Representing Knowledge Gaps Effectively

Alan Belasco, Jon Curtis, Robert C. Kahlert, Charles Klein,
Corinne Mayans, Pace Reagan

Cycorp Inc, 3721 Executive Center Drive S#100, Austin TX 78731
{belasco,jonc,rck,klein,cmayans,pace}@cyc.com

Abstract. In knowledge acquisition, one typically encounters two difficult situations: First, there are times when the system requests information that, due to a lack of information, the user is not in a position to provide at the level of precision requested. Second, there are situations where the system cannot capture information at the level of precision the user wishes to provide. We describe the techniques that have been developed for CYC to address these two cases during the extension of a variety of domains.

Introduction

In this paper we describe a Subject Matter Expert (SME) usable knowledge acquisition system that is used to extend the breadth of CYC's common sense knowledge verticals, some of them by half a million facts over the last two years. Increasing knowledge base verticals requires tools that allow for very high knowledge entry rates. Previous research suggests that entry rates drop when either the users or the system hit upon a knowledge gap. We therefore developed and deployed strategies to mitigate either contingency. We achieved average entry rates of 96 facts per hour, for entry sessions four hours long on average.

Knowledge Acquisition System Description[1]

For the purposes of extending CYC's knowledge verticals, a client-server knowledge acquisition system was developed. As the backend server, we used Cycorp's CYC knowledge base, inference engine and natural-language subsystem; as the front-end client, we developed a template-based knowledge acquisition tool known as the "Factivore." In the following, we will describe both components briefly.

[1] The key elements of the components described were developed by David Baxter, Tony Brusseau, Chris Deaton, Keith Goolsbey, Kevin Knight, Doug Lenat, Pierluigi Miraglia, Stephen Reed, Dave Schneider, Anuroopa Shenoy, Michael Witbrock, Ming Xu and the authors.

Cyc 10

CYC Version 10 is a knowledge base system designed for efficiently manipulating and reasoning with a large ontology (over 2.4 million assertions, 200000 terms and 9000 relationships). Its symbolic representation language, CycL, is based on first order predicate calculus with higher-order and modal extensions.

CycL contains both atomic and functionally-defined terms. As part of its higher-order extensions, CycL allows assertions in the Cyc knowledge base to occur as terms (arguments) for meta-assertions. CycL assertions are grouped into reasoning contexts called *micro-theories*, which in turn form hierarchies of assumption inheritance.[2]

