MindShare: Knowledge Sharing via Personalized Views on a Composite Ontology

by

Neil William Van Dyke

M.S., Computer Science, Brown University, 1997

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, in Partial Fulfillment of the Requirements for the Degree of Master of Science in Media Arts and Sciences at the Massachusetts Institute of Technology

February 2000

© Massachusetts Institute of Technology. All rights reserved.

Signature of Author

Program in Media Arts and Sciences
14 January 2000

Certified by Patricia Maes
Associate Professor of Media Technology
Media Arts and Sciences
Massachusetts Institute of Technology

Accepted by Stephen A. Benton
Chair, Departmental Committee on Graduate Students
Program in Media Arts and Sciences
Massachusetts Institute of Technology
MindShare: Knowledge Sharing via Personalized Views on a Composite Ontology

by

Neil William Van Dyke

Submitted to the Program in Media Arts and Sciences, School of Architecture and Planning, on 14 January 2000
in Partial Fulfillment of the Requirements for the Degree of Master of Science in Media Arts and Sciences

Abstract
Ontologies, or ways of categorizing and describing things, have application to organizing information for human access and allowing software agents to exchange knowledge. The traditional approaches to ontology development involve dedicated effort by a small team of knowledge engineers in a slow and expensive process. We show that a composite ontology of information resources can be constructed from the personal ontologies of individuals in an organization via an ongoing process that combines normal personal organizing activities with low-impedence computer mediation. The MindShare approach is designed to facilitate this process within an organization of people who wish both to organize internal and public information for their personal benefit, and to leverage information previously discovered by others within the organization. MindShare software provides the user with an animated graphical ontology browser for organizing Web documents according to his or her personal ontology, and also for accessing the composite ontology as extensions of the personal ontology with an assisted bimodal browsing interface. The personal ontologies are mapped to the composite ontology by a process that includes MindShare term-based information retrieval methods to suggest existing topics when a user chooses to add a topic, and a variety of user interface affordances that encourage the user to add exposed pieces of the composite ontology to their personal ontology (and later build upon those pieces). We implemented the MindShare software and conducted an initial trial with a small group of users. The MindShare approach behaved as hoped, suggesting that the approach has promise and merits further investigation.

Thesis supervisor: Henry Lieberman
Title: Research Scientist, MIT Media Laboratory
Readers

Research Advisor

Henry A. Lieberman
Research Scientist
The Media Laboratory
Massachusetts Institute of Technology

Academic Advisor

Patricia Maes
Associate Professor of Media Technology
Media Arts and Sciences
Massachusetts Institute of Technology

Reader

James A. Hendler
Full Professor
Institute for Advanced Computer Studies
University of Maryland at College Park

Reader

Ted Selker
Associate Professor of Media Technology
Media Arts and Sciences
Massachusetts Institute of Technology
Acknowledgements

In accordance with tradition, I would first like to thank my advisors—Henry Lieberman and Pattie Maes, my research advisor and academic advisor, respectively. I would also like to thank my thesis readers, James Hendler and Ted Selker.

My interest in ontology was piqued by Software Agents group discussions of a menace we came to call “The Ontology Problem,” and Pattie suggested I apply ontologies to helping people within a research group to share Web bookmarks. Although I must accept blame for the MindShare approach itself, the suggestion of trying new collaborative ontology development approaches came from Pattie or one of my fellow grad students in the group, and my ideas were surely influenced by them: Michael Best, Leonard Foner, Robert Guttman, Nelson Minar, Alexandros Moukas, Joan Morris, Sybil Shearin, Adriana Vivacqua, David Wang, Alan Wexelblat, Jim Youll, and Giorgos Zacharia. I’d also like to thank undergraduate research assistant Jesse Pavel, who worked on a C version of the MindShare client program. All of us in the group have all been well-served by our honorable succession of administrative ninja: Agnieszka Meyro, Julie Chassé, and Aileen Kawabe.

We started a Lab-wide discussion group on ontology, which yielded some interesting conversations. Some of the conspicuous group participants (i.e., people who attended at least one meeting) were Christopher Beland, Matthew Gray, Ken Haase, Eric Mueller, Carson Reynolds, Warren Sack, Oliver Selfridge, and Brygg Ullmer. I also had numerous chance-encounter hallway discussions on ontology with various other students. It just now occurs to me that the most memorable such conversations disproportionately involved Marvin Minsky’s students: Sasha Chislenko, Timothy Chklovski, and Push Singh.

Media Lab sponsor British Telecom supported me with a fellowship for most of the period of this work. Their liason, Barry Crabtree from BT Labs, deserves special mention for visiting several times and having good comments. I was also supported in part by the Media Laboratory’s Digital Life Consortium, News In the Future Consortia, and E-Markets SIG.

The most redeeming elements of my Media Lab experience while I was working on this thesis actually had nothing to do with the thesis, but rather involved impromptu 3am philosophical discussions in front of the coffee machine, and debates on various Lab email lists. I won’t implicate the guilty parties here; you know who you are.

My grad school career thus far was made possible by the support of several other friends and colleagues, especially KW and BH.
Contents

1 Introduction ........................................................................................................ 13
  1.1 Ontologies ........................................................................................................ 13
  1.2 Knowledge Management .................................................................................. 15
  1.3 MindShare ......................................................................................................... 16
  1.4 This Document ................................................................................................. 18

2 Usage .................................................................................................................... 19
  2.1 Interface Overview ............................................................................................ 19
  2.2 The Story ........................................................................................................... 21

3 Approach ............................................................................................................... 31
  3.1 Documents and Knowledge .............................................................................. 31
  3.2 The Human System ............................................................................................ 32
  3.3 Ontologies of Documents .................................................................................. 33
  3.4 Representation .................................................................................................... 34
  3.5 Concept Matching ............................................................................................... 35
  3.6 Personalized Ontology Browsing ....................................................................... 38
  3.7 Document Ranking ............................................................................................. 39
  3.8 Dynamic Graph Layout ...................................................................................... 43
  3.9 Client-Server Architecture ................................................................................ 48

4 Experience ............................................................................................................. 49
  4.1 Design ................................................................................................................. 49
  4.2 Observations ....................................................................................................... 50
  4.3 Conclusions ......................................................................................................... 53

5 Related Work ........................................................................................................ 55
  5.1 Ontology Construction ....................................................................................... 55
5.2 Communityware ........................................... 56
5.3 Knowledge Management .............................. 56
5.4 Information Retrieval ................................... 57

6 Future Work ................................................. 59
A OMT Notation .............................................. 63
B Sample Database ........................................... 65
C Network Protocol ........................................... 73

Bibliography ..................................................... 87
List of Figures

1.1 The MindShare interface on the desktop ......................... 17

2.1 Main window of MindShare ........................................ 20
2.2 Ana’s personal ontology ......................................... 22
2.3 Showing supertopics ............................................. 23
2.4 Seeing recommended documents ................................. 23
2.5 Browsing to the more specific .................................. 24
2.6 Browsing further .................................................. 25
2.7 Bookmarking something found .................................... 25
2.8 Disagreeing with a relation ...................................... 26
2.9 Returning to the familiar ........................................ 27
2.10 Ana’s new personal ontology ................................. 27
2.11 Adding a topic ................................................... 28
2.12 Suggested topics ................................................ 29
2.13 Topic added and document recommendations ................ 29

3.1 Composite ontology model ......................................... 34
3.2 Personal ontology model .......................................... 35
3.3 SUGGESTTOPICS algorithm ................................. 36
3.4 SIMILARITY function .............................................. 37
3.5 IDF function ........................................................ 37
3.6 RECOMMENDDOCS algorithm ............................... 40
3.7 FINDBESTTOPICPATHS algorithm .............................. 41
3.8 POPULARITYWEIGHT algorithm ............................... 42
3.9 Sample layout animation frame sequence ...................... 43
3.10 LAYOUT algorithm ............................................... 44
3.11 LAYOUTTREEIFY algorithm ................................. 45
3.12 LAYOUTTREEIFYKIDS algorithm ............................. 46
3.13 LAYOUTASSIGNCOORDS algorithm ......................... 47
CHAPTER 1

Introduction

The Lord said, “If as one people speaking the same language they have begun to do this, then nothing they plan to do will be impossible for them. Come, let us go down and confuse their language so they will not understand each other.” So the Lord scattered them from there over all the earth, and they stopped building the city. [29, Genesis 11:6-8]

1.1 Ontologies

If two agents—be they human or software—are to communicate and share knowledge effectively with each other, they must have a common conceptual model of the domain of their discourse, and common terms for concepts in that model. We call the common conceptual and terminological model an **ontology**.

Here we will use the term “ontology” more in the sense it is used by ontological engineers, as a machine-readable formal specification of a conceptualization, rather than in the metaphysical senses it is used by philosophers [8] [10] [36]. Another way to look at an ontology is as a model for categorizing or describing objects—their names, their properties, the relationships among them, the constraints they obey, etc. Some examples of ontologies within applications include:

- Web directories such as Yahoo [44] employ simple topical category ontologies of Web sites and pages, to help human agents locate desired information.

- If Web pages had semantic encoding in terms of a common ontology, rather than merely presentation-oriented HTML or ad-hoc XML DTDs, the Web could be treated as a massive-scale machine-accessible distributed knowledgebase, over which agents could perform more precise term-based information retrieval [23],
make inferences to answer questions [12], and present information in forms best suited to the user’s current interface modality.

- An ontology of consumer products permits description of product categories and subcategories, features, vendors, pricing structures, etc. If vendors’ and consumers’ agents commit to a common ontology, this can facilitate ecommerce services such as cross-vendor product selection aids, and the product offerings tailored to the needs and preferences of the individual consumer [11].

Common ontologies are essential for many desirable applications, but arriving at a common ontology is often difficult, sometimes prohibitively so. One of the greatest difficulties is in the development and maintenance of the ontology. Identifying the pertinent concepts and consistently describing or classifying objects in terms of the concepts is a major undertaking in sheer volume of work, and often requires special ontology engineering skills.

For example, the Yahoo ontology is carefully engineered by a team of dozens of ontologists, who have manually classified over a million Web site objects [38]. Yet, this ontology categorizes only a small percentage of the useful pages on the Web. A recent study estimates that the Web contains over 800 million public pages [19], and one can anecdotally observe the sparseness of Yahoo’s ontology by viewing Yahoo’s category for their field of expertise and noting that the majority of useful Web-available documents in their field are not categorized by Yahoo. It is not merely a matter of time before Yahoo’s ontologists can catch up with the Web, because, for the foreseeable future, the Web promises to grow at a faster rate than the current means of classification.

The example of an ontology of consumer products has a problem similar to that of Yahoo, in the sheer number of objects to be described. It also suffers from additional problems. One is that the product descriptions (e.g., of the features and performance specifications of a stereo system component) are likely to require much more detail and precision than the descriptions of to what coarse topical categories a Web page is relevant. Another problem is that market competition increases the importance of the ontology responding rapidly to innovation (e.g., representing a new class of product feature as soon as it is announced). A third problem is that there may exist incentives for a vendor to differentiate or promote a product by subverting the ontology to express something other than a reasonably objective truth (e.g., claiming that vendor A’s product feature is different than, and superior to, vendor B’s equivalent feature).

The magnitude of difficulty of developing an ontology for semantic encoding of arbitrary Web pages is suggested by the experience of the ambitious Cyc project [21].
Encoding a basic foundation of "commonsense knowledge" required 15 calendar years by a team of expert knowledge engineers, and resulted in $10^8$ axioms. Despite the Cyc project being an impressive accomplishment in many respects, it still cannot represent the knowledge conveyed on Web pages, and actually encoding the Web as an extension of the Cyc ontology remains an insurmountable task for the time being.

Traditional approaches to ontology development usually involve, like Yahoo and Cyc, the labor of centralized teams of dedicated expert knowledge engineers. Some attempts have been made to distribute the knowledge-engineering or classification effort among a decentralized human team of engineers [4]. The engineers either have domain knowledge they are encoding, or they elicit and encode the knowledge from a domain expert. Others have attempted to use automated means of interviewing domain experts who are not skilled in knowledge engineering, which requires substantial dedicated effort by both engineers and domain experts. Still others have worked on automated means of machine classification of documents according to some ontology, or automatically constructing ontologies via clustering and information theory, without any "understanding" of the ontology.

We would like to try an approach in a less-explored area of the design space of ontology development approaches: leveraging human domain knowledge to construct useful ontologies without employing knowledge engineers or significantly distracting the humans from their primary tasks.

Because ontologies are models or abstractions, and their usefulness is determined in part by how well the simplifications suit the application to which they are put, we must anchor and evaluate our approach within a particular application. We have selected a compelling application within the domain of organizational knowledge management.

1.2 Knowledge Management

The field of Knowledge Management concerns itself with facilitating the effective sharing of organizational knowledge. The organization is typically a commercial corporation or part of same, but can also be a distributed community of interest (e.g., a geodistant set of researchers in a field), or another collection of people with shared goals and intersecting areas of interest. The effectiveness of the organization is hindered by poor sharing of knowledge, often resulting in duplication of effort on both small and large scales, delays, missed opportunities, and otherwise poor solutions or practices. Knowledge that would improve the effectiveness of individuals within the organization often exists within the organization, but this useful knowledge is often
not available to the right people at the right times.

An organization’s collective knowledge is contained in the minds of its individual human members and in the digital “documents” (text, spreadsheets, multimedia, database views, etc.) authored by the members. Obviously it befits the organization to capture much of the useful human knowledge in documents, to facilitate long-term, asynchronous sharing. Individuals within the organization also study and refer to documents from sources outside the organization as a way of extending their knowledge.

If much of an organization’s knowledge is captured in documents, then one way to leverage that to make the organization more effective is to help individuals find or discover documents relevant to their current tasks. Another way involves capturing information about what information needs others within the organization have had for internal and public documents, and how those documents were related.

1.3 MindShare

MindShare is the name of our document ontology-based approach to facilitating knowledge sharing within an organization. It was inspired by the realization that useful ontologies of Web documents already exist—in a form distributed amongst the personal Web browser bookmark organizing schemes of sets of the Web users. Simply put, by matching equivalent topics and identical documents amongst each user’s personal organizing scheme, or personal ontology, we obtain a union, or composite ontology, that provides a richer network of topics and documents.

We therefore can consider “the MindShare system” to comprise the people within the organization, who are key computational objects, and the infrastructural software “glue” that mediates the interactions of these human objects and the composite ontology artifact.

Deploying a MindShare system involves providing each member of the organization with a graphical MindShare interface program (Figure 1.1) with which they organize Internet and intranet documents according to their individual needs and preferences—essentially the same activity they already do for their individual benefit with their Web browser’s bookmarking tools. Each of these MindShare interface programs are network clients of the organization’s MindShare server, which maintains the representations of the personal ontologies and the composite ontology.

