A User Interaction Model for Browsing Based on Iterative Category-Level Operations

Michelle Q Wang Baldonado and Terry Winograd

Abstract

We propose a user interaction model for browsing based on iterative category-level operations. The motivation comes from two observations: 1) people naturally think in terms of categories, and 2) in browsing, the types of categories that are salient to users change as they browse. We define a set of category-level operations that lets users iteratively view and find results in terms of these changing category types. We also show that we can express some standard IR operations as iteratively applied sequences of a fundamental category-level operation (thus unifying them). Finally, we describe SenseMaker, a prototype interface for browsing heterogeneous sources.

1.0 Introduction

Why develop a new user interaction model for browsing? As outlined in [4], browsing has very different properties from analytic search. Browsing strategies, unlike analytic search strategies, are opportunistic, interactive, and lacking in well-defined user goals. Users who are browsing continually find that new dimensions become important to them. As a consequence, user interaction models based on result-level operations do not synchronize well with users' needs. It is time consuming for users to analyze lists of results in order to choose new interesting avenues to pursue.[1] The user interaction model we define in this paper addresses this problem by allowing users to repeatedly specify dimensions of interest and to view and obtain results in terms of these changing dimensions. More specifically, our approach relies on categories as a natural and compact way of expressing results in terms of user-specified dimensions. In the model we propose, choosing a dimension of interest is equivalent to choosing a criterion for categorizing results.

In Section 2.0, we discuss in more detail what types of conceptual categories and category-level operations are useful for browsing, as well as issues involved in determining category membership. We then operationally define our user model and articulate a fundamental category-level operation. Finally, we take a brief look at how to handle categories in a heterogeneous digital library, where a change in the set of search services being queried can affect which categorization criteria the user finds important and which attributes are available for category approximation. In Section 3.0, we show that iterative text-based

1. Furthermore, the time needed to analyze results is certain to increase as users gain the ability to query large numbers of search services at once.

clustering methods and text-based relevance feedback techniques can be expressed in terms of our fundamental category-level operation. In addition, we use the fundamental category-level operation to give concrete definitions for the category-level operations suggested in Section 2.1. In Section 4.0, we present a scenario which shows how the iterative category-level operations we have defined might be used. We discuss SenseMaker, our prototype category-based interface for browsing heterogeneous sources, in Section 5.0. Finally, we discuss our conclusions and plans for future work in Section 6.0.

2.0 The Category-Based User Interaction Model

2.1 Overview

In browsing, the conceptual categories that users find relevant are those that reflect distinctions commonly made in our shared literary tradition. These distinctions tend to revolve around similarities in the intellectual content of works or similarities in their production history. We list examples of categories based on these distinctions below.

- works on the same subject
- works created by the same individual person
- works created by people at the same institution (e.g., Stanford)
- works that differ only in version (e.g., all the different editions of *Twelfth Night*)
- works created during the same time period
- works in the same language
- works in the same genre (e.g., novels vs. textbooks vs. technical papers)

A category-based browsing system can take advantage of a user's tacit understanding of literary tradition by incorporating these distinctions.

What types of category-level operations are needed by users who are browsing? As discussed in Section 1.0, users encountering a set of results for the first time will almost certainly want to see the results categorized in multiple ways. Hence, simple categorization operations are required. Furthermore, users may wish to expand searches as new dimensions of interest emerge. For example, a user who initially issues a query asking for technical papers on a particular subject may decide, after seeing a categorization of the results by author, that he really is interested in finding more works by some of those authors without losing the results he has already found. Thus, category-directed search expansion operations are also important. In Section 3.2, we will give precise definitions for the kinds of operations we have suggested here.

2.2 Determining category membership

In developing an interaction model that revolves around categories, we must recognize that determining membership in semantically-defined categories can be quite difficult. Traditionally, catalogers have been responsible for deciding when citations belong to the same category. An important resource for catalogers is the Library of Congress (LC) authority record database, which defines a one-to-one mapping between the set of LC authority records and the set of unique identities known to the LC. For example, LC "author" authority records uniquely correspond to individual authors known to the LC. A cataloger can use one of these "author" authority records to specify in a citation the real author of the corresponding work.

