Uniting *a priori* and *a posteriori* knowledge: A research framework

Michael Witbrock, Elizabeth Coppock, and Robert Kahlert*
Cycorp, Inc.
{witbrock,ecoppock,rck}@cyc.com

Abstract

The ability to perform machine classification is a critical component of an intelligent system. We propose to unite the logical, *a priori* approach to this problem with the empirical, *a posteriori* approach. We describe in particular how the *a priori* knowledge encoded in Cyc can be merged with technology for probabilistic inference using Markov logic networks. We describe two problem domains – the Whodunit Problem and noun phrase understanding – and show that Cyc's commonsense knowledge can be fruitfully combined with probabilistic reasoning.

1 Introduction

Machine classification is a general problem of fundamental importance to the field of artificial intelligence. The ability to harness the vast amount of information freely available on the World Wide Web, for example, depends on technology for solving the *entity resolution* problem: Determining whether two expressions refer to the same entity. Classification is important in military and law enforcement domains as well; consider the *Whodunit Problem:* given features of a criminal act, who is the most likely perpetrator? Classification problems in natural language processing include *word sense disambiguation* and *noun phrase understanding*: in the phrase, *Swiss bank,* what sense of *bank* is involved, and what relation to Switzerland does the referent have? With good machine classification technology, it will be possible to solve many important problems across a wide range of domains.

Our research agenda is to develop systems that are capable of taking into account both empirical statistics and *a priori* knowledge in order to solve classification problems. *Description logic* [Baader *et al.*, 2003] is an example of a purely logical, *a priori* approach to the problem of classification. Description logic provides a medium for encoding facts about the real world, and can be used to classify objects based on their attributes. This approach, however, is fundamentally too brittle; failure to meet any one of the conditions for classifying an object into a particular class is equivalent to meeting none of the conditions.

On the other end of the spectrum are machine-learning techniques. This type of approach succeeds in being more flexible than approaches like description logic, by having weighted constraints that may combine in a gradient fashion. The pure machine-learning approach suffers, however, from being overly reliant on having large quantities of training data. Training data is often lacking: It is costly to produce labelled data, and even labelled data may be sparse for other reasons. For instance, because most words are infrequent (by Zipf's law), training data for many natural language processing tasks is likely to be missing. Moreover, information that is already known should not have to be re-learned; it should be possible to combine what is known already with knowledge gained from empirical statistics.

In recent years, the gap between statistical and logical approaches to classification has begun to narrow. In the field of information extraction, statistical and non-statistical methods have been combined, for example in the TextRunner system [Banko and Etzioni, 2008] and SOFIE [Suchanek *et al.*, 2009]. The fields of *relational data mining* [Dzeroski and Lavrac, 2001] and *statistical relational learning* combine ideas from probability and statistics with tools from logic and databases [Getoor and Taskar, 2007]. A wide variety of techniques within these field have been developed, such as probabilistic relational models, knowledge-based model construction, and stochastic logic programs, and many of these techniques are special cases of *Markov logic* [Domingos and Richardson, 2007], [Domingos *et al.*, 2006]. In Markov logic, weights are attached to arbitrary formulas in first order logic, defining a probability distribution over possible worlds [Richardson and Domingos, 2006].

We propose to integrate Markov logic with the Cyc project, a large-scale effort to represent commonsense knowledge. The Cyc system cannot trivially be converted into a Markov logic network, because the Cyc knowledge base is quite large, and Cyc uses higher order logic. However, it is possible to create Markov logic networks over subsets of the Cyc knowledge base, and to bridge these two resources in a way that usefully combines logical and statistical approaches to artificial intelligence.

2 Background

2.1 Cyc

For over 20 years, the Cyc project has been devoted to the development of a system that is capable of reasoning with com-

monsense knowledge. At the core, Cyc consists of a powerful inference engine combined with a knowledge base (KB) that contains over 6 million assertions. These assertions are expressed in a language (CycL) based on first-order logic, enhanced by a quoting mechanism and higher-order extensions [Matuszek *et al.*, 2006]. In normal inference, the assertions in the Cyc KB function as "hard constraints" in the sense that if a formula contradicts an existing fact (within a given context), it is considered simply to be false. Thus, for the most part, Cyc represents the logical, symbolic, *a priori* approach to artificial intelligence.[1]