The composite ontology (CO) is represented in the MindShare server in terms of topics, documents, subtopic relations, and (obscured from the user) document classification relations. This simplified ontology meta-model achieves a balance between
Figure 1.1: The MindShare interface on the desktop

capturing useful information without being burdensome or error-prone for the user. Each personal ontology (PO) is represented as a visibility filter on the CO, with each CO element being *always*, *temporarily*, or *never* visible to the user.

The MindShare client provides a graphical personal ontology browser that allows users to “bookmark” documents they’ve found under topics, add new topics, and define subtopic relationships. In addition, the client allows the user to access information from the composite ontology relative to their personal ontology. For example, when a user wishes to find information about a topic that is not represented in her PO, she visually “browses out” from a topic that is in her PO, along subtopic relations in the CO. The client incrementally reveals topics and relations from the CO to the user, using animated graph layout.

This graph browsing works in conjunction with another key facility of the MindShare client, which is a *recommended documents list* that is continually updated in the client window as the user browses through the topic graph view. The documents recommended documents comprise a selection from the graph neighborhood of the
currently selected topic, ranked according the best weighted path from the topic to the document in the CO, with each path weighted according to absolute graph distance and popularity of the each relation along the path in the POs of all the users in the organization, and filtered through the user's PO. The effect is a kind of bimodal browsing, in which the user combines visual navigation through the topic space with the recommended documents list to get "close enough" to the document so that it appears near the top of her recommendation list. This interface mitigates some of the drawbacks of a complex composite ontology.

The topic-matching from POs to the CO in MindShare occurs primarily by two means: The first is that every time a user chooses to add a topic and provides a name and one-sentence "definition" for the topic, MindShare uses a term-based information retrieval method on the name and definition to find existing topics in the CO that seem similar to the one the user wishes to add. If any are found, MindShare pops up a GUI dialog allowing a user to say with one mouse click which of the existing topics is equivalent to what she intended, or whether neither of them are. The second means by which topics are matched arises from the fact that the user is frequently exposed by the MindShare interface to elements of the CO that are related to something in her PO, and there is a simple mouse operation by which she can choose to "add" an element from the CO to her PO (i.e., make always visible).

1.4 This Document

In the following chapter, Usage (Chapter 2), we illustrate user-visible aspects of MindShare by describing the user interface, the intended user's conceptual model of MindShare's operation, and a series of samples use cases involving of users interacting as part of the same MindShare system. The Approach chapter (Chapter 3) discusses the basic theory behind MindShare, the representation models, the key algorithms, and the client/server architecture. A discussion of our initial trial and what we learned is in the Experience chapter (Chapter 4). Related Work (Chapter 5) summarizes relevant work in fields such as ontological engineering, knowledge management, and information retrieval. Finally, Future Work (Chapter 6) introduces directions for future work—complementary aspects of the original vision, issues that were prompted by the results of our initial trial, and potential application of variations on the MindShare approach to other application domains.
CHAPTER 2

Usage

This section illustrates some user-visible aspects of MindShare's operation. We begin by introducing the GUI components and intended conceptual model of the interface as they would be explained to a new user. Then we describe a series of hypothetical sample uses involving an individual's interactions with a shared MindShare system.

2.1 Interface Overview

Irrespective of whatever benefit MindShare may have to other individuals or the organization as a whole, the individual user can think of MindShare as a personal tool for:

- bookmarking documents from organizational and public sources, including the documents that she writes in the course of her work,
- organizing the documents in ways that best suit her individual needs (i.e., areas of work and knowledge, perspectives, priorities, frequency of use, etc.),
- finding new useful documents and related topics on demand,
- serendipitously discovering useful relevant documents and topics, and
- being informed that others in the organization have had related information needs on a topic (i.e., they are working on or have solved similar problems, they are knowledgeable on a topic, etc.).

The MindShare GUI interface has a single main window that can sit on the user's desktop beside her Web browser (Figure 1.1). For purposes of the MindShare prototype, the Web browser is considered to be the tool through which the user views
organization-internal (intranet) and public (Internet) information, which take the form of multimedia documents.

The MindShare main window is divided into three panes—graph view, bookmark list, and recommendation list—with a status bar at the bottom (Figure 2.1). User input is via direct-manipulation mouse clicks, context-sensitive pop-up menus, and pop-up GUI dialogs.

![MindShare Main Window](image)

**Figure 2.1: Main window of MindShare**

The graph view presents what the user can think of as her topics and the subtopic relations between them. She can also use the graph view to "browse out" from one of her topics to see other topics and subtopic relations from MindShare, as described later.

Topics in the graph view are represented by colored rectangles, and subtopic relations ("links") are represented by colored arrows\(^1\) between the topic rectangles.

---

\(^1\)Some of the screenshots in this document were taken before the prototype MindShare client software could draw arrowheads in the graph view.
The color indicates to the user the long-term “visibility” of the topic or link in her graph view, and there are three possible colors. When the user adds a topic or link in her graph view, it appears green, which she understands to mean that it will always be in her graph view unless she takes explicit action to change its visibility. When she “browses out” from a topic, the topics and links that are temporarily added to her view appear blue, meaning that they are “temporarily visible” and removed as soon as she is done and chooses to return to seeing only things that are green (“hers”). The third possible color is red, which the user can turn something to communicate that she thinks it’s erroneous and wishes never to see again.

The graph view does automatic animated layout of topics and links as the user edits and browses. Along the top of the view is a gray bar, to which the user can attach topics that she wishes to always be presented as top-level topics. This helps to keep her most important areas of interest firmly rooted visually, influencing the shape of the automatic layout when there is a dense web of topics and relations to be presented.

One of the objects in the graph view, either a topic or a link, can be selected by clicking on it with the mouse, which causes a yellow “halo” to appear around it. When the user right-clicks in the graph view, the context menu that pops up applies to whatever object is selected, or the entire graph view if no object is selected.

When a topic is selected, the bookmark list in the green lower-left pane of the main window shows the names of the documents the user has bookmarked under that topic. Documents have names and URLs. Double-clicking on a name in the bookmark list causes the document to be viewed in the Web browser. Right-clicking pops up a context menu of operations on the document.

The recommendation list is the blue lower-left pane of the main window, and shows “smart” recommendations of other documents from MindShare that she has not bookmarked herself but that may be useful to her. Like the bookmark list, documents in the recommendation list can be viewed in the browser and have a context menu associated with them. The recommendations, which will be described in detail later, are updated whenever the user selects a new topic, and tend to be relevant to the selected topic.

### 2.2 The Story

Ana is a software engineer for International Tractors and Combines (ITC) corporation, where she develops embedded microcontrollers for farming machinery using Lisp. Her personal ontology view in MindShare looks like Figure 2.2.
One day, the CTO of ITC, Billy-Bob Barnes, returns from a gala industry conference and asks Ana to investigate rewriting the new spreader controller in Java. Ana dutifully agrees to learn about and experiment with Java. Naturally, she decides to leverage knowledge from people in other divisions of her company in finding useful Java information, by turning first to her MindShare interface.

Ana’s MindShare view doesn’t show Java right now, but she knows it’s a kind of computer programming language, like Lisp is, so her first thought is to “browse up” from Lisp to its supertopics. She right-clicks on “Lisp,” causing the Selected Topic menu to appear, and selects the Show Super-topics of this Topic item (Figure 2.3). A “Languages” topic and “Functional” topic in blue sprout from “Lisp” in her view.

Ana clicks on the “Languages” topic, and already sees some recommended documents about Java (Figure 2.4). She rapidly double-clicks on “Languages” to show subtopics. Several topics slide into her view, including a “Java,” which she clicks on, to reveal helpful recommended documents (Figure 2.5).
Figure 2.3: Showing supertopics

Figure 2.4: Seeing recommended documents
She sees the documents and decides she’s first most interested in getting her hands dirty with Java, but she doesn’t see a lot of information about Java tools or compilers. Taking a huge risk, she double-clicks on “Java,” and sees a “Compilers” topic, which she clicks on a recommended document about a Java compiler that will run on her Linux box (Figure 2.6). She also notices two company-internal documents relevant to Java. She double-clicks on one of the document items in the document list, and the document is loaded in her Web browser.

After perusing the document, Ana decides to bookmark it, and since she expects to be dealing with at least a few Java documents, she decides to permanently add a “Java” topic (and it might as well be this one) to her personal ontology. She right-clicks on the “Java” topic and selects **GREEN: Make this Topic Permanently Visible**, causing the topic to turn green. She then right-clicks on the document and selects **Bookmark this Document** (Figure 2.7), causing it to move over to her green bookmarks list.
Figure 2.6: Browsing further

Figure 2.7: Bookmarking something found
Ana decides to keep her personal ontology neat by now having a “Languages” topic, with “Lisp” and “Java” beneath it, so she decides to reuse the existing one, by turning it green. As she clicks on it, she notices a lot of documents about animals that really don’t seem to belong there. She then notices the “Animal Talk” category, and decides that she violently disagrees with the subtopic relation between this “Languages” topic and “Animal Talk.” She right-clicks on the relation, and selects RED: Make this Link Never Again Visible to turn the relation red and make it no longer affect her (Figure 2.8). When she clicks on “Languages” again, she no longer sees documents about talking to animals. Note that this action Ana took only for her own personal benefit, but it also benefits the entire organization, since paths through that relation will now have a slightly lower weight for everyone.

![Figure 2.8: Disagreeing with a relation](image)

Ana has found the information she needs for now, and decides to return to just her familiar personal ontology green-stuff. She right-clicks on the graph view, selects Hide Things that are Not Permanently Visible (Figure 2.9), and her view returns to that shown in Figure 2.10.
Figure 2.9: Returning to the familiar

Figure 2.10: Ana's new personal ontology
The next day, Mr. Barnes is so pleased to learn of Ana’s command of the Java language, that he tells Ana of an idea that he came up with while having drinks with the VP of Marketing: having the new high-end BFC-1000 line of autonomous robotic combines be administered through a remote Web browser interface. Ana realizes she’ll now need to thoroughly study Java servlet technology, to be sure that this safety-critical application will be free of defects, so she goes to MindShare to create a new topic just for servlets. She right-clicks on the graph view, selects *Add a New Topic*, and fills in the *Add Topic* dialog, as shown in Figure 2.11.

![Figure 2.11: Adding a topic](image)

Ana clicks *OK*, and MindShare pops up a new dialog with a set of buttons, each corresponding to an existing topic that Ana might mean (Figure 2.12). The first button is actually what Ana wanted, so she clicks on it. MindShare visually animates adding a *Servlets* topic to Ana’s view. Since it was already related to her Java topic in the composite ontology, those subtopic links are shown to her as well. She also sees some documents about servlets that might prove useful immediately.

Ana skims the servlets documents, then heads off for a well-deserved lunch, silently appreciating how MindShare has helped her both organize information relevant to her, and also quickly find relevant public and internal documents. Simultaneously, halfway around the globe, Ana’s colleague in the Irrigation subsidiary of ITC is appreciating that MindShare exposed him to some of Ana’s documents on her embeddable Lisp VMs, which he had assumed he would have to develop from scratch.
Figure 2.12: Suggested topics

Figure 2.13: Topic added and document recommendations
CHAPTER 3

Approach

The following sections discuss the MindShare sociotechnical design and details of some of the more interesting aspects of the MindShare software.

3.1 Documents and Knowledge

An organization's collective knowledge is contained in the minds of its individual human members and in the digital "documents" (text, spreadsheets, multimedia, database views) authored by the members. Obviously it befits the organization to capture much of the useful human knowledge in documents, to facilitate long-term, asynchronous sharing. Individuals within the organization also study and refer to documents from sources outside the organization as a way of extending their knowledge. This notion of a document is a useful one, and we take it as a primary form of packaging contextualized chunks of knowledge in MindShare.

If we are able to capture much or most of an organization's knowledge in documents, then one way to leverage that to make the organization more effective is to help individuals find or discover documents relevant to their current tasks. Another way is to inform the individual of related past and present efforts, and of persons knowledgeable on a topic, by exposing that others have or had related information needs.

In the case of documents authored outside of the organization, the fact that one individual in the organization found such a document useful in the course of their work suggests a better than average probability that the document will be useful to someone else in the organization. Therefore, information about which public documents have been used by members of the organization is also useful organizational knowledge.
3.2 The Human System

Human intelligence is a powerful thing, including commonsense and domain-specific knowledge and reasoning abilities that are not presently possessed by electronic computers. We would like to harness this human intelligence for our problem of connecting individuals and documents.

One way an organization could do this would be to have experts in domains of relevant knowledge do nothing but constantly study the work of everyone else in the organization to see if anyone could benefit from the expert's knowledge. But this would be expensive, even if a variation on the idea could be made feasible. Alternatively, every individual in the organization could be tasked to spend 20% of their time seeing what other people in the organization could use their knowledge, but that too is probably an impractical assignment of human resources in most organizations.

Possibly better approaches come from trying to find ways of using technology to leverage human intelligence within the organization while incurring negligible human resource cost. When doing this—designing a system of humans and technology—we can consider individuals within an organization to be computational objects with partially-specifiable behaviors. We can therefore look at what people normally do or do in response to certain input, and integrate these people objects with artificial software objects that observe and stimulate the human behavior.

Among the pertinent things people do is to bookmark Web pages (i.e., documents) that they find useful and to which they expect to want to refer later. Many people organize these bookmarks into categories according to topic and frequency of access. (The sophistication of the bookmarking varies depending on the individual and various other factors, such as the ease-of-use of their bookmark management tools.) Likewise, many people organize their computer files into subdirectories or folders. People tend to use their own organizing schemes, specialized to their priorities and perspectives, even if "standard" schemes exist. They are willing to use directory and keyword search engines in goal-directed ways to find Web pages to get answers to immediate questions, but generally do not look for Web pages when there is no pressing question to answer.

We can also observe more general values, such as a willingness to occasionally invest small amounts of effort if that results in a frequent and more effort-saving benefit. And we can observe willingness (if not desire) to assist others in the organization if only small effort is required and it is not a zero-sum game situation, or if there is significant direct return benefit.
The MindShare approach attempts to leverage these observed behaviors for organizational knowledge management. Sensitivity to human behaviors may also be detected in the design of some MindShare user interface elements, described later.

3.3 Ontologies of Documents

These individual bookmarking and organizing behaviors are expressions of human intelligence about the topical relevance of documents and the relationships among topics—effectively describing an ontology of documents. This ontology adds machine-accessible information about the documents that cannot be divined from the documents themselves.

The MindShare approach involves taking advantage of this ontology by replacing each individual's bookmarking and file-organizing tools with a personalized interface to the MindShare system. Each individual within an organization shares the same MindShare system, but retains their personal ontology view. By various means (described later), MindShare secretly knits together individuals' personal ontologies such that equivalent topics in different individuals' ontologies are represented as such. This results in what we call a composite ontology, which enables several kinds of automatic knowledge sharing in terms of a user's personal ontology and extensions of it.

(Note that one could employ traditional information retrieval methods, such as on-demand keyword queries, as an alternative means of finding documents in some cases. This would be complementary to the MindShare approach, and could benefit from some of the individual and organization profiles that are maintained by MindShare.)

