SUBMITTED TO DIGITAL LIBRARIES '96

Techniques and Tools for Making Sense out of Heterogeneous Search Service Results

Michelle Q Wang Baldonado and Terry Winograd
Department of Computer Science
Stanford University
Stanford, CA 94305-2140
{michelle, winograd}@cs.stanford.edu

Abstract

We describe a set of techniques that allows users to interact with results at a higher level than the citation level, even when those results come from a variety of heterogeneous on-line search services. We believe that interactive result analysis allows users to "make sense" out of the potentially many results that may match the constraints they have supplied to the search services. The inspiration for this approach comes from reference librarians, who do not respond to patrons' questions with lists of citations, but rather give high-level answers that are tailored to the patrons' needs. We outline here the details of the methods we employ in order to meet our goal of allowing for dynamic, user-directed abstraction over result sets, as well as the prototype tool (SenseMaker) we have built based upon these techniques. We also take a brief look at the more general theory that underlies the tool, and hypothesize that it is applicable to flexible duplicate detection as well.

1.0 Introduction

Imagine a reference librarian who responded to all queries from patrons by simply uttering one potentially relevant citation after another. Interacting with such a librarian would undoubtedly be a frustrating experience. In fact, it is hard to envision this scenario ever really taking place. Why not? The first and most obvious reason is that people are good at understanding what constitutes a reasonable answer to a question. In general, people have an intuitive understanding for when a high-level answer should be given in place of an abundance of low-level answers. The second reason, which may not be apparent to the naive patron, is that reference librarians have extensive training and experience in the art of the reference interview. Upon encountering someone with a broadly-stated information need, a good reference librarian will engage the patron in a dialogue designed to discover more precisely the patron's information needs.

Why analyze a scenario that sounds so patently absurd at first reading? Re-envision this scenario, if you will, with an on-line search service standing in for the librarian. Rather than sounding far-fetched, this scenario now describes the fashion in which most users interact with such search services today. Very few on-line search services respond to a query by presenting an abstraction over the matching citations — regardless of how many matching citations there may be. Furthermore, services that do provide users with abstracted views of result sets usually operate according to fixed rules and do not allow for much user control or interaction with the abstracted view. An example of a search service that does incorporate a limited degree of abstraction into its interaction model is Stanford University's Folio interface to its holdings database (Socrates)[1]. When users query this service for works by a particular author, they are presented with a list of authority records rather than individual citations matching the user's constraint. Selecting an authority

1. See also [4] for details of a library system that does abstraction based on Library of Congress classification numbers, subject headings and title keywords derived from MARC records.

record then reveals to the user the citations associated with that authority record. While the displaying of authority records for author-based searches often proves useful, it is still the case in this system that users have little flexibility in determining what kind of abstraction will be presented to them, and they have limited possibilities for interaction with the presented abstraction.

In reflecting further upon our scenario, we can make another observation about the differences between a reference librarian's presentation of results and a search service's presentation of results (especially a search service that gathers its information from many heterogeneous sources). In general, a good librarian refrains from pointing a patron to sources that the patron would consider indistinguishable. For example, if a reference librarian were asked by a high school student for a good play by Shakespeare, the librarian would probably not direct the student both to the First Folio Edition of *Twelfth Night* and to the Second Folio Edition of *Twelfth Night*. It is even less likely that the librarian would direct the student to otherwise-identical hardback and paperback versions of *Twelfth Night*. The point here is that librarians are very good at judging whether or not two citations refer to the same work in the eyes of the patron, i.e. they are good duplicate detectors. Current search services, on the other hand, do not take into account the spectrum that duplicate citation detection can span. Many search services do not address the question of duplicate citation detection at all, while those search services that do tend to have static, hard-wired definitions for what conditions must be satisfied in order for two citations to be considered identical.

We present in this paper techniques and tools that can be used to address the two shortcomings of search services that we have described in this introduction: namely, their inability to abstract dynamically over search results in accordance with a user's desires, and their inability to perform flexible duplicate detection. In fact, we will claim that the same techniques are applicable to both problems, though we will mainly concentrate here on the problem of abstraction. Furthermore, we argue that the reasons why these techniques work well for the tasks at hand can be found in our general theoretical model of searches for information artifacts.

We certainly do not mean to imply that our techniques make the experience of interacting with an automatic search service as rich as that of interacting with a well-trained reference librarian. What we do claim is that we can learn some very good lessons by observing how librarians and patrons interact — and that these lessons can inform our design of tools to improve interaction with automatic services. We also note here that the tool we describe in Section 3.0 is in fact designed with both sophisticated searchers and casual end-users in mind.