The problem with LC authority records is that creating them is currently a very labor-intensive process (given the conventions that govern what meta-information authors encode for their works). Hence, many of the databases and search services that will find their way into the digital library will not include this identity-establishing information. If we want to allow users who browse the digital library to do so in terms of categories, then we must operationally approximate categories on the basis of available syntactic information. We believe that the heuristic approximation of categories is technically feasible and can be supplemented with explicit relationship information when it is available. Measuring how well a particular approximation technique works will require user testing.

2.3 Operational definitions and a fundamental category-level operation

We now more formally define our user interaction model. The following operational definitions are necessary for approximating conceptual categories and for defining category-level operations.

- A *structured designator* denotes an information object (e.g., a document) in terms of a set of *attribute/value* pairs.[1] Example: bibliography entry.
- A *source* is a set of *structured designators.* Example: library card catalog.
- A *query* is a specification of desired *structured designator attribute/value* pairs. Example: ‘find author Jenkins’.
- A *filter function* takes as input a *query* and a *source* set. It returns a set of *structured designators* such that each member both satisfies the *query* and is a member of a *source* in the given *source* set (i.e., it applies a *query* to a set of *sources*).
- A *search service* performs *filtering* by applying a user-given *query* to a fixed set of *sources*. Example: WebCrawler.

1. We use *structured designator* as a more neutral term than *citation*, which is usually associated with traditional library catalogs. For example, the information returned by WebCrawler is not typically referred to as a set of citations.

- An *operational category* is a set of *structured designators* that approximates a conceptual category. Example: a list of citations, each of which records an author name of Chris Jenkins. This list approximates works by a unique person Chris Jenkins.

- A *categorization criterion* describes the conditions under which two designators belong to the same conceptual category. Example: same author.

- A *category set constraint* limits the set of allowable *operational categories*. Example: allow no more than 5 *operational categories*.

- A *categorization function* takes as input a *categorization criterion*, an initial *operational category* set, a *category set constraint*, a set of *attributes*, and the output of a *filter function*. It returns a new set of *operational categories* that satisfies the *category set constraint* and that approximates the conceptual categories a user would have determined on the basis of the *categorization criterion*.[1]

- A *category representation function* takes as input a set of *operational categories* and generates a canonical representation for each *operational category.*

We use the above entities and functions to define the fundamental operation for our interaction model.

- ***The Fundamental Category-Level Operation (FCO)***

```
Represent(Categorize(categorization criterion CC,
                     initial category set I,
                     category set constraint CSC,
                     attribute set A,
                     Filter(query Q,source set S)))
```

Note that the fundamental operation is made up of three sub-operations: filtering, categorizing, and representing. In Section 3.1, we will show how iterative text-based clustering and text-based relevance feedback can be expressed in terms of this basic category-level operation. In Section 3.2, we will see how we can use the *FCO* as the building block for iterative browsing operations of the type described in Section 2.1.

1. Note that an interface is likely to employ multiple categorization functions.

2.4 Categories in a heterogeneous browsing environment

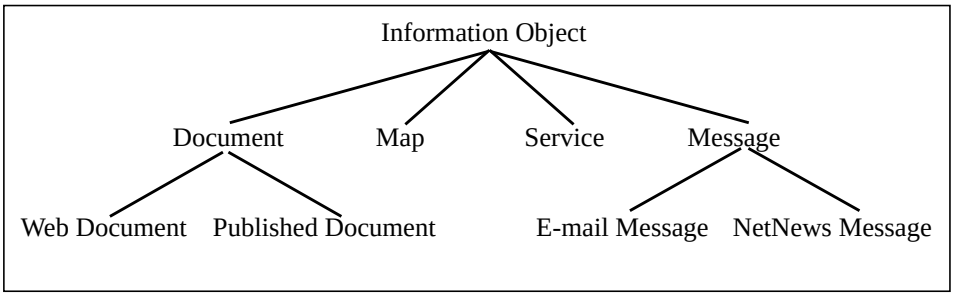
Before discussing applications of the fundamental category-level operation, we digress to consider how the design of category-based interfaces is affected by introducing heterogeneity into the source set. In a heterogeneous environment, the categorization criteria that are most relevant to the user may vary depending on which sources are being queried. Even if the categorization criteria stay stable, approximation is difficult because the sets of available structured designator attributes (as well as conventions for encoding attribute values) vary from source to source. We illustrate this problem with a scenario in which a user gives the same query both to the Dialog search service (targeting its Computer Database, which contains journal and magazine articles from the domains of computers, telecommunications, and electronics) and to the WebCrawler (a World Wide Web search service). Even assuming that we normalize attribute names,[1] we must still accept that certain attributes are appropriate for Web documents but not for magazine articles, and vice versa. For example, Web documents, but not magazine articles, have URLs (which are useful for approximating location-based categories). Furthermore, some attribute values are not readily available even when the attribute is appropriate. Intuitively, we feel that both Web documents and magazine articles should have an “author” attribute, but it is rare that we are able to determine automatically the author of a Web document.

