to as *propositionalization* (Russell & Norvig, 2001). During propositionalization, the sentences implied by the first order logic formulas are made manifest, for example, through computing the transitive closure latent in a subsumption hierarchy. The result is a set of fully-ground atomic sentences (sentences containing no variables or logical connectives), which may or may not be negated.

The fully-manifested contents of the relevant section of the KB are then converted to a feature representation. A feature representation such as an ARFF file is very much like a spreadsheet, with columns and rows. One can think of each cell in the spreadsheet as representing a separate proposition. In our case, each row represents a terrorist event, and each column represents an attribute of the event, whose value is given by the value in the corresponding cell. Cyc uses the so-called "Davidsonian" representation (Davidson, 1967) for events, which treats the event as a reified individual, about which facts can be asserted, such as date and location, so the events corresponding to the rows are individuals in Cyc.

We considered two main types of attributes: categorical and boolean. *Categorical attributes* correspond to relations that are *functional* in their second argument, i.e., relations such that there is one and only one value of the second argument for each value of the first argument. For such features, the column represents a binary predicate, and the value of the attribute corresponds to one of the possible instantiations of the second argument of the predicate. One categorical attribute used was the country in which the event occurred: For each event, there is *exactly one* country in which it occurred. The cells in this column take on values corresponding to the second argument of the predicate represented by the column, e.g.: Argentina, Brazil, Colombia<sup>1</sup>.

*Boolean attributes* are used for binary relations that are not functional in their second argument. For such relations, an attribute is created for each possible value of the second argument of the relation, and the cells in these columns take on boolean values. This is how location is represented when it is not restricted to countries; in this case, a single event can have multiple locations. For example, an event that occurs in Beirut also occurs in Lebanon, so two of the location attributes (spreadsheet columns) are `eventOccursAt_Beirut` ("Did the event occur in Beirut?") and `eventOccursAt_Lebanon` ("Did the event occur in Lebanon?"). Every event that occurs in Lebanon has a "Y" (representing truth) in the `eventOccursAt_Lebanon` column, and every event that does not has an "N" (representing falsehood) in that column.

In the "Full location information" model, location information at every geographic level was included: specific cities, as well as countries and continents; all inferrable locations were included. We also constructed a dummy model for comparison (the "Asserted location only" model), in order to test the effectiveness of using a geographical subsumption hierarchy. In this model, a feature like `eventOccursAt_Beirut` would take on the value of Y or N depending on whether it was *directly asserted* in the KB. We also constructed a model in

---

<sup>1</sup> This is a simplifying assumption. Some events take place in multiple countries. For example, take the incident that started the Israeli invasion of Lebanon in 2004. Hizballah operatives slipped over the southern border of Lebanon into Israel and kidnapped Israeli soldiers from an IDF outpost. An Israeli unit pursued the kidnappers back across the border and was drawn into an ambush in Lebanon. Consider this incident end-to-end, from kidnapping to ambush: did it happen in Lebanon or did it happen in Israel? Arguably, it happened in both.

which only countries were considered; this “Country only” model treats “Country” as a categorical feature, whose possible values are names of countries. In summary, there were three models:

1. Including only directly asserted `EventOccursAt` statements (the “Asserted location only” model)
2. Including only `EventOccursAt` statements where the location is a country (the “Country only” model)
3. Including all `EventOccursAt` statements, whether asserted or proven, and regardless of the nature of the second argument (the “Full location information” model)

Using this representation, we constructed decision trees using Weka's J48 algorithm, a version of Quinlan's (1993) C4.5 algorithm for constructing decision trees, based on the earlier ID3 algorithm (Quinlan, 1986). The nodes of a decision tree are tests for feature values, and leaves specify the class labels for the instances – in our case, who the perpetrator of the event was. The goal of any decision tree algorithm is to minimize the size of the decision tree, while maximizing classification accuracy. Generally this is done by maximizing information gain. The information gain of an attribute  $a$  can be defined as the reduction in entropy from partitioning on attribute  $a$ ; entropy is 0 when all examples belong to the same category, and 1 when examples are equally mixed (e.g. half category A, half category B). Our application of the J48 algorithm uses cross-validation to combat overfitting: a proportion, in this case 10% of the training data is iteratively held out, and overfitting nodes are pruned during this process.

## MLN Representation

Markov Logic Networks (MLNs) provide support for learning, representing, and performing inference using both “hard constraints” and “soft constraints”: if a world violates a “soft” constraint, it is less probable, but not impossible (Richardson and Domingos, 2006). In an MLN, each formula is associated with a weight that reflects the strength of the constraint. The stronger the constraint, the more unlikely an assignment of truth values to a set of propositions becomes when it is violated. Together with a set of constants, MLNs define Markov networks containing one binary feature for each possible grounding of each formula in the language. Each feature in the network is associated with the weight of the corresponding formula. By attaching weights to first-order formulas, Markov logic combines first-order logic and probabilistic graphical models. The *Alchemy* system (Kok et al., 2005) provides support for training, inference, and structure learning with MLNs.

An MLN specification consists of a set of type declarations and a set of (possibly weighted) formulas. Here is an example type specification:

```
EventOccursAt(event, location)
Perpetrator(event, agent!)
```

The first declaration specifies that `EventOccursAt` is a relation between something of type `event` and something of type `location`; the second declaration specifies that `Perpetrator`

is a relation between an event and an agent.<sup>2</sup> The exclamation point ("!") following `agent` indicates that there is exactly one agent who perpetrated each event; that the relation is *exclusive and exhaustive* in its second argument.<sup>3</sup> When `EventOccursAt` is limited to countries, the `EventOccursAt` relation is exclusive and exhaustive, because there is one and only one true value of the second argument for each distinct value of the first argument.