The rest of this paper is devoted to exploring how we can add interactive result analysis and abstraction to the Digital Library. We believe that the inclusion of these capabilities will make it easier for users to "make sense" out of the results that search services return to them. In Section 2.0 of this paper, we outline our approach to the abstraction problem. We have already taken this approach in our development of a prototype tool for the Stanford University Digital Library testbed, and we will describe its implementation and interface in Section 3.0 of the paper. In Section 4.0, we take a closer look at the theoretical model of information artifact search that underlies this work. In Section 5.0, we look at the problem of duplicate detection as a kindred problem to the abstraction problem, and point out how our implementation will be extended to address duplicate detection as well. Finally, Section 6.0 offers a summary of our current approach and identifies the areas in which we will be undertaking further research.

2.0 Abstracting over Result Sets from Heterogeneous Sources

As we have observed, librarians are good at giving out high-level answers to patrons who ask a question for which there are many possible relevant citations. Furthermore, librarians may try to figure out which type of high-level answer is most suited to the patron by engaging in a reference interview. How can we incorporate some level of flexible, customized abstraction into the tools we create for existing on-line search services?

2.1 How can we approximate abstraction?

We begin by looking first at how we might perform simple abstraction over result sets at all. Currently, many on-line search services respond to queries with lists of citations, where citations are represented by attribute-value pairs (e.g., WebCrawler, a World Wide Web spider, responds to queries by providing values for rank, title, and URL attributes). The availability of these attribute-value pairs suggests a good method for operationalizing and approximating the process of performing abstraction over the citations. We can group together citations on the basis of their attribute values, and then describe the result set in terms of the citation groups rather than in terms of the citations themselves. For example, we might group together all citations that have the same title.

The proposed approach (which could be augmented by other abstraction techniques as they become available) borrows both from SQL grouping facilities [1] and from statistical clustering techniques [12]. In SQL (and many other database languages), users can request that database rows (a database row is essentially a list of attribute-value pairs) be grouped together if they have identical values for a particular attribute (or attributes). Statistical clustering techniques, on the other hand, typically use more complex measures of similarity to cluster elements together. In addition, it is possible to have techniques that allow for an element to be assigned to more than one cluster (if certain conditions are satisfied).

2.2 How can we make abstraction flexible and customizable?

Now that we have seen how abstraction can be performed, we turn to looking at how a user might be able to control what kind of abstraction is performed. We have observed that in an SQL environment, grouping is performed with respect to a user's specification of attribute names. For SQL grouping, this is all that must be specified since grouping is performed only when rows have identical values for a particular attribute — no input is expected about similarity metrics. In contrast, statistical clustering algorithms can usually be parameterized in several ways (e.g., what is the similarity metric, what is the similarity threshold for clustering). We feel that expecting users to make low-level decisions about clustering algorithm details is not appropriate in the Digital Library. However, we do believe that it is important to grant a user control over what kind of abstraction is performed. The user must have control because he/she is interacting with an automatic service that lacks common sense. We cannot expect an automatic service to consistently determine the most appropriate abstraction for the situation, nor can we expect it to conduct a sophisticated reference interview.[1]

We remedy these two conflicting desires — the wish to avoid burdening the user with low-level decisions and the wish to grant users the power to determine how abstraction is performed — by introducing an intermediate conceptual level to our design. The distinctions made at this intermediate level are high-level, but they are mapped "under the hood" to a particular set of parameters given to our abstraction module. The choice of what distinctions are available should be dependent on good design principles and on studies of what distinctions are valuable to users. In Section 3.0, we will detail the choices that we have made for our prototype tool.

In addition to giving users control over what kind of abstraction should be performed, we would also like to give users the ability to interact with the created abstractions. Important types of interaction include the ability to refine the automatic groupings that are presented (e.g. by collapsing, exploding, or editing the groupings), the ability to focus on particular groups, and the ability to request recursive abstraction for a focus group. The latter two possibilities for interaction have already been demonstrated in the work at Xerox PARC on Scatter-Gather ([1], [2]). Scatter-Gather, used in conjunction with full-text searches, operates by clustering together texts that its algorithm judges to be similar, then allowing the user to "gather" together some of those clusters for recursive clustering. Since the Scatter-Gather algorithm was developed with homogeneous full-text sources in mind, it does not address the question of how to handle interaction in the face of heterogeneous sources with multiple clustering possibilities over multiple attributes.

