MetaScholar Initiative

General Libraries
540 Asbury Circle
Emory University
Atlanta, GA 30322

Phone 404 727 2204
Fax 404 727 0827

MetaCombine Project Interim Report

Mid-point project report on experiments at Emory University in combined searching, automated organization of OAI and Web resources, and new forms of scholarly communication

September 2004

Table of Contents

Executive Summary and Highlights

The MetaCombine Project, now at its mid-point, has extensive findings to report to the Andrew W. Mellon Foundation. The project has undertaken a series of experiments in new approaches to organizing information in digital libraries. Moreover, it has continued our ongoing work with an expanding group of Southern Studies scholars who are interested in utilizing the results of this project and previous efforts of Emory University's MetaScholar Initiative to create scholarly communications tools. Emory University wishes to thank the Mellon Foundation for its generous support of this important research; and is pleased to report that the results of this project already have many practical applications for digital libraries. The project website at http://www.metacombine.org now includes many example systems demonstrating the technologies developed to date. The following are highlights of the project findings:

- **Semantic Clustering:** The project has found that there are many effective techniques for semantically clustering both metadata aggregated through the Open Archives Protocol for Metadata Harvesting (OAI-PMH) and web pages aggregated through web crawls. The project team has been able to show through extensive usability tests that these techniques provide workable and scalable mechanisms for automating the organization and enhancement of harvested information for access purposes. These techniques compare favorably with existing means of organizing and harvesting metadata, and greatly complement such techniques.

- **Categories of Clustering Techniques:** An important basic finding is that such semantic clustering techniques fall into two broad, complementary categories: 1) clustering uncataloged information to identify underlying patterns in order to create new classification schemes, and 2) classifying (or understood another way, assigning metadata to) uncataloged information once one or more classification schemes have been selected or derived. We term the first variety *pre-classificatory* clustering (or simply *clustering*) and the second *post-classificatory* clustering (or *classification*) in this report.

- **Combined Searching:** The project has developed several effective systems for combined searching of harvested metadata and web pages. Integrated with the semantic clustering techniques, these systems will enable the creation of a new search and discovery service unlike anything else currently in operation: a service that organizes and searches subject domain specific information simultaneously from the separate realms of the visible and dark web.

- **Focused Crawling:** A basic finding related to the creation of such a combined function service is that it must include a technology called *focused crawling*, which provides the capability of selectively crawling only those websites relevant to the subject domain under consideration. Without focused crawling, the quality of web content aggregated for scholarly studies is greatly compromised. The project team has been successful in deploying prototype focused crawling systems as a component of the overall system architecture developed.

- **Visualization Tools:** The project has successfully produced a range of visualization tools and techniques that graphically display the results of semantic clustering systems. These tools are being vetted to focus groups of scholars. Early results have shown that scholars see value in the visualization techniques, and are also able to quickly suggest new scenarios for using the tools to conceptualize bodies of information.

- ***Southern Spaces*:** The MetaCombine project has built on strong collaborative relationships with scholars developed in the previous MetaScholar projects to create an Advisory Board with several new members. This group has not only advised the project concerning usability of the various systems devised, but has also acted as an Editorial Board for a new scholarly communication vehicle that they have entitled *Southern Spaces* (http://www.southernspaces.org). This internet journal and scholarly forum has been successfully launched and is now in the process of soliciting submissions from leading scholars in the field.

Section 1: Overview of Findings To Date

[Section author: Martin Halbert.]

With support provided by the Andrew W. Mellon Foundation, Emory University began work on the MetaCombine Project in 2003. The goals of this project were to develop, implement, and test a series of experimental approaches to organizing information in digital libraries, and to continue working with a select group of scholars to use the findings of this project to create new scholarly communications systems.

The project has now reached its mid-point and has completed a number of experiments with many findings. A project website has been established at http://www.metacombine.org to provide demonstrations of the experimental systems produced to date, which will contain timely updates as the work continues. In cases where the nature or results of tested experimental systems may be unclear in the narrative, we recommend a visit to the live demonstration site to get a sense of how the relevant technology operates.

This interim report includes many detailed sub-reports about the experimental work conducted in the last year. The nature of this work was in most cases extremely technical, and the reports include a significant amount of rigorous scientific narrative about the experiments. While every effort has been made to ensure that the individual sub-reports are clear and concise, we realized that a less technical narrative was also needed to summarize the main results and their implications for future digital library projects in a straightforward way. Section 1 of this report was prepared with this purpose in mind.

Emory University greatly appreciates the Mellon Foundation's ongoing support of this important research, and looks forward to demonstrating this year the variety of practical applications for digital libraries and scholarly communication systems that have been enabled by the research completed in the first half of this MetaCombine project.

1.1 Semantic Clustering

Automated systems for analyzing and organizing textual information fundamentally equate words to numbers and then apply various mathematical operations on the resulting data, typically in the form of various kinds of factor analysis of large matrices. There are many terms for such systems; in this project the phrase *semantic clustering* has been used as an umbrella term for all such systems, as they fundamentally sort information into clusters.

The project has found that there are many effective techniques for semantically clustering both metadata aggregated through the Open Archives Protocol for Metadata Harvesting (OAI-PMH) as well as web pages aggregated through web crawls. The project team has been able to show through extensive usability tests that these techniques provide workable and scalable mechanisms for automating the organization and enhancement of harvested information for access purposes. These techniques compare favorably with existing means of organizing harvesting metadata, and complement such techniques greatly.

The underlying problem addressed by semantic clustering technologies is the lack of consistent organization encountered in metadata and web pages harvested from disparate repositories of scholarly information. While most sites have some internally consistent organization, these internal organizational schemes are typically not consistent across institutional or project boundaries. When contents from many separate repositories are harvested and then re-assembled out of context in new search and discovery systems, they lack any overall organizational scheme for browsing and understanding relationships between individual records. Finding experimental ways to address this fundamental problem is the aim of the semantic clustering portion of this project.

The findings of the project to date are very promising, both because effective clustering systems were developed and because we have gained a much more sophisticated understanding of the capabilities and variety of clustering systems. This section of the report will summarize the basic types of clustering techniques studied, scenarios for their application, and next steps needed to deploy these experimental systems in operational systems.

1.1.1 Pre-Classificatory versus Post-Classificatory Clustering

An important basic finding is that techniques for semantic clustering fall into two broad, complementary categories:

1) **Clustering Techniques**, which create new organizational schemes for uncataloged information by identifying underlying patterns that may usefully serve as the basis for new classification schemes. This kind of clustering is **pre-classificatory** in that it occurs prior to the existence of a classification scheme for the information under study, and in fact is usually intended to create a new classification scheme for unorganized information. For convenience in our discussions, we have taken to referring to any organizational effort using this approach as **clustering** (although what we now term classifying is also technically a form of semantic clustering). There are many situations in which new classification schemes are needed for ad hoc assemblages of information; these are discussed in the section on scenarios below.
2) **Classification Techniques,** which sort uncataloged information into one or more pre-existing classification schemes selected for use in organizing the information. This kind of clustering is **post-classificatory** in that it occurs after the creation of a classification scheme for the information under study. In addition to newly derived classification schemes, there are obviously a large number of longstanding classification systems such as the Library of Congress Classification scheme that can be used to organize information for browsing purposes. This kind of activity can alternatively be understood as assigning new metadata to information items. Again for simplicity in discussion, we call any effort to automatically sort information into a classification scheme **classifying**.

The distinction between these two varieties of semantic clustering was not clearly understood when the project was proposed. Very different computational techniques are needed for these two different activities. The support vector machine (SVM) approach emphasized in the project proposal is indeed a useful technique, but it is for classifying information, as opposed to clustering or other relevant functions. For more detailed definitions of these and other technical terms, see the glossary in Appendix B of this report. The following is a brief list of the key mathematical techniques that we have used to date for various semantic clustering purposes:

<u>Clustering</u> techniques used:

- **NMF (Non-negative Matrix Factorization):** This is the main technique we have used for pre-classificatory clustering of unorganized information. This technique was reported in the research literature relatively recently (1998), and has been shown to be superior to all other comparable techniques in the research literature.

- **PCA (Principle Components Analysis):** This is the classic technique for pre-classificatory clustering information. For discussion purposes it can be considered essentially synonymous with all of the following frequently mentioned techniques for semantic clustering: latent semantic indexing (LSI), multi-dimensional scaling (MDS), and singular value decomposition (SVD). NMF has been shown theoretically to be a more effective technique than PCA for semantic clustering, a finding we can confirm through practical experiments. We nevertheless did use PCA in the course of the experiments, often as a reality check for NMF analysis.

Classification techniques used:

- **SVM (Support Vector Machines):** This is the main technique we used for classifying information using selected or derived classification schemes. SVM is generally considered the best tool for this purpose by practitioners in the DL research field today.

- **Bayesian Classification:** An example of a more traditional technique for classification, which we also used for comparison with SVM systems.

Different scenarios for using semantic clustering techniques became easier for us to understand after this clarification of the basic distinction between pre- and post- classificatory systems and how they function. Several scenarios for using semantic clustering have been explored so far in the project.

1.1.2 Importance of Labels

An unexpected finding of the clustering work was the importance of human-readable labels. Although clustering technologies like NMF are very effective for generating new ontologies for ad hoc assemblages of information, no clustering system is able to automatically create elegant labels for the resulting headings (i.e. artificial intelligence is still not here). The software can approximate a heading by identifying the most statistically significant keywords and word roots in the cluster, and we found that these approximations work effectively for rough-and-ready headings in various display schemes (especially dynamically generated visualizations as described below). But it was clear from our usability studies and focus groups that classification headings that were written by human beings for comprehensibility by other human beings were far better than anything automatically generated.

This suggests a logical workflow for the future, namely that we select a particularly good ontology generated by the clustering software and then devote the required time by project subject experts to assigning labels to the clusters. This is not a particularly time-consuming or onerous task (in fact, those who have so far done it found the exercise intriguing), but also is not something that we want to do for hundreds of separate ontologies.

1.1.3 Semantic Clustering Scenarios

There are a range of scenarios in which various semantic clustering technologies can usefully organize information aggregated from many sources:

- **Clustering Scenarios:** The biggest problem the MetaScholar projects have encountered with assembling ad hoc subject domain harvests is that the resulting virtual collections lack any overall organizational scheme. Using NMF techniques, we have found that we can address this problem by deriving new ontologies (classification schemes) for any assemblage of information, no matter how heterogeneous. The technology is not perfect, but it works well enough to immediately put an unorganized mass of information into functional and comprehensible order. To put it bluntly, this is a tremendously powerful new capability.

- **Classification Scenarios:** Although we can generate new classification schemes, we also want to be able to use existing schemes that particular groups of users are familiar with, such as the headings from the *Encyclopedia of Southern Culture* (ESC). The main requirement for using SVM or other classifiers is that a training set be available. Using a combination of the full text of the *ESC* plus direct feedback from scholars, we have developed such a training set that can be used both for classification purposes and for focused crawling. Now that we have done this for one classification scheme, we anticipate identifying subsequent existing schemes (the Library of Congress Classification is a prime example) and developing additional training sets for them. We intend to provide access to aggregated information through ESC, LCC, and other ontologies/controlled vocabularies.