Here is an example of formula in Markov Logic (`e`, `l`, and `a` are implicitly universally quantified variables):

```
EventOccursAt(e,+l) => Perpetrator(e,+a)
```

The `+` symbol denotes a “per-constant” formula: When a variable in a formula is preceded by `+`, a separate weight is learned for each formula obtained by grounding that variable to one of its possible values. Stating such a rule makes it possible to take advantage of statistics from the corpus about specific constants during learning — in this case, where particular agents tend to operate. The resulting weighted formulas can be used in inference to predict the perpetrator of the event. For example, the contrast among the weights learned for the following groundings captures the fact that the Revolutionary Armed Forces of Colombia is more likely to commit acts of terrorism in Colombia than in Israel:

```
1.843 EventOccursAt(id,"Colombia") =>
Perpetrator(id,"RevolutionaryArmedForcesOfColombia")
-1.872 EventOccursAt(id,"Israel") =>
Perpetrator(id,"RevolutionaryArmedForcesOfColombia")
```

In contrast, Hamas is more likely to operate in Israel than in Colombia, as captured by the weights associated with the following two formulas:

```
-1.821 EventOccursAt(id,"Colombia") =>
Perpetrator(id,"TerroristOrganization-Hamas")
0.604 EventOccursAt(id,"Israel") =>
Perpetrator(id,"TerroristOrganization-Hamas")
```

Following previous work on MLN representation (e.g. Singla and Domingos 2006), we include both positive and negative antecedents for the per-constant rules, giving both of the following for the example under consideration: (The exclamation point in this case represents negation.)

```
EventOccursAt(e,+l) => Perpetrator(e,+p)
!EventOccursAt(e,+l) => Perpetrator(e,+p)
```

<sup>2</sup> Note that the expressions `event`, `location`, and `agent` are types, whereas in the formulas below, the arguments of the predicates are implicitly universally quantified variables. By convention, variables will be indicated with single letters, and types will be indicated with full words.

<sup>3</sup> This is a simplifying assumption; an event can have more than one perpetrator in principle, but assuming that this is impossible has dramatic computational advantages. There is nothing in the intended interpretation of the perpetrator predicate to rule out the possibility of multiple perpetrators, but especially because the perpetrators asserted in the TKB tend to be terrorist groups rather than individual terrorists, this is a rare case.

This allows separate weights to be learned for the positive and negative cases.

Training uses an *evidence database*, which consists of a set of positive or negative ground atomic formulas (GAFs). Here are sample contents from an evidence database that includes the full extent of `EventOccursAt` for each event.:

```
EventOccursAt ("TerroristAttack-August-5-2006-Israel", "ContinentOfAsia")
EventOccursAt ("TerroristAttack-August-5-2006-Israel",
"ContinentOfEurasia")
EventOccursAt ("TerroristAttack-August-5-2006-Israel", "EasternContinent")
EventOccursAt ("TerroristAttack-August-5-2006-Israel", "Israel")
EventOccursAt ("TerroristAttack-August-5-2006-Israel", "PlanetEarth")

Perpetrator ("TerroristAttack-August-5-2006-Israel", "LebaneseHizballah")

...
```

Generating the full extent in this way is called *fully manifesting* a predicate. This requires deductive reasoning using the Cyc inference engine.<sup>4</sup>

The learned model is tested on held out data, also represented in evidence databases. The predicate corresponding to the predicted variable is held out from the test set as well, so evidence databases used for testing predictions of `Perpetrator` exclude groundings for that predicate.

Using the same basic MLN, we generated training data in three different ways, correspondingly with the feature representations:

- Including only *directly asserted* `eventOccursAt` statements (the "Asserted location only" model)
- Including only `eventOccursAt` statements where the location is a country (the "Country only" model)
- Including all `eventOccursAt` statements, whether asserted or proven, and regardless of the nature of the second argument (the "Full location information" model)

The Markov Logic Networks were trained using discriminative learning, and inference was performed using the MaxWalkSat algorithm, using the Alchemy software from the University of Washington. These models and their associated training databases were converted directly from their corresponding feature representations, reconstructing non-exclusive-and-exhaustive relations from sets of binary features, to ensure comparability across modes of learning and inference.