2.3 How can we make abstraction work in a heterogeneous environment?

The strategies we have outlined in the previous two sections work straightforwardly in the case where we are accessing only one search service or a set of homogeneous search services (services that use the same set of attributes and the same set of conventions about how attribute values should be encoded and queried). However, when we look at how these ideas apply in a heterogeneous environment, the situation rapidly becomes much more complex. We illustrate the problem by looking at a scenario in which we would like to query both the Dialog Computer Database (containing journal and magazine articles from the domains of

1. At the same time, we do expect that services will be able to make good "guesses" about what kind of abstraction is appropriate — and hence that users should always be supplied with a default abstraction choice.

computers, telecommunications, and electronics) and the WebCrawler index of World Wide Web documents. Even assuming that we normalize attribute names (in fact, WebCrawler does not actually give an attribute name for its title values, even though it is clear that they are titles), we are still faced with the fact that certain attributes make sense for World Wide Web documents but not for magazine articles, and vice versa. For example, World Wide Web documents, but not magazine articles, have URLs. Further complicating the matter is that some attribute values are not readily available even when the attribute does make sense. We intuitively feel that both World Wide 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 World Wide Web document. There are not yet standard conventions for how World Wide Web document authors should be declared, whereas we do have conventions in the traditional publishing world.

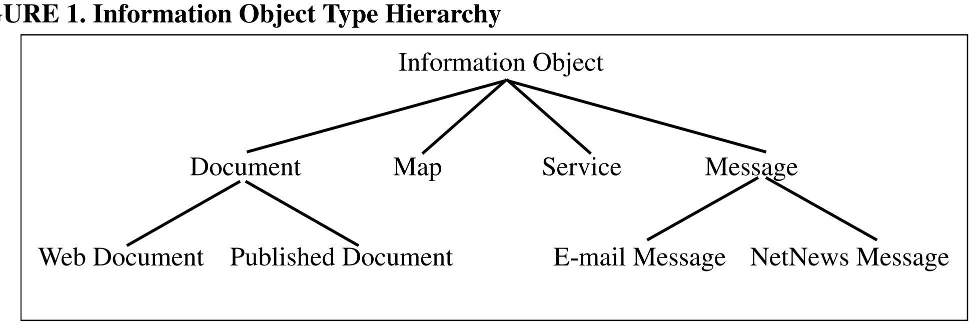
In essence, abstraction over the results from individual heterogeneous sources can only work if we have a canonical set of attribute-value pairs to which we can appeal. In order to obtain this canonical set, we must have methods for translating from the individual source attribute-value pairs to the canonical versions. This problem can be viewed as the inverse of the attribute translation component of query translation (in which queries written in a canonical front-end language are translated into queries that are native to individual heterogeneous sources). Two "forward" attribute translation approaches that have influenced our solution to the inverse attribute translation problem are those taken by the Xerox PARC GAIA effort ([8], [9]) and by Paepcke [7]. In GAIA, users may either make reference to universal attributes that are fixed for the front-end query language, or they may make reference to attributes that are native to a particular source. The query translator is responsible for translating universal attributes to their native equivalents. In contrast, the approach described by Paepcke moves away from the dichotomy of universal and specific attribute sets. Paepcke models sources as types, organizes the types into an inheritance hierarchy, and introduces a typed query language for the front-end language. This approach allows users to make reference to attributes that are neither universal nor native, but rather are common to a group of related sources.

As we pointed out earlier, abstraction over results from heterogeneous sources requires a shared attribute set. In the case where every query is sent to a fixed collection of source services, a well-designed universal attribute set (as in GAIA) is appropriate for abstraction purposes. If, however, the user is granted control over which search services should be consulted for a given query, then this solution has some important drawbacks. For example, a source collection might include WebCrawler, Lycos (another WWW spider), and the Dialog Computer Database. URL-based abstraction makes sense if the user has selected only Web-based services, but is not as useful if the Dialog Computer Database has also been selected.

Since one of our design goals is to increase the degree of possible user customization, the approach we adopt is more similar to that described by Paepcke. We introduce the concept of a type hierarchy into the design, where each node in the hierarchy corresponds to a particular information object type (e.g. World Wide Web document). Furthermore, each node has associated with it at least one canonical set of attributes. For each individual search service, we identify it with at least one particular node in the tree, and define mappings for the service that can translate its native attributes and attribute values into the canonical attribute descriptions associated with its corresponding node (as well as into the canonical attribute descriptions associated with the ancestors of that node). Adding a new source (or information object type) is easy to do with this design.

A simple version of an information object type hierarchy is presented in Figure 1.