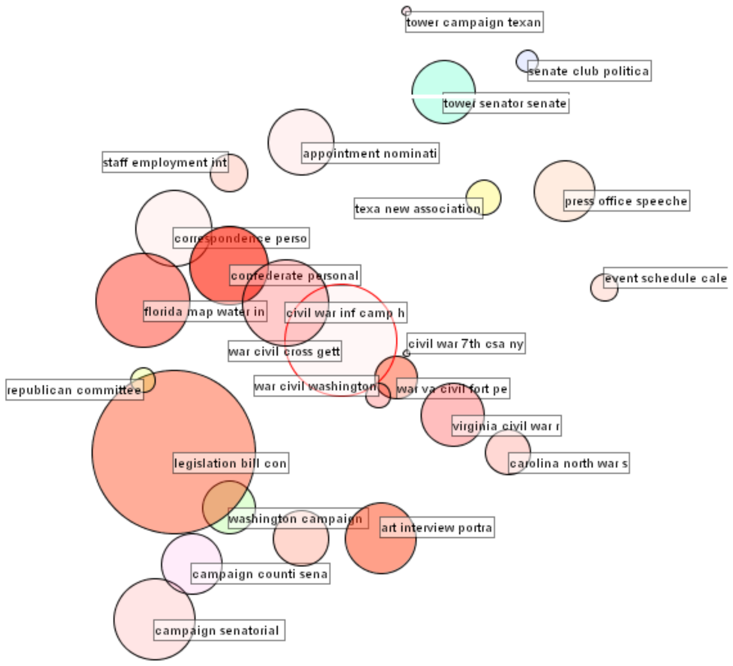
- **Hierarchical Scenarios:** The question of hierarchy in ontologies arose during the experiments. Clearly, a hierarchy of organization is needed once more than a few hundred information items have been aggregated. However, we have not yet identified guidelines for *how many* levels of hierarchy need be present for any given number of items. Generating hierarchical ontologies is straightforward with both the clustering and classification systems we have developed and we tested several; the question is mainly one of deciding how many levels of hierarchy are appropriate for a given aggregation of metadata and web pages. We plan on conducting more experiments in this area in the coming year, as well as exploring the information science literature to see if there has been theoretical work done in this area.

- **Hybrid Scenarios:** We found that hybrid combinations of the above scenarios could be a useful approach to organizing information, for example, combining the ESC headings with clustering. There are only two dozen ESC subject headings, and the resulting blocks of records become too large after a few thousand items are added to the collection being classified with the ESC headings, requiring at least one more hierarchical layer of organization. If each ESC heading area is then clustered (for example with the NMF technique) to identify sub-clusters, the resulting hybrid organization becomes much more browsable.

The findings of the project experiments suggest that all of the various semantic clustering strategies we tried are valuable and in fact complementary in that each addresses a particular kind of organizational access need by users. We therefore envision deploying a mixture of semantic clustering techniques to organize aggregated information in our future work. Concretely, this would provide multiple controlled vocabulary browsing schemes for the information we aggregate in new search and discovery systems, finally providing the kind of comprehensive ability to browse related subject headings across collections that we have been seeking for some time.

1.1.4 Cluster Visualization

The project has successfully developed a range of visualization tools and techniques to graphically display the results of semantic clustering systems. These tools are being vetted to focus groups of scholars. Early results have shown that scholars see value in the visualization techniques, and are also able to quickly suggest new scenarios for using the tools to conceptualize bodies of information.



Example of Radial Cluster Visualization for AmericanSouth Metadata

The visualization software developed in the project can generate a variety of graphical displays of the clusters, as well as dynamically linking the graphical display regions to lists of the metadata records. We have hosted a series of focus groups to discuss which displays and graphical conventions (color, orientation, etc.) are the most useful for users.

Some of the most interesting suggestions from focus group participants to date concerned the utility of radial visualizations (an example is shown in the figure above), in which the user could select a cluster to center the entire display around in order to show the relative relationships of all the information in the system to a particular subject. In this display, one of the civil war clusters was selected, resulting in a cluster of related topics around it. The visualization displays were found to be an excellent potential mechanism for gaining a quick overview of the collections aggregated, and a remarkable means of studying the relationships between the collections in a variety of ways.

We intend to further explore the scenarios for using the visualization displays in the coming year. We want to analyze what the most effective strategies are for immediately connecting the displays to the intuitive guesses that people make about what a graphical display like this tells them about information relationships.

1.2 Combined OAI / Web Searching

The project has developed several effective systems for combined searching of harvested metadata and web pages. Coupled with the semantic clustering techniques, these systems enable the possibility of a new search and discovery service unlike anything else currently in operation. This service would organize and search subject domain specific information simultaneously from the separate realms of the visible and dark web.

1.2.1 Focused versus Generic Web-Crawling

A basic finding related to the creation of such a combined function service is that it must inherently include a technology called focused crawling, which provides the capability of selectively crawling only websites relevant to the subject domain under consideration. The quality of aggregated web content for scholarly studies is otherwise greatly compromised. The project team has been successful in deploying prototype focused crawling systems as a component of the overall system architecture.

Focused crawling essentially works by giving a web spider the capability to assess whether links it follows are relevant to a subject domain or not. Without this ability, the spider follows links indiscriminately, recording all sorts of off topic web pages. For example, when test crawls were performed early in the project with unrefined generic spider software such as Swish-E, crawls of targeted university digital archive sites quickly followed links to other parts of the campus web sites resulting in an irrelevant mass of web pages.

Focused crawls on the other hand provide the capability for fine-tuning spiders to different levels of subject sensitivity and inclusiveness, and produce quite good aggregations of on-topic web pages. An unexpected benefit of the focused crawling technology is that it identifies additional websites related to the subject domain to consider for inclusion. This is a powerful technology for canvassing the vast number of possible web sites that might be relevant to include in a combined search and discovery system.

1.2.2 Search Engine Capabilities

The project has been able to perform combined searches of OAI metadata and web information through several different approaches in the course of the B1 and B2 series of project experiments. We now want to refine our experiments to select a preferred core of software tools for developing a new combined search engine.

Several search engines were therefore evaluated for use in the project. The capabilities of the engines were compared in terms of searching features and flexibility of integration with other tools. The main systems that we ultimately focused on were Swish-E, Lucene, Greenstone, and Essex. Each of these tools has particular strengths in either features or integration. We think there may be some utility in setting up reference implementations of each of these systems for long term comparison. We are currently inclined toward Lucene overall because of its combination of features and adaptability.

A key topic that we wish to investigate in the next year is displays of result sets, and how to most effectively organize them. The result set organizer approach developed in the previous MetaScholar projects forms our general framework, but there a number of specific usability questions that we intend to investigate.

1.3 Federated Digital Library Frameworks

This portion of research was always planned for the second half of the project, so there are no results to report as yet. Nevertheless, we wanted to share our early thoughts about this portion of the project as there has been some initial analysis of the topic as we have considered how to approach the question of frameworks for interoperation.

1.3.1 Web Services and the OCKHAM Library Network

Emory University is currently working on a complementary project concerning federated frameworks for distributed digital libraries for the NSDL. This project is called the OCKHAM Library Network, and intends to develop a web services framework for interoperation between digital library systems.

The W3C web services framework appears to be the logical technology to use for the kind of interoperability sought in the MetaCombine project. We therefore hope to develop some synergies in thinking between the NSDL work and the MetaCombine project in the coming year. Because the Fedora toolkit is a key element of the NSDL Core Integration team's thinking about interoperability, we intend to study Fedora very closely for possible connections with what we are trying to accomplish in MetaCombine and OCKHAM.

Beyond Fedora and NSDL, we also hope to establish some collaborative discussions with other projects such as OAIster, NITLE, CLIMB, Greenstone, and iVia in the next year to see if these projects would be willing to provide web services gateways to key software tools for processing and remediating the metadata that they have harvested. If they are interested in this possibility, we intend to test this approach to inter-institutional digital library information processing.

1.4 Advisory Board and Plans for New Scholarly Communication Systems

The project has continued to build on our successful collaborative relationships with scholars in Southern Studies developed in previous MetaScholar projects. We have expanded the group of scholars to form the MetaCombine Advisory Board, and have conducted several meetings of the group. Full details are provided in section 5 of this report.

In our discussions with the Advisory Board we have sought to further refine our shared understanding of what new scholarly communication systems we need to develop, scenarios for their use, and working specifications for such systems. This has led to several new developments.

1.4.1 Modularizing Portal Functions

The Advisory Board members have reiterated many of their conclusions voiced at the end of the AmericanSouth project. They assert that there is a real need for both A) a new form of online peer-reviewed scholarly communication, and B) a new search and discovery system for canvassing both OAI and web sources. They also assert that these two systems should be separately articulated and implemented. Discussions concerning both concepts have been very fruitful.

This modularization of functions previously conceived as one monolithic scholarly communication portal has proved very helpful. It allows various functional concepts to be designed independently in parallel development tracks. This allowed the group to make immediate progress on the concept of an online peer-reviewed forum while the research was still being done on the search and discovery system. The group has named the former *Southern Spaces* and the latter *Southern Searches*. Using the "Southern" prefix in these two services (as well as others in the future, potentially) enables us to link them while keeping them distinct conceptually.

1.4.2 *Southern Spaces* and *Southern Searches*

The MetaCombine Advisory Board have designed two new services in conjunction with the project team. These services address specific niches that have been identified as highly desirable by scholars in the area of Southern studies. In the case of *Southern Spaces*, the MetaCombine Advisory Board is referred to as the Editorial Board to reflect the function of the same individuals in that context.

The following are brief summaries of the most important aspects of these two new services.

<u>Southern Spaces</u>

- This is a peer-reviewed Internet journal and scholarly forum focusing on Southern culture and history research that integrally uses the digital medium to produce scholarship that cannot be published in print. The Editorial Board strongly felt that there was great need to pioneer a new model for how to conduct scholarship on the web, not by simply using the web as a dissemination medium for print publications, but by using the strengths of new media to go beyond the limitations of the print medium.

- *Southern Spaces* is founded on a new theoretical focus that the scholars involved in the MetaScholar Initiative have been developing for several years now, namely the study of how geographies (both real and imagined) affect the way we think about culture and history. This builds on what the Editorial Board sees as a very significant trend in the last few years of humanities scholarship to focus on discussions of spaces and places, and how overlapping spheres of influence impact and affect each other. They felt that the digital medium is particularly suited to analysis with this orientation, and in fact felt that this theoretical focus demanded the capabilities of the digital medium.

- The *Southern Spaces* content will be made available under an Open Access model, with no direct subscription fees. This was an important conceptual distinction because all the scholars agreed that they did not wish to see this content commodified and wished to reach as wide an audience as possible with this work. They did understand the importance of sustainability issues for the site, and explored at length models for sustainability based on a combination of institutional sponsorship and non-profit advertising, grounded in the discussions from the two previous MetaScholar projects. Officials of Emory University including Provost Earl Lewis and Dean of Graduate Studies Brian Noe have been brought into the discussion through briefings and demonstrations of *Southern Spaces*. Both the Emory libraries and the Graduate School have already contributed approximately $80K to the creation of Southern Spaces through fellowships, project assistant salaries, facilities support, and semester release time for faculty involved in the project. Emory is now considering the creation of an ongoing managing editor position as well as providing further institutional support for the project. All involved campus parties see *Southern Spaces* as a very important new endeavor that connects well with pre-existing university groups such as the Emory Graduate Institute for the Liberal Arts and the Southern Studies faculty.