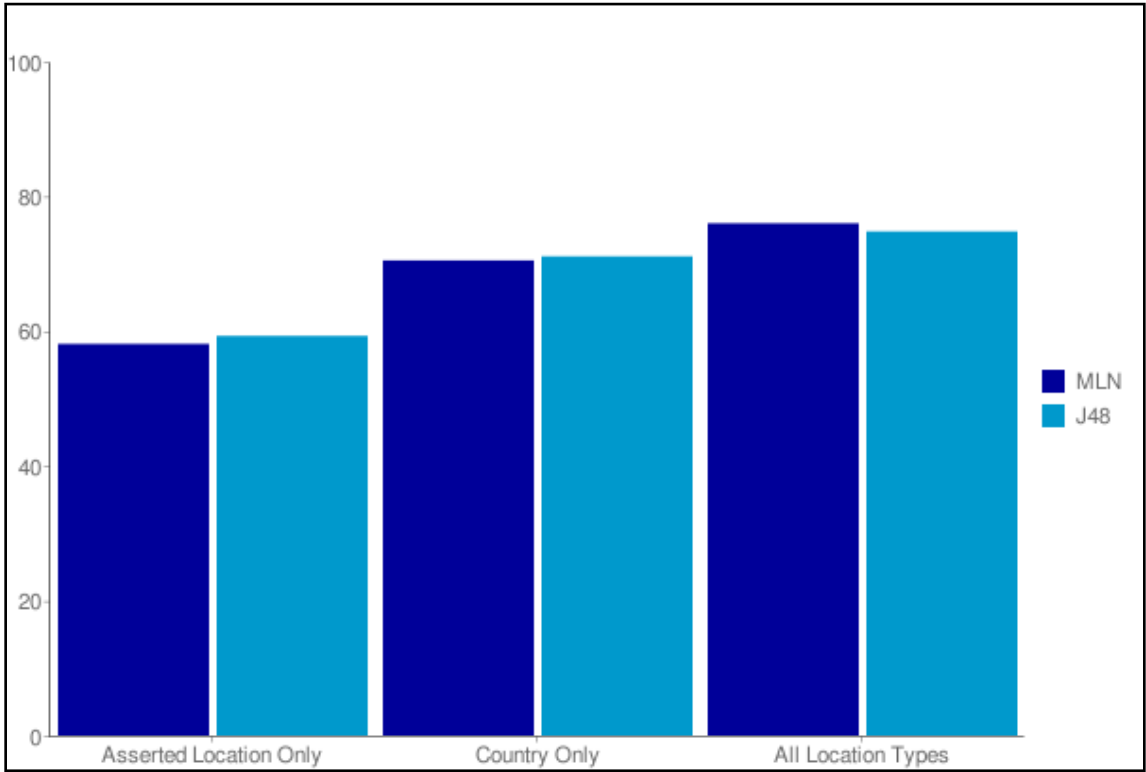
---

<sup>4</sup> By default, Alchemy makes the closed world assumption for predicates represented in an evidence database, so all groundings of `EventOccursAt` that are not explicitly listed as true or false are assumed to be false.

## Results

We found that it is possible to predict the perpetrator with roughly 60% accuracy on the top 12 perpetrator dataset, using asserted location alone.<sup>5</sup>

However, when geographical subsumption reasoning is used to identify the *country* in which the event occurred, accuracy jumps to 70%: among the events perpetrated by one of the top 12 perpetrators, it is possible to predict, with 70% accuracy, who the perpetrator was, using country as the only predictive feature. This number jumps again to 75% when the full geographical subsumption hierarchy is used to model location: This model uses information not only about what country the event occurred in, but also what continent, what city, what region, etc. J48 Decision Trees and Markov Logic Networks performed quite similarly after 10-fold cross-validation. These numbers are summarized in the graph below.



Although country is an excellent predictor on its own, these results show that there is an advantage to including richer geographical information. The conclusion we draw from this is that *subsumption reasoning is useful in probabilistic reasoning*.

---

<sup>5</sup> This number is surprisingly high, since it seems that it should be difficult for the system to correctly generalize over specific locations; we suspect that many of the asserted locations in the AKB are in fact relatively general; quite a few are countries.

To see why including richer geographical information is useful, it is instructive to compare the confusion matrices for the two models. Below is the J48 confusion matrix for the "Country only" model; the light blue squares on the diagonal contain correct classifications, where the class assigned by the model is identical to the true class.

Confusion matrix for the Country-only model

a	b	c	d	e	f	g	h	i	j	k	l	«classified as
27	0	3	0	24	0	0	0	0	0	0	4	a
0	0	0	0	1	0	0	0	0	60	0	1	b
0	0	125	0	5	0	0	0	0	0	0	0	c
0	0	0	81	0	0	0	0	0	0	0	0	d
3	0	3	2	217	0	0	1	0	86	0	0	e
0	0	0	0	0	3	0	87	0	0	0	0	f
0	0	0	0	4	0	0	0	0	97	0	0	g
0	0	1	0	1	5	0	209	0	0	0	0	h
0	0	0	0	1	0	0	0	0	42	0	0	i
0	0	0	1	3	0	0	0	0	352	0	0	j
0	0	0	0	0	0	0	0	0	0	0	12	k
0	0	0	0	0	0	0	0	0	0	0	65	l

The perpetrators are coded as follows:

a	AbuNidalOrganization
b	AlFatah
c	BasqueFatherlandAndLiberty
d	Iraqi-insurgents-Group
e	LebaneseHizballah
f	NationalLiberationArmyColombia
g	PalestineIslamicJihad
h	RevolutionaryArmedForcesOfColombia
i	TerroristOrganization-Al-Aqsa-Martyrs-Brigade
j	TerroristOrganization-Hamas
k	TerroristOrganization-National-Liberation-Front-of-Tripura
l	TerroristOrganization-United-Liberation-Front-of-Asom

The confusion matrix above shows (marked in Yellow) that the "Country only" model *always* misclassifies events perpetrated by TerroristOrganization-National-Liberation-Front-of-Tripura (k) as being committed by TerroristOrganization-United-Liberation-Front-of-Asom (l). This is because *the Tripura and Asom groups both operate*

in India, and Asom occurs more frequently in the data than Tripura. When country is the only distinguishing factor, as in the Country only model, these groups cannot be distinguished. On the other hand, when richer geographical information is included, these groups can be distinguished. In the "Full location information" model, the Asom group (l) is classified correctly in 11/12 instances, as shown in the confusion matrix for this model:

Confusion matrix for the model with full location information:

a	b	c	d	e	f	g	h	i	j	k	l	<< classified as
33	0	2	0	18	0	0	0	0	0	1	0	a
0	2	1	0	7	0	3	0	0	42	1	0	b
0	0	120	0	3	0	0	2	0	0	0	0	c
0	0	0	68	0	0	0	0	0	0	0	0	d
10	1	4	2	283	0	1	1	0	2	0	0	e
0	0	0	0	0	4	0	81	0	0	0	0	f
0	4	0	0	24	0	11	1	0	48	0	0	g
0	0	1	0	0	2	0	205	0	0	0	0	h
0	0	0	0	4	0	0	0	0	25	0	0	i
0	7	0	1	42	0	7	0	0	245	0	0	j
0	0	0	0	0	0	0	0	0	0	11	1	k
0	0	0	0	1	0	0	0	0	0	1	62	l

Event type

The AKB contains a variety of information about events, and it is useful to be able to use them as predictive features in conjunction with other kinds of information. One major class of information is *event type*: The events in the AKB are directly asserted to be instances of various types, via the Cyc *isa* predicate. For example, some are asserted to be instances of Bombing, thus:

```
(isa TerroristAttack-March-29-2002-Vista-Hermosa-Colombia Bombing)
```

Like locations, the types are arranged in a hierarchy; so for example, if an event is a Bombing, one can infer that it is an act of BlowingSomethingUp. This is expressed via *genls*:

```
(genls Bombing BlowingSomethingUp)
```

From this and the *isa* statement above, along with the following rule:

```
(implies (and (isa ?X ?COL1) (genls ?COL1 ?COL2)) (isa ?X ?COL2))
```

one can conclude:

```
(isa TerroristAttack-March-29-2002-Vista-Hermosa-Colombia
    BlowingSomethingUp)
```

We hypothesized that event type information should improve the accuracy of a probabilistic model of perpetrator identification.

## Methods

We used subsumption reasoning in this domain as well to fully manifest the `isa` predicate for each event. Again, we constructed both decision tree models and MLNs. The decision tree models contained, in addition to the event location features, a feature for each relevant collection, such as "isa\_Bombing" or "isa\_BlowingSomethingUp". A collection was considered *relevant* if there was any event in the dataset that was an instance of that collection.<sup>6</sup>

The MLN was written as follows (**Country and Event Type model**):

```
Isa(event, collection)
Perpetrator(event, agent!)
EventOccursAt(event, location!)

// co-occurrence matrix of location and perpetrator
EventOccursAt(e, +l) => Perpetrator(e, +a)

// co-occurrence matrix of event type and perpetrator
Isa(e, +c) => Perpetrator(e, +a)

// baseline frequency of each perpetrator
Perpetrator(e, +a)
```

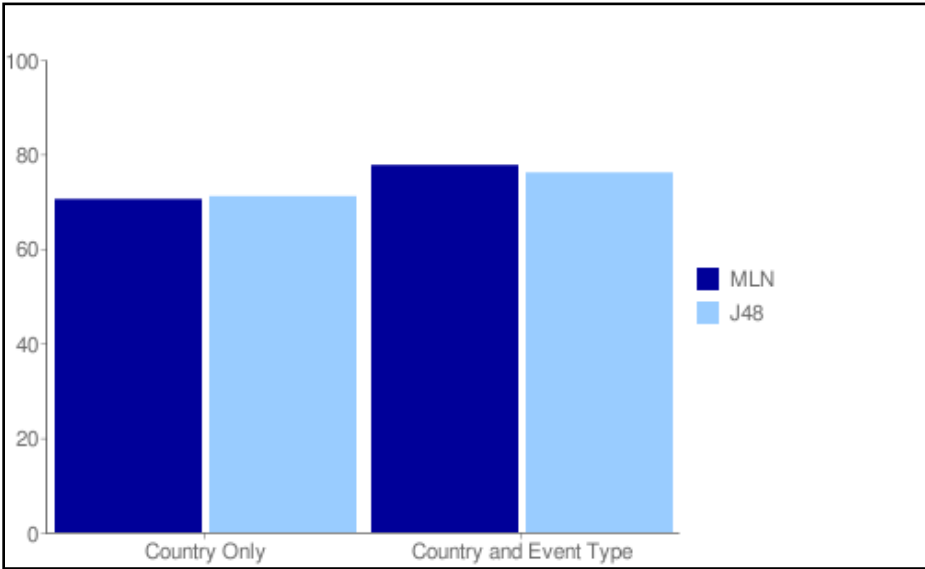
The first three lines provide the type specifications. The subsequent lines contain formulas. The double forward slash (`//`) begins a comment; anything following it is not interpreted by the parser. The plus symbol (`+`), again, denotes a per-constant formula; it directs the weight-learning process to learn a weight for each grounding of the variable marked with the plus symbol. The final formula, "Perpetrator(e,+a)", is a "unit clause", containing only an atomic formula; this allows baseline weights (priors) to be learned for each perpetrator. The extent of `EventOccursAt` was limited to cases in which the second argument is a country.

---

<sup>6</sup> We also tried pruning redundant collections; this was no more effective.

## Results

Both the MLN model and the J48 model showed an improvement in ability to predict the perpetrator when event type information was included. This is shown in the graph below; the "Country only" model, repeated from above, is given as a baseline.



This result shows that it is useful to combine various types of information in probabilistic inference, and that both Markov Logic Networks and J48 decision trees can successfully integrate multiple predictors.

## Date of event

Another factor that can be used to predict the perpetrator of a crime is when the crime occurred. We hypothesized that date information should improve the predictive accuracy of a perpetrator identification model.