FIGURE 1. Information Object Type Hierarchy



With such an information object type hierarchy in place, we can adopt a more dynamic solution to the problem of determining the appropriate attribute set for abstraction. After the user selects the search services of interest, the corresponding nodes on the tree can be located. The attribute set used for abstraction then becomes the canonical set that corresponds to the lowest common ancestor of the computed type nodes. If the user later focuses on a subset of the groups described in the abstracted view, the process can be repeated so as to allow for the most relevant attribute set to be in use at any point during the interaction. For example, we might have a situation in which a query is initially sent to both WebCrawler and the Dialog Computer Database, and only the "title" attribute is available for abstraction. If the user later happens to focus on groups of citations that come only from WebCrawler, recursive abstraction could be based upon both the "title" and "URL" attributes.

3.0 SenseMaker: A Tool for Interacting with and Abstracting over Result Sets

The ideas that we have presented in Section 2.0 have formed the conceptual basis for the development of SenseMaker[1], a prototype version of an interactive result analysis tool. Currently, the SenseMaker mediates between the user and five different sources — three World Wide Web search services (WebCrawler, Lycos, and InfoSeek), Stanford's Folio Inspec service (technical articles), and the Dialog Computer Database (File 275, computer magazines). In Section 3.1, we describe aspects of SenseMaker's design and implementation. In Section 3.2, we walk through an example of interaction with the SenseMaker.

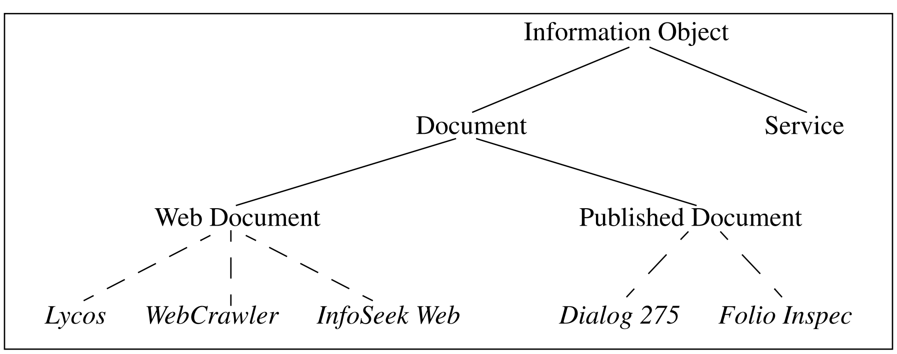
3.1 Design and Implementation of the Prototype

In this section, we describe the choices made in defining the prototype's information object type hierarchy, the intermediate conceptual abstraction distinctions, and the available styles of interaction.

3.1.1 Information Object Type Hierarchy

We show in Figure 2 the SenseMaker information object type hierarchy, augmented to reveal where the particular services available in the prototype fit in. SenseMaker uses this hierarchy at each new stage in the interaction to determine on the fly which set of attributes is currently most appropriate for cross-service citation comparison. For example, if the user has selected Lycos and Dialog 275 as targets for a query, but citations matching the query are found only in Lycos, then SenseMaker will correspondingly adjust the "best fit" in the information object type hierarchy from 'Document' to 'Web Document'.

FIGURE 2. Information Object Type Hierarchy with Sources



1. Russell et al.[10] bring to the fore the general notion of "sensemaking," which is defined 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.

3.1.2 Intermediate Conceptual Level Distinctions

In Section 2.2, we described the need for an intermediate conceptual level that maps a user's specification of how abstraction should be performed onto the actual heuristic that is used for doing grouping and clustering. The distinctions made at this intermediate conceptual level are those that best match a user's view and understanding of the search service results, rather than the lower level distinctions that are necessary to do abstraction approximation. We have designed the intermediate conceptual level for the SenseMaker with these principles in mind. The following table shows the abstraction choices available to the user for the 'Document', 'Web Document', and 'Published Document' nodes of the information object type hierarchy. We note here that the details of the heuristics to which these distinctions are mapped are not the focus — indeed, one of the advantages of this approach is that as better heuristics are defined, they can be substituted for the current prototype versions of the heuristics.

TABLE 1. Intermediate Conceptual Level Distinctions

Document	Web Document	Published Document
Bundle together documents with identical titles. Bundle together documents with similar titles. Bundle together documents with identical authors. Bundle together documents discovered by the same source. Bundle together documents discovered by sources of the same type.	Bundle together Web documents from the same last n domains. Bundle together Web documents that form a collection at a site. Bundle together Web documents with identical titles. Bundle together Web documents with similar titles. Bundle together Web documents discovered by the same source.	Bundle together published documents with identical titles. Bundle together published documents with similar titles. Bundle together published documents with identical authors. Bundle together published documents discovered by the same source.

