

Abstract

We describe a method for automatic discovery of generalizations about data residing in the Cyc Knowledge Base (KB). Data is exported from the Cyc KB into a feature representation, a decision tree classifier is used to discover generalizations in that data, and the learned generalizations are imported back into the KB using probabilistic vocabulary. Our case study is the *Whodunit* problem: guessing the perpetrator of a terrorism event.

We compare two methods of building decision-tree models from which rules can be imported. A *single-multiclass model* involves a single decision tree with multiple possible values for the predicted variable. A *multiple-binary model* involves multiple decision trees with binary predicted values, predicting, for a given instance i and class c , whether i has class c . We show that the single multiclass model is more prone to overgeneralization on larger datasets, due to its greater reliance on the closed world assumption. We conclude that for the purpose of rule extraction into a large knowledge base, it is better to use multiple binary decision trees.

Importing decision tree rules into Cyc

Single multiclass vs.
multiple binary classifiers
in the open world

Michael Witbrock, Elizabeth Coppock,
Keith Goolsbey, and Robert Kahlert*

Cycorp Technical Report 09-002, August 17, 2009

Copyright ©2009 Cycorp, Inc. All rights reserved.

*This work was partly supported under DARPA contract HR0011-06-C-0025.

Introduction

Reasoning systems equipped with large knowledge bases such as Cyc constitute vast resources of information, whose inferential power has not been fully realized. One major source of potential comes from generalizations that could emerge if the data within the knowledge base were subject to statistical analysis. With the ability to analyze their own content for statistical generalizations, knowledge-based reasoning systems would be able to answer more questions deductively, and provide reasonable estimates for answers probabilistically. If they could also identify questions that should be asked, and hypothesize what the answer might be, they would be capable of self-directed self-improvement. *Ontology evolution* – goal-directed autonomous self-improvement by a knowledge-based reasoning system, building on what is already known – is our ultimate research aim.

In this report, we describe a method for automatic discovery of generalizations about data residing in the Cyc Knowledge Base (KB). Data is exported from the Cyc KB into a feature representation, a decision tree classifier is used to discover generalizations in that data, and the learned generalizations are imported back into the KB using probabilistic vocabulary. Our case study is the *Whodunit* problem: guessing the perpetrator of a terrorism event, using data from the Cyc Analyst's Knowledge Base (AKB; Deaton et al., 2005).

After describing how Cyc data is exported into a feature representation, we compare two methods of building decision-tree models from which rules can be imported. A *single-multiclass model* involves a single decision tree with multiple possible values for the predicted variable. A *multiple-binary* model involves multiple decision trees with binary predicted values, predicting, for a given instance i and class c , whether i has class c . We show that the single-multiclass model is more prone to overgeneralization than the multiple-binary model due to its greater reliance on the closed world assumption. We conclude that for the purpose of rule extraction into a large knowledge base, it is better to use multiple binary decision trees.

Exporting Cyc data into a feature representation

Our case study is the problem of predicting the perpetrator of a terrorism event, given features of the event. The AKB data set has over 4500 terrorist events that belong to over 2000 classes and were performed by over 1000 agents. 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

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.

A feature representation such as an ARFF file is very much like a spreadsheet, with columns and rows, and 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 treat 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 reified 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, Colombia¹.

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"

¹ 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.

(representing truth) in the `eventOccursAt_Lebanon` column, and every event that does not has an "N" (representing falsehood) in that column.

Importing rules from decision trees

A decision tree classifier can be converted into a set of rules in first order logic. This is a similar process to that of generating *production rules* from decision trees. Quinlan (1987) examined methods for re-expressing a decision tree as a succinct collection of production rules of the form:

If left-hand side then class (certainty factor)

As Quinlan (1987) points out, converting decision trees into rules of this form is useful because these rules can be integrated into a reasoning system and modified individually by a human expert. Quinlan also shows that the transformation *"can improve classification performance by eliminating tests in the decision tree attributable to peculiarities of the training set, and by making it possible to combine different decision trees for the same task"* (p. 304).