## Methods

### Feature representation

A feature representation of event date was generated, very simply, by adding a column labelled "date" to the feature representation, whose cells were numbers corresponding to years. The conversion from CycL used Cyc's ability to reason about dates. For example, here is an event date assertion that is stated directly in the KB:

```
(dateOfEvent TerroristAttack-March-8-1999-Leioa
 (DayFn 8
  (MonthFn March
   (YearFn 1999))))
```

This means that the event in question occurred on March 8th, 1999. From that assertion, the following fact can be proven:

```
(dateOfEvent TerroristAttack-March-8-1999-Leioa (YearFn 1999))
```

This means that the event took place in the year 1999. As this example shows, dates are represented in CycL with complex data types composed of functions and numbers. For the purpose of creating a feature representation, it is desirable to represent the year not as a `YearFn` expression, but as a number, because this makes it possible to take advantage of ARFF's built-in support for numeric comparison (e.g. that  $1996 < 1998$ ). Therefore, the number "1999" was extracted and used as the value for the date.

## MLN representation

Unlike the types of features we have seen so far, numeric features such as dates cannot be translated readily from a feature representation to an MLN representation. MLN syntax does support numeric comparison between integers; this is not the problem. The problem is finding the right date "cut-offs". A decision tree model splits the data using a numeric cutoff, as will be shown in the following section. For example, a decision tree may split the events into two groups: those which occur in 2001 or afterwards, and those which occur before 2001. The choice of split point is made automatically in the decision tree learning algorithm. If any split points are to be included in a Markov Logic Network, they must be specified in the model for which weights are to be learned, when weight learning is the only kind of training that is done. One option is to manually choose arbitrary cut-offs, such as decades (the **Decade model**):

```
when(event,int)
who(event,agent)
isa(event,event_type)
where(event,location)

when(e,d) ^ d < 1990 => who(e,+a)
when(e,d) ^ d < 2000 => who(e,+a)

isa(e,+t) => who(e,+a)
where(e,+l) => who(e,+a)

who(e,+a)
```

In principle, another option is to include all possible cut-off points and learn weights for all of them.

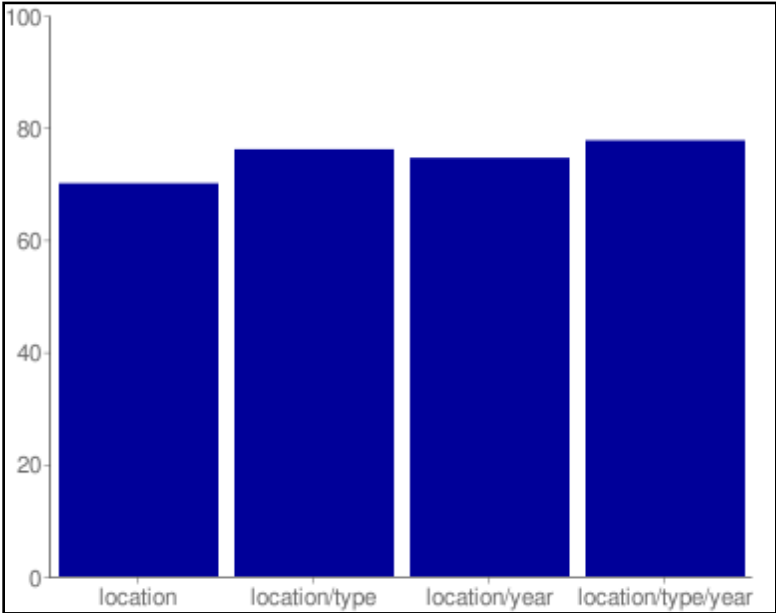
It is in this domain that the difference between Markov Logic Network weight-learning and decision tree learning becomes evident. So far, we have treated them similarly, but they are fundamentally quite different: In J48 decision tree learning, the structure of the model is learned, whereas in MLN weight-learning, the structure of the model is given in advance, and weights are learned for that model.

## Results

Here is a segment of the decision tree that was produced using all three types of attributes (location, event type, and year):

```
where = Colombia
| when <= 2001
| | isaE_PreventingFn_TransportationEvent = no
| | | isaE_IllegalAction = no: RevolutionaryArmedForcesOfColombia
(57.0/21.0)
| | | isaE_IllegalAction = yes
| | | | when <= 1999: RevolutionaryArmedForcesOfColombia (84.0/30.0)
| | | | when > 1999
| | | | | isaE_TakingHostage = no: NationalLiberationArmyColombia
(13.0/2.0)
| | | | | isaE_TakingHostage = yes:
RevolutionaryArmedForcesOfColombia (6.0/1.0)
| | | isaE_PreventingFn_TransportationEvent = yes:
NationalLiberationArmyColombia (2.0)
| when > 2001: RevolutionaryArmedForcesOfColombia (121.0/19.0)
```

This decision tree model achieves 77.7% precision. This is the highest precision model for this data we have generated. Its improvement over the "Country + Event type" model above is only slight, however; that model achieved 76.1% accuracy. It is possible that the learner is encountering a *ceiling effect*: It may not be possible to model more than about 80% of the variation in the data on the basis of readily available information about events. A further indication that date information is in fact predictive comes from the fact that 74.6% accuracy can be obtained by combining information about where the event occurred with date information; this is a 4.5% improvement over what can be obtained through country information alone (70.1%). These numbers are summarized in the graph below.



Markov Logic Networks performed similarly. The "Decade" model given above achieved 76% accuracy – a marginal gain on the model without year information.