The ontology model supported by MindShare consists of topics, documents, subtopic relations, and (obscured from the user) document classification relations. The reason that the model does not support more sophisticated knowledge representation concepts such as part-of and kind-of relations and user-defined relations is that these might be cumbersome and intimidating for the user, and results from the field of knowledge acquisition show that people who are not knowledge engineers tend to use the modeling concepts incorrectly. Therefore, we rely on the notion of a subtopic, which is familiar to the layperson from directories such as Yahoo, and which the user is less likely to use incorrectly.
3.4 Representation

The MindShare server, which is the software-based repository of knowledge captured by the MindShare system, contains the canonical representations of the organization’s composite ontology and each individual’s personal ontology.

A conceptual-level data model for the composite ontology is shown in Figure 3.1, using OMT notation ([33], summarized in Appendix A). The MindShare server represents personal ontologies as sparse visibility filters, in object graphs homomorphic to the object graph of the composite ontology, as shown in Figure 3.2.

![Composite ontology model](image)

Each element in a personal ontology (PO) corresponds to an element in the composite ontology (CO). Each PO element has a visibility attribute that can have one of three values: permanent, meaning that the CO element is a part of what the user thinks of as her personal ontology; temporary, meaning that the CO element has been temporarily shown to the user in the course of browsing extensions of their PO; and never, meaning that the user has explicitly disagreed that the PO element is valid, and wishes to never see it again during topic browsing or have it affect document recommendation.
3.5 Concept Matching

Central to the formation of a healthy composite ontology is that equivalent topics in separate individuals' minds tend to get mapped to the identical topic in the CO.

One very simple way in which this can happen is due to the fact that the browsing features of the MindShare interface will tend to expose the user to existing topics from the composite ontology that she may wish to reuse. The interface makes it easy for the user to toggle the PO visibility of a CO topic she finds very useful, to permanently add it to her PO view. Thereafter the user will build on that topic with relations and documents, further knitting it into the CO.

Another way of trying to match equivalent topics is to interpose some software smarts at the time the user chooses to create a new topic. When a user invokes the operation to create a new topic, and types a short topic name and definition phrase into a GUI dialog, MindShare tries to recommend existing CO topics that the user might be intending. If any are found, MindShare pops up a GUI dialog with a button
for each of suspected topics (up to around ten) and a button for the user to indicate
that none of the existing topics are what she intended. This GUI dialog is a small
imposition we make on the user when they add a new topic, and the user is motivated
to participate because they directly and personally benefit from correct matching of
topics.

The topic suggestion algorithm currently works via term vector similarity on the
topic names and definitions, as shown by the pseudocode\(^1\) for algorithm SUGGEST-
TOPICS (Figure 3.3). The **SIMILARITY** cosine similarity function (Figure 3.3) is taken
from Salton and McGill’s work in information retrieval techniques [34]. It in turn uses
the **IDF** inverse-document frequency function of [37] (Figure 3.5). The ‘document
collection’ from which corpus term frequency is calculated is the set of topics presently
in the composite ontology.

**Require:** parameter *name* is the name of the topic.
**Require:** parameter *desc* is the textual definition of the topic.

\[\text{△ Build a term vector from the name and definition.}\]
1: \(\text{queryTV} \leftarrow \text{NAMEANDDESCTERMVECTOR}(name, \text{desc})\)

\[\text{△ Build the initial list of suggested topics, using term vectors.}\]
2: \(\text{suggTopicList} \leftarrow \emptyset\)
3: \textbf{for all} coTopic \(\leftarrow\) topics in the composite ontology \textbf{do}
4: \(\text{tv} \leftarrow \text{TOPICTERMVECTOR}(\text{coTopic})\)
5: \(\text{similarity} \leftarrow \text{SIMILARITY(queryTV, tv)}\)
6: \textbf{if} similarity \(> 0\) \textbf{then}
7: \(\text{APPEND(suggTopicList, (coTopic, similarity))}\)

\[\text{△ Sort the topics list in order of decreasing similarity value, truncate it}\]
\[\text{△ at a max length, and return it.}\]
8: \(\text{SORTSUGGTOPICsLIST}(\text{suggTopicList})\)
9: \(\text{TRIMLIST(\text{suggTopicList}, 10)}\)
10: \textbf{return} \(\text{suggTopicsList}\)

**Figure 3.3: SUGGESTTOPICS algorithm**

\(^1\)Abstract imperative pseudocode is used in this document instead of the object-oriented Java
code in which the prototype system is implemented, for the sake of readability. In addition to hid-
ing various low-level language-specific performance and caching issues, the pseudocode also ignores
multi-threaded and multi-user concurrency issues, which are also addressed in the Java code.
SIMILARITY($Q, D$) = \[
\frac{\sum_{i=1}^{t} (w_{i,q} \cdot w_{i,j})}{\sqrt{\sum_{i=1}^{t}(w_{i,q})^2 \cdot \sum_{i=1}^{t}(w_{i,j})^2}}
\]

\(w_{i,j} = freq_{i,j} \cdot IDF(i)\)

\(freq_{i,j} = \) the frequency of term \(i\) in document \(j\)

\(IDF(i) = \) the IDF of term \(i\) in the entire collection

\(w_{i,q} = freq_{i,q} \cdot IDF(i)\)

\(freq_{i,q} = \) the frequency of term \(i\) in query \(q\)

Figure 3.4: SIMILARITY function

\(IDF(i) = \log_2 \frac{N}{n_i} + 1\)

\(N = \) the number of documents in the collection

\(n_i = \) the total number of occurrences of term \(i\) in the collection

Figure 3.5: IDF function

In preliminary informal experience, this method gives good results, usually suggesting an existing topic as one of the first matches, if one exists. Substantially better precision and recall is more likely to come not from minor tuning the similarity and IDF functions, but by using additional sources of information within the database and user interface. For example, one approach that may yield even better results is to provide an atomic add subtopic user interface operation, and use this information about the intended subtopic relation and its place in the composite ontology to bias topologically nearby topics, or have the term vector for the supertopic contribute a small amount of influence to the new topic’s vector. We might further extend that with some simple linguistic part-of-speech parsing of topic names and descriptions, and detecting when a subtopic name is modifying a supertopic name without explicitly repeating terms from the supertopic.
3.6 Personalized Ontology Browsing

While individual users might form their personal ontologies mostly as small hierarchies, MindShare’s unifying of personal ontologies is expected to result in composite ontology that contains a great deal of topics and documents with a much more dense web of relations among them. This is beneficial in that it aids useful ways of knowledge discovery and searching, especially when linked to the user’s personal ontology. It also poses some interesting problems for browsing.

In order to use a large common ontology, such as a Web directory like Yahoo, the user can encounter a difficulty of mapping their own conceptual model of the world to the common ontology – they know what concept they mean, but first they have to find an equivalent concept in the ontology. This difficulty can worsen as the size and complexity of the ontology grows. For example, some people already have difficulty finding topics within the Yahoo directory, and if Yahoo were to capture the full wealth of useful documents on the Web, the necessary ontology might be several orders of magnitude more complex.

MindShare addresses this problem of finding a starting point within a large common ontology by using the personal ontology as the gateway to the composite ontology. When a user wants to find some new topic, she conceptualizes what she is looking for in terms of something she already knows. If the something she already knows is represented in her MindShare personal ontology, she can use that familiar concept in her view of MindShare and “browse out” in the direction of the thing that she is looking for (i.e., by browsing in the direction of subtopics or supertopics).

Since the composite ontology is expected to be rather densely-connected and have some redundancies and perhaps some convoluted paths, navigating solely through the graph view might prove cumbersome. For example, a document the user needs may be several subtopic relations away selected topic in the CO, even though it is closely related to the selected topic. This is one of the main reasons that MindShare supports a kind of bimodal browsing, through the use of a smart ranked list of recommended documents that is constantly being updated as the user browses in the graph view (§3.7). At any time, the user can switch from navigating the graph view clicking on a useful document near the top of the recommendation list, or scrolling down in the document. Therefore, the user only needs to navigate “close enough” to the document for it to appear in the recommendation list.

We suspect that users will adopt strategies for the bimodal browsing in MindShare, and tend to use the same one or two at high speed most of the time. For example, one strategy might be to navigate for up to several clicks in the graph view, choosing more and more specific topics, until a promising-sounding document percolates to near the
top of the recommendation list. If such a document doesn’t reveal itself after a few clicks, or running out of topics, the user might then revert to scrolling down in the recommendation list.

3.7 Document Ranking

The document ranking algorithm, RECOMMENDDOCS, is shown in Figure 3.6. Given a topic in the composite ontology, a user’s personal ontology, and the personal ontologies of all the other users on the system, the algorithm yields an ordered list of documents that might interest the user in a browsing context. This is the algorithm that is invoked in the prototype whenever a user clicks on a topic in the graph view to select it, and the ranked list of documents returned by the algorithm appears in the Recommended Documents list of the user’s MindShare window.

Topic searching is done by FINDBESTTOPICPATHS (Figure 3.7). The ranking is based on neighborhood graph distance, popularity of subtopic relations (i.e., how many people have chosen to permanently-add and permanently-ellide (hide) the relation, given by POPULARITYWEIGHT in Figure 3.8), and popularity of document classification relations. Relations permanently-ellided by the user are ignored by the graph algorithm. In each case, the ranking of a document is based on the best weighted path to it. The algorithm is constraining its search to a subtopic relation distance of six at the moment. A path in the supertopic-to-subtopic direction (i.e., general to specific) has a somewhat higher weight than a path subtopic-to-supertopic (i.e., specific to general).

In preliminary tests with a small composite ontology, the algorithm runs in less than one second on the modest development PC, which is fast enough for speedy interactive browsing. Although the algorithm is exponential with respect to the number of subtopic and classification relations in the neighborhood, it is limited by the maximum search depth. Popularity information is maintained in a cache in the prototype system, so the number of users does not affect the complexity of the algorithm. We therefore suspect that the algorithm will scale reasonably for practical purposes.
Require: parameter user is the user.

Require: parameter startCOTopic is the user’s selected topic.

Recursively find the best weighted paths to all of the topics in the neighborhood, respecting any never-visibility elements in the user’s personal ontology.

1: \( \text{TopicWeight} \leftarrow \emptyset \)

2: \( \text{FindBestTopicPaths}\left( \text{startCOTopic}, 0, (\text{count of users in the database}) \right) \)

For each topic, add its documents associated by classification relations (and not ellided by never visibility in the PO) to the document list, along with the weight. If the document already exists in the list, but the weight is better, replace the weight with the existing weight.

3: \( \text{DocWeight} \leftarrow \emptyset \)

4: for all coTopic \( \leftarrow \text{TopicWeight} \) do

5: \( \text{topicWeight} \leftarrow \text{weight of coTopic tuple} \)

6: for all coClarel \( \leftarrow \text{classification relations for coTopic} \) do

7: \( \text{poClarel} \leftarrow \text{POcoClarel} \)

8: if \( (\text{poClarel} = \emptyset) \lor (\text{poClarel} \neq \text{never}) \) then

9: \( \text{coDoc} \leftarrow \text{document of the coClarel} \)

10: \( \text{poDoc} \leftarrow \text{POcoDoc} \)

11: if \( (\text{poDoc} = \emptyset) \lor (\text{poDoc} \neq \text{never}) \) then

12: \( \text{docWeight} \leftarrow (\text{POPULARITYWEIGHT(coClarel)} \cdot \text{topicWeight}) \)

13: if \( (\text{DocWeight}_{\text{coDoc}} = \emptyset) \lor (\text{docWeight} > \text{DocWeight}_{\text{coDoc}}) \) then

14: \( \text{DocWeight}_{\text{coDoc}} \leftarrow \text{docWeight} \)

Return a list of documents by sorting DocWeight by descending weight in each tuple.

15: return DocWeightSetAsSortedList(DocWeight)

Figure 3.6: RECOMMENDDOCS algorithm
Require: parameter $coTopic$ is the CO topic point in this path.
Require: parameter $distance$ is the unweighted path distance thus far.
Require: parameter $pathWeight$ is the weight of the path thus far.

\[ \triangle \text{If the PO topic for the CO topic is defined and is } never, \text{ then terminate this path.} \]

1: $poTopic \leftarrow PO_{coTopic}$
2: if $poTopic = never$ then
3: \hspace{1em} return

\[ \triangle \text{If the topic has been encountered, but the previous weight was as good as or better than this one, then terminate this path by returning.} \]

4: if $(TopicWeight_{coTopic} \neq \emptyset) \land (TopicWeight_{coTopic} \geq pathWeight)$ then
5: \hspace{1em} return

\[ \triangle \text{Assign this topic weight.} \]
6: $TopicWeight_{coTopic} \leftarrow pathWeight$

\[ \triangle \text{If we've hit our max distance, then terminate this path.} \]
7: if $distance \geq 6$ then
8: \hspace{1em} return

\[ \triangle \text{For each of the subtopic relations out of the topic that are not ellided by the PO, recursively call this function with a decaying path weight.} \]
9: for all $childCOSubrel \leftarrow COSubrel_{coTopic,*}$ do
10: \hspace{1em} $childPOSubrel \leftarrow PO_{childCOSubrel}$
11: \hspace{1em} if $childPOSubrel \in \{\emptyset, never\}$ then

\[ \text{FINDBESTTOPICPATHS} \left( \begin{array}{c}
\text{childCOSubrel.childTopic,}
\end{array}
\frac{\text{(distance} + 1),}{\text{pathWeight} + \text{POPULARITYWEIGHT(childCOSubrel)}},
\frac{\text{distance+2}}{}
\right) \]

\[ \triangle \text{Do the same for parent subtopic relations, but with a slightly lower weight, since topics in that direction will tend to be more general, rather than more specific.} \]
13: for all $parentCOSubrel \leftarrow COSubrel_{coTopic,*}$ do
14: \hspace{1em} $parentPOSubrel \leftarrow PO_{parentCOSubrel}$
15: \hspace{1em} if $parentPOSubrel \in \{\emptyset, never\}$ then

\[ \text{FINDBESTTOPICPATHS} \left( \begin{array}{c}
\text{parentCOSubrel.parentTopic,}
\end{array}
\frac{\text{(distance} + 1),}{\text{pathWeight} + \text{POPULARITYWEIGHT(parentCOSubrel)}},
\frac{\text{distance+4}}{}
\right) \]

Figure 3.7: FINDBESTTOPICPATHS algorithm
Require: parameter coElement is a CO Element.

Return the popularity weight as a function of the count of people who have permanently added the element in their POs and the count of people who have ellided the element in their POs.

1: return min \( \left( \frac{((\text{permanentsCount}_{\text{coElement}} + 0.5) * 2) - \text{neversCount}_{\text{coElement}})}{0.10} \right) \)

Figure 3.8: POPULARITYWEIGHT algorithm
3.8 Dynamic Graph Layout

As the user browses through the graph view, such as by double-clicking on a topic to reveal more subtopics from the composite ontology, the graph layout must be adjusted to accommodate the changed graph. Due to the potential density of the graph and the kind of rapid navigation, it is desirable that the user be able to visually recognize how objects in the old layout map to objects in the new layout, to avoid disorientation. Part of the approach employed by the MindShare client is to use animation to show the transition to the new layout. For example, the user can see two adjacent topics moving to either side to make space for a third topic that slides into place between them. A sample of frames in an animation sequence is shown in Figure 3.9.