Consider the following excerpt from a decision tree classifier for perpetrator, trained on the data from the "top 12 perpetrator" set:

```
where = Iraq
|   when <= 1998: LebaneseHizballah (3.0/1.0)
|   when > 1998: Iraqi-insurgents-Group (81.0)
where != Iraq
|   where = India
|   |   when <= 1989: AbuNidalOrganization (4.0)
|   |   when > 1989
|   |   |   CriminalAct = Y
|   |   |   |   when <= 2000: TerroristOrganization-United-Liberation-Front-of-Asom (3.0/1.0)
|   |   |   |   when > 2000: TerroristOrganization-National-Liberation-Front-of-Tripura
|   |   |   |   (11.0/4.0)
|   |   |   |   CriminalAct != Y
|   |   |   |   ControllingAPhysicalDevice = Y: TerroristOrganization-National-Liberation-Front-
|   |   |   |   of-Tripura (3.0/1.0)
|   |   |   |   ControllingAPhysicalDevice != Y: TerroristOrganization-United-Liberation-Front-
|   |   |   |   of-Asom (58.0/3.0)
|   |   where != India
|   |   |   when <= 1988
|   |   |   |   where = Lebanon: LebaneseHizballah (69.0/2.0)
|   |   |   |   where != Lebanon
|   |   |   |   |   when <= 1982: AbuNidalOrganization (15.0/2.0)
|   |   |   |   |   when > 1982
|   |   |   |   |   |   where = Italy: AbuNidalOrganization (8.0)
|   |   |   |   |   |   where != Italy
|   |   |   |   |   |   |   where = Austria: AbuNidalOrganization (3.0)
|   |   |   |   |   |   |   where != Austria
|   |   |   |   |   |   |   |   where = Sudan: AbuNidalOrganization (3.0)
|   |   |   |   |   |   |   |   where != Sudan
|   |   |   |   |   |   |   |   |   ControllingSomething = Y: LebaneseHizballah (3.0)
|   |   |   |   |   |   |   |   |   ControllingSomething != Y
|   |   |   |   |   |   |   |   |   |   AttackTypeByWeaponTypeFn_Bomb = Y: AbuNidalOrganization (4.0)
|   |   |   |   |   |   |   |   |   |   AttackTypeByWeaponTypeFn_Bomb != Y
|   |   |   |   |   |   |   |   |   |   |   where = Pakistan: AbuNidalOrganization (5.0/1.0)
|   |   |   |   |   |   |   |   |   |   |   where != Pakistan
|   |   |   |   |   |   |   |   |   |   |   |   ConflictEvent = Y: LebaneseHizballah (6.0)
|   |   |   |   |   |   |   |   |   |   |   |   ConflictEvent != Y
|   |   |   |   |   |   |   |   |   |   |   |   |   where = Greece: AbuNidalOrganization (6.0)
```


higher-order logic representation and programming languages. Consider the following sentence⁵:

```
(eventOccursAt TerroristAct-212 CityOfBaghdadIraq)
```

This sentence can be interpreted as a statement about the event: "The event TerroristAct-212 has the feature that it occurred in the city of Baghdad, Iraq." The `Kappa` function allows us to represent this feature. From the sentence, we can construct a unary predicate that denotes the feature "occurred in the City of Baghdad, Iraq", simply by substituting a variable for the event instance `TerroristAct-212` and wrapping the resulting expression with the `Kappa` function, which takes a list of variables as its first argument and an open sentence as its second argument:

```
(Kappa (?X) (eventOccursAt ?X CityOfBaghdadIraq))
```

The resulting predicate-denoting `Kappa` expression denotes precisely the unary predicate "occurred in Baghdad". Equipped with this `Kappa`, we can now rewrite the sentence about `TerroristAct-212` as sentence using a unary predicate.

```
((Kappa (?X) (eventOccursAt ?X CityOfBaghdadIraq))
 TerroristAct-212)
```

This sentence is logically equivalent to the original sentence using the binary predicate `eventOccursAt` directly.

Due to the differences in lexical expressivity between ARFF and CycL, a transformation from CycL to ARFF is lossy. It is therefore not always possible to write an invertible function that will correctly reconstitute the CycL terms using the terms that appear in a decision tree. This problem is circumvented by a translation table that maps the Cyc terms to ARFF terms and vice versa. In order to preserve the knowledge of the translation table within an ARFF file, pragmatic comments are emitted during its generation that specify the translations on a per-term basis. They are added only once, when the term is mentioned for the first time in the ARFF file. This makes models generated from the ARFF file re-importable into the KB without error.