## Using agent information: Where agents operate

One of the advantages of MLNs over established machine learning techniques is that they support *relational data*. In our case, we have data about events (where they occurred, etc.) along with data about agents (where they operate, who their enemies are, etc.). We would like to make use of the information about the agents in order to predict "Whodunit". We cannot make use of this kind of information in a decision tree, because the features that can be used in a decision tree classifier for events must be tied to the events.

One type of information available in the AKB about agents is the predicate `operatesInRegion`, which holds between an agent and a geographical region if the agent operates in that region. This information is not computed based on the location and perpetrator information that we have been describing so far, but rather was gathered independently through various news sources (143 sources in total). This information, combined with the location of the event, should help in identifying the perpetrator of an event.

## Methods

To test the usefulness of `operatesInRegion`, the following MLN was constructed (the **OperatesInRegion model**):

```
Perpetrator(event, agent)
EventOccursAt(event, location)
OperatesInRegion(agent, location)

OperatesInRegion(p, loc) ^ EventOccursAt(id, loc) => Perpetrator(id, p)

Perpetrator(id, +p)
```

The extent of `OperatesInRegion` was computed using a Cyc query, so that both asserted and derived facts would be included. The extent of `EventOccursAt` was limited to countries. Notice that the `OperatesInRegion` rule is not a "per-constant" rule<sup>7</sup>; it merely checks whether the locations are identical. The rule therefore only adds one parameter to the model, whereas per-constant rules add one parameter for each combination of constants. This MLN also includes a per-constant unit clause for `Perpetrator`, which allows a prior (weight) to be learned for each possible perpetrator. This adds one parameter to the model for each perpetrator (of which there are 12).

To evaluate the effectiveness of the `OperatesInRegion` rule, we compared the above MLN to another MLN which was identical except that this rule was not included (the **Baseline model**):

---

<sup>7</sup> That is, it is not instantiated, with a different weight, for every location and potential perpetrator.

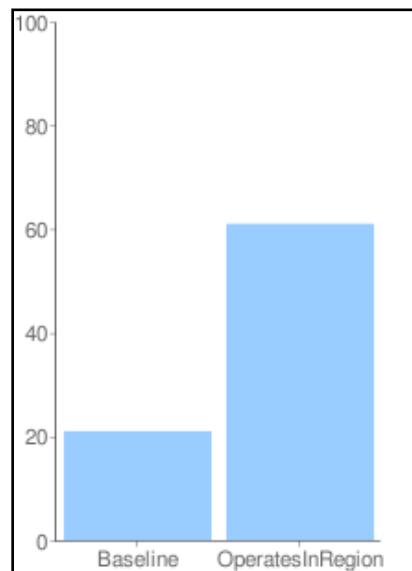
```
Perpetrator(event, agent)
EventOccursAt(event, location)
```

```
Perpetrator(id, +p)
```

This baseline model takes into account the frequency with which perpetrators operate, but nothing more.

## Results

The benefit of the `OperatesInRegion` rule turned out to be striking. Without the `OperatesInRegion` rule, the average precision of the model was 21% (i.e., the perpetrator was correctly identified in 21% of the events). With `OperatesInRegion`, precision jumps a full 40% to 61%.<sup>8</sup>



The `OperatesInRegion` model has only *thirteen parameters*: one for each perpetrator, and one for the `OperatesInRegion` rule, whereas the `EventOccursAt` model has *over 1,000 parameters*, including one for each perpetrator/location pair. This is an enormous advantage in simplicity. Simpler models are more robust against over-fitting than more complex models by definition, so there is a potential practical advantage to building simpler models.

Such a model could not have been constructed with a J48 decision tree, because this is a *relational* model, using facts about events (`EventOccursAt`, which takes an event as its first argument), and facts about agents (`OperatesInRegion`, which takes an agent as its first

---

<sup>8</sup> These percentages are averaged over 10 partitions between training and testing sets (“folds”); the per-fold percentages varied by no more than a few percentage points.

argument). This result demonstrates the usefulness of relational probabilistic models, and takes advantage of the power of Markov Logic Networks to represent them. Markov Logic Networks are not the only relational models, of course. Other machine learning techniques supporting relational data are reviewed in Džeroski and Lavrač (2001). Taskar et al. (2002), introduce Relational Markov Networks, which use conjunctive database queries as a logical language. MLNs are more general than these other relational techniques, however, because they model (weighted) full first-order logic.

Skeptics of the claim that such a model could not have been constructed with a decision tree might point out that in some cases, relational data can be converted into non-relational data. Indeed, we have shown examples of this already: For example, we know about some event that it took place in Lebanon, and we know about Lebanon that it is in Israel, so we conclude about the event that it took place in Israel, via something we knew about Lebanon. For another example, suppose one has a database with a table of medical visits, with one record for each medical event, and a table of patients, with one record for each patient, and these tables are linked, so that the patient ID associated with a visit can be located in the patient table. If one wants to know the age of the patient who was treated in a particular event, one finds the patient ID, and looks up the age of the patient in the patient table using the ID. The patient table can be eliminated by creating a "patient age" column in the medical visit table, and inserting the patient's age into that column for each visit that the patient makes. This strategy is crucially *not* available in this situation, however. Imagine what would happen if we created a predicate `*perpetratorOperatesInRegion`, which holds of an event and a region if the perpetrator of the event operates in the region in which the event took place, and used this predicate to predict the perpetrator. This would be circular and absurd: One cannot know `perpetratorOperatesInRegion(e, r)` if one does not already know the perpetrator of the event  $e$ , and that is exactly what we are trying to predict. When it comes to features of the values that are being *predicted*, it is not possible to convert relational data into non-relational data. A framework for relational probabilistic modelling is necessary for taking into account information about the predicted variable, while it is in the process of being predicted.