The "Factivore" Knowledge Acquisition Tool

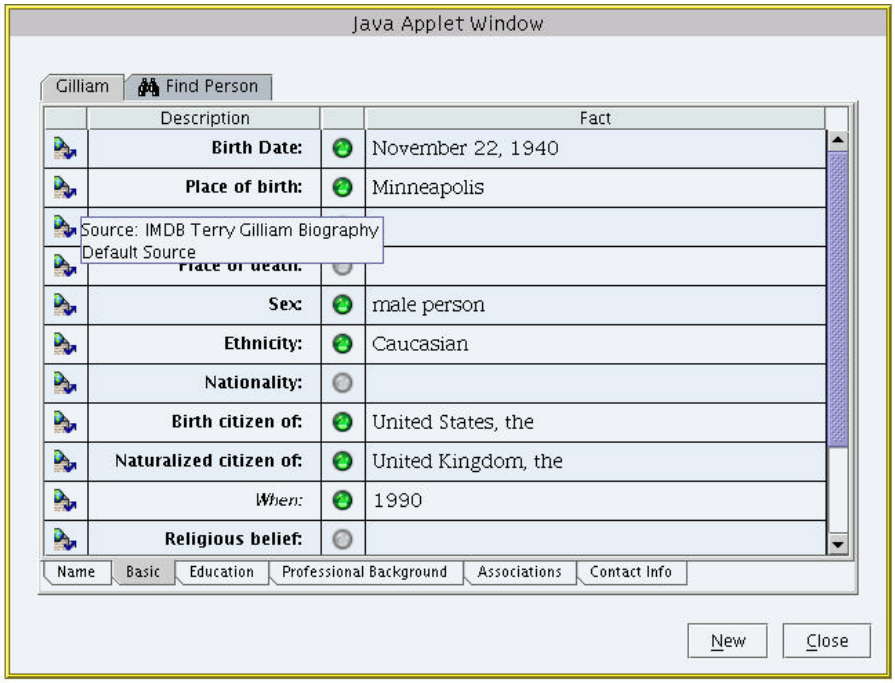

Fig. 1. The "Factivore" Knowledge Acquisition Interface.

The "Factivore" knowledge acquisition interface (see Figure 1) is the main tool employed for extending knowledge verticals. It presents a tabular user interface that is driven by fact-entry templates described in CYC's knowledge base.

---

[2] A substantial, taxonomic portion of Cyc's knowledge has been exported into OWL; a smaller version of the knowledge base, OpenCyc, is already freely available and freely usable (at http://www.cyc.com/2004/06/04/cyc).

One or more of the rows in the presented form correspond to argument positions in the underlying fact templates; the current topic of discussion, the so-called *focal term*, is implied by the title bar. Users either select possible values from drop-down lists or enter them, in which case CYC's natural language parsing mechanisms attempt to map the string entered via CYC's English lexicon to the appropriate CycL term. [2][3]

As previously indicated, all properties of the individual fact templates are described in the knowledge base itself, including the organization of the tabbed panels, the presentation order of the templates, the types of entities that are permissible, whether new entities can be introduced or only existing ones selected, and how often a fact can be stated. In the future, CYC will adjust the interface as it understands more about the focal term, pre-filling input fields or adding additional questions.

Because knowledge verticals typically combine information from a variety of document sources, each individual fact can be associated with a source. These sources are also described via appropriate "Factivore" templates and represented in the knowledge base. Inside CYC, each source is associated with it a micro-theory that represents the semantic content of the document; source assignments determine into which micro-theory the facts are placed (see tool-tip in Figure 1).

Representing what Users do not know

When entering information from individual sources, Subject Matter Experts (SMEs) attempt to capture their meanings as accurately and with minimal information loss. Consider the following example from a fictitious source, selected for expository purposes: "Brandishing AK-47s, the kidnappers pulled the diplomat from his car."

While this sentence names the type of weapon employed, it provides no information about the exact number of weapons that the kidnappers employed. However, the underlying fact template, as represented in CYC, expects not only the weapon type but also a non-negative integer.

```
(relationInstanceExistsCount Kidnapping4 weaponsUsed
AK47Gun number)
```
[3]

At this point, SMEs have three choices: They can skip the question; they can enter a number by making an educated guess; or, they can enter "unknown." Skipping the question altogether violates the stated goal of capturing the meaning. One would want this kidnapping to be in the result set if CYC was queried for criminal activities involving AK-47s. Making an educated guess is a more helpful but equally inaccurate description of the source, since it omits the fact that there was a lack of information in the source. The remaining choice is then to allow the SME to enter "unknown" and to make this information productive.[4]

The knowledge base description of a fact template includes information describing what should be done when a SME's input is malformed in specific ways. These repair

[3] Meaning that, in the event Kidnapping4, the count of AK47s playing the role of the weapons used was *number*.

[4] The user interface of the "Factivore" provides a way for inserting a sentinel "unknown" value, which is mapped to a value appropriate for the specific template to forestall ambiguity.

strategies are called *reformulation rules* and are semantically guided descriptions for how to perform syntactic transformations on the input. These rules target a CYC component appropriately called the *Reformulator*, a generic CycL to CycL transformer that uses semantic indicators to select applicable and valid transformations from a set of rules encoded in the knowledge base.