The information expressed in a decision tree node consists of three parts: An attribute (i.e., a column name), a comparison operator (equals, less than, greater than or equal to, etc.), and a value (i.e., a cell value). Each attribute is associated with a template for a `Kappa` expression that includes "slots" (denoted below with the syntax `:SLOT`) for the comparison operator and the value. For example, the attribute "when" in the decision tree above is associated with the following `Kappa` expression:

⁵ The discussion of representing decision tree features in the KB that follows is somewhat technical and involved, a first time reader may wish to skip forward to the section on representing probabilities.

```
(Kappa (?E)
  (thereExists ?DATE
    (and (dateOfEvent ?E ?DATE) (:COMP (YearFn :VALUE) ?DATE))))
```

The comparison operator ">" translates to the CycL term `laterThan`, so the decision tree node "when > 1989" in the decision tree above will be translated into the following Kappa expression:

```
(Kappa (?E)
  (thereExists ?DATE
    (and (dateOfEvent ?E ?DATE) (laterThan (YearFn 1989) ?DATE))))
```

This Kappa expression "where != Iraq" will be translated using the Kappa template associated with `where`, which is as follows:

```
(Kappa (?E)
  (thereExists ?LOC
    (and (isa ?LOC Country) (eventOccursAt ?E ?LOC)
      (:COMP ?LOC :VALUE))))
```

The comparison operator in this case is "!=", which is translated with the Cyc term `different`. The string "Iraq" is translated to the Cyc term `Iraq`:

```
(Kappa (?E)
  (thereExists ?LOC
    (and (isa ?LOC Country) (eventOccursAt ?E ?LOC)
      (different ?LOC Iraq))))
```

The Kappa expression associated with an event type feature such as `CriminalAct` has only one "slot":

```
(Kappa (?E) (:COMP (isa ?E CriminalAct)))
```

In this context, `:COMP` will be filled by either the second order predicate `trueSentence` or `unknownSentence`; "=" will be translated with `trueSentence` and "!=" will be translated with `unknownSentence`. The `:VALUE` slot can be eliminated in the translations of attributes that take on boolean values, in order to reduce complexity. The node "CriminalAct != N" should be equivalent to the node "CriminalAct = Y" ("N" meaning "no", or "false", and "Y" meaning "yes", or "true"). The representation can therefore dispense with the `:VALUE` slot by canonicalizing all logical operations such that the `:VALUE` is always "True", which places all variation into the `:COMP` position.

The algorithm for translating a decision tree rule to a CycL rule first constructs the antecedent of the rule, using the following process:

1. Antecedents that are implicit are added. In this case, the data was constructed from events that are acts of terrorism (instances of `TerroristAct`); this is not explicitly represented in the data, but assumed during decision tree learning; its absence in a CycL rule, however, would result in its inappropriate application.

2. The decision tree is harvested for antecedents represented by the intermediate nodes along the path to a leaf.
3. The leaf node labels are mapped to the correct Cyc term; in this case, the Cyc term is a particular perpetrator.
4. The leaf node counts are retained for the computation of the certainty factor; leaves that do not pass the coverage threshold are discarded.
5. All constraints are mapped to unary `Kappa` expression templates that can be applied to the variable representing the event in the antecedent of the rule.
6. With the `Kappa` template in hand, the comparison and the value operator of the node are extracted and mapped back to the appropriate Cyc terms. These terms are then substituted into the `Kappa` template, representing a unary function applicable to the event variable.
7. All of the constraints are put into a single conjunction. However, this conjunction is a very baroque form of an appropriate antecedent and compares poorly with what a human ontologist would have authored; this is addressed in step 9.
8. The process then performs across-constraint transformations that capture the semantics of the decision tree structure but that are not adequately expressed by the target vocabulary. For example, the decision tree treats both `where` and `when` as functional with respect to the second argument, but the underlying predicates `eventOccursAt` and `dateOfEvent` are not. The functional assumptions of the decision tree algorithm allow the process to rewrite multiple `where Kappa` expressions with one `different` clause each into one `Kappa` with a multi-argument `different` clause. The functional assumptions similarly allow simplifying sets of `when` constraints by eliminating subsumed cases.
9. The resulting conjunction is passed to a component of the Cyc inference engine, called the "simplifier", that is responsible for converting Cyc expressions into the most effective form for reasoning and which gives them a form that approaches hand-authored rule antecedents in quality.

With the antecedent in place, the process next constructs a consequent. From the coverage and accuracy annotations of the leaf nodes in the tree, the probability can be computed directly. The label of the leaf node is translated back into the Cyc term corresponding to the perpetrating party; the relationship itself is provided by the context.

