AAAI Fall Symposium on Question Answering Systems
Abstracts of presentations not appearing in the proceedings

Ontology Translation and Merging with OntoMorph

Hans Chalupsky
USC Information Sciences Institute

A central piece of the infrastructure of a question answering system is its knowledge base (KB) and the underlying ontology. Since constructing the KB is usually a very time-intensive effort, reuse of preexisting KBs and ontologies has the potential of significantly reducing system development time and cost. In most cases, however, reuse comes at a price, since importing a preexisting ontology almost always involves translation and adaptation operations to fit the syntactic and semantic constraints of the target system. If the translation cost is too high, it can render the reuse of a preexisting ontology nonviable.

Traditionally, ontology translation has either been performed manually with a text or KB editor or via special-purpose translation software. The manual approach is tedious, error prone, and not easily repeatable. Special purpose translation software is difficult to write and hard to maintain. These problems are exacerbated by the fact, that ontology import is often not a one-shot operation, but has to be performed repeatedly. For example, an imported ontology might be further elaborated by an independent developer warranting a repeated import to benefit from the elaboration, or, in an ontology merging situation, one might need to experiment with different translations to make the imported ontology fit with the host ontology. In such scenarios it is crucial to have automatically executable translation specifications that can easily be modified to fit the changed requirements of a repeated import.

To facilitate the rapid generation of automatic translators necessary for ontology merging and reuse, we developed OntoMorph, a rewrite system designed to match, de-structure and transform arbitrary expressions represented as syntax trees. OntoMorph provides a powerful pattern language, a semi-declarative rule language to represent rewrite rules and a rule interpreter with full backtracking. Rewrite rules can be organized into rule sets to facilitate modularity and search control. Since translations cannot always be performed based on syntactic criteria alone, OntoMorph is fully integrated with the PowerLoom knowledge representation and reasoning system. This makes it possible, for example, to perform logical inference to determine whether a particular rewrite rule is applicable or not.

Defining translators via declarative OntoMorph rewrite rules automatically documents the performed ontology transformations. This information is usually lost with manual translations or is not easily accessible with special-purpose translators. It also makes the translators easier to adapt and maintain. To date, we have successfully applied the OntoMorph system as a Cyc-to-Loom input translator for a critiquing system, and as the core technology of a translation service that translates messages between agents based on different planning ontologies.

Information Exchange Dialogues with Machines that Reason

Susann Luperfoy
Department Of Linguistics, Georgetown University
and
Information Extraction and Transport Corporation

Inherent to all information exchange dialogues is a discrepancy between what is said and what is known or believed by at least one of the dialogue agents. If you are teaching me new information, the assertions I hear you make will not map to anything in my belief system. So the dialogue involves a kind of negotiation over shared information following an interleaving of four dialogue modes: (1) Teacher asks student questions to discern the scope and structure of the student's domain understanding, and student responds to questions. (2) Teacher asserts new information to student and student acknowledges and/or asks for clarification. (3) Teacher checks comprehension by asking questions and posing tasks for the student, and student responds. (4) Teacher corrects misconceptions through one or more remedial assertions to undo false beliefs and impart new beliefs.

The prototype system described in this talk casts the human user as teacher and a spoken dialogue interface agent to the Cyc KB as the student. I will review the theoretical framework for information exchange dialogue, and a set of implementations that use this framework. The emphasis here will be on interaction between humans and automated dialogue agents that reason. The theoretical framework, based on Discourse Pegs, extends linguistic theories of semantics--namely File Change Semantics and Discourse Representation Theory-- and posits a partitioning of contextual information (accessed by any dialogue agent at run time) into three interconnected tiers: (1) belief system tier (2) discourse model tier (3) surface form tier. In the machine-based dialogue agent these are implemented as: (1) the ontology and reasoning engine(s) (2) a probabilistically maintained configuration of Discourse Pegs (3) the history of communicative events in the dialogue thus far.

The partitioned context representation is motivated by cognitive, linguistic, and engineering concerns, and is in service of discourse-level processes that track the evolving context, manage the dialogue, and pragmatically 'translate' between communicative events (from user or backend KB system) and resulting dialogue agent actions. A key contribution of this framework is robustness in the environment of (1) incomplete/flawed agent beliefs, (2) a dynamic world of reference, (3) imperfect speech recognition or sentence analysis, and (4) temporary misconceptions on the part of either the user or the automated dialogue agent itself.

I will show how this framework supports the real-time processing of spontaneous language phenomena: multi-turn dialogue sequences, attentional focus phenomena, complex dependence relationships between referring expressions, and multi modality and how the dialogue manager and pragmatic translation components address the handling of dialogue disruptions, and meta-dialogue segments of different sorts.

How to evaluate your system daily and still get real work done?

Eric Breck
John Burger
Lisa Ferro
Lynette Hirschman
David House
Marc Light
Inderjeet Mani
(MITRE)

It would be nice to know if the work you did yesterday improved the performance of your system. However, if you spend the day evaluating the system performance by hand today, tomorrow's system will be no better than today's. In more formal terms, automatic system evaluation is crucial to tight, efficient development loops.

In this talk, we will report on our attempts to develop an automated evaluation system for question answering systems. The goal of our effort was to provide a meaningful quantitative score for each version of our system and to minimize the human effort required to do so. We study how well such automated scores correlate with the results of human evaluators.

Very briefly, one methodology we explore is the following: a human creates an answer key for the question test set consisting of a short answer for each question. The system response for each question is compared to the answer key for that question using word recall and precision.