In essence, category approximation over heterogeneous source results can only work if we can translate native source attribute-value pairs into a canonical set of attribute-value pairs. This problem is the inverse of the attribute translation component of query transla-

1. Although WebCrawler does not actually supply an attribute name for its title values, it is clear from the context that they are titles.

tion (in which queries written in a canonical front-end language are translated into queries that are native to individual sources). Standard “forward” attribute translation approaches include designing a single universal attribute set ([8], [9]) and designing a hierarchy of attribute sets ([6]) in which each hierarchy node corresponds to a different information object type. We present a simple information object type hierarchy in Figure 1.

FIGURE 1. Information Object Type Hierarchy



For category-based interfaces operating in a heterogeneous environment, the hierarchy approach is preferable because it allows categorization criteria and categorization approximation strategies to vary appropriately during the course of the interaction. At any point in the interaction, the categorization criteria and set of attributes available for category approximation will be those associated with the lowest common ancestor of the nodes corresponding to the sources in use.

3.0 Applications of the Fundamental Category-Level Operation

3.1 Iterative text-based clustering and text-based relevance feedback

User interaction models designed to support iterative text-based clustering and text-based relevance feedback turn out to be special cases of our category-based user interaction model. More specifically, we can express the basic operations in these other models in terms of our model's fundamental category-level operation (*FCO*). In both cases, iterative categorization by 'similar content' plays an important role, and special techniques are used for approximating these content-based categories.

We first examine the model underlying the Scatter/Gather text-based clustering interface ([1], [2]), designed at Xerox PARC along with new efficient clustering algorithms. Initially, a particular text-based corpus is "scattered" into clusters (each represented to the user by a cluster "summary") according to statistical measures of full-text similarity. Subsequently, the user repeatedly "gathers" together clusters of interest and submits them to the system for re-clustering. Below, we express this interaction in terms of the *FCO*.

- ***Scatter/Gather Mapping***

```
Initial Step:
Represent(Categorize(CC="similar content",
                     I={},
                     CSC=nil,
                     A={"full text"},
                     Filter(Q=nil,S={text-based corpus})))
Iterative Step:
Represent(Categorize(CC="similar content",
                     I={},
                     CSC=nil,
                     A={"full text"},
                     User-Filter(Q=nil,S=previous step output)))
```

In the initial step, the system performs categorization by similar content on a text-based corpus. In the iterative steps, the category (cluster) set returned to the user becomes the new source set. Filtering is user-directed: the user must select the categories (clusters) of interest. The output of user filtering is then re-categorized by similar content.

Expressing relevance feedback operations in terms of the *FCO* is only slightly more complex. We describe here what happens in one particular relevance feedback interface, as the mapping does not vary much from one interface to another. In the interface we analyze, the user initially issues a query to a text-based corpus and receives back an uncategorized list of results (designators) matching that query. The user then examines these results, makes relevance assessments, and accordingly assigns each result to one of three different categories: *good*, *bad*, and *neutral*. Using statistical measures of full-text similarity, the system in turn presents the user with a new set of results in which the results are like those previously assigned to the *good* category, but not like those assigned to the *bad* category. In terms of the *FCO*, the interaction is made up of an initial operation followed by an iteratively applied sequence of two operations. We describe these operations below.