- The *Southern Spaces* website (http://www.southernspaces.org) has been carefully designed and built out during the last year with a basic body of content from the Editorial Board members. The Board is now in the process of soliciting submissions from a much wider group of additional nationally recognized scholars. A publicity campaign is being planned for the coming year to popularize the site and make it known to Southern Studies programs around the country.

- We would like the site to include a significant capability for displaying information concerning Southern regions through Geographic Information System (GIS) components. We have lacked the staff expertise in digital cartography to implement this functionality so far, but would like to enhance the site in the future with this capability.

Southern Searches

- During the next year we intend to begin prototyping a new search and discovery system tentatively named *Southern Searches* that will incorporate the new technologies developed in the MetaCombine Project. We are interested in developing a name beginning with *Southern* to reflect the conceptual relationship with *Southern Spaces* while still preserving some conceptual distinction between the two sites. We do not think that we can create a production system while completing the second half of the MetaCombine Project, but we can at least get some of these features up and running. We would very much like to continue this work by establishing a *Southern Searches* production site in a subsequent project. The URL would be http://www.southernsearches.org if we use the name *Southern Searches*.

- The service would provide the capability to simultaneously search metadata harvested from OAI data providers as well as web pages aggregated through focused crawling. The site might also harvest and index several other distinct sources of information such as selected records of rare books from library catalogs (harvested through Z39.50 queries), and information from news sources harvested through RSS feeds.

- The system would organize all aggregated information with a uniform set of ontologies, both newly derived through clustering, as well as pre-existing from classification system like the LCC and the ESC. All records will be browsable using these ontologies, in the same way that bibliographic entries in a library catalog are browsable by subject headings, see-also references, etc. In addition to descriptive metadata subject headings, we anticipate equivalents for some authority controlled browsing. An example of this would be browsing by the name of the parent institution that produced the material to see other related materials.

- Visualization schemes would also be provided for graphical display and browsing of the aggregated material. These schemes would provide the ability to gain a quick understanding of the contents of the site through visual displays, as well as via browsing citations. The Advisory Board saw this as a particularly innovative and intriguing means of organizing access to the site, and a logical further exploration of the spatial metaphors that interest the group members.

- All sites indexed by the service would be reviewed by the Advisory Board to ensure that the content harvested would be comprised of high quality and trustworthy sources of information. This does not mean that every OAI record harvested or page aggregated by focused crawling would be reviewed, simply that the source sites would be examined by qualified subject experts to ensure they were appropriate for scholarly research needs.

- The system would be designed as a model that could easily be replicated to create subsequent services oriented to other subject domains. For example, Emory and several other research libraries holding Irish materials have jointly considered building a similar service for Irish studies.

The MetaCombine project team and Advisory Board are very pleased with the results of the last year of work, and we hope that officials of the Mellon Foundation will likewise be satisfied with these outcomes. Beyond their value for digital library research, we consider *Southern Spaces* and *Southern Searches* to be very important contributions to scholarship in Southern studies. We look forward to continuing our collaborative group efforts on these services.

Section 2: Semantic Clustering Experiments

This portion of the report describes the various project development efforts, experiments, and findings to date concerned with semantic clustering topics (Part A of the MetaCombine Project).

2.1 Enabling Research on Clustering and Classification

This section describes basic research that was conducted in order to enable the planned clustering experiments of the project.

2.1.1 Clustering Systems

[Section author: Aaron Krowne]

2.1.1.1 Introduction

The MetaCombine NMF clustering software forms the centerpiece of our semantic clustering efforts. The goal of all semantic clustering is to bring into proximity artifacts that have related meaning. Clustering algorithms proper (as distinct from classification and other related methods) deal with the specific scenario in which there is no a priori ontology (organizational scheme) assumed for the artifacts. Instead, the ontology is inferred from the output (i.e., the clusters).

Clustering operates on features of the artifacts being clustered. These features can be thought of as attributes, and can take on different values. Clustering uses co-occurrence of specific attribute-values and combinations of attribute-values across many artifacts to determine relatedness. The assumption is made that these similarities in features translate into similarities in meaning.

In text clustering, the artifacts are assumed to be representations of text documents (or text metadata) and the features are the words (or transformations of them). For MetaCombine, we are attempting to achieve the desired grouping of metadata records that describe objects that have similar meaning through the use of clustering.

Below we discuss the algorithm and system we employed for clustering, as well as related systems we built to work with and utilize the output of this system.

2.1.1.2 The Algorithm

We selected the Non-negative Matrix Factorization (NMF) algorithm upon which to base our clustering system. NMF is a relatively new algorithm, introduced in the literature in 1999 [Lee, 1999]. In a literature review, we found that NMF has been shown to be the best performer in the class of latent semantic indexing (LSI) methods [Xu, 2003], as applied to document clustering. NMF is an improvement on classical LSI, both in terms of accuracy and simplicity. LSI is based on a run of Singular Value Decomposition (SVD) and then a post-processing phase where clusters are built. By contrast, NMF produces the information for the output clusters directly with the first application of the core algorithm. LSI and NMF have basically the same run-time complexity, based on the number of documents times the number of features times the number of clusters.

2.1.1.3 The Clustering System

In the following subsections we will describe our NMF clustering system. In fact, only the central part of the system (the clustering program) is NMF-specific. The other components prepare the input to this portion or interpret and make use of its output. We opted for this modular architecture because it enabled us to re-use the components separately for other phases of the project. In information retrieval, a number of common tasks and transformations have to be repeated often, thus we separate out the programs that deal with each task to minimize “reinventing the wheel”.

2.1.1.4 The NMF clustering program

In this section we describe the main part of the NMF clustering system, the actual clustering program. We will discuss why we developed our own tool for this portion of the system and what its characteristics are.

Justification of work

We faced a number of constraints that convinced us that our only option was to develop our own tool:

1) The tool had to be available for our own free use.
2) The tool had to be open source, so that others could benefit from it freely, potentially with our enhancements.
3) The tool could only rely on freely-available code libraries (preferably open source also).
4) The tool had to scale for the text clustering domain. This is due to the “curse of dimensionality” problem in text processing: the size of text processing problems always includes a factor of the number of features in the corpus, which is proportional to the dictionary size. This explosion in the number of features makes many classical methods and implementations intractable.

Surprisingly, the first and second items are almost always major problems in building on published computer science work. Most people who publish do not or cannot make their code available, though one would expect such a process would be due diligence for all computer scientists (we aim to buck this trend ourselves). In our case, luckily, we were able to get a reference implementation of NMF done in matlab, from Xu et al. However, this reference implementation suffered from problems three (dependence on matlab, a commercial computation system) and four (the code relied on a standard matrix implementation). Thus, we couldn't actually use it for the core of our research, though it was useful to check our program for correctness on small test data sets.

Unable to locate other implementations (after all, the method is very new), we realized we had to produce our own. With the reference implementation and fairly detailed descriptions of the algorithm, we felt this would not be a problem. We elected to utilize C++ as the implementation language, due to our familiarity with it, its speed, and the fact that some of our text processing components were already written in C++.

This left us with the problem of high dimensionality. We observed that, in metadata, each record is actually very brief, containing only a handful of words. Yet, in the standard matrix representation, each document takes up as much space as the entire dictionary. Most of each document entry in this representation consists of 0's. Smart implementations of algorithms in this situation will use something called a *sparse matrix representation*, whereby zero entries are implicit, and nonzero entries are stored as a list. This means that each document takes up space proportional to just the number of words in the original record, which in our case, was well under 100, after preprocessing.

Desiring not to code our own sparse matrix implementation, we located an existing one for C++ called SparseLib++, produced at NIST. SparseLib++ is actually just a C++ wrapper for the de facto standard numerical analysis library Sparse-BLAS (which is natively Fortran). Accordingly, we coded our clustering system for SparseLib++. While the usage of SparseLib++ did give us a tower of dependencies (Sparse-BLAS and other intermediate layers), each component was also free and open source, thus not frustrating problem three (see above).

Functionality

Our C++ NMF clustering system, in brief, takes as input

- a sparse vector file containing the representation of the documents
- some files giving some of the basic metadata for each document (identifier, title, and description)
- a determination of the number of clusters desired

and returns as output

- a classification file, mapping cluster IDs to document identifiers
- a clustering metadata output XML files

The sparse vector file for input is of a "standard" format for our project, consisting of one document entry per line, with each line consisting of space-separated (feature, weight) pairs. In this scheme, "feature" is an integer corresponding to a feature ID, and weight is a real-valued strength of that feature in the document, from (0,1).

The classification file is extremely simple, containing lines of the form integer cluster ID, real-valued classification weight, and string-valued document identifier (based on mapping to the identifier metadata file which was input). This file enables weighted mappings of documents to clusters based on their identifiers.

The clustering metadata XML file contains much useful information about understanding the clustering run. For each cluster, it gives:

- the total number of documents in the cluster
- a list of the top ten most important words/features in the cluster, with weights
- a relative measure of the number of important features for the clustered
- a list of the top 20 documents in the cluster, with ranks (each is represented using its title, ID, and description metadata)
- optionally, it can include a centroid for each cluster, which is a single vector that mathematically estimates the center of the cluster in feature space (i.e., the "average" document in the cluster)

With all of this output, the NMF clustering system provides everything needed to form a basic high-level understanding of a clustering run, and to actually utilize the detailed clustering output.

The NMF clustering system can also be run with a range of cluster counts specified in the form of lower and upper determinations. When run in this mode, it repeatedly clusters the input, providing versions of all the output files for each value of the number of output clusters. When this is done, the user can fairly quickly review reports based on the XML output files to determine the best clustering run (the optional number of clusters, which is a challenge we discuss below).

Hierarchical Clustering

The above describes a basic one-level, or "flat" clustering situation. This is what we first wrote the system to do, as NMF by itself is just a flat clustering algorithm. That is, it separates the content into a number of mutually-exclusive sets, none of which contains the other. However, we desired to form more detailed organizational schemes, like the hierarchical classification schemes that are common in many subject domains. Such schemes allow intuitive navigation from the general to the specific. Users understandably expect such schemes for browsing, though they are expensive to manually produce and maintain for ad hoc subject domains.

To attain this goal, we modified the NMF clustering system so it could perform *hierarchical clustering*. We call the simplest adaptation of NMF *adaptive hierarchical clustering*, which can in fact be applied to any flat clustering algorithm. In adaptive hierarchical clustering, the clustering program operates recursively. This means that after performing a clustering at one level, it descends into each output cluster and performs clustering again for that cluster's subset of the original collection. This is done recursively until a stopping criterion is met.

The stopping criterion is why we call it this simple method of hierarchical clustering adaptive: a cluster is descended into only when the number of documents in it exceeds some pre-determined threshold. This makes intuitive sense: the very reason more detail is needed in a hierarchical scheme is to subdivide categories which contain too many items to browse or otherwise consider all at once. When this happens, the ontology designer tries to find facets common to all of the items in that category which can act as discriminants for separation into new subcategories. We emulate this process via subclustering recursively.