Finally, the antecedent and the consequent are combined to form an implication. The process is then repeated for the next path in the decision tree, until all leaf nodes have been visited.

Representing probabilities in the KB

In order to represent the "soft" nature of the decision tree rules, we treated the consequent of the rule as a probability assertion, for example:

```
(implies
  (and
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT Lebanon)
    (laterThan (YearFn 1989) ?DATE)
    (dateOfEvent ?EVENT ?DATE))
  (probability (perpetrator ?EVENT LebaneseHizballah) 0.9718))
```

This style of representation can be classified as *extensional*, as opposed to *intensional* because it attaches an uncertainty value to a formula rather than to a state of affairs or set of possible worlds (Pearl 1988). Extensional systems (tend to) have the property of being *modular*, in that they show the properties of *locality* and *detachment*: "A implies B" can be interpreted, "If you see A anywhere in the knowledge base, then regardless of what other things the knowledge base contains [locality] and regardless of how A was derived [detachment], you are given the license to assert B and add it to the database" (Pearl 1988: 5). Although modularity has computational advantages, it can lead to semantic problems. Pearl (1988: 7) explains the issue as follows:

For example, suppose we have a rule R1 = "If the ground is wet, then assume it rained (with certainty c1)." Validating the truth of "The ground is wet" does not permit us to increase the certainty of "It rained" because the knowledge base might contain strange items such as K = "The sprinkler was on last night." These strange items [are] called *defeaters* or *suppressors*...

The rule concluding to the probability statement can be seen as a statement of conditional probability, which could be represented thus:

$$P(\text{Perpetrator}=\text{Hizballah} \mid \text{TerroristAct,Lebanon},>1989) = 0.9718$$

The problem is that the probability that the perpetrator is Hizballah is not necessarily conditionally independent of everything not mentioned in the antecedent. In other words, the probability does not necessarily remain the same when more information is added. For example, suppose a new group appeared in Lebanon starting in 2005, and dominated the scene thereafter. Then the probability of Hizballah could go down, even if the first statement remains true:

$$P(\text{Perpetrator}=\text{Hizballah} \mid \text{TerroristAct,Lebanon},>1989,>2005) = 0.2$$

We are now in danger of allowing the following two statements to be simultaneously provable within the same context, for some particular value of ?EVENT:

```
(probability (perpetrator ?EVENT LebaneseHizballah) 0.9718))
(probability (perpetrator ?EVENT LebaneseHizballah) 0.2))
```

These two statements are blatantly contradictory; an event can have only one probability. Pearl claims that this problem cannot be remedied within an extensional system, but we propose a method for doing so, or at least for its mitigation. The solution is simple: we treat rules concluding to probability statements as *default rules* in the sense of Reiter's (1980) default logic; default logic is an integral part of Cyc (Matuszek et al., 2006). Default rules are associated with exceptions in Cyc. This makes it possible for probabilities to be adjusted when more is known. For example, if a new group appeared in Lebanon starting in 2005, then an exception to this rule could be stated as follows:

```
(exceptWhen (laterThan (YearFn 2005) ?DATE)
  (implies
    (and
      (isa ?EVENT TerroristAct)
      (eventOccursAt ?EVENT Lebanon))
```

```
(laterThan (YearFn 1989) ?DATE)
(dateOfEvent ?EVENT ?DATE))
(probability (perpetrator ?EVENT LebaneseHizballah) 0.9718))
```

This exception statement will prevent the rule from concluding that Lebanese Hizballah has a 97% probability of having perpetrated an event when the event occurred after 2005.

An alternative way of combining default reasoning and probability is discussed by Pearl (1988, ch. 10, sec. 2). In Pearl's system there are default statements of the form $P(x) \rightarrow Q(x)$, which stand for "the probability of $P(x)$ given $Q(x)$ is greater than or equal to $1 - \epsilon$ ", where ϵ is a quantity that can be made arbitrarily small. Pearl defines a notion of *ϵ -entailment*, which "guarantees that an ϵ -entailed statement S is rendered highly probable whenever all the defaults in [the set of default statements] are highly probable" (p. 485). Thus, an alternative method of representing probabilistic rules in the KB would have been simply as default rules, understood as conditional probability statements with probability $1 - \epsilon$. Further exploration of the relative merits of these two strategies should be done in a larger, follow-on project.