3.1.3 Styles of Interaction

The visualization and interaction components of the current SenseMaker interface have been influenced to a great extent by the decision to make the prototype version of SenseMaker available on the World Wide Web to our community of testbed users. In fact, several aspects of the interface are treated differently in the more sophisticated interface under development as part of the work in progress by Steve Cousins of Stanford University on designing a task-based interface to the Digital Library.

In the World Wide Web interface to the SenseMaker, it is possible for users to control what kind of abstraction will be performed and to do recursive abstraction. There is not yet support for dynamically editing abstracted views.

Interaction takes place through three different kinds of Web pages — one for users to articulate their queries, one for users to choose the type of abstraction to be performed, and one for SenseMaker to present the abstracted view and for users to focus in on the elements of interest. More specifically, the query specification page allows users to supply a set of query keywords[1] and to select which of the available search services should be queried. The abstraction customization page portrays to the users those intermediate conceptual level distinctions that are appropriate for the current interaction stage (computed from the services selected for an initial query, and from the services represented in the focus set for a recursive query). Finally, the pre-

1. Although the current prototype allows only for keyword-based queries, we will soon incorporate the more complex query language being developed by Kevin Chang of Stanford University.

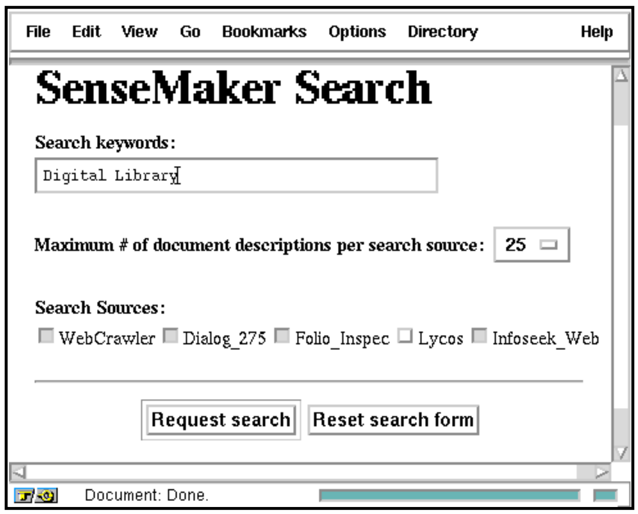
sentation/selection page produces a table-based visualization of the generated abstract result view (enabling users to form a focus set by selecting individual rows).

3.2 Interaction Example

In order to give the reader a better feel for what it is like to use SenseMaker as a tool, we give here a detailed illustration of a real interaction with the prototype system.

Initially, the user is asked to provide a set of search keywords, a specification for the maximum number of hits that should be returned from each service, and an indication of which search services are desired for the current query. Each of these values is specified through a Web forms interface. In this example, the user enters the search keywords 'Digital Library', a maximum number of hits per service of 25, and requests that WebCrawler, InfoSeek Web, Dialog 275, and Folio Inspec all be queried (see Figure 3).

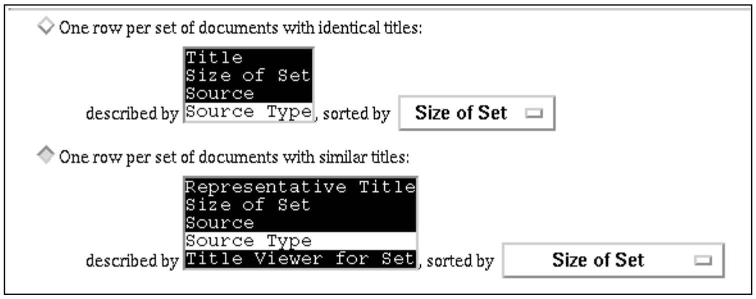
FIGURE 3. SenseMaker Query Specification



At this point, negotiation takes place behind the scenes to determine the relevant set of attributes for cross-service result comparison. The lowest common ancestor of the nodes associated with the requested search services is 'Document'. Therefore, the abstraction choices made available to the user are those described in the 'Document' column of Table 1.