### Looking for a motive: Who has a negative vested interest in the target?

In many cases, an attack can be linked to a perpetrator using the perpetrator's *motives*. For example, the attack named `TerroristAttack-October-7-2000-Shebaa-Farms`, and perpetrated by `LebaneseHizballah`, killed several Israeli soldiers. Cyc can infer that `LebaneseHizballah` has a negative vested interest in the government of Israel, and that Israel has a positive vested interest in its soldiers, by virtue of the fact that citizens are "parts," broadly speaking, of their governments, and that agents have positive vested interests in their parts.

For another example, the attack named `TerroristAttack-October-27-1989-The-Hague`, perpetrated by `BasqueFatherlandAndLiberty`, was a bombing of the Spanish embassy in the Netherlands. Cyc can infer that `BasqueFatherlandAndLiberty` has a negative vested interest in the Spanish government, and that the Spanish government has a positive vested interest in its embassy in the Netherlands, by virtue of the fact that Spain owns the Spanish embassy. This type of reasoning is particularly important when it comes to events like embassy bombings, because here, location is not quite as good a predictor as it is in other

kinds of attacks. For embassy bombings, knowing the location in which they take place is not as informative about the identity of the perpetrator as knowing the country whose embassy was bombed is.

To use this kind of information in an MLN, we created the predicate called `harmed`, which holds of something if it itself was harmed, or if something it has a positive vested interest in was harmed. It is related to the predicates `negativeVestedInterest` and `perpetrator` thus:

```
harmed(e, x) ^ negativeVestedInterest(p, x) => perpetrator(e, p)
```

This reads: "If someone has a negative vested interest in something that was harmed during an event, then that person is the perpetrator of the event." It is worth reiterating at this point that rules for probabilistic inference, may be very different from the sort of rules that are admissible for deductive inference. They do not need to be uniformly true (this rule plainly is not); they merely need to capture some statistical regularity.

The predicate extent of `harmed` and `negativeVestedInterest` is fully manifested for the case of agents that are countries in order to reduce the size of the model while capturing the interesting cases. Indicating the virtues of combining probabilistic and deductive knowledge, the formulas in the extent of these predicates that were used for training and testing were far from directly asserted in the Cyc KB; rather, each grounding was derived through a complex proof, using a large number of rules and facts. These computations sometimes used quite domain-general knowledge in Cyc. For example, the proof for the following assertion:

```
(negativeVestedInterest TerroristOrganization-Hamas Nicaragua)
```

used the following facts:

- `(isa TerroristOrganization-Hamas (BelieverFn IslamicSocietyIdeology))`
- `(relationAllInstance negativeVestedInterest (BelieverFn IslamicSocietyIdeology) AttemptToDemocratizeIraq)`
- `(implies (and (relationAllInstance ?PRED ?COL ?VALUE) (isa ?OBJ ?COL)) (?PRED ?OBJ ?VALUE))`
- `(cooperatingParticipants AttemptToDemocratizeIraq Nicaragua)`
- `(implies (and (negativeVestedInterest ?AGENT ?COOP) (cooperatingParticipants ?COOP ?PARTICIP)) (negativeVestedInterest ?AGENT ?PARTICIP))`

Here is an example of how the MLN predicate `harmed` was computed. The CycL equivalent of the MLN predicate `harmed` is a composite of the CycL predicates `objectHarmed` and `postiveVestedInterest`; if `harmed` holds of an event `?EVENT` and an entity `?ENTITY` then:

```
(thereExists ?HARMED
  (and (objectHarmed ?EVENT ?HARMED)
       (positiveVestedInterest ?ENTITY ?HARMED)))
```

This definition can be read "*something in which ?ENTITY has a positive vested interest is harmed in ?EVENT*".

Here are the facts that Cyc used to prove that Spain is a possible binding for ?ENTITY in this expression in the event in the 1989 bombing of the Spanish embassy building in the Hague:

1. (owns Spain (EmbassyBuildingFn Spain CityOfHagueNetherlands))
2. (damages TerroristAttack-October-27-1989-The-Hague (EmbassyBuildingFn Spain CityOfHagueNetherlands))
3. (genlPreds damages objectHarmed)
4. (genlPreds owns positiveVestedInterest)

To explain what the last two mean: The predicate `genlPreds` relates two predicates with the same number of arguments (the same arity), and asserts that the first predicate holds in a subset of the cases in which the second predicate holds. Thus, the `genlPreds` assertion relating `owns` and `positiveVestedInterest` above is equivalent to the following:

```
(implies
  (owns ?AGENT ?THING)
  (positiveVestedInterest ?AGENT ?THING))
```

That is, "whenever some agent owns something, the agent has a positive vested interest in it". Again, these facts and rules are not all specific to terrorism, and the full proof in which they are used is quite intricate.

To illustrate the usefulness of this kind of reasoning, we did the following experiment. We restricted the dataset to those events in which an embassy was bombed. There are approximately sixty such events. For each of these events, we included information as to who was directly or indirectly harmed in the event, restricted to countries, and as to who carried it out. For each relevant perpetrator, we also included information about the countries in which the perpetrator has a negative vested interest (`negativeVestedInterest`).