There are a few problems with the naive approach to recursive subclustering based purely on the above description. First, subclustering documents at a higher level but in the exact same form as their previous clustering is not comprehensible. Within the “civil war” cluster, a recursive clustering would likely produce all subclusters described primarily by the term “civil war.” In other words, despite the fact that all subcategories of a parent category in a scheme have the semantics of the parent category in common, we want to actually describe them by the semantics of the subcategories. The solution to this problem is to remove the information of the parent clustering from the documents before generating subclusters. This leaves behind only information that makes documents in a cluster different from each other. We approximate this by actually subtracting the centroid of a cluster from each document within it before processing that document for any subclustering. This approach works well for clarifying the semantics of the subclusters, and improving the quality of recursive clustering.

A second problem is that cluster size by itself is not a good stopping criterion. This is because, often times, a cluster will contain a “lump” of documents which have very strong similarity to each other, and this lump will happen to be larger than the cluster size limit. Subclustering this cluster will fail to split off many documents into additional clusters. This makes sense, as we would expect an arbitrary set of documents to sometimes represent just one dominant topic. To solve this problem, the recursive clustering system is sensitive to the possibility of a failure upon subclustering. If the largest cluster of a subclustering is not sufficiently smaller than the parent cluster, the subclustering output will be thrown away and the parent cluster will be left to stand.

Design

When our program starts up, it first parses the command-line arguments. From these it determines what mode it is running in (flat or hierarchical) and what the parameter values and input files are. It then loads the metadata files to in-memory lookup tables, and loads the document to a huge sparse matrix, called the term-document matrix, or just the “document matrix.”

It then enters the main NMF procedure. The output matrices (U and V) are initialized to random (0,1) values, and normalized. The sizes of these matrices are based on the number of clusters the clustering is being performed for, the number of features, and the number of documents. The term-document matrix is also NC-weighted at this time [Xu, 2003].

The core of the flat NMF clustering algorithm is a very simple iteration of some mathematical transformations of a handful of matrices. This iteration terminates when the output matrix values (in U and V) fail to change a certain amount after updating. This usually occurs within 10-20 iterations, regardless of the size of the document matrix.

After the NMF iteration finishes, a classification is produced for the documents. This is simply a map of cluster IDs to document IDs. Multiclassification is performed based on parameters supplied by the user. After this, centroids are calculated. Finally, the XML clustering metadata is written out to a file.

If the user specified hierarchical clustering, at this point, additional processing is done. The classification produced for the clustering run is analyzed to determine if any clusters exceed the size limit supplied by the user. If so, a subclustering for that cluster is initiated.

To perform a subclustering, the input document and metadata representations must all be translated to properly encode a subset of the original corpus. This involves re-indexing the document IDs, the metadata lookup tables, and even the word IDs for the document vectors. In addition, the document vectors are adjusted for centroid removal, for reasons discussed above. This translated data for the cluster is written out to a subdirectory with the name of the parent cluster ID. The clustering program then launches a child process within this subdirectory, which proceeds basically in the same fashion as the parent clustering process.

In this way, a directory hierarchy is output, containing classification and clustering metadata in a structure that mirrors the cluster hierarchy's structure.

2.1.1.5 Preprocessing

In this section we discuss portions of our clustering system which come before the core NMF program. These programs perform critical pre-processing steps, which have tractability and semantic ramifications.

Phrase-finding

Early on in our research we noticed that we were suffering some disadvantages in output because the features processed by our clustering system could only be words (or word stems, see below). The problem with this is that much meaning in text is carried not by words but by word pairs, triplets, or more. The assumption with text information retrieval (IR) is generally that a prohibitive amount of meaning will not be lost when text is interpreted as a “bag of words,” that is, an unordered list of the words in the originally documents. This makes sense inasmuch as many phrases in text can be treated as simple context-free combinations of their component words, and these component words do not appear in many other phrases with different meaning. However, often this assumption does not hold.

Text IR normally gets by because, for retrieval or classification purposes, the benefits of the bag-of-words representation far outweigh the drawbacks of ignorance of phrases. However, for clustering, we have the added challenge of needing to assist the user in efficiently understanding the content of a cluster. Below we discuss how we do this utilizing the words weighted as most-important for each cluster. Where phrases come into play is that, in this representation, it is very desirable to present phrases in a form humans will easily recognize. That is, terms in phrases should be connected, and in their proper order.

For example, it would be optimal for the phrase “civil war” to be recognized as a phrase, rather than just the words “civil” and “war” separately. Considering that the phrase “civil rights” is also common in our corpus, and the two phrases share the word “civil,” it would be very useful to disambiguate these two phrases and recognize (and present) them as atomic entities.

In sum, we wanted some recognition of phrases in the clustering system, both to make the output more comprehensible, and also to enhance the system's representation of underlying document semantics for the actual clustering itself. We needed to meet these goals while not abandoning the standard text representation model used throughout MetaCombine and assumed in our NMF clustering system. Typically, when text processing systems handle phrases, the underlying representation and algorithms and data structures needed to handle it balloon to unmanageable complexity. To address this, we built a phrase-finding preprocessor which converted raw document wordlists to pseudo-word lists, where some items are joined series of words, connected by an underscore character ('_'). As far as later processing steps are concerned, these pseudo-words can be treated just like words, with no modification.

The phrase-finding process is split into two steps:

1) the identification of the most common, statistically significant phrases in the corpus
2) the transformation of the original text into pseudo-word lists, based on selected significant phrases.

The first step is handled by a program called "phrasefinder", and the second by a program called "phrasemaker". This separation into two programs is convenient because it allows us to output, examine, and even modify a list of candidate phrases, before passing the list on to "phrasemaker" for the actual transformation of the corpus.

Readers familiar with natural language processing may have noted that what we have described above sounds a lot like an activity called *noun-phrase parsing*. In noun phrase parsing, linguistic analysis is performed to break text down into tagged parts-of-speech, with noun-phrases explicitly denoted. However, the benefit of our approach is that it efficiently archives our objective of recognizing common phrases, without the complexity and overhead of the actual linguistic analysis. Our method is based purely on text statistics and data mining techniques - we simply use the measures of *support* and *confidence* to determine which phrases are important. We sacrifice the recognition of infrequent noun-phrases, but as they aren't useful for clustering, we don't need them in the first place. Another problem we avoid is the intellectual property encumbrance of the available noun-phrase parsing systems.

Stemming and Stopping

Stemming and stopping are important and widespread procedures performed to improve both the tractability and quality of the text mining process. In our system, both stemming and stopping are performed by the same program: the *vectorizer*. The vectorizer takes a corpus in the form of document wordlists and outputs a mathematical representation of the corpus as sparse, numeric vectors (feature, weight pairs). In the middle, the vectorizer performs stemming and stopping, described below.

Stemming is the process of removing the stems from words, so that only word roots are used in representing the text. This effectively replaces words with equivalence classes of all words that have the same root. Thus, "walk," "walking," and "walkers" would all be stored as the single feature "walk". Usually this is a preferable situation for text mining, as most of the actual semantics are stored in the word root. For example, a query about alternatives to commuting to work by car might be phrased as "walk to work," "walking to work," or "commute walkers." Ultimately, the same meaning is being sought, and performance is increased by translating queries and document text to their fundamental roots.

Stopping refers to the process of removing "irrelevant" words from documents. This chiefly means common articles such as "and" and "the," which do not add high-level semantics to a document, but also includes less-common, collection-specific words like "we" and "present" (which, for example, might not be particularly useful features in a research publication collection). The underlying rule is that most of the semantics useful for information retrieval are actually stored in the verbs and nouns, and adjectives of the text. Removal of stopwords can better than halve the size of the text processed or stored in a text mining system, usually with an *increase* in speed and accuracy (this is because the most frequently occurring words in a collection carry the least information). The list of stopwords words to remove from text is called the *stoplist*.

Our vectorizer system, in addition to utilizing its built-in stoplist, can take as an argument a file containing additional ad hoc stopwords. This enables a feedback loop from clustering, whereby the user can use the clustering output (in the report form described below) to determine additional collection-specific words to add to the stoplist. After doing this, a new vectorized representation of the collection can be generated, and clustering can be re-performed.

Combined with the previously-described phrase-finding system, the use of this ad hoc stoplist functionality is further leveraged. By using pseudo-words produced by the phrasefinder, entire irrelevant phrases can be added as "stop-phrases". However, they need only be represented and treated as normal words by the stopper.

2.1.1.6 Post-Processing

In this section we discuss portions of our clustering system which come after the NMF program and utilize its output. These programs have to do with interpreting and utilizing the output of the clustering program.

Understanding Clustering Outputs

As discussed above, the clustering program outputs an XML file containing metadata for each clustering, corresponding to the number of clusters requested. As the raw XML is a somewhat inefficient presentation, we created a system to transform the XML into an HTML clustering report page. The core of this system is an XSLT stylesheet which remaps the XML data, and stylizes the output in a more easily comprehensible form. Some of the things this HTML report has that make it easier to understand are:

- attractive layout
- separation into executive summary section (containing highest-level view of clusters *qua* clusters) and detailed section (containing details of top documents)
- “graphical” representation of most important terms in each cluster, where weights are cast as the shading of the word
- hyperlinks between sections
- external links to full metadata

An expert can scan a number of these reports quickly, and determine which clusterings look the best. In addition, problems which can be resolved with alterations in the preprocessing or underlying data itself followed by reclustering can be identified.

Utilizing the Cluster Hierarchy

To utilize the output of a clustering (either hierarchical or flat), both the scheme information (the clusters) and the classification information need to be interpreted and loaded into a database. From there, digital library services can utilize the scheme and classification data in a common and efficiently-represented format. We used the clustering information in our database for both a demo hierarchical browsing service and our scheme evaluation experimental systems.

The loading/interpretation process is done with a Perl script we wrote. The script utilizes the classification output files, the clustering metadata XML file, and the directory structure itself. The nontrivial operations the script performs are:

- reading in the most-important words for the cluster and automatically generating a succinct category label
- recurring through the directory structures intelligently, only loading classifications at a level if there are no sublevels for a cluster (no directory with that cluster ID is present)
- generating category codes (X.Y.Z...) recursively based on cluster IDs
- mapping categories to internal database category codes
- loading classifications as category id, weight, identifier triplets.

Later, we discuss some of the challenges involved in one of the first of these procedures – the generation of cluster/category labels.

Visualization

The visualization portion of this project interacts heavily with the clustering portion. Semantic clusters have a very intuitive rendering visually, as well as many variants in ways to show various attributes of the underlying data and structure, and many potential ways that the display can be manipulated interactively.

Schematically, the canonical semantic cluster visualization is a two-dimensional field, with clusters represented as scatter-plotted circles. The positions of the circles are meaningless, but the distance between them carries meaning about their similarity (the closer together the circles, the more similar they are). The diameter of the sphere is proportional to the size of the cluster (the number of resources classified into that cluster). Cluster labels are somehow associated with each cluster, either as a text label overlaid on the region, or a popup message upon mouseover. One would expect that clicking on a cluster "opens up" to subclusters, shifting the field to show the relationships between these subclusters in the same manner as for the parent cluster and its siblings.

A variety of other manipulations and variations of attributes represented and ways of representing them are discussed in the section of this report on multidimensional visualization.