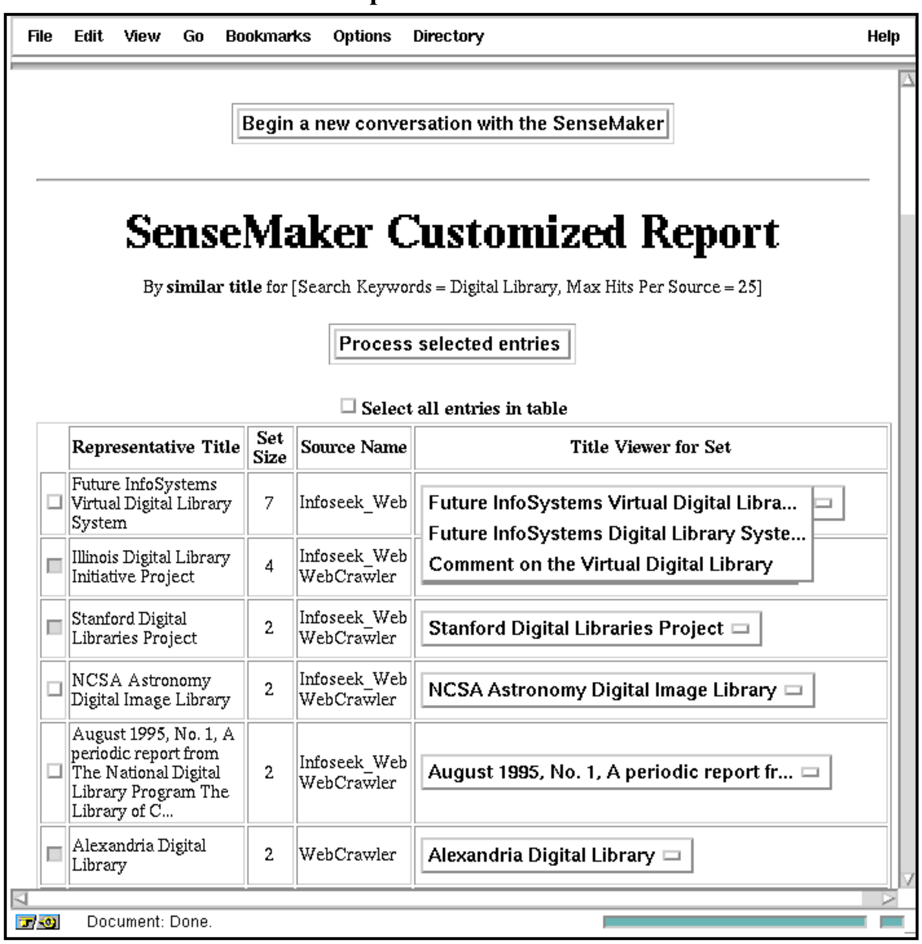
In the Web interface, these choices are explained using the familiar language of report generation. SenseMaker explains to users that a "customized report on the search source results" will be generated on the basis of the selections made on this page. Each row in this "report" corresponds to an individual abstraction group, and thus a decision about what makes up a row is directly analogous to a high-level decision about how abstraction should be performed. Each of these row formation (grouping) choices is presented to the user as a radio button entry on the page. In addition, users have the ability to control which attributes should be used to describe the computed abstraction groups (i.e., what are the columns in the "report"). Users make these column (representative attribute) decisions by choosing options from the scrolled list of relevant display attributes that appears underneath each row formation option. Users may select one or more of these attributes, and may also specify that the rows be sorted by a particular attribute. Figure 4 is an excerpt from this customization page (showing two of the row formation options with their corresponding column controls). For row, column, and sorting choices, default values are always supplied. In our example, the user overrides the default row formation choice ("one row per individual document" — i.e., no abstraction) and decides to ask for results with similar titles to be grouped together. However, the user accepts the default choices for the "report" columns.

FIGURE 4. Excerpt of SenseMaker Abstraction Choices



Finally, Figure 5 shows the "customized report" that is generated in response to the user's query specification and abstraction choices. As expected, each row corresponds to a group of results, while each column corresponds to a particular attribute for that group. The left-hand column of the table contains checkboxes, which can be used to select individual rows. In our example, the columns give a representative title for each group of results, the number of results included in each group, the names of the sources from which the group members originated, and a "title viewer" for the group. The "title viewer" is simply an option menu. When the user positions the mouse on the menu button and clicks, the full list of the distinct titles for the members of the group appears. Figure 5 shows what this expansion looks like for the first row.

FIGURE 5. SenseMaker Customized Report



We can see in Figure 5 that the user has decided to focus on the Illinois, Stanford, and Alexandria rows. Once this selection has been made, negotiation can take place again to determine the new set of relevant attributes. Since the results described by the selected rows include only sources of type 'Web Document', the set of abstraction options will now correspond to those presented in the 'Web Document' column of Table 1. Although we do not show it here, we can imagine that our user might now ask for the grouping together of descriptions that have the same final two domain name components in their URL (e.g., the URLs http://www-pcd.stanford.edu and http://www-diglib.stanford.edu would satisfy this criterion). Finally, our user would be able to focus on just the most promising looking descriptions and ask to see them without any kind of abstraction. Since these are all Web documents, the URL will be displayed and the user can direct the Web browser to go to look at the text.