- ***Relevance Feedback Mapping***

```
Initial Step:
Represent(Categorize(CC=nil,
                     I={},
                     CSC=nil,
                     A={},
                     Filter(Q=query,S={text-based corpus})))
Iterative Sequence Step 1:
Represent(User-Categorize(
                     CC="relevance",
                     I={},
                     CSC=limit category set to {good,bad,neutral},
                     A={"full text"},
                     Filter(Q=nil,S=output of either initial step
                     or iterative sequence step 2)))
Iterative Sequence Step 2:
Represent(Categorize(CC="similar content",
```

```
                    I=output of iterative sequence step 1,
                    CSC=limit category set to {good},
                    A={"full text"},
                    Filter(Q=nil,S={text-based corpus})))
```

In the initial step, the system returns an uncategorized set of results that satisfies the user's query. In the first step of the iterative sequence, the user categorizes the returned results by relevance using the three categories *good*, *bad*, and *neutral*. These categories form the *initial category set* for the second step of the iterative sequence, in which the system builds upon the supplied categories by looking for results in the text-based corpus with similar content. In this step, the generated category set is limited to the *good* category. The user takes these *good* results, re-categorizes them by relevance, and the cycle begins again.

Thus, we have seen that the *FCO* can indeed serve as a common building block for these standard IR operations. The value in this type of unification lies in its implication that we can develop an interface that not only incorporates both iterative text-based clustering and text-based relevance feedback but also has a consistent underlying conceptual model. In the next section, we will generalize from the *FCO* sequences used in the Scatter/Gather model and the relevance feedback model to generic iterative category-generating and category-assigning *FCO* sequences.

3.2 New iterative browsing operations

In Section 2.1, we gave an overview of the types of category-level operations that are valuable to users. In this section, we define and discuss three generic *FCO* sequences which encode these operations. Used iteratively, these operations can form the basis for new, useful styles of browsing interfaces.

- ***Dynamic Category Generation***

```
Initial Step:
Represent(Categorize(CC=categorization criterion,
                     I={},
                     CSC=nil,
                     A=attribute set,
                     Filter(Q=query,S=source set)))
Iterative Step:
Represent(Categorize(CC=new categorization criterion,
                     I={},
                     CSC=nil,
                     A=new attribute set,
                     User-Filter(Q=nil,S=previous step output)))
```

- ***Free Category-Directed Search Expansion***

```
Initial Step:
Represent(Categorize(CC=categorization criterion,
                     I={},
                     CSC=nil,
                     A=attribute set,
                     Filter(Q=query,S=source set)))
Iterative Step:
Represent(Categorize(CC=same categorization criterion,
                     I=previous step output,
                     CSC=limit category set to I,
                     A=same attribute set,
                     Filter(Q=nil,S=new source set)))
```

- ***Fixed Query Category-Directed Search Expansion***

```
Initial Step:
Represent(Categorize(CC=categorization criterion,
                     I={},
                     CSC=nil,
                     A=attribute set,
                     Filter(Q=query,S=source set)))
Iterative Step:
Represent(Categorize(CC=same categorization criterion,
                     I=previous step output,
                     CSC=limit category set to I,
                     A=same attribute set,
                     Filter(Q=same query,S=new source set)))
```

Dynamic category generation can be seen as a generalization of the operations used in Scatter/Gather. The user repeatedly filters categorizations of a result set and chooses new criteria for categorizing the output of this filtering action. *Dynamic category generation* enables users to cut down on the number of entities requiring scanning since the number of categories is almost always smaller than the number of results described by the catego-

ries. More importantly, it allows users to “make sense” out of their results by viewing them in terms of conceptual categories. For example, a user might first use *dynamic category generation* to view the results of a query for technical papers on ODA (Open Document Architecture) in terms of unique author. By applying *dynamic category generation* again to a selected subset of the author categories, the user can consider a different conceptual dimension, perhaps author institution. In this example, the user would learn that much of the work on ODA has taken place in Europe — a fact which might not have emerged otherwise.

Free category-directed search expansion can be seen as a generalization of the operations used in relevance feedback, with the exception that it does not include a user categorization step.[1] After seeing a categorization, a user asks that more results be found that can be incorporated into the already generated set of categories. What is novel about this approach in its generic form is that it gives users the possibility of switching the specification that new results must satisfy without explicitly breaking away from the interaction to launch an entirely new query. We illustrate this operation by returning to the example used in Section 2.1, in which a user who initially issues a query asking for technical papers on a particular subject may decide, after seeing a categorization of the results by author, that he really is interested in finding more works by some of those authors without losing the results he has already found. With *free category-directed search expansion*, the user can simply specify a new source set and request that the author categories already discovered be used as the initial category set.