Here, we briefly discuss how the output of the clustering program becomes the input to such a visualization system. As mentioned above, the NMF clustering program can optionally output metadata for cluster centroids within its XML metadata report. The centroids are simply the "coordinates" of the center-point of each cluster in very high dimensional feature-space. The visualization system then projects these centroids into two-dimensional space so that they can be displayed on a plot (using either NMF or SVD, as we discuss in the MDV section). The viz system utilizes most of the other data in the XML file; important terms for derivation of cluster labels, metadata for top resources, count of items in a cluster, count of important terms for the cluster.

However, due to the recursive self-calling nature of the clustering program, a fully hierarchical visualization will have to deal with many XML output files (one for each subcluster/subdirectory). At this point, we deal with this, and the transmission of the hierarchical structure, by simply archiving the directory structure for transmission to the clustering system.

2.1.1.7 Continuing Challenges

In this section we discuss challenges we have encountered related to clustering, our attempts to date to deal with them, and possible future directions.

Labeling

It cannot be understated how important the process of generating category labels (or names) for the clusters is. The label is supposed to succinctly communicate the majority of the semantics of the category (or cluster, in this case). The semantics of subcategories are supposed to be a chain of general-to-specific refinements based on the names of their parents. This is all difficult to attain automatically. However, we must make an attempt, as manually generating labels for what is typically hundreds of category nodes is extremely labor-intensive. At worst, we can expect that an automated cut at labels can be used to speed-up a manual refinement.

The most convenient data from which to derive these labels is the word-cluster weight information output by the clustering system. As mentioned before, the clustering metadata XML file contains the top 10 words and their weights for each cluster. Obviously, a scheme where each category had a label exactly 10 words long would be ridiculous, not to mention that the weights of these words can fall off very precipitously. Usually, the most significant words are within the first three, by a large margin. Because of this, we apply a simple method of selecting a variable number of the top most-significant words for a cluster to generate the label. We simply take all words from the weight of the top word, to the weight of the top word times a factor of .9. The maximum number of words we select is give. This gives us on average about 2-3 words, which is more what one would expect from a scheme.

This method works very well as long as the drop-off in weights is rather steep (that is, the “best” terms for a cluster have weights that form a plateau), but otherwise, it is rather arbitrary. Some more analysis of the natural statistics underlying this problem could be useful to improve the automated portion of the process. However, we propose that automatically-generated cluster labels will never be “perfect,” and there will always be the opportunity for manual refinement by an expert. We would like to concentrate on convenient interfaces to do so in the future.

Hierarchical clustering algorithms

Above, we described one method of performing hierarchical clustering, a natural adaptation of NMF to hierarchies called NMF adaptive hierarchical clustering. However, this is not the only possible approach to hierarchical clustering. We have envisioned another method we call graph hierarchical clustering. This method would also be based on the same NMF core, and would work as follows.

Maslowska uses a flat clustering method to generate a large number of clusters, and then subsumption relationships between the clusters are calculated [Maslowska, 2003]. These subsumption relationships are used to re-form the clusters into a hierarchy. We propose to do something similar with NMF. NMF always outputs a matrix, V, which maps documents to clusters. For standard classification, only the top few mappings for each document are typically used. However, if we take all of these cluster mappings into account, we can calculate subsumption relationships for the output NMF clusters. This would then allow us to re-organize the clusters into a hierarchy.

At this point, we have not implemented this idea, and are not sure whether the output would be of quality comparable to NMF hierarchical clustering. However, it should not be too much work to implement, and is a “low-hanging fruit” in the exploration of multiple hierarchical clustering.

As time permits, we could also investigate clustering with non-NMF based methods. For example, Maslowska explores suffix-tree base clustering [Maslowska, 2003], and we could acquire such an implementation from colleagues in the digital library field. However, methods like these typically suffer from being extremely slow, and so they tend to be used for a posteriori clustering on relatively small retrieval sets. We would have to explore whether dimensionality reduction would allow us to apply such methods to entire collections.

Finding the Optimal Number of Clusters

We alluded above to the fact that finding the optimal number of clusters for a dataset is a nontrival task. Recall that NMF takes as *input* the number of clusters to produced, disavowing responsibility to make such a determination automatically. The burden is instead placed on the user of the clustering system. The problem with requirement is that a user who is seeking to *discover* the latent topical content of does not know this parameter beforehand. The user must instead guess, and try re-clustering for a variety of values of cluster count. This is why extensive and efficient output reports are critical for the clustering process; they enable the user to get a high-level sense of the quality of clustering for many clustering runs with different cluster counts. We have observed that one can intuitively infer clustering quality from metrics like:

- number of extremely similar clusters (that is, clusters having similar important terms)
- size of clusters (clusters with zero or a few items could be considered very weak)
- number of topics corresponding to clusters in large cluster count runs which are missing from runs with fewer clusters

These metrics make the process tractable, but still a considerable amount of work, with a steep learning curve. Our attempt to have a scholarly expert make optimal cluster count determinations through such reports based on one instruction session was not a success.

It would therefore be desirable to eliminate the need for the user to specify the number of clusters as input. Below, we discuss two approaches that could help.

Automation

It would obviously be most desirable to fully-automate the optimal cluster count determination process. However, such an ability has not yet been discovered in even the most cutting-edge document clustering research, as far as we know. The main problem is that the core clustering algorithm operates at a very low level, consisting of operations on a matrix with values that indirectly represent the semantics of documents. By contrast, the optimal number of clusters is a very high-level notion, involving the semantics of entire clusters and a complex relationship with the documents within them.

We have made some attempts at solving this problem by casting optimality in terms of various a posteriori numeric metrics, which serve as heuristics for the quality of a clustering. The basic strategy is to cluster for a range of values, for each clustering, calculating a quality metric. The clustering with the highest quality is then picked as the output clustering. Such methods could also use gradient-descent heuristics to quickly "zoom-in" on the optimal number of clusters without having to explore a large range of potential values. The starting point could be initialized to some arbitrary but common number of optimal clusters (such as 10), decreasing the average number of clusterings performed before the optimal count is found. The user could still specify a starting point or a range, but the output number of clusters could differ from the starting point or be outside of the range. This would drastically lower the burden on the user.

One attempt at a quality metric we've tried is to produce, for each cluster, a ratio of the average distance between documents *within* the cluster to documents *outside* of the cluster. Once these ratios are calculated for each cluster, a single number for the entire clustering can be produced in turn by averaging the individual cluster scores. Obviously, low values are better, with near-unity or over-unity values indicating very poor clustering. The main problem with this approach is intractability. It requires us to find the distance between every pair of documents in the cluster, a task on the order of the square of the number of documents times the number of features. Were it not for this "small" problem, the approach would do almost exactly what we want.

The other quality metric we have attempted to use as a clustering goodness discriminant is *reconstruction error*. Reconstruction error is a measure which is proportional to the difference between the original document matrix and the reconstruction of that same matrix based on the factors calculated by NMF. The reconstruction error can be efficiently calculated with little work beyond the work already done in the NMF iteration. Thus, there is no tractability problem. One would expect that the clusterings using the optimal number of clusters for a data set would have a minimum in reconstruction error. However, in our investigations, we found that this was not the case. It turns out that reconstruction error basically falls monotonically with increasing cluster count, with poor ability to distinguish especially good intermediate values. Therefore, we believe this metric is clearly too tied to the matrix representation of the data, and does not adequately represent cluster optimality. Reconstruction error, relative to different numbers of clusters, simply tells us that bigger factorized matrices approach the information content of the original document matrix. Thus, it remains relegated to its conventional role of determining when to terminate NMF iteration for a particular clustering.

Another metric we tried utilizes the output classifications to determine the quality of the clustering. The basic idea is that for a high-quality clustering, the document-cluster mapping matrix (V) will be very sparse, containing a value of "1" for the appropriate cluster for each row (document), and zeros elsewhere. Thus, if one took a matrix C generated from V which contained "1" in place of the max column value for each row, and "0" elsewhere, one would expect the difference between V and C to be small for a high-quality clustering. In practice, this assumption did not hold true. The first problem is that, the more clusters requested, the more columns the matrix V will have which will typically contain non-classifications (only one column is the chosen category for each row/document). Thus, the dominant effect of adding more clusters is to add more columns which are expected to correspond to "0" (non-classifications). The cumulative effect of noise values in these zero cells causes the output quality measure to fall near monotonically with increase cluster count: more clusters add more approximations to non-classifications, which are normally not considered.

A simple modification of this approach we tried was to replace the difference between zero entries in C and the non-maximum column values in V with a difference between the "1" value of C and the max value of V, averaged with the difference between the average non-max value of V and 0. This bins all non-classifications together, preventing the residual error of an increasing number of non-classifications from accumulating.

However, this simply reversed the problem: the quality now monotonically *increased* with the number of clusters, similar to the situation with the reconstruction error-based metric. Once again, this just tells us that classifications will improve as we approach the same number of matrix entries as the original document matrix. This metric had to be thrown out.

After the above approaches, we realized that something was missing in our quality metric: a penalization factor for increased data structure sizes for the representation of the collection. In information theory, this is called an *ockham factor*. In other words, there must be a quality metric component that punishes a lack of economy in data representation. Otherwise, quality metrics will indicate that the most verbose representations more faithfully represent the "input signal." This is the precise problem we were encountering with the two previous approaches.

We looked deeper into information theory and data mining, and decided to try something called the Bayesian Information Criterion (BIC). The BIC, based on Bayesian statistical theory, is composed of a *likelihood factor* and an ockham factor. The likelihood factor basically measures how statistically likely a representation of the input signal your output data is. We discovered a way to efficiently calculate the BIC for a clustering, avoiding scalability problems, and plotted the BIC for runs of clustering across a wide range of cluster count values. We did discover that BIC dictated a distinct maximum in quality, however, this maximum came at values of *hundreds* of clusters. We are unsure of why the optimal BIC should come at such a high value, especially when intuition tells us that that just a handful of clusters best represent the underlying topics in the collection.

What the BIC may be telling us is that the "optimal" number of clusters in the information-theoretic sense is actually quite high. Indeed, we note that in a fully-fleshed out hierarchical scheme, there actually *are* hundreds of nodes in the hierarchy. Perhaps what the BIC is telling us is how big the hierarchy of clusters needs to be. We are unsure if this is a fair assumption, however, because there is no notion of hierarchy anywhere in the underlying representational model (we only add it post-hoc with our adaptive hierarchical method).

Interestingly, there may be an opportunity to test this assumption about the BIC. As we discussed above, the graph hierarchical clustering method would actually work best with a large number of clusters specified. Instead of making this determination manually, the output of the BIC, and the clusters produced at its peak value, could be used. The graph hierarchical clustering processor would then take these clusters and organize them into a hierarchy. If the output makes logical sense and is useful for retrieval by end-users, then we will have solved the problems of cluster count specification, optimal cluster count determination, and improved hierarchical clustering quality. Naturally, we are excited about developing graph hierarchical clustering to investigate this possibility.

Contraction

Failing the discovery of a fully-automated method to determine the optimal number of clusters, we have conceived of a way to lower the importance of the input cluster count value, and hence reduce the burden on the clustering system user. We call this approach *contraction*.

Contraction would be a post-processing step built into the NMF system. Basically, the contraction processor would check the U matrix output by NMF (the weighted mappings of terms to clusters) and use it to determine which clusters are very similar. Similar clusters would be merged into equivalence classes of clusters, and document classifications would be accordingly remapped. For example, three clusters out of 10 that happen to be strongly described by "civil war" would all be merged into one cluster. The documents placed in these clusters would be placed in a new cluster representing all three. Assuming no other contraction took place, the clustering system would return 8 clusters instead of the requested 10, only one of which is the "civil war" cluster.