4.0 Underlying Theoretical Model of Information Artifact Search

The main thrust of our theoretical analysis of information artifact search is that it is important to distinguish among *designators* and *conceptual information items.* What do we mean by these terms? A *conceptual information item* is the true object of an information artifact search — it is the intellectual content that the user is seeking. We carefully distinguish this *conceptual information item* from its physical (or electronic) representations. In essence, a *conceptual information item* has an identity that is external to any given description or representation for it. Although this point is subtle, it is crucial to our theory. A *designator,* on the other hand, has a much more clear-cut definition. It is simply a description that denotes, upon interpretation, one or more *conceptual information items.* The title *Twelfth Night* is an example of a *designator.* We illustrate this distinction between *designators* and *conceptual information items* by imagining two people, one of whom read an English version of *Twelfth Night*, and the other of whom read a French version of *Twelfth Night*. Clearly, it would be possible for these two people to have a conversation about *Twelfth Night* even though they did not read the same physical book, and in fact did not even read the exact same words. They could also have this conversation in the case where one of them had read a later edition than the other, where one had read a differently formatted version than the other (e.g., an HTML version found on the Web vs. a reproduction of an original printing), and so on. When they refer to *Twelfth Night* in this hypothetical conversation, they are actually referring to the *conceptual information item* that is *Twelfth Night* rather than to any particular manifestation of it.

We observe that *designators* are very different from *conceptual information items* in that they are syntactic objects, which can be easily manipulated. The mechanisms used for abstraction by SenseMaker, for example, all rely on analyzing properties of *designators.* However, we believe that the reasons why users find our abstraction approximation to be useful are that: 1) different *designators* for the same *conceptual information item* are likely to have properties in common; 2) *designators* for related c*onceptual information items* are also likely to have properties in common; 3) *designators* for unrelated *conceptual information items* are less likely to have many properties in common. Furthermore, we can see that automatic duplicate detection is typically based on nothing more than comparing *designators,* and thus we can envision flexible duplicate detection being carried out using similar techniques to those used for abstraction.

5.0 Detecting Duplicate Results from Heterogeneous Sources

In Section 1.0, we discussed the fact that librarians (and people in general) are naturally good duplicate detectors. In the language of Section 4.0, they are good at determining when two *designators* denote the same *conceptual information item.* Furthermore, we believe that given an arbitrary two *designators,* it is not possible to make a **universal** decision about whether or not they denote the same *conceptual information item* — the decision depends on the people involved, the task at hand, and the overall context. Levy [5] gives an insightful description of this point of view, and we gave some illustrative examples involving editions in Section 1.0.

We have also already hinted that it is possible to approximate duplicate detection in the same fashion that we approximated abstraction — by comparing the attributes and attribute values described by result citations. Furthermore, this approach allows us to have the flexibility that we need in performing duplicate detection that is user-dependent and context-dependent.

How then should we proceed to build a tool that allows for flexible duplicate detection? At the lowest level, it is apparent that we can rely on heuristics that make equivalence decisions based on comparing attribute values. For example, we might consider two citations to be equivalent if they have the same values for their "title" and "author" attributes[1]. However, we still maintain that it is inappropriate to ask users to choose from among these heuristics (especially for duplicate detection, where we assume that the choice will not be changed on a frequent basis — resulting in limited user familiarity with the heuristics). Thus, we once again need to develop an intermediate conceptual level at which broader distinctions can be expressed. These are clearly not the same as those needed for abstraction, in large part because users prefer abstracted views to be based on a small number of attributes (if not just one), while duplicate detection is more likely to be based on multiple attributes. Furthermore, choice of duplicate detection criteria differs in that users generally want to be able to specify the criteria in terms of the narrowest information object type possible, rather than in terms of the lowest common ancestor of all the chosen sources.

In addition, duplicate detection must be handled differently than abstraction in the "characterization" and "visualization" stages of interaction. For a view containing abstraction groups, recursive interaction often results in one of the original abstraction groups being broken up into its components. In contrast, for a view containing duplicate groups, we will need to formulate stricter constraints governing when an original duplicate group may be broken apart (since a duplicate group usually forms a single entity in a user's conceptual model).

Thus, we see that even though duplicate detection can be handled in the same manner as abstraction at the lowest levels, it necessitates that we make a number of changes at the user interaction level. We have not yet implemented these changes, but we plan to explore their requirements more fully as part of our future work.