Style 1: Single multiclass decision tree

We considered two styles of decision trees as the basis for importing rules into Cyc. In this section, we describe the first style, in which a single decision tree classifier is used. An excerpt from this tree was shown above. The rules automatically extracted from this tree look like this:

```
(implies
  (and
    (isa ?EVENT TerroristAct) (eventOccursAt ?EVENT Iraq)
    (laterThan ?DATE (YearFn 1998)) (dateOfEvent ?EVENT ?DATE))
  (probability (perpetrator ?EVENT Iraqi-insurgents-Group) 1))

(implies
  (and
    (unknownSentence (isa ?EVENT CriminalAct))
    (unknownSentence (isa ?EVENT ControllingAPhysicalDevice))
    (isa ?EVENT TerroristAct) (eventOccursAt ?EVENT India)
    (laterThan ?DATE (YearFn 1989)) (dateOfEvent ?EVENT ?DATE))
  (probability
    (perpetrator ?EVENT
      TerroristOrganization-United-Liberation-Front-of-Asom)
    0.9508))

(implies
  (and
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT Lebanon)
    (laterThan (YearFn 1989) ?DATE)
    (dateOfEvent ?EVENT ?DATE))
  (probability (perpetrator ?EVENT LebaneseHizballah) 0.9718))
```

```
(implies
  (and
    (unknownSentence (isa ?EVENT ControllingSomething))
    (unknownSentence (isa ?EVENT (AttackTypeByWeaponTypeFn Bomb)))
    (unknownSentence (isa ?EVENT ConflictEvent))
    (isa ?EVENT TerroristAct)
    (thereExists ?LOC
      (and
        (eventOccursAt ?EVENT ?LOC) (isa ?LOC Country)
        (different ?LOC Greece Pakistan Sudan Austria
          Italy Lebanon India Iraq)))
    (laterThan (YearFn 1989) ?DATE) (laterThan ?DATE (YearFn 1982))
    (dateOfEvent ?EVENT ?DATE))
  (probability (perpetrator ?EVENT LebaneseHizballah) 0.7551))
```

Robust generalizations are being acquired from the data, but if one looks closely at the rules produced, one notices that they are a bit odd: They contain a large number of negations (`unknownSentence` and `different` formulas) that seem irrelevant. They are included in the rule because they were useful in deciding between two other possible perpetrators, but they are incidental to the task of deciding whether or not it was the perpetrator mentioned in the consequent of the rule. A particularly egregious case is in the rule for `LebaneseHizballah`, which emerges relatively far down into the tree. This rule says, "if the event is not an event of controlling something, and not a bombing attack, and not a conflict event, and it is a terrorist act, and it occurs in a location that is distinct from Greece, Pakistan, Sudan, Austria, Italy, Lebanon, India, and Iraq, and it occurs after 1989, and it occurs after 1982, then the probability that the perpetrator of the event is Lebanese Hizballah is 0.75." This does not seem plausible, as a causal description, and it does not seem likely to generalize appropriately to new cases.

Style 2: Binary decision trees for each perpetrator

The problem of irrelevant conditions was noticed by Quinlan (1987), who addressed it by eliminating conditions from rules: When eliminating a condition did not negatively affect the certainty factor of a rule, the condition was eliminated. In this section, we discuss a different approach to the problem of irrelevant conditions. Our solution relies on converting the multiclass problem into k binary problems, where k is the number of classes (perpetrators, in this case). We call this the **multiple-binary model**, in contrast to the **single-multiclass model** above. Under this strategy, a separate decision tree is generated for each perpetrator. The predicted variable is binary: whether the event was carried out by the perpetrator in question or not. Converting a single multiclass problem into multiple binary problems is a well-studied technique, and several variants have been explored (Nilsson, 1965; Dietterich and Bakiri, 1995; Allwein, Schapire and Singer, 2000). This can be useful when the classifier in question does not extend naturally from binary classification tasks to the multiclass case, as in the case of AdaBoost (Freund and Schapire, 1997; Schapire and Singer, 1999), the SVM algorithm (Vapnik, 1995; Cortes and Vapnik, 1995). Decision trees are among the models that *can* be naturally extended to the multiclass case, but we will show that there is an advantage to dividing the problem into a series of binary problems anyway, when it comes to rule extraction.