1. The extra categorization step can be modeled as an interruption of *free category-directed search expansion* for a single application of user-directed *dynamic category generation*.

Finally, *fixed query category-directed search expansion* is a variation on *free category-directed search expansion*. The difference is that we require the results incorporated into the already generated category set to continue to match the query specification. This lets a user who is satisfied with a particular query and categorization incrementally add results from new sources. In essence, this approach offers a user a natural way of expanding her search results without needing to drop down to the level of individual results. This operation is especially useful in cases where certain search services are either slow or expensive. In these situations, a user may opt to perform *dynamic category generation* until a useful and narrow set of categories for that query has been determined. At that point, additional results matching the query and fitting into the specified initial category set can be obtained from the more costly search services and incorporated through *fixed query category-directed search expansion*.

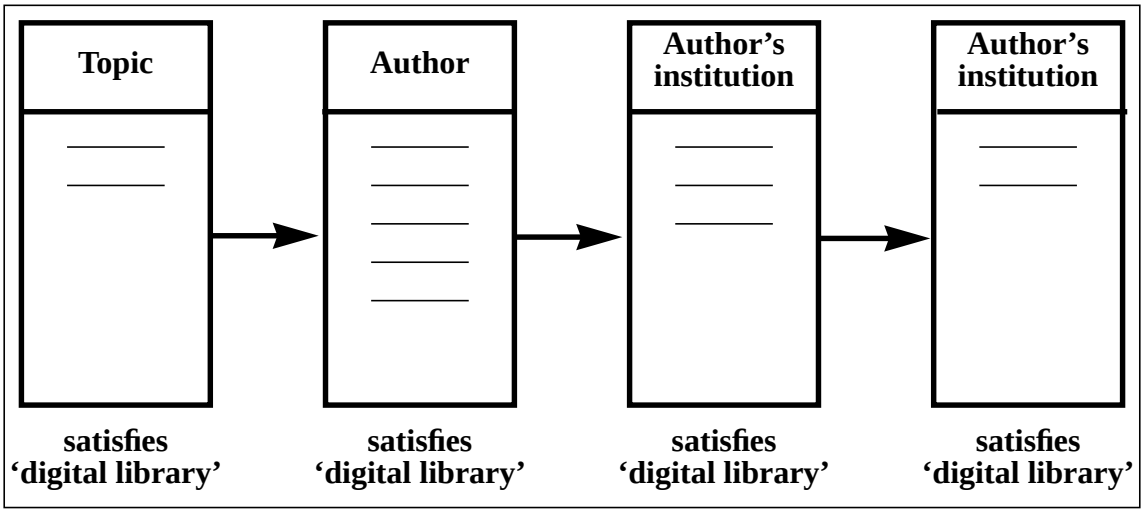
Other iteratively applied *FCO* sequences may also be useful for interfaces built around the category-based user interaction model (for example, it may be useful to have a variant of *fixed query category-directed search expansion* that can generate new categories in addition to assigning new results to old categories). For now, however, we turn to a scenario that shows how the iterative sequences we have already defined could be integrated and used in a browsing interface.

4.0 Scenario

We present here a scenario that shows how the three operations defined and discussed in Section 3.2 might be used interdependently in a unified category-based browsing inter-

face. In this scenario, our user's initial goal is to learn about digital libraries. She begins with the simple phrase query 'digital libraries' and a set of search services that covers sources of technical literature citations. She then performs a sequence of *dynamic category generation* operations, the results of which are depicted in Figure 2 (in which each line denotes a category). The first *dynamic category generation* operation has the categorization criterion of topic (which is approximated by the system using Library of Congress subject headings). She keeps all of these results and uses *dynamic category generation* to categorize by author, then keeps all the results and uses *dynamic category generation* again to categorize by author's institution. At this point, she decides to focus only on the categories corresponding to two institutions, namely Stanford and Illinois.