A portion of the information in the Cyc KB is *taxonomic,* expressing (i) the class membership of terms, using the binary predicate `isa`, which relates an *instance* to a *collection* e.g. (`isa Snoopy Dog`), where `Snoopy` is an individual dog and `Dog` stands for the collection of all dogs; (ii) the subsumption relationships among those classes, expressed with `genls`, relating a *subcollection* to a *supercollection* e.g. (`genls Dog Animal`); (iii) disjointness information, expressed with `disjointWith`, which holds of collections that do not share any members. Cyc predicates (including `isa` and `genls`) are associated with *definitional* information, which constrain the types of entities that may appear as arguments to the predicate. Consider some of the argument constraint information for the predicate `biologicalMother` (read "has as the biological mother").

(`arg1Isa biologicalMother Animal`)
(`arg2Isa biologicalMother FemaleAnimal`)

The argument constraint information states that the notion "has as a biological mother" is defined for pairs of instances whose first member is an `Animal`, and whose second member is a `FemaleAnimal`. Definitional information, combined with the taxonomic hierarchy, makes Cyc into a higher-order system.

Another higher-order feature of Cyc is that predicates are also arranged in a generalization hierarchy; `biologicalMother` is a more specific predicate than `relatives`. This relation between the two predicates is expressed with the second-order predicate `genlPreds` as follows:

(`genlPreds biologicalMother relatives`)

This means that `biologicalMother` inherits all of the constraints on the predicate `relatives`, including the following rule (CycL variables, noted with question marks, are implicitly universally bound by default):

(`implies`
(`isa ?COL BiologicalSpecies`)
(`interArgIsa1-2 relatives ?COL ?COL`))

The predicate `interArgIsa1-2` specifies a type constraint on one argument, given the type of another. This rule requires, for example, that if X is Y's relative, and X is a bird, then Y is also a bird.

This higher-order information is used extensively by the Cyc inference engine to prune search when answering queries. The reduction in search space makes it feasible to perform inference over a KB of the size of Cyc. This means that the Cyc KB cannot be converted as a whole into Markov logic, but it is possible to use this higher-order information to identify subsets of the KB with which to build Markov logic networks.

2.2 Markov Logic Networks

Markov logic is a language that unifies first order logic with probabilistic graphical models [Richardson and Domingos, 2006]. In Markov logic, logical formulas are associated with weights. Intuitively, the higher the weight is for a given formula, the less likely it is to be contradicted. Formally, weights are interpreted using a Markov logic network, which defines a probability distribution X over assignments of truth values to propositional variables, or *worlds*. Given a set of formulas and their associated weights, the probability of a world x is defined as:

$$P(X = x) = \frac{1}{Z} \exp(\Sigma_{i=1}^{F} w_i n_i(x))$$

where F is the number of formulas, Z is a normalization constant ensuring legal probabilities, w_i is the weight of the ith formula, and $n_i(x)$ is the number of true groundings of the ith formula in x. This means that the more times a world violates a formula, the less likely the world is (when the weight is positive), and the higher the weight, the stronger the effect. When the weight is infinite, violations of the formula are impossible; this is how "hard constraints" are modelled.

Software for weight learning and inference with Markov logic networks is provided through the Alchemy system [Kok *et al.*, 2007]. Alchemy is a flexible software package providing generative and discriminative methods for weight learning and several methods of performing probabilistic inference, including MC-SAT, Gibbs Sampling, and Belief Propagation (ibid). In what follows, we describe a framework for integrating Cyc with Alchemy.

3 Merging Cyc with Markov Logic

3.1 The Whodunit Problem

The Cyc Analyst's Knowledge Base (AKB) is a portion of the Cyc KB that contains over 4500 events of terrorism, with information about each event including the type of attack, the location, and the agent. Given facts about an event, the goal is to predict who was the perpetrator – the *Whodunit Problem*.[2]

[1] Some assertions in Cyc are defeasible; CycL contains five possible truth values: *monotonically false, default false, unknown, default true,* and *monotonically true.* Default assertions can be overridden when two rules conclude P, but one concludes that P is monotonically true and the other concludes that P is default false. Then, all else being equal, Cyc sets the truth value of P to the one suggested by the monotonic rule [Panton *et al.*, 2006]. However, formulas are not associated with probabilities in Cyc.

[2] Several approaches to this problem were presented by [Halstead and Forbus, 2007].