Here are sample entries from these evidence databases:

```
negativeVestedInterest (AbuNidalOrganization, Israel)
negativeVestedInterest (AlQaida, UnitedStatesOfAmerica)
negativeVestedInterest (BasqueFatherlandAndLiberty, Spain)
...

harmed (AlQaidaUSEmbassyAttack-Tanzania, UnitedStatesOfAmerica)
harmed (TerroristAttack-1992-Argentina, Israel)
harmed (TerroristAttack-April-18-1983-Beirut-Lebanon,
UnitedStatesOfAmerica)
...

perpetrator (TerroristAttack-March-2002-Lima-Peru, SenderoLuminoso)
perpetrator (TerroristAttack-October-27-1989-The-Hague,
BasqueFatherlandAndLiberty)
perpetrator (TerroristAttack-November-8-1993-Teheran, LebaneseHizballah)
...
```

The hand-specified MLN that we used to train weights was as follows (this is what we will refer to, henceforth, as the **negativeVestedInterest model**):

```
negativeVestedInterest(perp, country)
harmed(event, country)
perpetrator(event, perp)

negativeVestedInterest(p, c) ^ harmed(e, c) => perpetrator(e, p)

perpetrator(e, +p)
```

Again, the type specifications are in the top three lines, and the formulas for which weights should be learned are specified on the following lines. Notice that the rule of interest is again not a "per-constant" rule, but rather introduces a single parameter into the model. The perpetrator per-constant unit clause "perpetrator(e, +p)" was used to calculate the baseline frequency of each perpetrator.

For comparison, we constructed a baseline model with just the perpetrator per-constant unit clause (the **Baseline model**):

```
negativeVestedInterest(perp, country)
harmed(event, country)
perpetrator(event, perp)

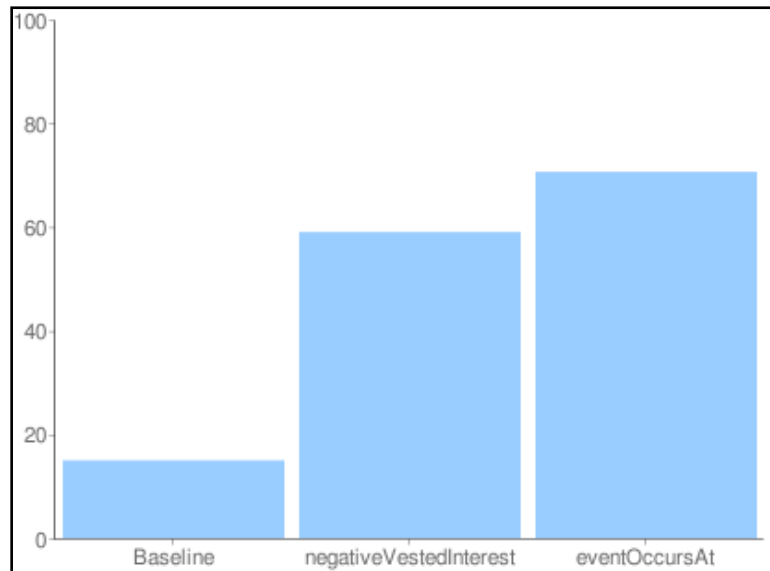
perpetrator(e, +p)
```

We also constructed a location model for the same data (the **eventOccursAt model**):

```
eventOccursAt(event, location)
perpetrator(event, perp)

eventOccursAt(e, +l) => perpetrator(e, +p)
```

The model with the `negativeVestedInterest` rule outperformed the baseline model 59% to 15%: Taking only perpetrator frequency into account, the baseline model was able to predict the perpetrator with only 15% accuracy. However, the `negativeVestedInterest` rule increases accuracy to 59%.



Granted, the `eventOccursAt` model from above achieves higher precision than the `negativeVestedInterest` model, but it also has far more parameters: The location model in this case has *over 400 parameters* (including one for each perpetrator/event combination), while the `negativeVestedInterest` model makes do with only one more parameter than the baseline model, which had one parameter for each perpetrator. In this dataset, there are 27 perpetrators, so the `negativeVestedInterest` model has only 28 *parameters*.

This example illustrates particularly well how a structured ontology, logic and probability can work together to solve classification problems. The structured ontology provides the model with predictive features (e.g. by providing the concept of `negativeVestedInterest`). Logic is used to instantiate those features providing appropriately construed training and testing data (in this case by permitting inference to the predicate extent of `negativeVestedInterest`), and probability is used for identifying and representing the association between predictive features and the dependent variable.

## Conclusion

For the problem of perpetrator identification, use of Cyc's ontology and reasoning abilities improves the accuracy of both decision tree and Markov Logic Network classifiers. These results suggest that the Cyc ontology and inference engine should be used in machine classification more generally.

## Bibliography

Deaton, C., Shepard, B., Klein, C., Mayans, C., Summers, B., Brusseau, A., et al. (2005). The Comprehensive Terrorism Knowledge Base in Cyc. In *Proceedings of the 2005 International Conference on Intelligence Analysis*. McLean, Virginia.

- Halstead, D. T., & Forbus, K. D. (2007). Some Effects of a Reduced Relational Vocabulary on the Whodunit Problem. In *Proceedings of IJCAI-2007*. Hyderabad, India.
- Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). *The Alchemy system for statistical relational AI*.
- Lavrač, N., & Dzeroski, S. (1994). *Inductive Logic Programming: Techniques and Applications*. New York, NY.
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.
- Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.
- Richardson, M., & Domingos, P. (2006). *Markov Logic Networks*.
- Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence* (pp. 485-492).
- Witten, I. H., & Frank, E. (1999). *Data Mining: Practical Machine Learning Techniques and Tools*. San Diego: Academic Press.