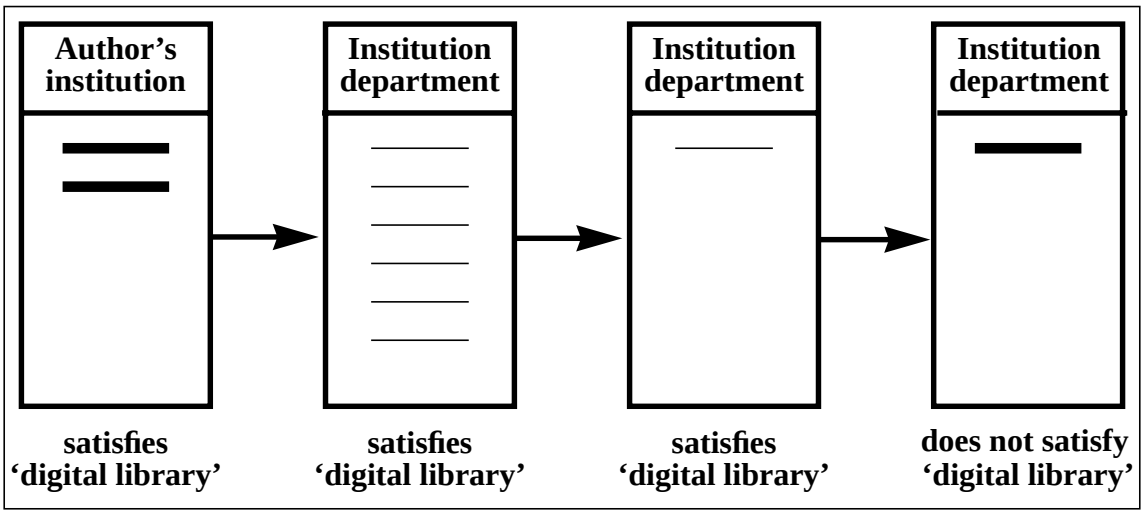
FIGURE 2. Scenario Actions



With this new focus, the user performs another sequence of actions, this time including *dynamic category generation*, *fixed query category-directed search expansion*, and *free category-search expansion*. The results of these actions are depicted in Figure 3 (in which

thin lines denote categories and thick lines denote expanded categories). In particular, our user begins by taking the institution categories she has selected and performing *fixed query category-directed search expansion* in order to add more results to them — this time by including some Web-based search services. She then uses *dynamic category generation* again, this time categorizing by department within an institution. After perusing the resulting categories, she decides that the category corresponding to works on digital libraries done in the CS department at Stanford is particularly interesting. In fact, she becomes curious about what other research has been done in Stanford's CS department. She accordingly performs a final *free category-directed search expansion* on the Stanford CS department category, and requests an uncategorized view of the results. She finishes the session by requesting the full text associated with three of the result designators.

FIGURE 3. Scenario Actions Continued



Thus, in the course of a browsing session, our user was fluidly able to change the original dimension of interest from a particular subject to a department at an institution, as well as to find a set of useful documents in an opportunistic, interactive fashion (the hallmarks of

browsing, as described in [4]). We believe that this scenario illustrates the value of adopting a consistent category-based user interaction model as the foundation for browsing interfaces.

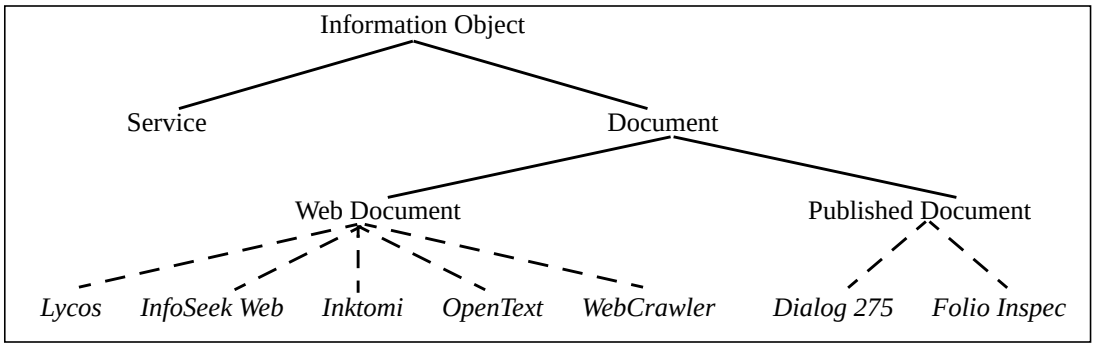
5.0 SenseMaker: A Prototype Browsing Interface

SenseMaker[1] is a prototype category-based system which adds category-level operations to a browsing interface. It is a working service that enables users to perform interactive high-level analysis of results from heterogeneous search services, including five WWW search services (WebCrawler, Lycos, Inktomi, OpenText, and InfoSeek), Stanford's Folio Inspec service (technical articles), and Dialog (restricted to the Computer Database, File 275). It accesses these services through the Stanford interoperability protocol ([7]).