![Sample layout animation frame sequence](image)

Figure 3.9: Sample layout animation frame sequence

MindShare uses a layout algorithm tailored to the MindShare representation model, which tends to result in minimal layout changes when changes are made to the graph. This simple approach allows incremental layout changes to be accomplished by simply re-running the layout algorithm over the entire graph, and then stepping through an animation of each object moving from its coordinates in the old layout to its new layout coordinates. One additional consideration is to animate automatic scrolling to keep focused elements in view—as shown in the sample above, in which the selected topic stays in the same screen position, while some other portions of the graph seem to move out of the way to make space.
The LAYOUT algorithm, shown in Figure 3.10 and following figures, has several major steps. The first, LAYOUTTREEIFY, overlays the disconnected and cyclic subtopic relation directed graph with a prioritized breadth-first tree traversal graph from an imaginary root vertex. The next step, LAYOUTASSIGNCOORDS, does a traversal of the resulting tree to assign coordinates to each vertex. Finally, the coordinate systems are adjusted, and the display animates each object moving along its vector from previous coordinates to new coordinates, in approximately 30 frames, which automatically scrolling the view as necessary.

**Require:** each topic has valid current layout coordinates

![Diagram](image)

1: LAYOUTTREEIFY()

△ Map the vertices and some of the edges of the topic-subrel arbitrary directed graph to a tree graph for layout purposes. This sets treeRoot, which is a placeholder topic object that will not be drawn.

2: newLayoutBounds ← ∅

3: LAYOUTASSIGNCOORDS(treeRoot, 0, 0)

△ Adjust the new coordinates of the vertices to their absolute position on the unscreened canvas, set the shelf's newBounds, and set newLayoutBounds to be the canvas bounds including shelf and margins. (This step was added to eliminate dealing with four separate coordinate systems and constant translations.)

4: LAYOUTFINISH()

△ Animate the layout by stepping along the vectors for each vertex from old coordinates to new coordinates. Edge arrows are always drawn along lines that pass through the centerpoints of both parent and child vertex node. In some cases, such as a topic node being selected, also animate scrolling the view to keep important things on screen.

5: ANIMATELAYOUT()

![Figure 3.10: LAYOUT algorithm](image)
Require: called from LAYOUT algorithm

△ Sort the sequence of topics primarily by visibility (permanent precedes temporary, precedes never), and secondarily by alphabetic ordering of topic names.

1: \texttt{SORT(topicsVector, LAYOUTSORTCOMPARE)}

△ Create a first pass of “root” topics from those marked alwaysRoot, and move all other topics into treeOrphans. (Note that the actual prototype implementation maintains a hasTreeParents flag to save frequent membership tests in treeOrphans.)

2: \texttt{treeRoot ← new Topic}  
3: \texttt{treeRoot.treeKids ← 0}  
4: \texttt{treeOrphans ← 0}  
5: \texttt{for all topic ← topicsVector* do}  
6: \texttt{topic.treeKids ← 0}  
7: \texttt{if topic.alwaysRoot then}  
8: \texttt{APPEND(treeRoot.treeKids, topic)}  
9: \texttt{else}  
10: \texttt{APPEND(treeOrphans, topic)}  

△ Add topics without subrel parents to the tree root.

11: \texttt{for all topic ← treeOrphans* such that topic.parentSubrels = 0 do}  
12: \texttt{REMOVE(treeOrphans, topic)}  
13: \texttt{APPEND(treeRoot.treeKids, topic)}  

△ Assemble connected topics in a tree below treeRoot via breadth-first traversal.

14: \texttt{LAYOUTTREEIFYKIDS(treeRoot)}

△ Any topics still in treeOrphans are in cycles. Pick the first untreeified topic, add it as a child of fakeRoot, and treeify its children. Repeat until there are no untreeified topics left.

15: \texttt{while treeOrphans ≠ 0 do}  
16: \texttt{topic ← POP(treeOrphans)}  
17: \texttt{APPEND(treeRoot.treeKids, topic)}  
18: \texttt{LAYOUTTREEIFYKIDS(topic)}  

Figure 3.11: LAYOUTTREEIFY algorithm
Require: called from LAYOUTTREEIFY algorithm
Require: parameter topic is the topic that roots this subtree

△ Initialize a queue of topics to traverse with topic or (if it is treeRoot) its current tree children. (treeRoot must be special-cased due to the way data structure referential integrity is maintained in the prototype implementation.)

1: queue ← ∅
2: if topic = treeRoot then
3:     for all child ← topic.treeKids* do
4:         APPEND(queue, child)
5: else
6:     APPEND(queue, topic)

△ Do a breadth-first traversal of previously-unvisited topics from each of the elements in queue, connecting orphan subrel children as layout tree children.

7: while queue ≠ ∅ do
8:     visitedTopic ← POP(queue)
9:     if visitedTopic.treeKids = ∅ then
10:        while kidTopic ← visitedTopic.childTopics* do
11:           if kidTopic ∈ treeOrphans then
12:              REMOVE(treeOrphans, kidTopic)
13:              APPEND(visitedTopic.treeKids, kidTopic)
14:           end if
15:        end while
16:     end if
17:     if kidTopic.treeKids = ∅ then
18:         APPEND(queue, kidTopic)

Figure 3.12: LAYOUTTREEIFYKIDS algorithm
**Require:** called from LAYOUT algorithm or self
**Require:** parameter topic is the topic that roots this subtree
**Require:** parameter x is the initial X coordinate of topic
**Require:** parameter y is the initial Y coordinate of topic

- Save the old coordinates and set the new coordinates of this topic (oldX is animation beginning, curX is current, newX is animation end). Also set set coordinates for children and horizontal growth.

1: \((topic.\text{old}X, topic.\text{old}Y) \leftarrow (\text{cur}X, \text{cur}Y)\)
2: \((topic.\text{new}X, topic.\text{new}Y) \leftarrow (x, y)\)
3: \text{local} \((\text{childStart}X, \text{childStart}Y) \leftarrow (x, (y + \text{layoutVerticalSpacing}))\)
4: \text{local} \(\text{last}X \leftarrow x\)

- Recursively position any children of the topic. First sort the sequence of children. Then position each child by invoking this algorithm recursively. Then center each parent horizontally above its children.

5: \text{if} \ topic.\text{treeKids} \neq \emptyset \text{ then}
6: \text{SORT}(topic.\text{treeKids}, \text{LAYOUTSORTCOMPARE})
7: \text{local} \ firstChild \leftarrow \emptyset
8: \text{local} \ lastChild \leftarrow \emptyset
9: \text{while} \ \text{childTopic} \leftarrow topic.\text{treeKids}^* \ \text{do}
10: \text{if} \ firstChild = \emptyset \text{ then}
11: \ firstChild \leftarrow \text{childTopic}
12: \lastChild \leftarrow \text{childTopic}
13: \ text{last}X = \text{LAYOUTASSIGNCOORDS}(\text{childTopic, childStartX, childStartY})
14: \ \text{childStartX} = \text{last}X + \text{layoutHorizontalSpacing}
15: \text{if} \ topic.\text{treeKids} \neq \emptyset \text{ then}
16: \text{local} \ leftX \leftarrow firstChild.\text{new}X
17: \text{local} \ rightX \leftarrow lastChild.\text{right-hand X}
18: \ topic.\text{new}X = leftX + \left(\frac{rightX - leftX - \text{topic width}}{2}\right)

- Expand the layout bounds to include this topic.

19: \text{if} \ topic \neq \text{treeRoot} \text{ then}
20: \text{EXPANDRECTANGLE}(\text{newLayoutBounds, bounds of topic})

- Return the rightmost coordinate of this subtree.

21: \text{return} \ lastX

Figure 3.13: LAYOUTASSIGNCOORDS algorithm
3.9 Client-Server Architecture

A MindShare system will have many people using it at once. To facilitate evaluation in real-world situations, the MindShare prototype has been implemented as a client-server system, with a single central server for the organization and multiple user interface client programs. Together they comprise a bit over thirteen thousand lines of Java code. All of the significant representation and computation occurs on the server. The client provides graphical user interface elements, including dynamic animated layout of the topic graph, and makes server requests for information and computation to support most user interface operations. A specification of the network protocol between client and server is given in Appendix C. While client-server partitioning has many well-understood practical benefits in a production system, an additional reason we used a “thin client” approach was so that the client program could be implemented by an undergraduate research assistant according to the rather well-defined protocol spec (although ultimately the author ended up writing both server and client).

The server-side representation is kept in an in-core graph of native Java objects for reasons of performance and ease of prototyping. Roughly once every minute, the prototype server dumps the database to an ASCII interchange file using a human-readable Lisp-like syntax. This provides persistence and safety for when the server needs to be upgraded or restarted during evaluation, and also allows manual editing of the database by experimenters, to construct test cases, repair errors, port to a new representation model, etc. A sample database in this syntax is shown in Appendix B.

It bears noting that if a version of MindShare were to be implemented as a large-scale production system, it would likely use a conventional object or relational database management system. When deployed in a real-world context, there would also be significant privacy issues to address and disclose, since the centralized server captures information about individual behavior and interests that is usually not as easily accessible as without MindShare.
CHAPTER 4

Experience

To gain experience with the MindShare approach, we conducted an initial trial using six subjects and a shared MindShare system.

4.1 Design

Two subjects participated in a ‘real-world’ trial, using MindShare in their daily work over a period of weeks. We had planned to have many more subjects participate in a real-world trial, but encountered difficulty finding volunteers who used the required operating system during the necessary time period. The remaining four subjects each individually used MindShare under observation for 30–60 minutes in sequence, in an accelerated approximation of normal activities. All subjects shared a MindShare server with a single composite ontology and separate personal ontologies.

The latter four subjects were asked to use MindShare to construct a personal ontology of their existing Web browser bookmarks (and any other Web pages they cared to), while the experimenter observed. It was emphasized that the personal ontology should reflect their own preferred way of organizing bookmarks for their own use, as if they planned to continue using MindShare in their daily work. The subjects were told how to perform operations such as adding a topic, bookmark a document, link to a subtopic, etc., but were not coached as to how organize their personal ontologies or respond to MindShare.

In addition to capturing the personal and composite ontologies in the MindShare database, we also kept a timestamped event log of all communication between MindShare server and clients during the trial, for later analysis.
4.2 Observations

The MindShare database that resulted from this trial is shown in Appendix B. The composite ontology contains 401 objects: 66 topics, 62 subtopic relations, 130 documents, and 143 document classification relations. There are 13 more classification relations than there are documents, because a single document can be classified under multiple topics. The following observations come from watching and questioning the users, and from later examination of the database.

Correctness

In a subjective examination of the composite ontology, it appeared almost entirely correct—by which we mean that for each relationship expressed in the composite ontology, a reasonable-person argument could be made that it is not incorrect. Put another way, except as noted below, there were no relationships that appeared incorrect.

There was one instance in which a subject created a subtopic relation in the direction opposite what they had intended. However, the user corrected this by changing the undesired relation to be never visible to them (after asking the observer how they could undo the operation).

There were a small number of instances in which it appeared that one subject might be treating a topic to mean something substantially different than what the original creator of the topic intended, but there was no clear evidence of this.

The only obvious instances of cavalier user behavior that would be likely to adversely affect the correctness of the composite ontology came from two subjects. In an attempt to see some topics from the composite ontology, they began using the Add Topic operation as a search feature, without actually intending to add a topic themselves. The subjects would type a keyword in both the topic name and description fields of the Add Topic dialog, then select one of the existing topics from the suggested topics dialog. Fortunately, in each of the 8–10 instances this was tried, the composite ontology actually contained good matches for the keywords; if none had existed, then the unintended new topic would be added to the composite ontology, cluttering it. Note that the subjects had been informed previously that everything added to MindShare is permanent, even though it can be hidden by individual users, and its influence on things such as document recommendations can decrease over time. This observation suggests a user desire for a keyword search feature, and also the need for MindShare mechanisms to better accomodate cavalier behavior.
**Topic Matching**

The matching of equivalent topics (i.e., that two people who have the same concept in mind will tend to use the same MindShare topic for it, rather than two separate topics) worked surprisingly well in this limited trial.

In most cases, the user would choose to add a topic, enter the name and description, and if an equivalent topic existed, it would be one of the first few suggested topics. However, it should be noted that the topic suggestion algorithm relies upon intersection of term vectors, and that no more than ten suggestions are ever shown to the user. Since any intersection, even with less significant terms, results in a match, this greater recall compensates for some poor precision. However, as the size of the composite ontology increases but the results list remains truncated, the result will be lower recall and therefore lower precision in some respects. Partly counterbalancing this is that as the size of the ontology increases, there is more term-frequency information, and therefore the suggestion algorithm will do a better job of ranking less common terms higher, increasing the precision for the many cases that lend themselves to TFIDF.

There was also one observed instance in which a subject was confused because the suggested topics list included two topics that seemed equivalent, so the subject did not know which was preferrable. This problem might be addressed by improving the overall topic-matching, or providing a visualization or hints as to which of the suggested topics were better-connected in the composite ontology or more popular in the user base’s personal ontologies.

As expected, there were many instances of subjects encountering topics while browsing the topic space and choosing to permanently-add those to their personal ontologies.

The only observed case of separate but equivalent topics in our trial occurred when one of the first subjects consciously chose to create an *Agents Research* topic after being exposed to an existing *Software Agents* topic, feeling that the topics had different meaning. The observer felt that the two topics were serving equivalent purposes in the composite ontology, but did not interfere. Later, one of the last subjects happened to encounter both topics at the same time, and asked the observer why there were “two topics for the same thing.” The observer answered that there were a variety of reasons, including that the topics were added by two separate people. The subject then asked how they could combine those two topics, and was told that one could be made a subtopic of the other, if that’s how the user wanted it to appear in their personal ontology. The user added a subtopic relation between the two, thereby linking the two topics. Although the MindShare meta-model does
not presently support equivalence relations, this use of the subtopic relation achieves much of the benefit of two separate topics—documents classified under one topic will tend to be recommended when a user selects the other topic, and the user will tend to be shown both topics when browsing through the topic space.

**Discovery and Search**

The use of MindShare for information search and serendipitous discovery was not exercised very heavily in the initial trial, but there were several encouraging anecdotes. For example, one of the real-world subjects reported adding a *Comics* topic one day, along with bookmarks for several comic strip Web sites, and that when he was viewing his comics several days later, he noticed a recommendation of a comic strip he had once seen and was glad to now have a bookmark for. Another subject remarked that the recommendations were “good for serendipity,” another said that the ranking order of the recommended documents seemed intuitive to him, and another said that the exposure to other people’s bookmarks was the “real strength” of the system.