6.0 Conclusions and Future Work

Tools that allow users to obtain overviews of their results (through abstraction as well as through merging together citations that are duplicates in the eyes of the user) will become increasingly necessary as the number of heterogeneous sources included under the "Digital Library" umbrella grows ever larger. Providing users with high-level descriptions of the results that match their constraints will allow them to zero in more quickly upon the most useful dynamically-determined categories of results.

With this belief as motivation, we have described the development of a new style of interactive result analysis, made technically possible by manipulating the *designators* returned by on-line search services. We have outlined our prototype tool, SenseMaker, which we designed in order to experiment with this new style of interaction. Finally, we have also given a brief articulation of the distinction between *designators* and *conceptual information items* upon which our approach depends. We hypothesize that our *designator*-based framework will extend to handle the question of flexible duplicate detection as well.

Our plans for future work on this subject span several areas, some of which have been touched upon already during the course of this paper. The areas that we will focus upon in our current research include:

1. Incorporating flexible duplicate detection into our prototype tool.
2. Exploring issues that arise in the development of the "inverse" attribute translation component.
   a. Addressing the question of how to ensure that "inverse" attribute translation has good scaling properties.
   b. Articulating and modularizing the required sub-components of "inverse" attribute translation (e.g., encoding translation, convention translation, etc.).
3. Refining and extending our theoretical analysis of *designators* and *conceptual information items* (including looking to see how these ideas are applicable to the realm of universal identifiers).

1. Another very useful and popular way of judging whether or not two results are identical is to compare the full ASCII text associated with them in order to determine the degree of textual overlap. Further details of a sophisticated instance of this approach can be found in [11].

4. Exploring in a more principled fashion the techniques required for generating representative designators for groups formed by the abstraction process.
5. Refining the set of distinctions made at the intermediate conceptual level.
6. Incorporating more complex query languages into our prototype tool, which is currently keyword-based (work by Kevin Chang at Stanford).
7. Developing a more sophisticated, task-oriented user interface for our tool (work by Steve Cousins at Stanford).
8. Investigating how to incorporate abstraction ranking and query refinement into our model.

Acknowledgments

We are indebted to Scott Hassan of Stanford University both for building the stateful WWW server on which the SenseMaker interface depends and for developing the protocol and interfaces necessary for uniform communication with the heterogeneous search services. We also thank Steve Cousins, Hector Garcia-Molina, Scott Hassan, Frankie James, Andreas Paepcke, Martin Röscheisen, and Trace Wax (all of Stanford University) for valuable discussions about both theoretical and design issues related to this work.

This material is based upon work supported by the National Science Foundation under Cooperative Agreement IRI-9411306. Funding for this cooperative agreement is also provided by ARPA, NASA, and the industrial partners of the Stanford Digital Libraries Project. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the other sponsors.

The prototype was developed on machines supplied by the Digital Equipment corporation, the Hewlett Packard corporation, and the IBM corporation.

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*, pages 318-329.
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*, pages 126-134.
3. George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press, Chicago, 1987.
4. Ray R. Larson. Managing Information Overload in Online Catalog Subject Searching. In *Managing Information and Technology, Proceedings of the ASIS Annual Meeting*, 1989, pages 129-135.
5. David M. Levy. What Do You See and What Do You Get? Document Identity and Electronic Media. In *Screening Words: User Interfaces for Text, Proceedings of the Eighth Annual Conference of the UV Centre for the New OED and Text Research*, Waterloo, Canada, 1992, pages 109-117.
6. Jim Melton and Alan R. Simon. *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann Publishers, San Mateo, California, 1993.
7. Andreas Paepcke. An Object-Oriented View Onto Public, Heterogeneous Text Databases. In *Proceedings of the Ninth International Conference on Data Engineering*, 1993, pages 484-493.
8. Ramana Rao, Bill Janssen, and Anand Rajaraman. GAIA Technical Overview. Xerox PARC technical report, 1994.
9. 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.
10. Daniel M. Russell, Mark J. Stefik, Peter Pirolli, and Stuart K. Card. The Cost Structure of Sensemaking. In *INTERCHI '93*, pages 269-276.

11. Narayanan Shivakumar and Hector Garcia-Molina. SCAM: A Copy Detection Mechanism for Digital Documents. In *Digital Libraries '95*, pages 155-163.

12. Peter Willett. Recent Trends in Hierarchic Document Clustering: A Critical Review. In *Information Processing & Management*, Volume 24, Number 5, 1988, pages 577-597.