Contraction would solve the problem whereby "too many" clusters are requested and the clustering system is forced to split up topics that should naturally be represented as a single cluster. Large clusters should not be split up at the peer level like this; instead, they should be divided later as subclusters, after their centroids are subtracted (see our discussion of adaptive hierarchical clustering).

A great property of this approach is that it performs a “search” in number-of-clusters space without actually reclustering. Schematically, it “skips” directly from the K clusters specified by the user to an output with K-C clusters, where C is the number of clusters “combined away” with others through contraction. Thus, contraction would be a very efficient way to semi-automatically improve the determination of the optimal number of clusters.

From the user's standpoint, with contraction, it would be in their interest to specify an overly-large value for the cluster count, which needn't be as accurate as the former guess for the precise number of clusters had to be. The system could optionally return fewer clusters than requested, if the underlying data justified it. Thus, the user could perform clustering runs for fewer cluster count values, and have fewer clustering reports to examine (for example, 10, 15, and 20, instead of 10 through 20 inclusive). The user could be confident that runs for count values not explicitly specified would be “covered” through contraction.

2.1.1.8 Conclusion

We have built a full-featured, NMF-based hierarchical clustering system already at this point in the project. We are currently engaged in evaluation of its output, as we discuss next in this report. We believe that the tool is already very useful, if not for research production systems, then as inputs for further refinement by experts. There is much to do in terms of improving the quality of output, making the tool easier to use, and requiring less input and guidance from deployers of the tool. However, it is becoming apparent to us that the outputs will never be entirely without the need for refinement by experts. Thus, aside from refinement of the central computational tool, a goal moving forward in the second half of this project is to construct tools for experts to interact with the clustering system. We would hope such tools would help make the final output of clustering a higher-quality hybrid of automatic and human efforts.

2.1.2 Evaluation of the Effectiveness of Text Classifiers on Digital Library Metadata

[Section authors: Aaron Krowne and Saurabh Pathak.]

In recent years, automated text categorization has received much attention as a way to efficiently organize objects within of information systems. Digital libraries have not been exempt from this trend, with classification-related works a mainstay of conferences such as TREC, SIGIR, JCDL, ECDL, and many others. There have also been numerous algorithmic developments in the field of text categorization of late, such as Support Vector Machines (SVMs) and Bayesian classifiers. These have lead to comprehensive re-evaluations of classifiers [Sebastiani, 2002].

However, it is still not clear that a “best" classifier has been found, and it is likely that there is no such thing. Trials of classifiers on different types of content have found different results. Usually, one of SVMs, Naive Bayes, or kNN classification performs the best (most accurate) on the target content. We suspect that differing quantities of text, as well as varying intrinsic statistics of the content, explain the conflicting findings. In addition, text classification algorithms have different properties in terms of quantity of training data needed, time to train, and time to evaluate novel objects. These varying properties suggest that users of classification will continue to have to evaluate various classifiers and algorithms for their specific situations.

Some researchers have suggested using *committees* of classifiers (each with a different algorithm) to label documents, and have found this meta-approach yields the highest accuracy [Sebastiani, 2000]. While this makes intuitive sense, we suggest that it is simpler and more tractable for many users performing classification to select just one, most-optimal method.

Here we present an evaluation of text classifiers for our situation in the context of the MetaCombine project. MetaCombine is an effort to more meaningfully combine digital library resources and services. We have been employing classification on MetaCombine because, ideally, classification gathers together otherwise disparate objects based on their similar topical focus, via examination of their text content. Once associations between DL objects are made, they can be presented in proximity for browsing, link as related during searching, and connected in other forms of navigation or utilization.

2.1.2.1 Content

In this section we discuss the characteristics of the content used with our classifiers, for both the evaluation set and training set. We also discuss the classification scheme we used for this task.

Content Characteristics

The content we wanted to classify was drawn from the AmericanSouth.org site, which holds a union catalog-style collection. This collection was built by aggregating the metadata of a number of southern history and culture digital collections via the Open Archives Protocol for Metadata Harvesting (OAI-PMH) The resulting collection is very heterogeneous -- diverse and "lumpy," with varying size subcollections (which range from less than 20 items to about 20,000). In addition, we only had Dublin Core metadata to represent the records, thus, the text to be classified had to be derived from this. The resulting text representation of each record was very "sparse;" containing few words, with a low amount of meaningful overlap between records (particularly between subcollections).

These characteristics set our evaluation apart from classical evaluations of classifiers, which have had the luxury of a large quantity of text for each record (thousands of words instead of tens of words), high statistical overlap between records in the same category, and relatively even and complete coverage of the classification scheme. We feel that, due to these drastic differences, it behooved us to re-evaluate text classification for our setting. We think others building the same sort of digital libraries will benefit from such a fresh evaluation.

Training Data and Classification Scheme

Every classifier requires a *training set*; a set of "labeled" records. This simply means that the records in this set are properly associated with the categories that reflect their topics. Of course, a classification scheme is required as a source of the labels and, more abstractly, to serve as an ontology for item organization.

For our classification scheme, we used the named sections of the *Encyclopedia of Southern Culture* (*ESC*). For the training set, we used the full text of the articles within the encyclopedia. The label of each article was simply the section it appeared in, thus, this placement was considered a "canonical" classification.

2.1.2.2 Classification Systems

In this section we discuss the classification systems we utilized, along with the algorithms we selected in each for testing.

BOW

BOW is a text mining and retrieval library produced by Carnegie Mellon [McCallum, 1996]. It consists of C libraries and a number of programs to use the algorithms in the libraries. One of these programs, rainbow, is a classifier which supports all of the algorithms in BOW: naive Bayes, SVM, on-line linear, and many more.

The rainbow program has a convenient interface: the training set is read in as directories (one per category) containing text files which serve as examples for those categories. The rainbow program reads in this directory and builds a statistical model of the corpus, which is stored on disk. This model is used by all classifiers. To classify novel examples, we run rainbow as a query server, specifying the location of the model and the desired classification algorithm, and send it document text over sockets. For each document, rainbow returns a sorted, weighted list of categories, which we can select from and then store.

WEKA

WEKA is a text mining system from the University of Waikato [Garner, 1995]. It has a laboratory-in-a-box style, complete with GUI, meant for doing classification in general. It is not geared specifically to text classification. WEKA takes its input as an ARFF file, which contains formal declarations of the attributes, classes, and samples making up the classification problem. It has functions to perform classification and summarize and visualize the results.

WEKA can also output a text-based report giving individual classifications, as well as some overall statistics. We make use of this report to extract and load classifications into our analysis system.

Because of the ARFF format and generalization of the system, a bit of work must be done to use WEKA for text classification. The vocabulary of the text must be interpreted as attributes and fully declared in its entirety. Each document must then be converted to a vector referring to these attributes, and declared in turn.

More importantly, WEKA seems to have trouble with a large number of features (attributes), which in our case is a text vocabulary. While WEKA can accept data points declared as sparse vectors, its internal representation seems to be non-sparse. Thus, dimensionality reduction must be done for WEKA to run successfully on typical text-based data sets.

2.1.2.3 Experiment

In this section we discuss the experiment conducted to find the best classifier and algorithm on our digital library's content.

Methodology

The Test Set: Because our training set (the *ESC* articles) was of a different nature than our test set (AmericanSouth.org metadata records), it was not simply enough to perform cross-validation on the training set to evaluate the classifier. This would allow us to make only weak claims at best as to the applicability of the classifier to organizing the DL records. Instead we had to turn a subset of the actual DL records themselves into a labeled test set.

We produced this labeled set in the course of conducting another MetaCombine experiment on the efficacy of a variety of browse ontologies (see Section 2.2). In the first phase of this experiment, the paths of users were recorded as they navigated a variety of browse schemes in search of specific records. In the second phase, they were asked to select from a list of reasons for each category they browsed in, for each resource. The list of reasons they could select from were:

1) I thought this category was about the resource.
2) I guessed the computer would put the resource in that category.
3) The category had words in common with the resource.
4) Process of elimination/random guessing.
5) Accidental selection/ miss-click.

Reasons 1 through 3 can be interpreted as judgments about where the resource belongs, of course, with 1 being the strongest. Reasons 2 and 3 could still be considered valid judgments for classification, as they are still useful for organization due to the fact that users could anticipate placement. However, for the results presented in the rest of the paper, we only select reason 1, to be as conservative as possible.

We took all of the instances of this feedback provided by users for resources within the ESC scheme and interpreted them as classification labels for those resources. In essence, this gave us a “free" labeled test set for our actual digital library content. In addition, the use of multiple users helps to make any results derived from the test set more general. The test set size, number of user judgments, and other data set statistics are given in Table 1 . We also note that the judgments were performed on resources which were selected from the full collection randomly, yet balanced across subcollections, which provides an unbiased test set.

Table 1: Classification Experiment, by the numbers

Item	***Count***
Collection size	35,374 records
Test set size	99 records
User judgments	179 judgments
Training set size	1404 articles

Multiclassification

An effect of the inclusion of multiple users and imperfect navigation is that there are multiple test set labels for most of the records. We decided to include these multiple labels, as they can be thought of as making up a *multiclassification* of the items. In fact, our findings in Section 2.2 of this report clearly indicate that, not only is there not a single “right" answer as to the proper category for the typical resource, but users become very irate when resources are not present under any categories they feel are appropriate for them (even if they feel there are many appropriate categories). This is easy to understand; backtracking wastes time, and multiclassification is easy to do, given the non-physical nature of digital libraries.

Thus, we elected to work with multiclassification in our evaluation, rather than the simpler case of uniclassification that most researchers have reported on. As discussed above, this opens the door to a more satisfactory, efficient browsing experience. However, there are considerations to make for evaluation of classification accuracy with multiclassification, which we discuss later. In addition, one must keep in mind the “economy" of resource organization: most classifiers return a sorted, ranked list of *all* categories for each resource. In the extreme case, one could multiclassify simply by taking this entire list, and assigning all categories to every resource. Clearly this would result in a mess; every resource would be in every category. Thus, we limited multiclassification to the top three labels, and tried to take fewer categories based on a simple thresholding method.

In multiclassification, the crux of what we want is to multiclassify a resource only when its top-ranked categories are similarly weighted. Thus, we thresholded based on proximity to the top-ranked category for each resource via a simple linear multiple of its weight. This multiple was adjustable. For example, if it was set to 0.9, categories were assigned to a resource if their weights were between the weight of the top-ranked category and a score of 0.9 times its weight (up to a limit of three categories). We tried various values of this multiple for each classification algorithm. Interestingly, we found more of a spread of these binding weights for some algorithms than others, so the best threshold multiple was higher for some algorithms than others. We suggest that a reasonable threshold multiple for an algorithm is selectable based on how many multiclassifications overall are performed, with 1-2 times the number of distinct records being optimal. Multiclassification within this range results in multiple categories being assigned only when their weights are close, while simultaneously not cluttering the browse system with too many category assignments.