For a later subject, we conducted a preliminary exercise of the bimodal browsing approach. With a rough idea of the kinds of topics that were in the composite ontology (including topics on personal finance and investing), we asked the subject to pose an information retrieval question and then try to use MindShare to answer it. The subject stated the question as “How can I get in on IPOs?” then double-clicked on a personal finance topic in their personal ontology to show subtopics, then double-clicked on one of the newly-revealed subtopics, then single-clicked on a subtopic of that. The subject did this with little pause, but hesitated when they noticed the recommended documents list, and stated that they had trouble deciding which document to try first. After a few seconds, the subject chose the first document, a Web page about IPOs on an online brokerage site. The user skinned down this page in a few seconds, then clicked on a link near the bottom of the page that took them to a page that told how to participate in IPOs with that brokerage.

**Altruism**

One unexpected behavior we observed seemed to be altruism, even though the subjects were instructed that they should only do things for their own personal benefit. One of the first subjects entered a long, descriptive title for a document. When asked afterwards by the observer why they did that, the subject said that the descriptive title would help other people have a better idea what the document was about. We later observed two other subjects doing the same thing. Although the idea of very
descriptive titles might’ve been suggested to them by examples they’d seen from the previous subject, both subjects said they were providing descriptive titles to help out other users. One of the three subjects added that they thought the descriptive titles would also benefit themself in remembering what the document was about.

As other evidence of altruism, one subject asked how to see the entire composite ontology. When asked why they wanted to do that, the subject said it was so that they could see where the clusters of knowledge were, and therefore “where it would be most useful to the group to add my knowledge.”

Although the MindShare approach was designed to assume no altruism, it was encouraging to see altruistic behaviours in this subject pool. This suggests that in some organizations, a small degree of altruism will help facilitate the goal of MindShare in sharing knowledge.

4.3 Conclusions

In this initial trial, what we were able to observe of the MindShare approach worked surprisingly well. Users constructed very different personal ontologies that were nevertheless well-matched in the composite ontology, and the various human-machine mechanisms worked very close to our hopes. Additionally, we discovered unexpected altruism behaviors and desire for other additional modes of searching and browsing.

Although this trial was with a small subject pool under constrained conditions, the results thus far are encouraging, and suggest that the MindShare approach has promise and warrants further investigation.

A series of controlled experiments to evaluate the MindShare approach are nontrivial due to the number of variables, but possible. For example, one possible experiment to evaluate some of the utility of MindShare for finding information would involve taking a MindShare database developed by a community. All of the subjects are then assigned the task of finding information that is contained in various of the organization’s documents. One group is given the existing MindShare system as a tool. The other group is given more conventional tools, such as keyword-based search.
CHAPTER 5

Related Work

This MindShare work touches upon several fields, including ontology construction, communityware, organizational knowledge management, and information retrieval.

5.1 Ontology Construction

As mentioned in Chapter 1, traditional ontology and knowledge engineering is performed by having centralized teams of trained ontologists or knowledge engineers encode information extracted from domain experts. The most prominent effort in this area is the CYC commonsense knowledgebase [21]. CYC was developed by a team of engineers for over a decade, and work is ongoing. The tools developed for the CYC knowledge engineers includes ontology browsers and visualizations, but are designed for the needs of knowledge engineers.

Ontolingua [4] is a Web-based collaborative ontology editing system, intended to help users build and reach consensus on formal ontologies. In contrast to MindShare, Ontolingua is intended for developing rich ontologies, rather than classifying documents; and it is oriented towards people with ontology-engineering training to engineer shared ontologies, rather than individuals who are merely concerned with organizing their own bookmarks and discovering new information.

The Pangloss Ontology [18] is constructed by merging various existing knowledge sources, such as online dictionaries and semantic networks, using semi-automatic ontology merging approaches.

An ontology somewhat more similar to MindShare’s is that of Yahoo [44], which is a large category-based directory of Web sites that has been engineered by a centralized team of editors for over five years, and presently references 1.2 million pages [38]. The Open Directory Project [2], is a Yahoo-like directory of Web resources, constructed by
volunteers distributed around the globe, rather than employed by a single company (even though ownership of the ontology is claimed by a company). A more ambitious approach to collaborative ontology construction is CoMeMo [28], which provides a mechanism for associative linking and merging of ontologies.

5.2 Communityware

An emerging field sometimes called communityware, community computing, or digital ecologies [24] considers the combination of worldwide virtual communities and augmented communication to accomplish tasks—having a different scope and perspective than traditional work in computer-supported collaborative work. [14] and [15] are notable references, collecting a variety of work in the area. [27] discusses some of the sociology of combining humans with technology for tasks.

5.3 Knowledge Management

More established than communityware is the field of knowledge management (KM), which concerns information in more traditional organizational structures, such as for-profit corporations. [39] distinguishes them as “management of information” and “management of people.” According to this view, MindShare is presently concerned primarily with the former—constructing software systems to capture and share information within the organization. These software systems sometimes have been called artificial knowledge management systems [5]. The field of KM has also been described with a catchphrase of goal of “The right information to the right people at the right time,” and we might also define it in a way appealing to business executives as, “Leveraging the knowledge that already exists in your organization.” There is also a large set of research by computer scientists that could be considered KM, even though the creators did not identify their work as such.

One relatively early effort in automated Web bookmarks sharing was Siteseer [32], which compared the URLs in users’ Web browser bookmark categories to find similar categories and recommend URLs. Grassroots [16] provided a structured environment for messages and resource noting that allows users to subscribe to collections of other users’ information.

A number of systems use automated collaborative filtering (ACF) [35] among the set of users to rank the usefulness of information. PHOAKS [13] used frequency-of-mention of URLs in Usenet articles to rank results from keyword queries, in addition
to using the Usenet hierarchy as an organizational structure. GroupLens [30] applies automated collaborative filtering of user ratings to rank Usenet posts, rather than Web documents. [25] describe a system for active collaborative filtering, supporting explicit recommendations of resources. [17] and Google[7] use the structure of hyperlinked references to Web documents as a means of ranking their popularity. Footprints [43] performs a kind of ACF on navigational paths within the information space that other users have taken to answer related questions.

Like one of the goals of MindShare, Answer Garden [1] combines information retrieval from a knowledgebase and matchmaking with people who have knowledge in the area.

MindMeld [42] allows people to quickly submit Web page URLs to “buckets” that represent communities of interest. Other people within those communities can then do on-demand information retrieval on the database of URLs, with the results ranked by keyword relevance and the reputation rating the person querying has assigned to the submitter of each URL. This differs from the MindShare approach in that URLs are contributed for the benefit of the group, rather than individual benefit; URLs are located via explicit keyword queries, rather than ontology-based navigation or recommendation; and MindMeld uses reputation of the submitter as a basis for ranking, whereas MindShare uses the popularity of ontology elements along a path as a basis for ranking.

TheBrain [40] is a bookmark manager with a graphical interface to a very loose semantic network that provides a simple means of publishing ones bookmarks on the Web so that others can browse it. Individual “brains” are not linked together, and the inconsistently-used semantic model appears to be a barrier to effective merging. However, the popularity of the TheBrain graphical interface is evidence that many users do prefer that view to the built-in hierarchical list views of current Web browsers.

Large commercial knowledge management systems such as Autonomy [3] combine a variety of features for capturing and sharing knowledge in an attempt to provide integrated practical solutions.

5.4 Information Retrieval

The field of information retrieval (IR) is concerned with means of finding documents within an electronic library, or items within a knowledge base, that are relevant to a query. A survey of the field and detailed discussion of the various algorithms can be found in [6]. MindShare borrows the popular IR algorithm TFIDF for its topic-suggestion algorithm.
IR algorithms typically work by extracting characterizing keyword vectors from each document, and comparing the similarity of two documents (one document often being a query document) to see if they are relevant to each other. SHOE [12] provides a knowledge representation markup language for Web pages, to enable inference-based document retrieval and knowledge querying. Amalthea [26] uses software agents to treat the Web as a loose knowledge base and both discover and monitor resources.

IR usually involves on-demand retrieval via explicit queries. The Remembrance Agent [31] extends this by continuously monitoring the part of a document a user is viewing or editing, and suggesting possibly relevant documents in an unobtrusive manner.
CHAPTER 6

Future Work

There are a variety of possible future directions for MindShare and outgrowths of it. The ones noted here can be categorized as further evaluation and refinement of the MindShare approach, extension of MindShare to more comprehensive knowledge management systems, MindShare-related ontology construction for other domains, and investigation into this area of sociotechnical systems design.

The trial we have done within the thesis timeframe is far from a thorough evaluation of all the interesting possible implications of the artifact and approach. The results thus far should be be considered preliminary. One methodology that may yield interesting results is to conduct real-world trials of MindShare with different kinds and numbers of users. This will suggest which aspects of MindShare's performance are sensitive to the technical training of the users, how the system is used by people with different classes of information need (e.g., research vs. engineering vs. customer service vs. marketing, etc.), how MindShare behaves when the user base has a mostly homogenous vs. heterogenous set of interests, and how the MindShare approach scales to user bases of one hundred, one thousand, ten thousand, and more.

Regarding how the ontology will scale, our suspicion is that the size of the ontology will grow at a rate no more than linear to the number of users. The growth could in fact be geometric or exponential (e.g., the number of potential subtopic relations is the square of the number of topics). However, the intuition is that the amount of material that each user adds for their own benefit will vary by less than an order of magnitude with respect to the size of the composite ontology or number of users. Additionally, as the composite ontology becomes larger and therefore more inclusive, it will increasingly tend to already contain things a user wishes to add.

The trials will also suggest how other various elements of the approach scale to larger ontologies. For example, we foresee that the current graph browser interface will become less manageable as the composite ontology becomes large and the degree
of subtopics per topic increases, and therefore it will be necessary to introduce more usability features. Also, as previously noted, it will be interesting to see how the topic suggestion and document ranking algorithms scale to a much larger composite ontology. The topic suggestion algorithm might be improved by adding additional sources of information. The document ranking algorithm might be similarly improved. Tuning the weighting factors for the ranking algorithm might also yield better scalability.

The trials should also suggest ways in which the prototype system can be improved. For example, the automatic graph layout currently in the prototype will likely need some improvement to scale to large and complex composite ontologies. Some incremental improvements that can be made as necessary include adding a conventional tree compaction algorithm from the field of graph drawing, adding variably-sized nodes and scalable fonts\textsuperscript{1}, and extending the LAYOUTTREEIFY algorithm to emphasize putting subtopics of the selected topic directly beneath it. Within the area of graph visualization, one further idea we would like to explore is extending the MindShare visualization to track the user's Web browsing behavior and increasingly emphasize the topics corresponding to each page the user visits, such that at any point in time, there is a visual characterization of the semantic neighborhood of recent browsing activity. This would both suggest to the user related topics and documents, and also suggest topics in which the user might wish to bookmark the current document. For another example of how the prototype can be improved, large amounts of real-world data will allow the document and topic recommendation algorithms to be tuned or rewritten. The weights used by the algorithm could also be adapted via reinforcement learning from observed user behavior. A related area of improvement of the prototype involves extending the ontology model, such as to allow equivalence relations and multiple names for the same object.

Another area of work involves incorporating the MindShare approach as part of a larger organizational or enterprise knowledge management system. For example, MindShare captures information about the information needs of people within the organization over time, and it would be helpful to expose this to the user, to more easily identify others within the organization who are knowledgeable about a given area or have previously done related work. The system could also capture structured discussions that reference documents within MindShare, and make past messages in the discussions available to future users encountering the documents. The system might also notify a user when new documents are added to a topic of especial interest to them, or when an active discussion might interest them (somewhat similar to the

\textsuperscript{1}The font and scaling support in the version of Java currently used by the MindShare prototype, JDK 1.1 with plain AWT, limited what could be done at this time.
real-time conversation-finding of Butterfly [41]). The MindShare ontology approach can also co-exist with conventional information retrieval mechanisms, such as full-text keyword-based retrieval and automatic document classification. MindShare-like interface might be one of the primary views to the knowledge management system.

As with other systems that capture information about people, privacy issues in real-world knowledge management systems must be addressed. This is important both for user acceptance, and more importantly, to ensure that any sacrifices of privacy are truly worthwhile. Consideration of privacy issues may lead to a different software architecture, from a centralized representation to a distributed one.

As mentioned in Chapter 1, a variety of application domains would benefit from new ontology development approaches. The MindShare approach addresses the construction and use of a particular kind of ontology for a particular application. Ontologies, and the means by which they are constructed, are necessarily application-sensitive. It would be interesting to attempt to apply variants on the MindShare approach to ontologies for Web semantic markup or product encoding. The eventual approaches might be fundamentally different than MindShare, yet retain some aspects of accidental collaboration and personalized perspectives.

A final area of further work was suggested by reflecting upon the nature of the MindShare approach during development. In essence, we have tried to treat individual humans as computational objects, with useful semi-predictable behaviors in response to interactions with other computational objects, and combine them with MindShare mediation to achieve a particular composite behavior. There is some work in the areas such as sociotechnical systems and operations research to understand how humans behave in a technological organization, and component-based systems design has been an area of particular interest to software engineering researchers in the past few years. We would like to investigate how these fields can be combined, to see how formal analysis and design methods can be applied to constructing computational systems of humans, computer networks, and smart mediation to achieve well-described emergent computational effects.
APPENDIX A

OMT Notation

Following is a key of the OMT [33] static object modeling notation used in this document. The concepts will probably be familiar to people with backgrounds in knowledge representation or database design.
Object Class:

<table>
<thead>
<tr>
<th>MyClass</th>
</tr>
</thead>
<tbody>
<tr>
<td>myAttribute1</td>
</tr>
<tr>
<td>myAttribute2</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>myAttributeN</td>
</tr>
</tbody>
</table>

Generalization:

```
A

X       Y       Z
```

Multiplicity of Association:

```
A -- B
A -- B
A -- B
A -- X -- B
```

Aggregation:

```
A -- B
```

Association:

```
Class A -- Role A -- Class B
```

Association as Class:

```
A

X

B
```

"The association of A and B is an X"

"A is the superclass of X, Y, and Z"

"For each A there is exactly-one B"

"For each A there are zero-or-one B's"

"For each A there are zero-or-more (many) B's"

"For each A there many B's, qualified by X"

"A is the whole, B is the part"
APPENDIX B

Sample Database

This appendix contains an edited dump of the MindShare database from the initial trial. In the interests of privacy, users have been pseudonymized with numbers (denoted #1 through #6), and other identifying or personal information has been replaced with a dot ellipsis in square brackets (“…”) Section headings have been added.

Top Matter

(mindshare-database "0.1")
(host  "mindshare.media.mit.edu")
(port 7783)
(lastid 401)
(user #1 (pass "[..]")) (user #2 (pass "[..]")) (user #3 (pass "[..]"))
(user #4 (pass "[..]")) (user #5 (pass "[..]")) (user #6 (pass "[..]"))