At present, SenseMaker incorporates *dynamic category generation*, but neither form of *category-directed search expansion*. The categorization criteria and attributes available for category approximation vary according to the search services in use. As described in Section 2.4, we use an information object type hierarchy to accomplish this. We show in Figure 4 the prototype information object type hierarchy used by SenseMaker, augmented to show where the search services available fit.

1. Russell et al.[10] bring to the fore the general notion of "sensemaking," which they define as the "process of searching for a representation and encoding data in that representation to answer task-specific questions." Here, we use the term more narrowly to suggest the process by which people come to have a high-level understanding of search results.

FIGURE 4. SenseMaker Information Object Type Hierarchy



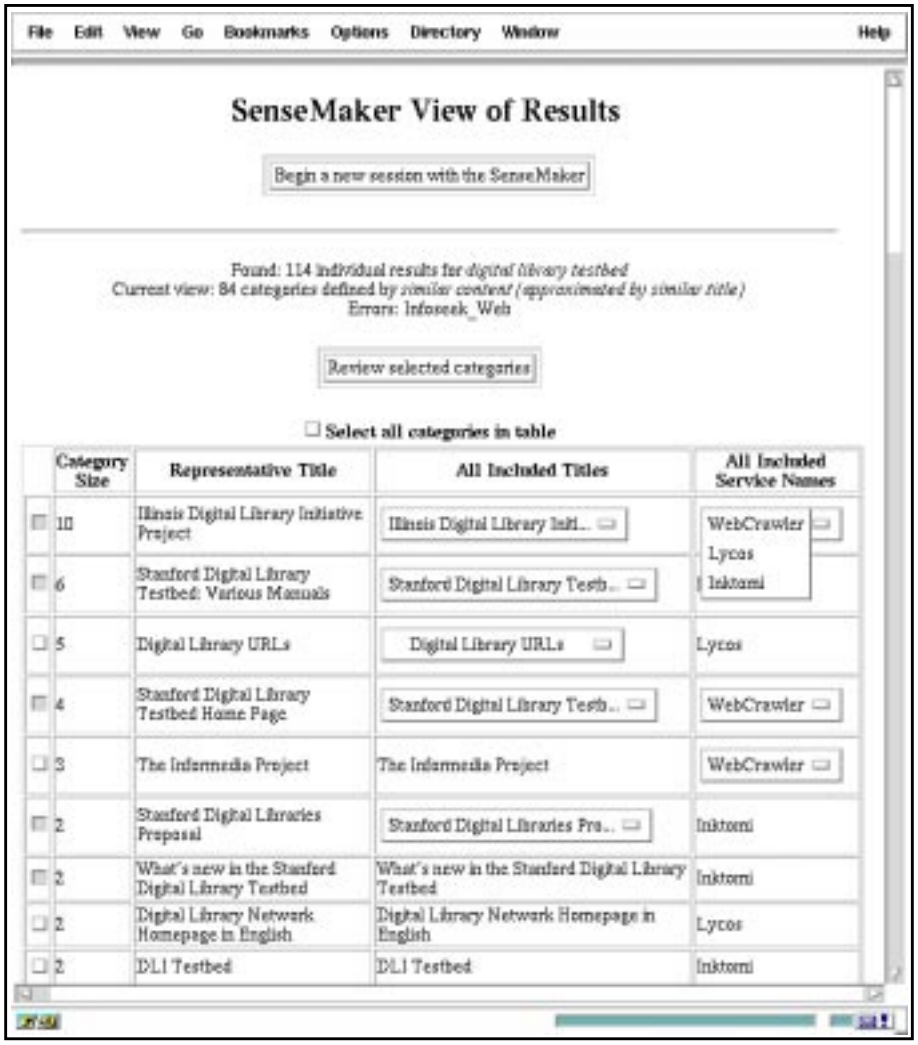
We note that the category approximation techniques based upon these type-dependent attribute sets are currently quite simplistic in SenseMaker. Indeed, one of the advantages of this approach is that as better heuristics for approximation are defined, they can be substituted for the prototype versions.