The decision trees produced to make a boolean decision about whether a particular agent was the perpetrator or not are simpler, and they yield simpler rules, and ones with fewer

negative antecedents. For example, the decision tree for the Iraqi insurgents group is simply this:

```

where = Iraq
|   when <= 2004.0
|   |   when <= 1998.0: other (3.0)
|   |   when > 1998.0: Iraqi-insurgents-Group (2.0)
|   when > 2004.0: Iraqi-insurgents-Group (79.0)
where != Iraq: other (1558.0)

```

The translation process involved in importing this style of decision tree mirrors the one for the first style. Again, each decision tree is translated by incorporating rules only for leaf nodes with 30 or more corresponding instances. However, rules are restricted to those that positively conclude to the perpetrator in focus. After filtering out rules failing to meet these conditions, only one rule was left in the case of the Iraqi insurgents group:

```

(implies
  (and
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT Iraq))
  (probability (perpetrator ?EVENT Iraqi-insurgents-Group)
    0.9655))

```

The rules that this strategy yields for LebaneseHizballah are also much simpler than one derived in the multiclass case:

```

(implies
  (and
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT Lebanon))
  (probability (perpetrator ?EVENT LebaneseHizballah) 0.962))

(implies
  (and
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT Syria))
  (probability (perpetrator ?EVENT LebaneseHizballah) 0.9756))

```

So is the rule for the Asom group:

```
(implies
  (and
    (unknownSentence (isa ?EVENT KillingPerson))
    (isa ?EVENT TerroristAct)
    (eventOccursAt ?EVENT India))
  (probability
    (perpetrator ?EVENT
      TerroristOrganization-United-Liberation-Front-of-Asom)
    0.8734))
```

Not only are these rules simpler, they also seem to capture more intuitive and plausible generalizations. These are similar in nature to what an informed human expert might assert, and they are comprehensible to human experts and non-experts. To the extent that "KDD [Knowledge Discovery in Databases] is the non-trivial process of identifying valid, novel, and potentially useful, and ultimately *understandable* patterns in data" (Fayyad et al, 1996; emphasis added), these rules more successfully achieve the goals of KDD, particularly with respect to the criterion of understandability. Because these rules have been induced from data, they also have strong empirical support, and counterexamples that a human expert might not have considered are automatically taken into account. In this sense, they are more demonstrably valid and hence potentially more useful than the rules that a human expert might assert. These rules were also unknown to us (and, more importantly, to the Cyc AKB) before they emerged from the data, so they are novel as well.

Generalizing to larger datasets and avoiding the closed world assumption

The rules extracted from separate decision trees for each classifier (the multiple-binary model) have already been shown to be qualitatively superior to the rules extracted from a single decision tree that globally classifies all perpetrators (the single-multiclass model). Is the multiple-binary model also quantitatively superior to the single-multiclass model? We hypothesized that it might be, on the following grounds. Decision trees are built using the "closed world assumption": that the set of values present in the dataset exhausts the set of possible values. But in any large-scale domain, there will be cases in the world that haven't been seen, let alone included in a training set, so the closed world assumption is false. It might be expected that the multiple-binary model would be less reliant on this false assumption, because it provides *positive characterizations* of the perpetrators, rather than *negative characterizations*. For example, recall one of the antecedents in one of the rules concluding perpetration by LebaneseHizballah in the single-multiclass model, stating that the location of the event is *not* Greece, *not* Pakistan, *not* Sudan, etc.:

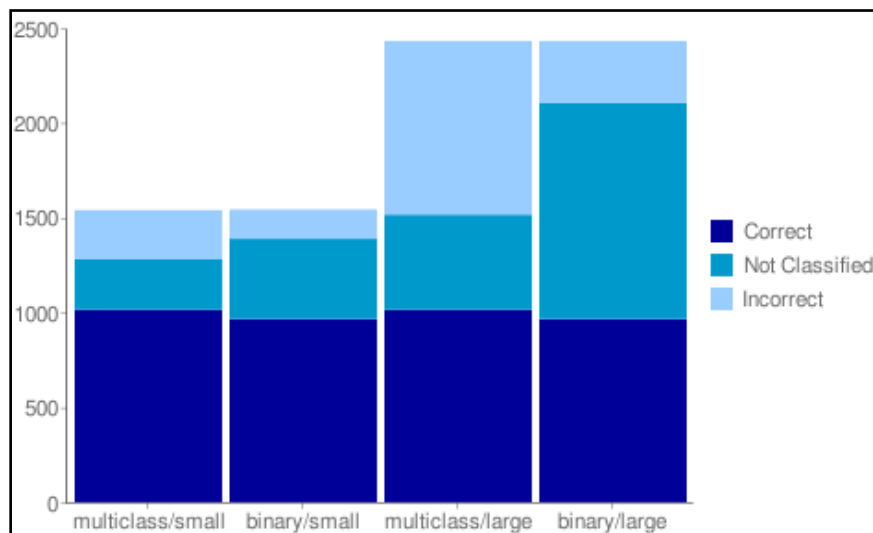
```
(different ?LOC
  Greece Pakistan Sudan Austria Italy Lebanon India Iraq)
```