Composite Ontology Topics

(co topic 1 (by #1) (name "Communityware") (desc "computer software and networking that facilitates virtual communities"))
(topic 2 (by #1) (name "E-Commerce") (desc "electronic commerce or commerce over the Internet"))
(topic 6 (by #1) (name "E-Commerce Groups") (desc "groups doing E-commerce research"))
(topic 8 (by #1) (name "Communityware Groups") (desc "research groups in communityware"))
(topic 10 (by #1) (name "Research Groups") (desc "various research groups"))
(topic 13 (by #1) (name "People") (desc "home pages of individual people"))
(topic 16 (by #1) (name "Conferences") (desc "conferences on communityware"))
(topic 18 (by #1) (name "Software Agents") (desc "software agents"))
(topic 19 (by #1) (name "E-Commerce Agents") (desc "software agents for E-commerce"))
(topic 22 (by #1) (name "Multi-Agent Systems") (desc "multi-agent-systems"))
(topic 32 (by #1) (name "E-Commerce News") (desc "news on electronic commerce"))
(topic 48 (by #1) (name "Groupware") (desc "computer software that helps groups of people work together"))
(topic 52 (by #1) (name "Chat Systems") (desc "networked real-time text chat systems"))
(topic 84 (by #1) (name "IRC") (desc "the Internet Relay Chat system"))
(topic 59 (by #1) (name "Collaborative Hypermedia") (desc "systems that support multiple people collaborating on hypermedia or hypertext"))
(topic 65 (by #1) (name "Collaborative Web Browsing") (desc "software that supports collaborative web browsing"))
(topic 73 (by #1) (name "Knowledge Management") (desc "systems that leverage knowledge within an organization"))
(topic 83 (by #1) (name "Development") (desc "development commerce"))
(topic 85 (by #1) (name "Comics") (desc "comic strip urls"))
(topic 92 (by #1) (name "Search Engines") (desc "Web search engines"))
(topic 99 (by #1) (name "Media Lab") (desc "Media Lab Sites"))
(topic 107 (by #1) (name "[..]") (desc "[..]"))
(topic 117 (by #1) (name "Humor") (desc "humorous texts and cartoons"))
(topic 125 (by #1) (name "Fun Stuff") (desc "humor, comics, and other fun stuff"))
(topic 130 (by #1) (name "Linux") (desc "the Linux operating system"))
Composite Ontology Subtopic Relations

(topic 135 (by "#2") (name Remembrance Agents) (desc Remembrance Agents))
(topic 143 (by "#2") (name "RA Papers") (desc Papers on Remembrance Agents))
(topic 155 (by "#1") (name Wearables) (desc "wearable computers"))
(topic 158 (by "#1") (name Displays) (desc "displays for wearable computers"))
(topic 162 (by "#2") (name Linux Software) (desc "Software pages for Linux"))
(topic 167 (by "#1") (name Ubiquitous Computing) (desc "ubiquitous computing - computers everywhere"))
(topic 169 (by "#2") (name Copyright) (desc "Copyright laws, articles, and other material"))
(topic 174 (by "#1") (name "Money") (desc "all about personal finance and investing"))
(topic 175 (by "#1") (name "Investing") (desc "personal money investing"))
(topic 177 (by "#1") (name IPOs) (desc "initial public offering (IPO) research"))
(topic 191 (by "#1") (name HTML) (desc "world wide web (wWW) hypertext markup language"))
(topic 194 (by "#1") (name Internet) (desc "Internet technologies"))
(topic 196 (by "#2") (name "Shopping") (desc "shopper and price comparison sites"))
(topic 199 (by "#3") (name HCI) (desc "Human-Computer Interaction"))
(topic 210 (by "#3") (name Agents Research) (desc "Topics"))
(topic 222 (by "#4") (name Insurance) (desc "Insurance for business, home, auto, life"))
(topic 226 (by "#4") (name Real estate) (desc "Real estate as an investment"))
(topic 233 (by "#4") (name Law) (desc "General information about law"))
(topic 234 (by "#4") (name Real estate law) (desc "Specific information about real estate law in the US"))
(topic 254 (by "#1") (name Courses) (desc "MIT classes of interest to Media Lab students"))
(topic 259 (by "#5") (name "Band Hold Devices") (desc "Palm and other band held device information"))
(topic 270 (by "#5") (name MIT) (desc "MIT resources and links"))
(topic 275 (by "#1") (name Interface Agents) (desc "Agents that assist the user in the user interface"))
(topic 299 (by "#6") (name "Info retrieval agents") (desc "Agents that do information retrieval"))
(topic 305 (by "#5") (name Mutual Funds) (desc "Mutual Funds"))
(topic 310 (by "#5") (name Banks) (desc "Banks"))
(topic 316 (by "#6") (name Investing Advice) (desc "Investing Advice"))
(topic 326 (by "#2") (name Wearable News) (desc "Have about Wearable Computers (announcements, etc)"))
(topic 328 (by "#1") (name "Boston Housing") (desc "Housing in Boston, MA, USA"))
(topic 334 (by "#1") (name Apartments) (desc "apartments in boston, ma, us"))
(topic 338 (by "#1") (name Complexes) (desc "apartment complexes in boston, ma, us"))
(topic 348 (by "#1") (name "Computer Buying") (desc "information on buying computers"))
(topic 349 (by "#1") (name Local Resellers) (desc "resellers of computers in the cambridge/boston area"))
(topic 351 (by "#1") (name MailOrder Systems) (desc "Mail-order computer resellers"))
(topic 367 (by "#1") (name "PC Parts") (desc "computer parts resellers"))
(topic 365 (by "#1") (name PC Cases) (desc "PC computer case parts"))
(topic 381 (by "#1") (name Digital Cameras) (desc "digital cameras, or digitcams"))
(topic 382 (by "#1") (name Photography) (desc "all about taking photographs"))
(topic 387 (by "#1") (name Computer Science) (desc "the field of computer science"))
(topic 392 (by "#1") (name Lifestyle) (desc "living and lifestyle things"))
(topic 396 (by "#1") (name Sociotechnical Systems) (desc "integrative design of human and machine systems"))

(subrel 3 (by "#1") (parent 1) (child 2)) (subrel 7 (by "#1") (parent 2) (child 6))
(subrel 9 (by "#1") (parent 1) (child 8)) (subrel 11 (by "#1") (parent 10) (child 8))
(subrel 12 (by "#1") (parent 10) (child 6)) (subrel 17 (by "#1") (parent 1) (child 16))
(subrel 20 (by "#1") (parent 10) (child 18)) (subrel 21 (by "#1") (parent 13) (child 18))
(subrel 23 (by "#1") (parent 18) (child 22)) (subrel 24 (by "#1") (parent 22) (child 18))
(subrel 33 (by "#1") (parent 2) (child 32)) (subrel 40 (by "#1") (parent 1) (child 48))
(subrel 53 (by "#1") (parent 1) (child 82)) (subrel 56 (by "#1") (parent 82) (child 54))
(subrel 60 (by "#1") (parent 1) (child 59)) (subrel 60 (by "#1") (parent 81) (child 59))
(subrel 74 (by "#1") (parent 1) (child 73)) (subrel 84 (by "#1") (parent 2) (child 83))
(subrel 118 (by "#1") (parent 117) (child 85)) (subrel 126 (by "#1") (parent 125) (child 85))
(subrel 127 (by "#1") (parent 125) (child 85)) (subrel 130 (by "#2") (parent 18) (child 135))
(subrel 144 (by "#2") (parent 136) (child 143)) (subrel 159 (by "#1") (parent 155) (child 158))
(subrel 163 (by "#2") (parent 150) (child 152)) (subrel 168 (by "#1") (parent 167) (child 158))
(subrel 176 (by "#1") (parent 174) (child 175)) (subrel 178 (by "#1") (parent 175) (child 177))
(subrel 195 (by "#1") (parent 194) (child 195)) (subrel 211 (by "#3") (parent 210) (child 199))
(subrel 212 (by "#3") (parent 210) (child 2)) (subrel 213 (by "#3") (parent 210) (child 1))
(subrel 223 (by "#1") (parent 174) (child 222)) (subrel 226 (by "#4") (parent 175) (child 225))
(subrel 235 (by "#4") (parent 226) (child 234)) (subrel 237 (by "#4") (parent 233) (child 234))
(subrel 271 (by "#5") (parent 270) (child 254)) (subrel 250 (by "#5") (parent 210) (child 18))
(subrel 288 (by "#5") (parent 18) (child 287)) (subrel 300 (by "#6") (parent 287) (child 299))
(subrel 306 (by "#5") (parent 305) (child 174)) (subrel 307 (by "#6") (parent 174) (child 300))
(subrel 311 (by "#6") (parent 310) (child 174)) (subrel 312 (by "#6") (parent 174) (child 310))
(subrel 317 (by "#5") (parent 174) (child 316)) (subrel 327 (by "#2") (parent 155) (child 326))
(subrel 335 (by "#1") (parent 334) (child 336)) (subrel 337 (by "#1") (parent 334) (child 336))
(subrel 350 (by "#1") (parent 348) (child 349)) (subrel 352 (by "#1") (parent 348) (child 351))
(subrel 358 (by "#1") (parent 348) (child 357)) (subrel 366 (by "#1") (parent 357) (child 365))
(subrel 368 (by "#1") (parent 366) (child 381)) (subrel 388 (by "#1") (parent 387) (child 367))
(subrel 388 (by "#1") (parent 387) (child 18)) (subrel 388 (by "#1") (parent 387) (child 18))
(subrel 395 (by "#1") (parent 395) (child 396)) (subrel 395 (by "#1") (parent 395) (child 396))

66
Composite Ontology Document Classification Relations
Personal Ontology of Subject #1

(po "#1")

(topic 1 perm (aroot false)) (topic 2 perm (aroot false)) (topic 13 perm (aroot false)) (topic 18 perm (aroot false))
(topic 73 perm (aroot false)) (topic 85 perm (aroot false)) (topic 117 perm (aroot false)) (topic 128 perm (aroot false))
(topic 130 perm (aroot false)) (topic 158 perm (aroot false)) (topic 158 perm (aroot false)) (topic 158 perm (aroot false))
(topic 174 perm (aroot false)) (topic 175 perm (aroot false)) (topic 177 perm (aroot false)) (topic 191 perm (aroot false))
(topic 194 perm (aroot false)) (topic 333 perm (aroot false)) (topic 334 perm (aroot false)) (topic 336 perm (aroot false))
(topic 345 perm (aroot false)) (topic 349 perm (aroot false)) (topic 351 perm (aroot false)) (topic 357 perm (aroot false))
(topic 365 perm (aroot false)) (topic 381 perm (aroot false)) (topic 382 perm (aroot false)) (topic 357 perm (aroot false))
(topic 392 perm (aroot false)) (topic 396 perm (aroot false))
(subrel 3 perm) (subrel 74 perm) (subrel 118 perm) (subrel 120 perm) (subrel 159 perm) (subrel 168 perm)
(subrel 176 perm) (subrel 178 perm) (subrel 195 perm) (subrel 328 perm) (subrel 337 perm) (subrel 350 perm)
(subrel 352 perm) (subrel 358 perm) (subrel 366 perm) (subrel 383 perm) (subrel 386 perm) (subrel 388 perm)
(subrel 389 perm) (subrel 389 perm) (subrel 391 perm) (subrel 393 perm) (subrel 394 perm) (subrel 395 perm)
(subrel 397 perm)
(doc 4 perm) (doc 14 perm) (doc 25 perm) (doc 28 perm) (doc 30 perm) (doc 34 never) (doc 37 perm)
(doc 39 perm) (doc 41 perm) (doc 43 perm) (doc 50 perm) (doc 56 perm) (doc 61 perm)
(doc 63 perm) (doc 67 perm) (doc 69 perm) (doc 71 perm) (doc 75 perm) (doc 77 perm) (doc 79 perm)
(doc 81 perm) (doc 90 perm) (doc 113 perm) (doc 115 perm) (doc 119 perm) (doc 121 perm) (doc 123 perm)
(doc 125 perm) (doc 131 perm) (doc 133 perm) (doc 151 perm) (doc 153 perm) (doc 156 perm) (doc 160 perm)
(doc 172 perm) (doc 179 perm) (doc 181 perm) (doc 183 perm) (doc 185 perm) (doc 187 perm) (doc 189 perm)
(doc 353 perm) (doc 355 perm) (doc 359 perm) (doc 391 perm) (doc 393 perm) (doc 395 perm) (doc 397 perm)
(doc 399 perm) (doc 399 perm) (doc 399 perm) (doc 399 perm) (doc 399 perm) (doc 399 perm) (doc 400 perm)

Personal Ontology of Subject #2

(po "#2")

(topic 18 perm (aroot false)) (topic 85 perm (aroot false)) (topic 85 perm (aroot false)) (topic 99 perm (aroot false))
(topic 107 perm (aroot false)) (topic 130 perm (aroot false)) (topic 130 perm (aroot false)) (topic 130 perm (aroot false))
(topic 155 perm (aroot false)) (topic 155 perm (aroot false)) (topic 156 perm (aroot false)) (topic 160 perm (aroot false))
(topic 196 perm (aroot false)) (topic 336 perm (aroot false))
(subrel 136 perm) (subrel 144 perm) (subrel 159 perm) (subrel 163 perm) (subrel 165 perm) (subrel 327 perm)
(doc 56 perm) (doc 86 perm) (doc 88 perm) (doc 90 perm) (doc 92 perm) (doc 96 perm) (doc 97 perm)
(doc 100 perm) (doc 103 perm) (doc 105 perm) (doc 109 perm) (doc 111 perm) (doc 126 perm) (doc 131 perm)
(doc 133 perm) (doc 137 perm) (doc 139 perm) (doc 141 perm) (doc 145 perm) (doc 147 perm) (doc 149 perm)
(doc 155 perm) (doc 160 perm) (doc 165 perm) (doc 170 perm) (doc 197 perm) (doc 205 perm) (doc 320 perm)
(doc 87 perm) (doc 89 perm) (doc 91 perm) (doc 94 perm) (doc 96 perm) (doc 98 perm)
(doc 101 perm) (doc 102 perm) (doc 104 perm) (doc 106 perm) (doc 108 perm) (doc 110 perm)
(doc 112 perm) (doc 129 perm) (doc 132 perm) (doc 135 perm) (doc 140 perm) (doc 162 perm)
(doc 144 perm) (doc 148 perm) (doc 150 perm) (doc 151 perm) (doc 164 perm) (doc 166 perm)
(doc 171 perm) (doc 173 perm) (doc 206 perm) (doc 228 perm) (doc 330 perm)
Personal Ontology of Subject #3