To give the reader a feel for what using SenseMaker is like, we conclude by describing a real interaction with the system.

- The user issues the query ‘digital library testbed’ to all of the search services available through SenseMaker.
- SenseMaker determines that the appropriate categorization criteria are those corresponding to the ‘Document’ node of Figure 4, and presents the choices to the user.
- The user requests ‘similar content’ categorization (very simplistically approximated using titles).
- SenseMaker presents the output of this *dynamic category generation* action.
- The user decides on several categories for further perusal. Figure 5 shows what the WWW interface looks like at this point.
- SenseMaker re-negotiates and determines that the appropriate categorization criteria are now those corresponding to the ‘Web Document’ node of Figure 4. The choices are presented to the user.
- The user asks for ‘same location’ categorization (approximated using URLs).
- SenseMaker presents the output of this *dynamic category generation* action.

- The user focuses on the most promising looking category descriptions and requests to see an uncategorized view of their included designators.
- The user directs the WWW browser to go to look at the text for several of these uncategorized designators.

FIGURE 5. SenseMaker WWW Interface



6.0 Conclusions and Future Work

We have developed a category-based user interaction model for browsing. We believe that users will find this model powerful because it enables them to browse using distinctions

commonly made in our shared literary tradition and to fluidly change focus from one of these distinctions to another. Interfaces designed according to this model can unify several standard IR user operations as well as new types of iterative browsing actions. We have focused on three category-level operations as the foundation for these interfaces: *dynamic category generation*, *free category-directed search expansion*, and *fixed query category-directed search expansion.* We have illustrated what such an interface might be like through the discussion of a scenario and the presentation of SenseMaker, an implemented prototype system which incorporates *dynamic category generation*.

In the future, we plan to extend SenseMaker to include the capability for both *free* and *fixed query category-directed search expansion*. We will develop more sophisticated category approximation techniques in order to make the interface less subject to breakdown. Also at the interface level, we will more carefully examine the issues involved in representing categories to users. In addition, we will address the question of how to make the information object type hierarchy more apparent to users as they navigate through the interface. At the model level, we will explore the potential value of other category-level operation sequences. Finally, we plan to do some user testing to gain design insights and to generate feedback on the model itself.

7.0 References

1. Douglas R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In *SIGIR '92.*
2. Douglas R. Cutting, David R. Karger, and Jan O. Pedersen. Constant Interaction-Time Scatter/Gather Browsing of Very Large Document Collections. In *SIGIR '93.*
3. George Lakoff. *Women, Fire, and Dangerous Things.* The University of Chicago Press, Chicago, 1987.
4. Gary Marchionini. Information Seeking in Electronic Environments. Cambridge University Press, Cambridge, England, 1995.
5. Bonnie Nardi and Vicki O'Day. Intelligent Agents: What We Learned at the Library. Draft, 1995.
6. Andreas Paepcke. An Object-Oriented View Onto Public, Heterogeneous Text Databases. In *Proceedings of the Ninth International Conference on Data Engineering,* 1993.
7. Andreas Paepcke, Steve B. Cousins, Hector Garcia-Molina, Scott W. Hassan, Steven K. Ketchpel, Martin Röscheisen, and Terry Winograd. Towards Interoperability in Digital Libraries Overview and Selected Highlights of the Stanford Digital Library Project. To appear in *IEEE Computer.*
8. Ramana Rao, Daniel M. Russell, and Jock D. Mackinlay. System Components for Embedded Information Retrieval from Multiple Disparate Information Sources. In *UIST '93,* pages 23-33.
9. Ramana Rao, Bill Janssen, and Anand Rajaraman. GAIA Technical Overview. Xerox PARC technical report, 1994.
10. Daniel M. Russell, Mark J. Stefik, Peter Pirolli, Stuart K. Card. The Cost Structure of Sensemaking. In *INTERCHI '93,* pages 269-276.
11. Peter Willett. Recent Trends in Hierarchic Document Clustering: A Critical Review. In *Information Processing & Management,* Volume 24, Number 5, 1988, pages 577-597.