In the current situation, the Reformulator would produce the strictly weaker but inferentially productive fact:

```
(relationInstanceExists Kidnapping4 weaponsUsed AK-47)
```

Adding this fact ensures that this crime will be in the result set when querying for kidnapping events involving AK-47s, yet it is syntactically different – and thereby appropriately semantically distinct – from the case where exactly one AK-47 was employed:

```
(relationInstanceExistsCount Kidnapping4 weaponsUsed
AK47Gun 1)
```

In addition, the "Factivore" makes a meta-assertion about the added fact, linking it to the original SME-authored formula and to the reformulation rule used to affect the syntactic transformation. Using this two-step method of transforming the user's input and recording the transformation itself allows the "Factivore" to successfully add knowledge contained in the source – that AK47s were used in the kidnapping – yet retain what amounts to the SME's identification of a knowledge gap. In effect, the meta-assertion and its subject retain the knowledge that the source did not provide all of the information the system desired. [5]

Representing what CYC does not know

Acquiring "Need to Know" Information

In a situation of knowledge acquisition, the system will very often not know what the SME has to teach. Typical entities for which the system is unlikely to have complete coverage are persons, companies, cities and villages. Therefore, it is of particular importance to deal gracefully with missing knowledge, especially when high rates of knowledge acquisition are desired.

In some situations it is appropriate for the system to assist the SME in drilling more deeply into a subject immediately, such as when describing the board members of a commercial organization; here the "Factivore" can simply create the missing officers and launch additional knowledge acquisition windows for describing them. In practice, one wants to restrict this approach to the important cases, because feedback indicates that topic-shifts disrupt the SMEs' focus and reduce the overall entry rate.

Conversely, there are situations where unrecognized entities indicate an entry error, such as misspellings. Whether the CYC system enforces that its input be valid and

[5] Should the SME find the number later, then CYC's Truth Maintenance System will automatically remove the dependent meta-assertion when the old assertion is replaced during the edit operation.

recognized, or not is again dependent on the template definition in the knowledge base, which in turn is driven by whether CYC knows that it can make the assumption that it knows all of the instances of a type.[6]

Again, it may not be worth the time of a subject matter expert to flesh out minor details. For these cases it is sufficient to allow the SME to create a stub-term that denotes the appropriate information and has the appropriate type to make the resulting fact semantically well-formed (see row three in Figure 1):

```
(placeOfBirth JosephStalin
  (InstanceNamedFn "Gori, Georgia"
     GeographicalRegion))
```

A stub-term consists of a type-specifying "function" and its arguments: a natural language string used to introduce the entity, and a collection that "types" the resulting term – in this case a geographical region named "Gori, Georgia." The word "function" is employed loosely here; there is no guarantee that the string designation of the entity is in fact unique (and therefore invertible), as a true function would require: the string "Gori" may denote any of a number of cities.

The representation makes use of CYC's ability to reason about the result types of functional terms; we simply state that the type of entity denoted by the functional term is the type of the 2nd argument. This is, however, insufficiently descriptive in the cases where the SMEs want to introduce a new subtype, such as a new type of weapon. Here we employ a different, type-denoting function:

```
(relationInstanceExists Kidnapping4 weaponsUsed
(SubCollectionNamedFn "Stazer" Weapon))
```

Stub-terms were originally developed for semantically integrating external knowledge sources such as SQL databases [1] for use during inference. In databases, string denotations are the predominant type of information, and many terms have to be classified while processing an SQL result set without access to SMEs to answer questions.

Resolving the "Need to Know" Information

At this level of representation, it is no longer possible to distinguish misspellings from new entities; therefore, all stub terms are reviewed and triaged by the ontological engineers at Cycorp overseeing the construction of the knowledge verticals. The stub-terms therefore function as markers in the workflow between the SMEs and the ontological engineers. The proper typing and the descriptive natural language of the functional term, plus its example usage make it straightforward for the ontological engineer to properly place the missing term in CYC's ontology.

Reviewing these terms provides critical feedback on what the SMEs are trying to represent and helps to identify limitations in the usability of the Factivore forms.