(po "#3")
(topic 1 perm (aroot false)) (topic 2 perm (aroot false)) (topic 6 perm (aroot false)) (topic 18 perm (aroot false))
(topic 19 perm (aroot false)) (topic 22 perm (aroot false)) (topic 32 perm (aroot false)) (topic 83 perm (aroot false))
(topic 135 perm (aroot false)) (topic 199 perm (aroot false)) (topic 210 perm (aroot false))
(subrel 3 perm) (subrel 7 temp) (subrel 20 temp) (subrel 21 temp) (subrel 23 temp) (subrel 24 temp)
(subrel 33 temp) (subrel 84 temp) (subrel 136 temp) (subrel 211 perm) (subrel 212 perm) (subrel 213 perm)
(doc 200 perm) (doc 202 perm) (doc 204 perm) (doc 206 perm) (doc 208 perm)
(clarel 201 perm) (clarel 203 perm) (clarel 205 perm) (clarel 207 perm) (clarel 209 perm)

Personal Ontology of Subject #4

(po "#4")
(topic 18 perm (aroot true)) (topic 19 perm (aroot false)) (topic 22 perm (aroot false)) (topic 174 perm (aroot true))
(topic 175 perm (aroot false)) (topic 222 perm (aroot false)) (topic 226 perm (aroot false)) (topic 233 perm (aroot true))
(topic 234 perm (aroot false))
(subrel 20 perm) (subrel 23 perm) (subrel 24 perm) (subrel 176 perm) (subrel 223 perm) (subrel 226 perm)
(subrel 235 perm) (subrel 237 perm)
(doc 214 perm) (doc 216 perm) (doc 218 perm) (doc 220 perm) (doc 227 perm) (doc 229 perm) (doc 231 perm)
(doc 240 perm) (doc 242 perm) (doc 244 perm) (doc 246 perm) (doc 248 perm) (doc 250 perm) (doc 252 perm)
(clarel 218 perm) (clarel 219 perm) (clarel 219 perm) (clarel 221 perm) (clarel 224 perm) (clarel 228 perm)
(clarel 230 perm) (clarel 232 perm) (clarel 232 perm) (clarel 236 perm) (clarel 239 perm) (clarel 241 perm)
(clarel 243 perm) (clarel 245 perm) (clarel 247 perm) (clarel 249 perm) (clarel 251 perm) (clarel 253 perm)

Personal Ontology of Subject #5

(po "#5")
(topic 1 perm (aroot false)) (topic 2 perm (aroot false)) (topic 16 perm (aroot false)) (topic 18 perm (aroot false))
(topic 19 perm (aroot false)) (topic 199 perm (aroot false)) (topic 210 perm (aroot false)) (topic 259 perm (aroot false))
(topic 259 perm (aroot false)) (topic 270 perm (aroot false)) (topic 274 perm (aroot false)) (topic 294 perm (aroot false))
(subrel 3 never) (subrel 17 perm) (subrel 20 perm) (subrel 21 perm) (subrel 211 perm) (subrel 212 perm)
(subrel 213 perm) (subrel 271 perm) (subrel 280 perm)
(doc 204 perm) (doc 206 perm) (doc 255 perm) (doc 257 perm) (doc 260 perm) (doc 262 perm) (doc 264 perm)
(doc 266 perm) (doc 268 perm) (doc 272 perm) (doc 274 perm) (doc 276 perm) (doc 278 perm) (doc 281 perm)
(doc 283 perm)
(clarel 205 perm) (clarel 207 perm) (clarel 208 perm) (clarel 210 perm) (clarel 211 perm) (clarel 263 perm)
(clarel 265 perm) (clarel 267 perm) (clarel 269 perm) (clarel 273 perm) (clarel 275 perm) (clarel 277 perm)
(clarel 279 perm) (clarel 282 perm) (clarel 284 perm)

Personal Ontology of Subject #6

(po "#6")
(topic 18 perm (aroot false)) (topic 19 perm (aroot false)) (topic 174 perm (aroot false)) (topic 287 perm (aroot false))
(topic 287 perm (aroot false)) (topic 299 perm (aroot false))
(topic 305 perm (aroot false)) (topic 310 perm (aroot false)) (topic 316 perm (aroot false))
(subrel 288 perm) (subrel 300 perm) (subrel 306 never) (subrel 307 perm) (subrel 311 never) (subrel 312 perm)
(subrel 317 perm)
(doc 218 perm) (doc 289 perm) (doc 291 perm) (doc 293 perm) (doc 295 perm) (doc 297 perm) (doc 301 perm)
(doc 303 perm) (doc 308 perm) (doc 313 perm) (doc 318 perm) (doc 320 perm) (doc 322 perm) (doc 324 perm)
(clarel 290 perm) (clarel 292 perm) (clarel 294 perm) (clarel 296 perm) (clarel 298 perm) (clarel 302 perm)
(clarel 304 perm) (clarel 309 perm) (clarel 314 perm) (clarel 315 perm) (clarel 319 perm) (clarel 321 perm)
(clarel 323 perm) (clarel 325 perm)
APPENDIX C

Network Protocol

This appendix contains a reformatted and abridged version of the original MindShare protocol specification, which was originally written with the intention that it allow the prototype client and server to be implemented in parallel by different programmers. The protocol was revised during prototyping, but remains essentially the same. This specification is included to better illustrate the nature of the MindShare representation and communication between client and server.

Introduction

The MindShare prototype is architected such that the representation and computation happen in the server, and the clients need only concern themselves with the graphical user interface and talking with the server. The protocol between the client and server is designed to be programmer-readable (so that the pieces can be tested and debugged rapidly with Telnet, for example), easy to implement, and easy to extend.

Connection

Communication between the MindShare server and client is via TCP. A session between the client and server exists within the duration of the TCP connection. A TCP connection may contain no more than one session during its lifetime. A TCP connection is initiated by the client connecting to the server listening port (default 7783). The connection is not required to be 8-bit clean. The protocol is synchronous, such that it is always exactly one party’s turn to speak. Upon connection, the server is in the ClientIdentify state, from which a session is created.
Encoding

All messages are composed of lines of case-sensitive 7-bit ASCII printable and horizontal whitespace characters. Each line is terminated by an end-of-line symbol, \texttt{\langle eol \rangle}, which is the ASCII linefeed character (0A\text{\text{16}}).

Sequences of one or more horizontal whitespace characters—space (20\text{\text{16}}) and tab (09\text{\text{16}})—separate tokens in a line. Leading and trailing whitespace on each line is ignored. Though not strictly legal, both server and client should gracefully treat all nonprintable characters except for linefeed (i.e., 00\text{\text{16}}-19\text{\text{16}} and 7F\text{\text{16}}-FF\text{\text{16}}) as whitespace. Blank lines (i.e., \texttt{\langle eol \rangle} optionally preceded by whitespace) are ignored.

A string literal, \texttt{\langle String \rangle}, is delimited by double-quote characters (22\text{\text{16}}). Legal within a string literal are space and printable characters except for the double-quote character itself (i.e., 20\text{\text{16}}-21\text{\text{16}} and 23\text{\text{16}}-7E\text{\text{16}}), all of which are preserved as the lexeme of the token when parsing. The only exception to this is the percent character (\texttt{\%}, 25\text{\text{16}}), which escapes a two-character hexadecimal code that specifies an ASCII character that is to replace the three-character sequence after tokenization (e.g., \texttt{\%5A} becomes \texttt{Z}).

The grammar is defined in this document using EBNF syntax. The horizontal whitespace separating each token on a line in the right-hand side of each production is assumed. Square brackets ('[ ' and '] ') delimit an optional pattern. Curly braces ('{ ' and '} ') followed by an asterisk ('*' ) delimit a pattern that occurs zero or more times. Curly braces followed by a plus sign ('+' ) delimit a pattern that occurs one or more times.

In this grammar, \texttt{\langle WholeNumber \rangle} is an sequence of one or more digit characters (30\text{\text{16}}-39\text{\text{16}}) encoding a decimal integer one or greater. \texttt{\langle Boolean \rangle} is the literal keyword \texttt{true} or \texttt{false}.

Errors

There are some "internal" error conditions that should not happen, but that nevertheless can if the client or server is behaving incorrectly. This does not include errors in user input. For example, the client may invoke an operation on an object that doesn't exist in the server, or may use an invalid request name or parameter syntax. In these cases, the server will send an \texttt{\langle Error \rangle} message with a human-readable string describing the cause. It is the responsibility of the client to present the error message to the user and take appropriate error-recovery actions. For each client request that can result in a semantic error, this fact is noted in the description of the request.

\texttt{\langle Error \rangle} := \texttt{error} \texttt{\langle String \rangle} \texttt{\langle eol \rangle}

Users

Each user account on the system is identified by a unique user ID, which for the sake of simplicity in this experimental prototype is their E-mail address in "user\texttt{@domain}" form.
All addresses must be absolute within world scope, not relative to the media.mit.edu domain. The user ID is case-insensitive, and should be forced to lowercase. Within the grammar, the user ID is represented as \((UserID)\) and encoded as \(\langle String \rangle\).

Each user has a password for crude authentication purposes, which is sent in plaintext during the UserIdentify server state.

\[
\langle UserID \rangle \quad ::= \quad \langle String \rangle
\]

\[
\langle Password \rangle \quad ::= \quad \langle String \rangle
\]

**Graphs**

Subsets of the user's personal ontology view of topics and subtopic relations are sent by the server in certain situations as \((Graph)\). It comprises a serialized set of \((GraphItem)\)s, each of which can be a topic \((\langle Topic \rangle\)) or a subtopic relation \((\langle SubRel \rangle\)). Each \((GraphItem)\) has a numeric ID that uniquely identifies it within the system. Within a \((Graph)\) sequence of \((GraphItem)\)s, an item can be referenced by its ID before it is defined. After a \((Graph)\) is complete, any items with unresolved references can be ignored.

\[
\langle Graph \rangle \quad ::= \quad graph \langle eol \rangle
\]

\[
\quad \{(GraphItem)\}^* \quad \langle eol \rangle
\]

\[
\quad \langle Graph \rangle \quad ::= \quad \langle GraphItem \rangle \quad | \quad \langle SubRel \rangle
\]

Each \((Topic)\) has an ID, a human-readable name, a visibility attribute \((\langle Vis \rangle)\), and an always-root flag attribute \((\langle TopicRoot \rangle)\). \((Vis)\) specifies whether the topic is visible permanently ('perm'), temporarily ('temp'), not right now ('not'), or never ('never'). \((TopicRoot)\) specifies whether the topic should always be laid out as a "root" node ('aroot'), or that it does not need to be a root node ('naroot').

\[
\langle Topic \rangle \quad ::= \quad topic \langle TopicID \rangle \langle TopicName \rangle \langle Vis \rangle [\langle TopicRoot \rangle] \langle eol \rangle
\]

\[
\langle TopicID \rangle \quad ::= \quad \langle WholeNumber \rangle
\]

\[
\langle TopicName \rangle \quad ::= \quad \langle String \rangle
\]

\[
\langle Vis \rangle \quad ::= \quad perm \quad | \quad temp \quad | \quad never
\]
Each \(\langle\text{SubRel}\rangle\) has an ID, ID-based references to its supertopic and subtopic, and a \(\langle\text{Vis}\rangle\) attribute like that of \(\langle\text{Topic}\rangle\).

\[
\langle\text{SubRel}\rangle \quad ::= \quad \text{subrel} \; \langle\text{SubRelID}\rangle \; \langle\text{SuperTopicID}\rangle \; \langle\text{SubTopicID}\rangle \; \langle\text{Vis}\rangle \; \langle\text{eol}\rangle
\]

\[
\langle\text{SubRelID}\rangle \quad ::= \quad \langle\text{WholeNumber}\rangle
\]

\[
\langle\text{SuperTopicID}\rangle \quad ::= \quad \langle\text{TopicID}\rangle
\]

\[
\langle\text{SubTopicID}\rangle \quad ::= \quad \langle\text{TopicID}\rangle
\]

→ graph
→ topic 32 "Pets" perm aroot
→ topic 24 "Dogs" perm naroot
→ topic 36 "Cats" perm naroot
→ subrel 10 32 24 perm
→ subrel 99 32 36 perm
→ end graph

Documents

While documents are part of the ontology graph in the MindShare server, from the user's and client's perspective, they are handled separately and more simply. A document \(\langle\text{Doc}\rangle\) has a unique ID, a human-readable name (e.g., a title of an article), and a URL. Note that in the MindShare server, a given document may have multiple names and various classification relations, but this is hidden from the client. Some server replies that need to communicate a sequence or set of documents use \(\langle\text{DocList}\rangle\).

\[
\langle\text{Doc}\rangle \quad ::= \quad \text{doc} \; \langle\text{DocID}\rangle \; \langle\text{DocName}\rangle \; \langle\text{URL}\rangle \; \langle\text{eol}\rangle
\]

\[
\langle\text{DocID}\rangle \quad ::= \quad \langle\text{WholeNumber}\rangle
\]

\[
\langle\text{DocName}\rangle \quad ::= \quad \langle\text{String}\rangle
\]

\[
\langle\text{URL}\rangle \quad ::= \quad \langle\text{String}\rangle
\]

\[
\langle\text{DocList}\rangle \quad ::= \quad \text{doclist} \; \langle\text{eol}\rangle
\]

\[
\quad \langle\text{Doc}\rangle^* \quad \text{end} \; \langle\text{doclist}\rangle \; \langle\text{eol}\rangle
\]

→ doc 12210 "Much Ado About Nothing"/
   "http://www.shakespear.com/products/maan/"
State: ClientIdentify

Once the TCP connection is established, the server is in the ClientIdentify state. In this state, the server first waits for \((\text{ClientIdent})\), which describes the version of the protocol that the client supports and also provides a human-readable string identifying the variety and version of the client software. If the server receives a correct \((\text{ClientIdent})\) with a supported \((\text{VersionNum})\), then the server sends \((\text{VersionOk})\) and transitions to the UserIdentify state.

\[
\begin{align*}
(\text{ClientIdent}) & \quad ::= \text{mindshare client} \ (\text{VersionNum}) \ (\text{SoftwareID}) \ (\text{eol}) \\
(\text{VersionNum}) & \quad ::= \ (\text{String}) \\
(\text{SoftwareID}) & \quad ::= \ (\text{String}) \\
(\text{VersionOk}) & \quad ::= \text{hello} \ (\text{eol})
\end{align*}
\]

\[
\to \text{mindshare client "0.1" "MindShare Linux GTK client 1.5"}
\to \text{hello}
\]

If the server does not receive \((\text{Version})\), or the specified \((\text{VersionNum})\) is not supported, then the server sends \((\text{ClientBad})\) with an optional human-readable \((\text{ClientBadWhy})\) and closes the connection.

\[
\begin{align*}
(\text{ClientBad}) & \quad ::= \text{clientbad} \ [{(\text{ClientBadWhy})}] \ (\text{eol}) \\
(\text{ClientBadWhy}) & \quad ::= \ (\text{String})
\end{align*}
\]

\[
\to \text{mindshare client "13.0 E&E" "Microlimp MindShare 99 SP3"}
\to \text{clientbad "Your client wants an unsupported protocol." Disconnect by server.}
\]

\[
\to \text{HELO}
\to \text{clientbad "You do not appear to be a MindShare client." Disconnect by server.}
\]

State: UserIdentify

In the UserIdentify state, the server waits for either \((\text{Login})\) or \((\text{NewUser})\). If neither is received within a 60-second timeout period, the server disconnects.
Login