In contrast, the multiple-binary model has two rules, one with an antecedent stating that the event *did* take place in Lebanon, and another one with an antecedent stating that the event *did* happen in Syria. In a closed world, negative characterizations can be equivalent to positive characterizations. But in an open world, these two kinds of characterizations are not equivalent. Suppose that a new event comes along, which takes place in Finland. None of the events in the training data took place in Finland. The single-multiclass model, with its negative characterizations, would lump this event into the *LebaneseHizballah* category, because Finland is not among the countries eliminated by the antecedent. In contrast, the multiple-binary model, with a positive characterization, would avoid this trap. In general,

we expect that a single-multiclass model is more likely to over-generalize than multiple-binary models.

To test this hypothesis, we evaluated the accuracy of the rule sets extracted from the top-12 dataset both on the set of events perpetrated by the top 12 perpetrators, and on the set of events perpetrated by the top 60 perpetrators. The latter condition tests how the two models fare on larger datasets. What we are interested in is not only the number of correctly classified instances, but also the number of instances that were not classified, and the number of instances that were falsely classified ("false positives"). A high rate of false positives implies over-generalization. It is better not to classify an instance than to classify it falsely, *ceteris paribus*.

The results are shown in the graph below. The height of the bars represents a count of events. The smaller dataset, including only the events perpetrated by the top 12 perpetrators is labelled "small"; the larger dataset, including the set of events perpetrated by the top 60 perpetrators is labelled "large". Thus, "multiclass/small" stands for "the single-multiclass model, evaluated on the small dataset," etc. Note that the additional cases provided by the top-60 set involve *other* perpetrators than those in the top-12 sets, so the top-12 classifier should not be expected to classify them correctly; success for these additional events lies in not incorrectly classifying them as having been performed by one of the top 12 perpetrators.



The graph illustrates that the single-multiclass model is *far more susceptible to false positives* (incorrect classifications, indicated with light blue) in the expanded dataset than the multiple-binary model. They perform approximately equally well on the small dataset, but their generalization performance differs greatly. This confirms our hypothesis.

Note that all of the decision trees in question were built using an n -fold cross-validation technique which prunes nodes from the tree that indicate over-fitting. Despite the use of this technique, the single-multiclass model is still prone to severe over-fitting, compared to the multiple-binary model.

The lesson we can learn from this is that even though decision trees are built based on the closed world assumption, the models we extract from them can be made robust against this assumption. Doing so only requires creating separate decision tree models for each level of the dependent variable, and extracting rules that conclude to the target category. The rules extracted through this method will tend to positively characterize the class in question, and therefore be more robust in the open world.

The rules in action

Once entered into the KB, the learned rules can be used to answer queries about likely perpetrators. For example, one can ask the following query in order to find out the most likely perpetrators of the terrorist attack in the West Bank on March 26th, 2004.

```
(probability
  (perpetrator TerroristAttack-March-26-2004-West-Bank ?PERP) ?PROB)
```

One of the bindings returned is TerroristOrganization-Hamas, who, thanks to the newly imported rules, committed the act with 93% likelihood according to Cyc. In order to understand how that conclusion was reached, it is useful to view the *justification* for that answer, which gives the facts and rules that were used to conclude it. The justification is shown in the screenshot on the next page.

The query under "EL Query" can be paraphrased, "who are the likely perpetrators of the March 26th 2004 attack in the West Bank?" The answer is: Hamas, with 93% probability. The reasoning by which Cyc made this conclusion is given under "Full Justification."

The first rule under "Full Justification" can be paraphrased, "For terrorist events that occur in the Palestinian West Bank or Gaza strip after 2003, the probability that the perpetrator is Hamas is 0.93." Cyc concludes that this rule applies to the event in question using ground facts about the event and hierarchical subsumption reasoning.