A particularly noteworthy class of stub terms are neither ontology gaps nor input errors but cases of SMEs pushing the envelope of the template infrastructure. Consider the following example:

---

[6] For example, if the SME attempted to supply the month in which a meeting took place, a misspelling of "December" would not trick the CYC system into thinking that a new month has been introduced, since it knows – and knows that it knows – all of the Julian months.

```
(objectFoundInLocation MackTheKnife (InstanceNamedFn
"some bar in London or Paris" GeographicalRegion))
```

An appropriate CycL representation of the SME's intended statement would be:

```
(thereExists ?BAR
  (isa ?BAR Bar)
  (or (geographicalSubRegions ?BAR CityOfLondonEngland)
      (geographicalSubRegions ?BAR CityOfParisFrance))
  (objectFoundInLocation MackTheKnife ?BAR))
```

A solution to this class of problem is not within the scope of our current research, but some preliminary approaches suggest themselves. In order to be able to transform the first representation into the second representation, it seems important to recognize that the SME is completing a natural language sentence that the presentation of the template in the "Factivore" is suggesting to them, for example "Mack the Knife is hiding in _____.", a very focused parsing problem.

If the representation of the natural language syntax tree for the suggested sentence were a property of the template, then the natural language processing system could attempt to parse the SME's completion of the sentence in the syntactic context of the overall sentence:

```
(TheDisjunctiveCoordinationSet
  (NLQuantFn-3 Some-NLAttr
   (NLNumberFn Singular
    (SubcollectionOfWithRelationToFn Bar
      in-UnderspecifiedContainer CityOfLondonEngland))
1) CityOfParisFrance)[7]
```

This reduces the problem to adding to the Reformulator's rule set so that an appropriate representation can be computed from these parts, a relatively straightforward task. Again, meta-assertions would track the dependency between what the SME entered and how the statement was interpreted and made inferentially productive by CYC.

Conclusions and Outlook

Even when a knowledge acquisition system and its users cannot completely understand each other, there is valuable information to be gained from what *can* be understood. We argue that the explicit representation of the exchange between SMEs and system can result in natural system interactions and high rates of knowledge acquisition, as well as the production of inferentially productive knowledge up to a maximum set by the incompleteness of information.

---

[7] Notice that the interpretation as shown here treats the phrase as if it had been "some bar in London or in Paris", i.e. it gets confused about the attachment.

This approach to representing knowledge gaps in the information contained in a source or the result of system misunderstanding, also serves to treat gaps as markers in the knowledge acquisition workflow; they represent to-do items for either the SME to investigate or the system to learn about.

More generally, such knowledge-gap markers can function as workflow markers for CYC itself, enabling CYC to focus its efforts in its general striving to extend what it knows. Each stub-term indicates an ontology gap that CYC can actively seek to fill. Each missing information piece is an indication that the wrong source or SME was consulted, allowing CYC to remain on the lookout for more details about the entity described. Such learning could happen through interaction with more appropriate users or even through CYC performing research on its own. Preliminary work at Cycorp is currently experimenting with ways in which CYC can try to harvest knowledge about novel terms for itself from the World Wide Web and other text sources. After all, the first step to achieving wisdom lies in knowing what one does not know.

References

1. Chip Masters, Zelal Güngördü, "Structured Knowledge Source Integration: A Progress Report". *Proceedings of the Internationa Conference on Integration of Knowledge Intensive Multi-Agent Systems (KIMAS 03).* Piscataway, N.J.: IEEE Press. (562-566), 2003.
2. Michael Witbrock, David Baxter, Jon Curtis, et al, "An Interactive Dialogue System for Knowledge Acquisition in Cyc" in *Proceedings of the IJCAI-2003 Workshop on Mixed-Initiative Intelligent Systems*, Acapulco, Aug 9, 2003.
3. Burns, K., and Davis, A. 1999. "Building and Maintaining a Semantically Adequate Lexicon Using Cyc", in E. Viegas, *Breadth and Depth of Semantic Lexicons*, Kluwer.