The \( \langle \text{Login} \rangle \) message in the \textit{UserIdentify} state specifies the user ID and password of the user. The server replies with \( \langle \text{LoginBad} \rangle \) or \( \langle \text{LoginOk} \rangle \). If \( \langle \text{LoginBad} \rangle \), then the server closes connection after sending the message with optional human-readable \( \langle \text{LoginBadWhy} \rangle \) reason. If \( \langle \text{LoginOk} \rangle \), then the server sends zero or more \( \langle \text{Notice} \rangle \)s to be presented to the user, and then a \( \langle \text{Graph} \rangle \) of the initial ontology view. After sending these, the server transitions to \textit{RequestWait} state.

\[
\begin{align*}
\langle \text{Login} \rangle & \quad ::= \text{login} \langle \text{UserID} \rangle \langle \text{Password} \rangle \langle \text{eol} \rangle \\
\langle \text{LoginBad} \rangle & \quad ::= \text{loginbad} [\langle \text{LoginBadWhy} \rangle] \langle \text{eol} \rangle \\
\langle \text{LoginBadWhy} \rangle & \quad ::= \langle \text{String} \rangle \\
\langle \text{LoginOk} \rangle & \quad ::= \text{loginok} \langle \text{eol} \rangle \\
& \quad \quad \{\langle \text{Notice} \rangle\}^* \\
& \quad \quad \langle \text{Graph} \rangle \\
& \quad \quad \text{end [loginok]} \langle \text{eol} \rangle \\
\langle \text{Notice} \rangle & \quad ::= \text{notice} \langle \text{String} \rangle \langle \text{eol} \rangle 
\end{align*}
\]

\[
\begin{align*}
\rightarrow & \text{ login "root@mit.edu" "ph34r"} \\
\leftarrow & \text{ loginok} \\
\leftarrow & \text{ notice "We're upgrading for a few minutes at 4am EDT."} \\
\leftarrow & \text{ graph} \\
& \quad \text{ topic 31337 "Computer Security" perm aroot} \\
& \quad \ldots \\
\leftarrow & \text{ end graph} \\
\leftarrow & \text{ end loginok} \\
\rightarrow & \text{ login "ayn~objectivism.com" "letthemeatcake"} \\
\leftarrow & \text{ loginbad "Your user ID or password was incorrect."} \\
\end{align*}
\]

\textit{Disconnect by server.}

NewUser

The \( \langle \text{NewUser} \rangle \) message in the \textit{UserIdentify} server state allows the client to request that a new user account be created with the specified \( \langle \text{UserID} \rangle \) and \( \langle \text{Password} \rangle \). The server will either create the account and respond with \( \langle \text{LoginOk} \rangle \), or respond with \( \langle \text{LoginBad} \rangle \) and a disconnect. If \( \langle \text{LoginOk} \rangle \), the \( \langle \text{Graph} \rangle \) may be empty or contain a default initial set of \( \langle \text{GraphItem} \rangle \)s.
\(\langle NewUser\rangle \quad ::= \text{newuser} \langle UserID\rangle \langle Password\rangle \langle eol\rangle\)

\[\rightarrow \text{newuser} \quad "\text{pawl@linux.hypnotic.org}" \quad "\text{outlate}"
\quad \leftarrow \text{loginok}
\quad \leftarrow \text{notice} \quad "\text{Welcome to MindShare!}"\]
\[\quad \leftarrow \text{graph}\]
\[\quad \leftarrow \quad \text{topic} \quad 1011 \quad "\text{Software Packages}" \quad \text{perm aroot}\]
\[\quad \leftarrow \quad \text{topic} \quad 42 \quad "\text{Philosophy}" \quad \text{perm aroot}\]
\[\quad \leftarrow \quad \text{topic} \quad 69 \quad "\text{Recreation}" \quad \text{perm aroot}\]
\[\quad \leftarrow \quad \text{end graph}\]
\[\quad \leftarrow \quad \text{end loginok}\]

\[\rightarrow \text{newuser} \quad "\text{cypherpunks@cypherpunks.org}" \quad "\text{cypherpunks}"
\quad \leftarrow \text{loginbad} \quad "\text{Sorry, that user ID already exists.}"\]
\quad \text{Disconnect by server.}\]

\[\rightarrow \text{newuser} \quad "\text{bubba@whitehouse.gov}" \quad "\text{interns}"
\quad \leftarrow \text{loginbad} \quad "\text{Sorry, that password is too obvious.}"\]
\quad \text{Disconnect by server.}\]

**State: RequestWait**

After the *UserIdentify* state, the rest of the session is spent by the server in the *RequestWait* state. In this state, the server is driven by requests from the client. This section of the document lists the available request protocol.

**Logoff**

The client should send \(\langle Logoff\rangle\) before initiating a disconnect. The server will send \(\langle LogoffReply\rangle\) and drop the connection.

\[\langle Logoff\rangle \quad ::= \text{logoff}\]
\[\langle ByeReply\rangle \quad ::= \text{bye}\]

\[\rightarrow \text{logoff}\]
\[\quad \leftarrow \text{bye}\]
\quad \text{Disconnect by server.}\]
Ping

The `<Ping>` request can be used to determine whether the server is responding and what the response latency or lag is. It has an optional arbitrary `<String>` parameter that is returned verbatim to the client in `<PingReply>`. If the client suspects network problems, it could conceivably send multiple `<Ping>`s, timing-out after each, breaking with the usual synchronous mode of this protocol.

```
<Ping> ::= ping [String] <eol>
<PingReply> ::= pong [String] <eol>
```

→ ping 2800347
← pong 2800347

ViewSubs

The client can request that all the subtopics of a specified topic be added to the view with the `<ViewSubs>` request. The server responds with a `<Graph>` of the requested topics and the subtopic relations between them and any other topics visible to the user. It is important to note that the `<Graph>` may include `<GraphItem>`s that are already visible, although it also has the option of not including things it knows are already visible to the user. `<Error>` results if the specified topic does not exist.

```
<ViewSubs> ::= viewsubs <TopicID> <eol>
<ViewSubsReply> ::= <Graph>
| <Error>
```

→ viewsubs 10
← graph
← topic 20 "Foo" temp naroot
← topic 30 "Bar" temp naroot
← topic 40 "Baz" temp naroot
← subrel 91 10 20 temp
← subrel 92 10 30 temp
← subrel 93 10 40 temp
← subrel 94 30 20 temp
← subrel 95 30 40 temp
← subrel 96 80 20 temp
← end graph
ViewSupers

The \((\text{ViewSupers})\) request is like the \((\text{ViewSubs})\) request, except it refers to supertopics rather than subtopics.

\[
\begin{aligned}
\langle \text{ViewSupers} \rangle &::= \text{viewsupers} \langle \text{TopicID} \rangle \langle \text{eol} \rangle \\
\langle \text{ViewSupersReply} \rangle &::= \langle \text{Graph} \rangle \\
&\quad| \langle \text{Error} \rangle
\end{aligned}
\]

SuggTopics

When the user says to add a topic named \((\text{TopicName})\) with description \((\text{TopicDesc})\), the client makes a \((\text{SuggTopics})\) request to get a ranked list of likely existing topics. The client can optionally specify a \((\text{TopicID})\) of the topic that is the intended supertopic of the new one (note that this is potentially very useful information, so a “create subtopic” operation is much better than “create topic”). The server responds with \((\text{SuggTopicsReply})\), which contains zero or more \((\text{SuggTopicsItem})s\). Each of those items has a topic ID, a \((\text{TopicPath})\), and a \((\text{TopicDesc})\). The idea is for the client to then present a popup menu of the suggestions to the user and allow the user to click on an existing suggested topic (causing the client to send \((\text{AddExistTopic})\), or on an “other” menu item (causing \((\text{AddNewTopic})\)).

\[
\begin{aligned}
\langle \text{SuggTopics} \rangle &::= \text{suggtopics} \langle \text{TopicName} \rangle \langle \text{TopicDesc} \rangle [\langle \text{TopicID} \rangle] \langle \text{eol} \rangle \\
\langle \text{SuggTopicsReply} \rangle &::= \text{suggtopicsreply} \langle \text{eol} \rangle \\
&\quad\{[\langle \text{SuggTopicsItem} \rangle]^{*} \\
&\quad\text{end} \text{[suggtopicsreply]} \langle \text{eol} \rangle
\end{aligned}
\]

\[
\begin{aligned}
\langle \text{SuggTopicsItem} \rangle &::= \text{sugg} \langle \text{TopicID} \rangle \langle \text{TopicPath} \rangle \langle \text{TopicDesc} \rangle \langle \text{eol} \rangle
\end{aligned}
\]

\[
\begin{aligned}
\langle \text{TopicPath} \rangle &::= \langle \text{String} \rangle
\end{aligned}
\]

The \((\text{TopicPath})\) can be equivalent to a \((\text{TopicName})\), or it can include a partial path through the subtopic relations in the the ontology that the server thinks provides relevant context.

\[
\begin{aligned}
\rightarrow \text{suggtopics} "\text{Lisp}" "\text{the Lisp language}"
\leftrightarrow \text{suggtopicsreply} \\
\leftrightarrow \text{sugg} 375 "\text{Computers \to Languages \to Lisp}" \\
\quad\quad \text{"The Lisp computer programming language"}
\leftrightarrow \text{sugg} 409 "\text{Functional Languages \to Common Lisp}" "\text{Lisp}"
\leftrightarrow \text{sugg} 747 "\text{Lisps}" "\text{the speech impediment}"
\leftrightarrow \text{end sugtopicsreply}
\end{aligned}
\]
AddExistTopic

After the client has done \textit{SuggTopics}, it can do \textit{AddExistTopic} to add an existing topic to the user’s view by making it permanently-visible and making subtopic relations between it and currently-visible things temporarily-visible (if they’re not already visible).

\begin{align*}
\langle AddExistTopic \rangle &::= \text{addexisttopic } \langle \text{TopicID} \rangle \ (\text{eol}) \\
\langle AddExistTopicReply \rangle &::= \langle \text{Graph} \rangle
\end{align*}

\begin{verbatim}
... \\
→ addexisttopic 375 \\
← graph \\
← topic 375 "Lisp" perm naroot \\
← subrel 123 1024 375 temp \\
← end graph
\end{verbatim}

AddNewTopic

The client can add a new topic to the composite ontology and the user’s view with \textit{AddNewTopic}. This should only be used after doing \textit{SuggTopics}.

\begin{align*}
\langle AddNewTopic \rangle &::= \text{addnewtopic } \langle \text{TopicName} \rangle \ (\text{TopicDesc}) \ (\text{eol}) \\
\langle AddNewTopicReply \rangle &::= \langle \text{Graph} \rangle
\end{align*}

\begin{verbatim}
... \\
→ addnewtopic "Lisp" "the Lisp language" \\
← graph \\
← topic 666 "Lisp" perm naroot \\
← end graph
\end{verbatim}

AddSubRel

The \textit{AddSubRel} adds a subtopic relation between two specified topics.

\begin{align*}
\langle AddSubRel \rangle &::= \text{addsubrel } \langle \text{SupertopicID} \rangle \ (\text{SubtopicID}) \ (\text{eol}) \\
\langle AddSubRelReply \rangle &::= \langle \text{SubRel} \rangle
\end{align*}

\begin{verbatim}
→ addsubrel 13 50 \\
← subrel 1000 13 50 perm
\end{verbatim}
AddDoc
A document is added to the composite ontology and user visibility with \(<AddDoc>\).
\[
<AddDoc> ::= adddoc \langle DocName \rangle \langle URL \rangle \langle TopicID \rangle \langle eol \rangle
\]
\[
<AddDocReply> ::= <Doc>
\]

→ adddoc "Winona Ryder’s Diary" "http://wrd/" 444
← doc 897 "Winona Ryder’s Diary" "http://wrd/"

SetVis
\(<SetVis>\) is used to change the visibility of a topic, subtopic relation, or document. There is no reply.
\[
<SetVis> ::= setvis \langle ObjectKind \rangle \langle ID \rangle \langle Vis \rangle \langle eol \rangle
\]
\[
<ObjectKind> ::= topic
| subrel
| doc
\]
\[
:ObjectID ::= \langle WholeNumber \rangle
\]

→ setvis topic 5638 perm
→ setvis subrel 3540 never

HideNonPerm
The \(<HideNonPerm>\) request removes from visibility everything in the graph that is not permanently-visible. The server sends \(<HideNonPermReply>\), which is a \(<Graph>\) containing items to remove from view.
\[
<HideNonPerm> ::= hidenonperm \langle eol \rangle
\]
\[
<HideNonPermReply> ::= \langle Graph \rangle
\]

→ hidenonperm
← graph
← topic 3876 "Pow" not
← topic 3845 "Zap" not
← topic 3890 "Thud" not
← end graph

83
GetPermDocs

The \(\text{GetPermDocs}\) request gets a \(\text{DocList}\) of documents that the user has permanently classified under the specified \(\text{TopicID}\).

\[
\text{GetPermDocs} \quad ::= \quad \text{getpermdocs} \ \text{TopicID} \ \text{eol}
\]

\[
\text{PermDocs} \quad ::= \quad \text{DocList}
\]

\[\rightarrow \text{getpermdocs 365} \quad \leftarrow \text{doclist} \quad \leftarrow \text{doc 87634 "What's an Agent, Anyway?" "http://foo/"} \quad \leftarrow \text{doc 87635 "Oh, So That's What It Is!" "http://bar/"} \quad \leftarrow \text{end doclist}\]

RecoDocs

The \(\text{RecoDocs}\) request causes the server to compute and return a ranked \(\text{DocList}\) of documents relevant to the specified \(\text{TopicID}\). The relevance is a function factors such as of graph distance, number of people who have permanently added it, etc.

\[
\text{RecoDocs} \quad ::= \quad \text{recodocs} \ \text{TopicID} \ \text{eol}
\]

\[
\text{RecoDocsReply} \quad ::= \quad \text{DocList}
\]

\[\rightarrow \text{recodocs 555} \quad \leftarrow \text{doclist} \quad \leftarrow \text{doc 735 "History of Western Philosophy" "http://howf/"} \quad \leftarrow \text{doc 750 "Cocktail Party Prepbook" "http://vacuous/"} \quad \ldots \quad \leftarrow \text{end doclist}\]

Security and Privacy

Some minor security and privacy issues of the protocol have been identified so far. Note that issues of the broader MindShare architecture and usage are not discussed here.

There is an obvious denial-of-service attack if a user is able to create an account with someone else’s E-mail address. To partly reduce this vulnernability, and to encourage use of valid E-mail addresses by test users, we may choose to implement E-mail confirmation on the creation of a new account.
The user's password is sent in plaintext, the biggest risk of which is that this will expose a password that's also used for more private systems to sniffing. This is not felt to be a serious security vulnerability for purposes of the research prototype, however. In a real-world implementation, the password should be encrypted and perhaps the entire communication channel should be as well.
Bibliography