In addition to specific facts about specific events, the AKB contains a rich hierarchy of event types; for example, a CarBombing is a type of Bombing, which is a type of IncurringPhysicalDamage, etc. It also contains a rich hierarchy of agent types. For example, any UrbanGuerillaGroup is also a RevoltOrganization, and by virtue of that, a PoliticalOrganization. These relationships are stated via genls assertions in the KB, and this information can be leveraged to construct Markov logic networks that accurately model the probability distribution over possible event perpetrators.

The event type and agent type hierarchies are both part of a single collection hierarchy in Cyc, but they could in principle be separated into two hierarchies, one for agents, and one for events. It turns out that the latter approach is more efficient for Markov logic networks, because it reduces the size of the model that must be constructed. With two separate hierarchies, it is possible to rule out from consideration the possibility that a something is both a RevoltOrganization and a Bombing within the Markov logic network. It is important to prevent Markov logic networks from considering such impossible states of affairs, because the network contains a node for every grounding of every formula, and this can easily grow quite large. (In contrast, Cyc's inference engine uses the fact that agents and events are disjoint to restrict search to only those groundings that are immediately relevant to its current inference problem.) Thus, in place of Cyc's isa predicate, we introduced two separate MLN predicates, IsaA and IsaE, which relate agents to agent types, and events to event types, respectively.

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

$$\texttt{IsaE}(\texttt{event}, \texttt{event\_type})$$
$$\texttt{Perp}(\texttt{event}, \texttt{agent!})$$

The first declaration expresses that IsaE is a relation between something of type event and something of type event_type; the second declaration expresses that Perp is a relation between an event and an agent. (Note that the expressions event, event_type, and agent are types, whereas in the formulas below, the arguments of the predicates are implicitly universally quantified variables.) The exclamation point ('!') following agent indicates that there is exactly one agent who perpetrated each event; the relation is *exclusive and exhaustive* in its second argument. (This is a simplifying assumption; an event can have more than one perpetrator in principle, but assuming that this rare case is impossible has dramatic computational advantages.)

Here is an example of an MLN formula:

$$\texttt{IsaE}(\texttt{e}, +\texttt{et}) \Rightarrow \texttt{Perp}(\texttt{e}, +\texttt{a})$$

The '+' symbol makes this 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 values. This notational device makes it possible to gather statistics from the corpus during learning about what agents, specifically, tend to perform what types of events. The resulting weights associated with each formula can be used in weighted inference to predict the perpetrator of the event. For example, consider the following two formulas:

$$\texttt{IsaE}(\texttt{e}, \texttt{MortarAttack}) \Rightarrow \texttt{Perp}(\texttt{e}, \texttt{AlFatah})$$
$$\texttt{IsaE}(\texttt{e}, \texttt{MortarAttack}) \Rightarrow \texttt{Perp}(\texttt{e}, \texttt{LebaneseHizballah})$$

The weights associated with these formulas are -0.0078 and 1.16332, respectively. The contrast captures the fact that LebaneseHizballah is more prone to commit events of type MortarAttack than AlFatah is.

Using only event type and location information, it is possible to predict the perpetrator with approximately 70% accuracy. But this number can be improved by adding in "hard constraints". For example, the date of an event can be used as a soft indication of who perpetrated the event, but combined with knowledge about when a perpetrator existed, can be used to rule out a perpetrator absolutely: Events are not perpetrated by organizations/individuals that do not exist yet. For example, it is asserted in the Cyc KB that LebaneseHizballah was founded in 1982, and for a certain terrorist act, it is stated that it occured on May 25th, 1977. Reasoning with a rule stating that any agent who comes into being after an event takes place cannot be the agent of that event, it is possible to infer that LebaneseHizballah cannot have been the perpetrator of the 1977 event.

This section has demonstrated two ways in which Cyc's commonsense knowledge can be combined with probabilistic reasoning. First, Cyc provides a hierarchically structured ontology that underlies the content of the "soft constraints". Second, it provides rules that can be treated as "hard constraints".