Conclusion

As a result of this work, Cyc is able to analyze its own content for statistical generalizations, and provide reasonable probabilistic estimates for answers it deduces. This greatly enhances the inferential power of the knowledge base.

The use of multiple-binary models not only increases the accuracy of Cyc's estimates, but also makes it possible for Cyc to justify its answers in terms understandable to a user, because of the simple and intuitive nature of the rules extracted.

Inference Answer Full Justification [58770.0.0.0] for answer [58770.0.0]**Mt** : TKBProbabilityRules-Top12-Positive-Mt**EL Query** :

(probability

(perpetrator TerroristAttack-March-26-2004-West-Bank ?PERP) ?PROB)

Answer Bindings :

?PERP → TerroristOrganization-Hamas

?PROB → 0.9337

Full Justification :

●(implies

(and

(isa ?EVENT TerroristAct)

(eventOccursAt ?EVENT PalestinianWestBankAndGaza)

(dateOfEvent ?EVENT ?DATE)

(laterThan ?DATE

(YearFn 2003)))

(probability

(perpetrator ?EVENT TerroristOrganization-Hamas) 0.9337)) in TKBProbabilityRules-Top12-Positive-Mt

●M(dateOfEvent TerroristAttack-March-26-2004-West-Bank

(DayFn 26

(MonthFn March

(YearFn 2004))))

in (ContextOfPCWFn TKBFactEntrySource-WireService-Israel-Thwarts-Hamas-Retaliatory-Attacks)

:EVAL (laterThan

(DayFn 26

(MonthFn March

(YearFn 2004)))

(YearFn 2003)) in TKBProbabilityRules-Top12-Positive-Mt

●(isa TerroristAttack-March-26-2004-West-Bank TerroristAttack) in SAICLegacyAssertionsMt

●(genls TerroristAttack TerroristAct) in UniversalVocabularyMt

●M(eventOccursAt TerroristAttack-March-26-2004-West-Bank WestBank)

in (ContextOfPCWFn TKBFactEntrySource-WireService-Israel-Thwarts-Hamas-Retaliatory-Attacks)

●(transitiveViaArgInverse eventOccursAt geographicallySubsumes 2) in UniversalVocabularyMt

●(genlInverse territoryOf geographicalSubRegions) in DualistGeopoliticalMt

●(genlPreds geographicalSubRegions geographicallySubsumes) in UniversalVocabularyMt

●(transitiveViaArgInverse eventOccursAt geographicalSubRegions 2) in UniversalVocabularyMt

●(genlPreds territoryOf geographicalSubRegions) in DualistGeopoliticalMt

●(territoryOf WestBank

(TerritoryFn WestBank)) in WorldGeographyMt

●(isa genlPreds TransitiveBinaryPredicate) in UniversalVocabularyMt

●(genlPreds geographicalSubRegionsOfCountry properGeographicalSubRegions) in UniversalVocabularyMt

●(genlPreds properGeographicalSubRegions geographicalSubRegions) in UniversalVocabularyMt

●(geographicalSubRegionsOfCountry PalestinianWestBankAndGaza WestBank) in WorldGeographyDualistMt

References

- Allwein, E. L., Schapire, R. E., & Singer, Y. (2000). Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. In *International Conference on Machine Learning*.
- Cortes, C., & Vapnik, V. N. (1995). Support Vector Networks. *Machine Learning*, 20, 273-297.
- Davidson, D. (1967). The Logical Form of Action Sentences. In N. Rescher, *The Logic of Decision and Action* (pp. 81-95).
- 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.
- Dietterich, T. G., & Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2, 263-286.
- Fayyad, U., Pietetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. (1996). *Advances In Knowledge Discovery and Data Mining*. Menlo Park, CA: AAAI Press.
- Freund, Y., & Schapire, R. E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*. Bari, Italy.
- Nilsson, N. J. (1965). *Learning Machines*. New York.
- Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Francisco, CA, USA.
- Quinlan, J. R. (1987). Generating production rules from decision trees. In *Proceedings of the Tenth International Conference on Artificial Intelligence* (pp. 304-307). Los Altos, CA:.
- Russell, S. J., & Norvig, P. (2001). *Artificial Intelligence: A Modern Approach*. Cambridge, MA: MIT Press.
- Schapire, R. E., & Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37(3), 297 - 336.
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*.
- Witten, I. H., & Frank, E. (1999). *Data Mining: Practical Machine Learning Techniques and Tools*. San Diego: Academic Press.