Haystack: Per-User Information Environments
MIT9904-08

Progress Report: January 1, 2002—June 30, 2002

David Karger

Project Overview

The Haystack project is investigating the ways in which electronic infrastructure can be used to triangulate among the different knowledges of an individual, of collegial communities to which we belong, and of the world at large. Its infrastructure consists of a community of independent, interacting information repositories, each customized to its individual user. It provides automated data gathering (through active observation of user activity), customized information collection, and adaptation to individual query needs. It also facilities inter-haystack collaboration, exposing (public subsets of) the tremendous wealth of individual knowledge that each of us currently has locked up in our personal information spaces. The Haystack-NTT project involves augmenting its customization, learning and adaptation, and inter-haystack communication.

The Haystack project stands upon three research pillars. We study semi-structured databases and semantic-web ontologies as tools for representing all of the knowledge useful to an individual. We explore user interfaces that present this rich information to the user in an effective way, letting them search, navigate, and manipulate their information. And we investigate deductive agents and machine learning as tools to reduce an individuals information management burden.

Progress Through June 2002

The past six months have been devoted to fleshing out the user interface infrastructure prototyped in the previous six months. Our user interface architecture records, within our semistructured repository, information about how to render each type of object in the repository. Views of complex objects (made up of simpler objects) are constructed by properly aggregating views of their component parts---thus, for example, a view of a document may be aggregated from a view of the title for that document, a view for the author of the document, and a view of the abstract of the document. Views also provide opportunities for the user to manipulate information; for example, the user could correct the author of the document by typing over the author-view within that document.

We started this period with the infrastructure for constructing these views, and have spent this period using the infrastructure to build views for a variety of object types, including text documents, music files, people, collections, and calendars. While the view for a document is a simple list of title and author, the viewer for a calendar provides the functionality of a typical calendar manager, and the viewer for a music file provides the functionality of a music player. Much of our attention, however, has focused on collection viewing. As a primary goal in haystack is the management of information, we expect collections to play a critical role. As typical user query will return a collection of results which the user will then need to navigate to home in on the object they want. We have been investigating user interfaces that assist such navigation. For example, our collection viewer examines the objects in its collection and suggests additional attributes that can be used to refine the collection to a smaller one or divide the collection into a number of categories. We came up with the Query/Browse paradigm, which generates a Yahoo-like "table of contents" dynamically for arbitrary collections and gives users increased flexibility in terms of the way they choose to narrow their search parameters. We also developed several paradigms for presenting these collections, from simple list and icon views to slide sorter and complex directed

graph displays. Among the applications of these collection navigation techniques is a photograph collection manager.

In order to ease the development of such viewers, as well as agents that manipulate information for the user, we have been developing a programming language specifically targeted at manipulation of RDF. It embeds the natural operations of following RDF triples and adding new triples as primitives in the language, thus reducing the amount of syntax needed to carry out RDF operations. The language also compiles into RDF, reflecting the same kind of program-data equivalence as lisp. This in turn allows the compiled program to be annotated with metadata comments or execution optimization information.

We've also been solidifying the overall interface paradigm, defining the kinds of operations a user might typically carry out. We've developed drag and drop mechanisms for annotating objects with attributes. We've developed a context-menu mechanism that, when triggered on an object, looks up in the data repository and lists all the operations that might naturally be applied to that object.

Research Plan for the Next Six Months

A good deal of the next six months will be devoted to preliminary user studies. We are currently designing studies to learn how users interact with collections, how they prefer to categorize information, and what contextual clues help them navigate a semistructured data model. As we put the finishing touches on some of the Haystack "applications" such as a calendar viewer and an music organizer and player, we hope to deploy these systems with a number of users to collect initial feedback on our system.

Another important upcoming thrust is the incorporation of machine learning. Several of the Masters thesis projects planned for this year focus on machine learning. One project aims at categorizing email and, in particular, separating out spam email. Another will incorporate techniques from Winnow (described in a previous report) to seek out information on the web that is of interest to its user (with a preliminary focus on news items). Yet a third aims to take some ideas from query refinement and relevance feedback in text retrieval and apply them in the retrieval of semistructured information.