3.2 Noun phrase interpretation

We now turn to another domain entirely, in order to demonstrate a more complex type of problem. One of the advantages presented by Markov logic networks over simpler machine learning techniques is that they jointly predict multiple variables. This is of crucial importance for the problem of *noun phrase interpretation.* For example, in a noun phrase such as *fire bomb,* (i) What does the head noun (e.g. *bomb*) mean in this context? (ii) What does the modifier (e.g. *fire*) mean in this context? (iii) What is the relation between the noun phrase and the modifier? In other words, what does the entire phrase mean? The noun phrase *fire bomb* describes a bomb that creates fire, but consider the many kinds of relations that can hold between modifiers and *bomb* (all taken from descriptions of terrorism events in the Analyst's Knowledge Base):

- *main ingredient:* petrol bomb, tear gas bomb, shrapnel bomb, nail-filled bomb
- *mechanical component:* pipe bomb
- *result:* fire bomb, smoke bomb, sound bomb
- *manner of camouflage:* car bomb, suitcase bomb, videocassette bomb, book bomb, parcel bomb
- *triggering mechanism:* time bomb, remote-control bomb

The problem of understanding noun phrases like this requires joint inference because the relation between the noun phrase and the modifier depends on the interpretation of the head noun and the modifier, but the interpretation of the head noun and the modifier can also be influenced by the relation; these problems are mutually interrelated.

Creating hand-labelled data is expensive, and there is already information about noun phrases within the Cyc knowledge base that can be used as training data. In particular, the Cyc lexicon contains thousands of lexical entries for strings such as *western film.* Under certain assumptions, this information indirectly reveals how the modifier and the noun are related. The assumptions are as follows:

(i) Western films are films. More precisely, if the meaning of the whole phrase bears the `genls` relation to some denotation of the head word, then the latter is what the head word denotes in this context. Possible denotations of *film* include 'photographic film', 'the act of filming', 'a sheet or coating', and 'movie'; since what *western film* denotes is a type of movie, it is clear that *film* means 'movie' in this context.

(ii) Western films have something to do with westernness. More precisely, if there is an assertion mentioning the meaning of the whole phrase and a concept that can be denoted by the modifier, then the latter concept is what the modifier denotes in this context. In our example, the modifier *western* can denote either the cardinal direction west, a western story, or a western conceptual work. The meaning of *western movie* is related to western stories and western conceptual works via the `genls` relation. Thus we can assume that in the phrase *western film,* the modifier string *western* has one of those two meanings.

In general, the semantic relation between the modifier and the phrase can be any binary relation. For another example, in the phrase *blackberry bush,* the relation is a `fruitOfType` relation, which holds between the meaning of *blackberry* – (`FruitFn BlackberryBush`) – and the meaning of the whole phrase: `BlackBerryBush`.

This corpus can be used as data to train a Markov logic network that simultaneously disambiguates the head noun and the modifier and identifies the relation that holds between the meaning of the modifier and the meaning of the phrase. Here is a sample set of type declarations for predicates capturing the relevant information about noun phrases:

$\texttt{reln}(\texttt{np}, \texttt{reln})$
$\texttt{modDenotes}(\texttt{np}, \texttt{sense})$
$\texttt{headDenotes}(\texttt{np}, \texttt{sense})$
$\texttt{genls}(\texttt{sense}, \texttt{concept})$
$\texttt{denotes}(\texttt{word}, \texttt{sense})$
$\texttt{modWord}(\texttt{np}, \texttt{word})$
$\texttt{headWord}(\texttt{np}, \texttt{word})$
$\texttt{modGenls}(\texttt{np}, \texttt{concept})$
$\texttt{headGenls}(\texttt{np}, \texttt{concept})$
$\texttt{wellFormedArg1}(\texttt{sense}, \texttt{reln})$
$\texttt{wellFormedArg2}(\texttt{sense}, \texttt{reln})$

These capture what the semantic relation involved is (`genls`, `fruitOfType`, etc.), the meaning of the head and the modifier, the taxonomic hierarchy and denotational information about each word, how the meanings of modifier and the head noun fit into the taxonomic hierarchy, and argument type constraints on semantic relations.

Here are sample hard constraints that could be expressed using these predicates:

$\texttt{modDenotes}(\texttt{p}, \texttt{s}) \wedge \texttt{modWord}(\texttt{p}, \texttt{w}) \Rightarrow \texttt{denotes}(\texttt{w}, \texttt{s})$.
$\texttt{headDenotes}(\texttt{p}, \texttt{s}) \wedge \texttt{headWord}(\texttt{p}, \texttt{w}) \Rightarrow \texttt{denotes}(\texttt{w}, \texttt{s})$.
$\texttt{modGenls}(\texttt{p}, \texttt{c}) \wedge \texttt{modDenotes}(\texttt{p}, \texttt{s}) \Rightarrow \texttt{genls}(\texttt{s}, \texttt{c})$.
$\texttt{reln}(\texttt{p}, \texttt{r}) \wedge \texttt{modDenotes}(\texttt{p}, \texttt{s}) \Rightarrow \texttt{wellFormedArg1}(\texttt{s}, \texttt{r})$.