The Text Representation

As mentioned before, our items to be classified were DL records in the form of XML-encoded Dublin Core metadata records. To apply text classification methods, we had to generate a text-only representation of these records. The first step of this was to parse the XML records, and extract a few fields salient for classification:

- title
- description
- subject

These fields were determined to have the most free text rich in information about the topic of the record. The text from these fields was extracted and concatenated to produce a single string for the record. This string was cleaned by normalizing all whitespace to a single space, stripping it of most punctuation, and eliminating non-alphanumeric characters and words shorter than three characters. The full text of the *Encyclopedia of Southern Culture* articles (the training set) was also cleaned and normalized similarly.

For the BOW classifier, we could directly utilize this normalized text. To train the classifier, we simply made a directory for each category, and dumped the normalized *ESC* text into the appropriate directories as text files. For the classification, we ran rainbow as a daemon, and sent it the record text over a socket.

For WEKA, we had to do a bit more preprocessing, to use its specially-delimited “ARFF" file. We wrote a converter that transformed our raw, normalized text into an ARFF file. In addition, we had to concatenate the training and test set records so that WEKA would use the same vocabulary for both during the dimensionality reduction.

2.1.2.4 Results

For classical (uni-) classification, it is enough to simply present as overall accuracy the fraction of resources in the test set which were assigned the correct label by the machine learning system. This evaluation model is insufficient for our situation, where instead of comparing a correct with a proposed category, we have a *set* of valid categories to compare with a *ranked list* of proposed categories. In fact, this looks very much like the search engine evaluation situation, where one has a ranked list of search results, and users supply binary relevance judgments upon which performance evaluation is based. For this task, metrics based on *precision* and *recall* have been devised, and we can re-apply them here.

For evaluation of our multiclassifiers, we can use presence in the set derived from user browse judgments to substitute for *a posteriori* user relevance judgments. We then define the following metrics for multiclassification evaluation:

1) precision (p) : the fraction of assigned categories which are present in the user judgments set.
2) recall (r) : the fraction of categories in the user judgments set which is present in the assigned categories set
3) R-precision : the precision taken only for the top R assigned categories (by rank), where R is equal to the number of user judgment categories.
4) f_1 : a metric combining both precision and recall, equal to $\frac{2}{\frac{1}{p}+\frac{1}{r}}$.

We desire high precision because we do not want to clutter the browse system (or any other information system) with classifications that are not appropriate. This can manifest as false associations or long, mostlyuseless lists. We desire high recall because we want to minimize searching for an item in a category where it *should* be present, but is not.

There is a trade-off between precision and recall in information retrieval: the more items included in a ranked list, the greater recall becomes, but the lower precision becomes. This trade-off is important for multiclassification, where selection of a small number of categories out of a large number of candidates for each resource must be done by thresholding. This is why we report on f_1, which takes into account both precision and recall, and R-precision. R-precision gives us an idea of precision in a way that is customized to the "true" number of categories for each resource, which is often lower than our global cutoff of three.

We also note that, in the results presented below, we micro- versus macro-average. That is, we first get precision, recall, and f_1 for each individual resource, then average all of these scores together to get overall scores for the classifier. This makes intuitive sense, as we view performance at the item-level, rather than the collection level. In Figure 1, we present a chart of f_1 scores for various classifiers and classifier variants with different multiclassification threshold multiples.

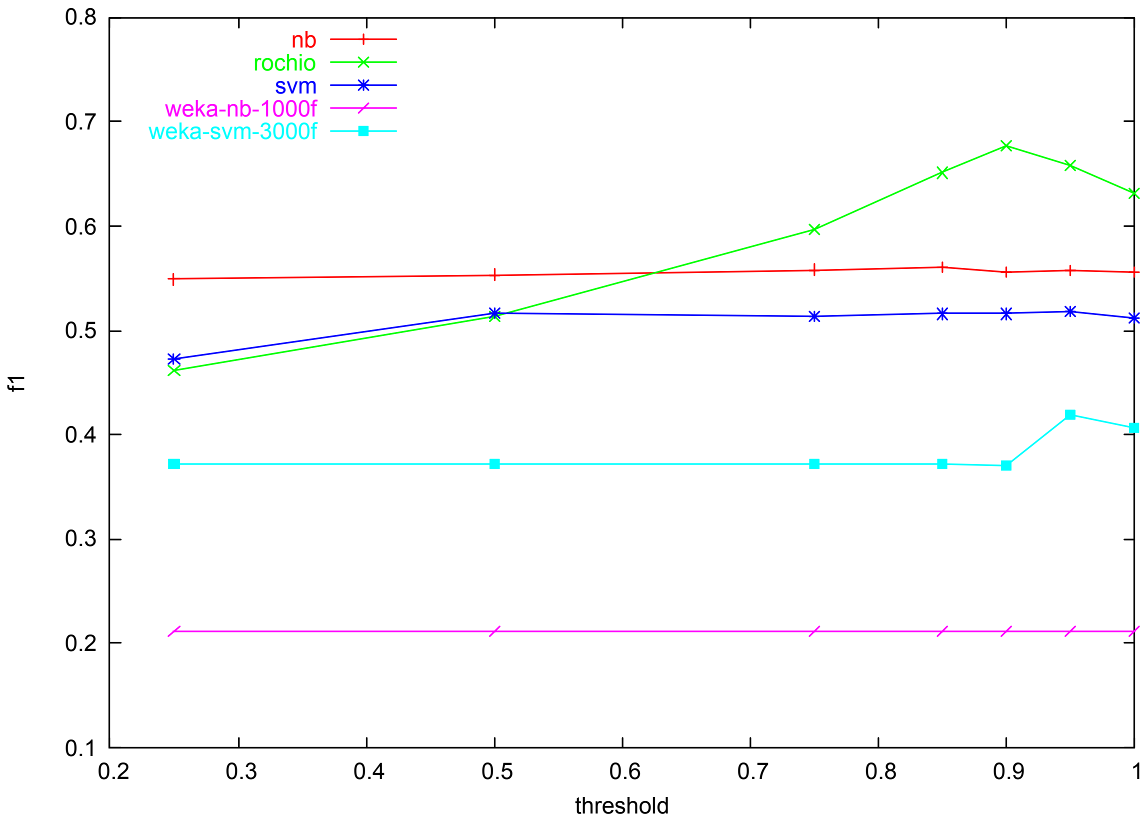


Figure 1 : f_1 scores for classifiers at different thresholds.

For perspective, we also graph the average multiclassification factor in Figure 2. The average multiclassification factor is simply the number of categories assigned divided by the number of distinct resources. This number is bounded above by three, due to our hard limit of three categories per resource.

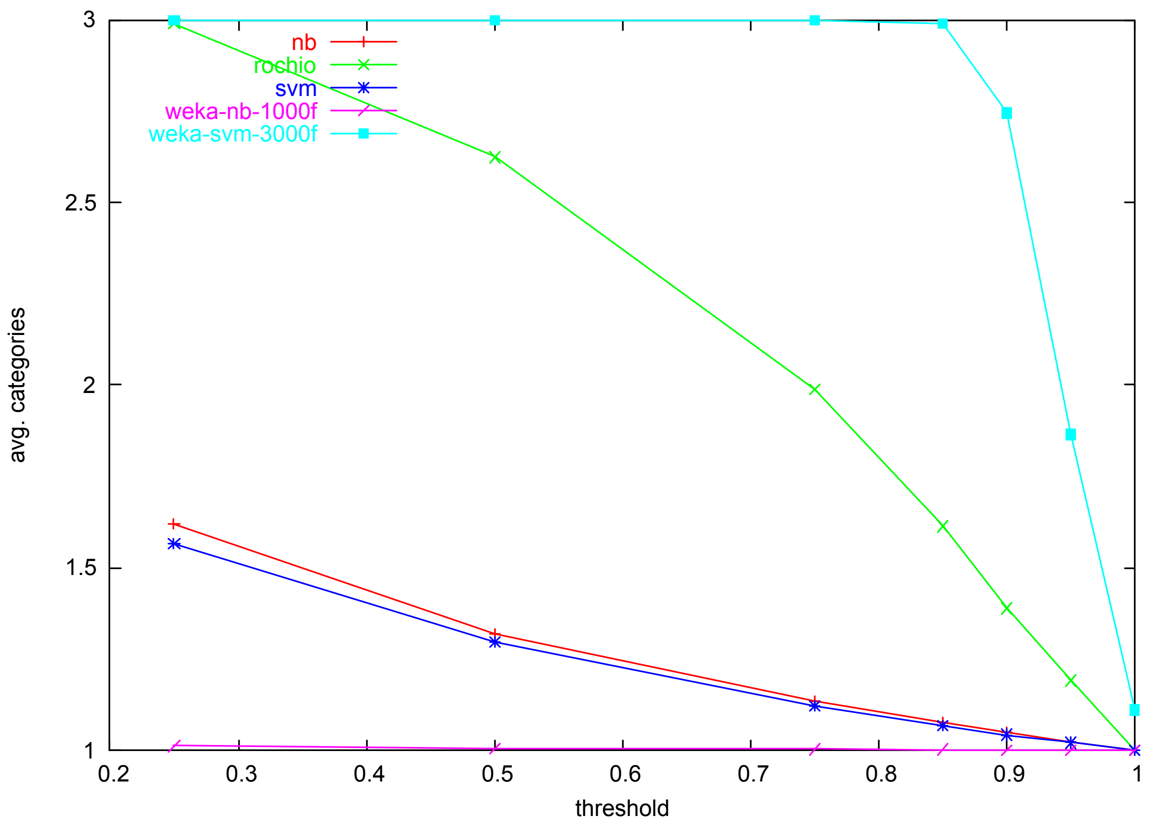


Figure 2: Multiclassification factor for classifiers at different thresholds.

2.1.2.5 Discussion

It came as a surprise to us that the the Rochio-linear classifier was overall the best, and had the highest peak f_1 by a wide margin. Recent research has heavily emphasized SVMs and Bayesian classifiers, often times even omitting a linear classifier. It is possible that the lack of use of Rochio weighting in linear classifiers has lead to a decline in interest in them.

Rochio weighting grew out of research into relevance feedback [Rochio, 1971], and strengthens queries by weighting them positively against the desired portion of a results set, and negatively against the rest of the results set. Translated to classification, each training set document can be augmented by the documents in its category, and reduced by the documents from other categories. This emphasizes the features which distinguish between categories, and enhances classification.

It is also possible that the content used in other evaluations was so different in nature that Naive Bayes and SVM did perform better. However, this illuminates the fact that classifier performance is very situational.

We hypothesize that SVM performs relatively poorly because the support vectors coincide minimally with very many document vectors. In our case, the document vectors are formed from record metadata, and are very sparse. Naive Bayes may perform relatively poorly due to its multiplicative formulation, in combination with the sparsity of our data. Besides the weighting enhancement in the Rochio variant, linear classifiers benefit from the fact that every coinciding feature of the evaluation document and test set is considered.

We also note drastic differences between BOW and WEKA. These seem to stem largely from the fact that WEKA's design does not let it scale well for a large number of features. We had to reduce the number of features used in our WEKA classification through a preprocessing round, ending up with 1000 or 3000 features (the difference in results between the two was not large). This was much less than the 50,000+

features in the overall corpus. Still, we are surprised by how much worse than BOW WEKA performed, as feature selection should generally keep most of the important information for class discrimination. This may in fact have something to do with the characteristics of our vocabulary.

