Database Learning for Software Agents

Mike Perkowitz  Oren Etzioni
Department of Computer Science and Engineering, FR-35
University of Washington, Seattle, WA 98195
{map, etzioni}@cs.washington.edu

With the amount of information available rapidly outstripping the ability of individuals to use it, we wish to explore how a software agent can learn a description of an information resource (such as a database on the internet) in order to turn it into a well-understood tool at the agent’s disposal. An agent who could do this would have access to all the information it could find without having to cache the internet.

As the agent makes queries to an information resource, it will generalize from those queries and generate hypotheses about the structure and content of the database. We therefore formulate this problem as a learning problem in which the input is (1) the agent’s model — its representation of the world; and (2) a series of queries to and responses from a database. The output is a mapping from fields in the information resource to predicates in the model.

Our approach to this learning problem relies on overlap between the agent’s model and the information in the database. The agent will use its own knowledge to form hypotheses about the structure of the records. We have developed the correspondence heuristic, which states that a correspondence of tokens between the agent’s world model and the information resource indicates a correspondence between types. The agent matches the values of the fields in the database against facts in its model. The relationships that hold among these facts in the model are assumed to correspond to relationships in the database.

Suppose that the agent makes a query to staffdir, the UW personnel directory, and gets back “Oren Etzioni 206”. The agent would have facts in its model like (lastname person37 Etzioni) and (office person37 206). From this query and this knowledge, the agent could conclude that the second field of the output is lastname and the third field is office.

Our work has many similarities to structure-mapping work (Falkenhainer, Forbus, & Gentner 1986). Both approaches rely on discovering correspondences between separate domains. Structure-mapping, however, seeks correspondence between underlying structure, while the correspondence heuristic relates tokens in order to make inferences about the structure.

The correspondence heuristic is an inductive bias which can be formalized as a determination:

$$\forall (x, y)\left[T(x) \land T(y) \land (S(x) = M(x)) \rightarrow S(y) = M(y)\right]$$

$T$ is a type predicate such as “on the UW faculty”, $S$ is a syntactic predicate like “the first field in the output of staffdir $x$”, $M$ is a semantic predicate (i.e. from the agent’s model) such as “the first name of $x$”.

This formalization clearly indicates three areas for work. Learning $T$ could be handled by standard inductive learning algorithms. We assume a syntactic model of ordered fields to account for $S$. Future work may pursue other kinds of syntax, such as keyword-based syntax. The focus of our work is learning the appropriate $M$ predicate.

In particular, we have been exploring the problem of Predicate mismatch, which occurs when instances of one type in the database are instances of a different type in the model, or when relations in the database do not correspond to primitive relations in the model. For example, imagine that the agent gets back “Oren Etzioni FR-35” from a query. FR-35 is actually the mail stop of Etzioni’s department and so there is no fact to link the person Etzioni to the string FR-35 directly. Instead, the agent must realize that the entry in the database corresponds to a chain of predicates in its model linking Etzioni to Computer Science and Computer Science to FR-35. We have devised a way of doing this using a method reminiscent of spreading activation, in which a link between two tokens is found by exploring outward from the tokens until an intersection is found.

Given simplifying assumptions about the syntax, our implemented algorithm has learned staffdir as well as ls and finger (UNIX commands with tabular output can be treated as query/response databases). In the future, we will extend this to be able to handle information resources found on the World Wide Web by programs that traverse the web automatically.

References