These express the following constraints: words only denote senses that they have; if the modifier denotes a given type of concept and the modifier has a given sense, then that sense is that type of concept; if the phrase involves a certain semantic relation, and the modifier has a given sense, then that sense fits the argument constraints for the semantic relation.

Here are sample per-constant formulas that yield soft constraints:

$\texttt{modGenls}(\texttt{p}, +\texttt{c}) \Rightarrow \texttt{headGenls}(\texttt{p}, +\texttt{c}')$
$\texttt{headGenls}(\texttt{p}, +\texttt{c}) \Rightarrow \texttt{modGenls}(\texttt{p}, +\texttt{c}')$
$\texttt{modGenls}(\texttt{p}, +\texttt{c}) \Rightarrow \texttt{reln}(\texttt{p}, +\texttt{r})$
$\texttt{headGenls}(\texttt{p}, +\texttt{c}) \Rightarrow \texttt{reln}(\texttt{p}, +\texttt{r})$

These will capture regularities such as "the head tends to denote a kind of person when the modifier denotes an ethnicity"; "the relation tends to be a `fruitOfType` relation when the modifier denotes a fruit," etc.

The resulting model combines the Cyc ontology, logic, and probability to solve the problem of noun phrase interpretation. Taxonomic information (`isa` and `genls`) underlies the content of the soft constraints, and definitional information (argument type constraints) comes into play in defining hard constraints. These pieces of information are combined with statistics in a single unified model of this restricted domain.

4 Conclusion

We believe that a successful strategy for merging background knowledge and empirical statistics lies in unifying Cyc's strengths with those of Markov logic networks. Our initial case studies have shown that the knowledge in Cyc comes into play both as a way to categorize the data so that useful statistics can be computed, and as a way of enforcing regularities in the model that must hold. We have addressed efficiency issues by finding ways of minimizing the size of the Markov logic network that is constructed, for example, by using two separate predicates where Cyc uses only one. We see great potential in creating a bridge these two resources, for the sake of solving these problems, along with countless other problems of classification.

References

[Baader *et al.*, 2003] Franz Baader, Diego Cavanese, and Deborah L. McGuinness, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Banko and Etzioni, 2008] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June 2008. Association for Computational Linguistics.

[Domingos and Richardson, 2007] Pedro Domingos and Matt Richardson. Markov logic: A unifying framework for statistical relational learning. In Getoor and Taskar [2007], pages 339–371.

[Domingos *et al.*, 2006] Pedro Domingos, Stanley Kok, Hoifung Poon, Matt Richardson, and Parag Singla. Unifying logical and statistical AI. In *Proceedings of the Twenty-First National Conference On Artificial Intelligence*, pages 2–7. AAAI Press, Boston, MA, 2006.

[Dzeroski and Lavrac, 2001] Saso Dzeroski and Nada Lavrac, editors. *Relational Data Mining*. Springer, Berlin, 2001.

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

[Halstead and Forbus, 2007] Daniel T. Halstead and Kenneth D. Forbus. Some effects of a reduced relational vocabulary on the whodunit problem. In *Proceedings of IJCAI-2007*, Hyderabad, India, 2007.

[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, 2007. http://alchemy.cs.washington.edu.

[Matuszek *et al.*, 2006] Cynthia Matuszek, John Cabral, Michael Witbrock, and John DeOliveira. An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, 2006.

[Panton *et al.*, 2006] Kathy Panton, Cynthia Matuszek, Douglas Lenat, Dave Schneider, Michael Witbrock, Nick Siegel, and Blake Shepard. Common sense reasoning – from Cyc to intelligent assistant. In Y. Cai and J. Abascal, editors, *Ambient Intelligence in Everyday Life*, pages 1–31. Springer-Verlag, Berlin, 2006.

[Richardson and Domingos, 2006] Matt Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006.

[Suchanek *et al.*, 2009] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: a self-organizing framework for information extraction. In *Eighteenth International World Wide Web Conference (WWW2009)*, 2009.