An interesting discovery, enabled by the multiclassification graph in Figure 2, is that SVM and Naive Bayes (in both classifiers) seem to have either top-right or bottom-left tracing curves, as compared to Rochio-linear's middle-diagonal curve. This seems to suggest that linear classifiers have more relative and useful information in classification weights that SVMs or Naive Bayes classifiers, which can be used gainfully for multiclassification.

Overall, it is clear that the Rochio-linear classifier thresholded at a multiclassification multiple of .90 is optimal for classifying our metadata. We suggest that other projects with similarly sparse metadata look into classifying their records similarly.

2.1.2.6 Future Work

One basic way to improve this study would be to enlarge the test set. We could also test more classification algorithms and classification systems. An omission was that we did not test the effect of feature reduction on the BOW classifiers. However, it is preferable to avoid throwing away features if this is not necessary for tractability.

What we have done so far is called “content-based classification." This means that the text content of a record is the sole data that goes into determining its class. However, there are other portions of the metadata that may be useful for classification. For example, relational links between records (containment, citation, dependence) should be exploited when available. This idea is a generalization of the notion of using hyperlinks to enhance ranks in web searches (in fact, web-derived digital library records could directly incorporate this). Some work in using various types of links to enhance ranking and classification has been explored in a preliminary investigation [Xi, 2004].

2.1.2.7 Conclusion

In conclusion, we find that a basic Rochio-linear classifier performs best on our metadata records. We also discover that multiclassification is feasible for this type of content, and a simple strategy for selecting multiple categories yields significant accuracy improvements over uniclassification. Our evaluation is particularly pertinent for portal-style digital libraries, as (1) the sparsity of the underlying metadata records is a common feature, and (2) our classification test set was derived from feedback by real users in the digital library setting. In general, we have made a case for renewed attention to linear classifiers and case-by-case evaluation of classifier algorithms for different situations.

2.2 Experiment A1: AmericanSouth Metadata Clustering

[Section author: Aaron Krowne.]

2.2.1 Introduction

The rapid acceptance and deployment of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) has enabled the rise of digital library “portals". These portals usually specialize within some particular subject domain, in which case we refer to them as “subject portals". The Open Archives Initiative has fostered the rise of these systems by making the metadata for digital library records extremely portable, and the underlying content databases transparent. Specifically, OAI-PMH standardizes on a protocol, a data representation format (Dublin Core), and a design which puts low demands on the providers of content. Thus, digital libraries with special or otherwise sought-after collections can make their metadata available for re-use by relatively unaffiliated peers, allowing the sharing of content and addition of value within the digital library community in a *loosely-coupled* fashion.

A major example of a digital library portal built this way is the National Science Digital Library (NSDL). The NSDL's constituent digital libraries are usually examples of subject portals (such as CITIDEL for computing) or producers of original content (such as PlanetMath for mathematics). These portal-style digital libraries, subject or general, share as a common basis a *union catalog* of metadata records harvested from remote collections.

While OAI-PMH has been a great advance in that it has made it easy for such DLs to arise, the picture is not so rosy when it comes to building digital library services that are as unified as the underlying catalog itself. Although it is good that Dublin Core is a very flexible standard, this tends to lead to varying conventions and interpretations in its use at various digital libraries. In addition, often the best practices for field semantics are not followed. Yet, this does not “break" any of the technical elements of harvest and aggregation systems, which makes the resulting heterogeneity problem a latent one. Thus, when it comes time to build services on the union catalog, the system architect finds that the lack of a uniform semantic basis is a major bulwark to functionality.

The MetaCombine project seeks to address this problem of building services over heterogeneous metadata, in addition to other aspects of digital library integration. This article describes one phase of the project, for which we have focused on a *browsing* service.

2.2.2 Browsing

We define digital library *browsing* as an exploration or retrieval of a resource or resources through a navigable ontology. The ontology, which can be hierarchic, may also be referred to as a *subject hierarchy*, *classification scheme*, or various hybridizations of these terms. Typically browsing through such an interface is used when the end user does not know how to express the resource they are looking for in terms of keywords, or when they do not have a specific resource in mind. In addition, the intellectual extent and hierarchic structure of an ontology is in itself a learning and exploration tool, a fact exploited by end users even if they do not consciously realize it. Thus, ontological browsing is a distinct and valuable service for digital libraries to provide.

The browse ontology (or *browse scheme*) may be presented in a variety of ways. Typically one encounters a directory-style interface (as in Yahoo or Open Directory), with levels in the hierarchy displayed as clickable category names, and DL items in that category shown below. Alternatively, the categories can be rendered graphically. We plan to study such interfaces in a later phase of the MetaCombine project. However, for this phase, we used a simple text-based directory-style listing.

The problem of metadata heterogeneity in the subject portal DL setting manifests itself in full force in the case of browsing services. In order to place resources into an ontology (browse or otherwise), some sort of *classification* must be a part of each resource's metadata. Further, the classification metadata must uniformly correspond to the ontology used for the browse interface. In the loosely-coupled union scenario, however, there are a number of problems which undermine this assumption:

- Specialized DLs tend to use a variety of ontologies, standard or ad hoc. They are different and incompatible, but have the same coverage, as far as the subject domain is concerned.
- Some DLs will use no classifications at all, foregoing ontologically-based services.
- Some DLs will not propagate this portion of the metadata, perhaps because it is too much work to encode it, or there is simply no standard way.
- Some DLs might employ classification schemes that are too coarse for a specialized domain (such as UDC or LCCS), rendering them of minimal use for information organization within the scope of the subject portal DL.
- The subject domain covered by the DL may be completely novel, and no prior scheme may exist.

Without ontologically-based browsing, users of subject portals may be reduced to browsing via divisions into *subcollections*, with each “folder" representing an Open Archives (or other form of) content provider. This kind of organization (which we will also call the *trivial scheme*) is most useful for the rare instances where the user cares more about the collection of origin than the topics latent in the actual content. Common topics which appear in many subcollections are not proximate in the ontology, violating one of the basic principles of ontology design and the foundation their utility. This is the current situation on AmericanSouth.org, the collection we are using as the basis of our study.

To solve this browse ontology problem, a few things need to be done:

1) Ontologies need to be created (or discovered) where they are lacking.
2) Resources must be assigned category labels corresponding to the browse ontologies desired.
3) The previous two must be done automatically, or mostly-automatic, due to the quantity of content which must be labeled or relabeled (for example, around 40,000 resources for AmericanSouth.org).

We investigate meeting these conditions with both *text clustering* and *text classification* methods. It is our hypothesis that clustering and/or classification can provide functional browse schemes for subject portal DLs which exceed the functionality of the trivial browse scheme.

Some readers may ask why we cannot simply rely on manual, expert-generated ontologies and classifications. We concede that such a vehicle of producing content organization is currently the ideal and will likely remain so for quite a while. However, expert, manual construction of ontologies is known to be astronomically expensive. In addition to the construction of the ontology, resources are optimally placed into it by manual classification, which in turn dwarfs ontology-construction in cost, and clearly grows without bound in the size of the collection (hence the reason for condition #3). Thus, we take it for granted that manual browse schemes would outperform any of the automatic schemes and methods we are studying here, but must point out that the economics of the situation requires that our approach be considered.

2.2.3. Background

In this section we give an overview of text classification and clustering in general, as well as some essentials of which specific algorithms and methods we employed in our test system. The key difference between text clustering and classification is that classification assumes the existence of a classification scheme (or ontology *qua* classification scheme) and places items within it. By contrast, clustering takes a set of items *ab initio* and arranges them into subsets in such a way as to illuminate “latent topics." From this, a classification scheme can usually be derived with a small amount of work. Depending on whether a browse ontology already exists or whether one is desired, both clustering and automatic classification may potentially be useful to digital library portals for browse organization.

2.2.4 Classification

Text classification is a well-studied field, with extensive applications to digital libraries already. In the past 10-15 years, there have been a number of advances that have accelerated increases in effectiveness of text classification [Sebastiani, 2002].

Text classification refers to a task, however, and not a method. There are a number algorithms and methods based on them which all perform text classification. The particular method one should use often varies depending on the situation. Some of the most prominent methods are Naive Bayes, Support Vector Machines (SVMs) [Joachims, 1998], k-Nearest Neighbor (kNN), and on-line linear (tf-idf vector space).

In specific, text classification assigns to a document d a category label c from the set of all categories, C. A text classification algorithm may in fact produce a ranked list of all $c \in C$ for each d, with numeric ranks typically of the form $0 < r < 1$. Text classification requires a *training set*, that is, a set of already-labeled (categorized) documents. From this set, a model is built which allows the classification of novel documents, not in the training set.

Text classification is actually just an instance of the classification problem in general, where *objects* having *attributes* (or *features*) are placed into *categories* based on the values of these attributes. To fold text classification into this family of algorithms, document words are interpreted as the features, and entire documents are interpreted as numerical vectors. The indices of these vectors correspond to features, and the values correspond to the influence or importance of that feature.

However, what separates text classification from traditional classification tasks is the “high-dimensionality" of the problem; the number of features (or dimensions) is proportional to the dictionary size of the corpus being classified. To help mitigate this problem, some measures can be applied, such as stemming and stopping [Salton, 1989], and dimensionality reduction [Koller, 1997]. However, dimensionality usually remains in the hundreds to tens of thousands. This has in turn affected which algorithms are typically selected for text classification [Sebastiani, 2002].

In our case, we elected to use an on-line linear, Rochio-weighted classifier. We used the BOW text mining package for our classifier [McCallum, 1996], and in our tests of a number of leading algorithms, found that the Rochio classifier performed the best by a significant margin. To discover this, we used ten-fold cross-validation on our training set, the *Encyclopedia of Southern Culture* articles, and compared SVM, Naive Bayes, Rochio, kNN, and others. This result conflicts with many reports that find SVMs perform best, as well as our own subsequent investigations with the WEKA classification package [Garner, 1995]. We will likely integrate WEKA into future studies as a result. However, we do not believe the modest difference in cross-validation accuracy (Rochio with about 70% versus SVM with about 77%) alters the general thrust of our study.

2.2.5 Clustering

Text clustering shares many commonalities with text classification. However, there is a major philosophical and functional difference: clustering is *preclassificatory*. With clustering, one seeks to classify documents when no prior classification scheme exists. Thus, clustering both generates an ontology and places documents within it.

Similar to classification, clustering is actually a general task, applied to text documents via a similar rendering of their human-readable content. In addition, there are a number of algorithms and methods to perform the clustering task. Clustering has been studied for decades, largely in the social sciences. However, there has recently been a blossoming of new methods, as clustering moves into computing and information sciences. The dimensionality problem has been an even greater impetus for the discovery of new methods here.

Clustering methods break down into two families: agglomerative (bottom-up) and partition-based. Agglomerative clustering iteratively groups individual data points (documents) into clusters based on their similarity, producing a tree structure. However, the order of complexity of agglomerative algorithms is $\mathrm{O}(n^2 \log n)$ for n items, which is prohibitively expensive for large collections [Xu, 2003].

Because of this performance limit, we focus on partition-based methods. Some well-known clustering methods by partitioning are k-means [Willett, 1990], Naive Bayes [Baker, 1998], the Gaussian mixture model [A. Liu, 2002], Latent Semantic Indexing (LSI) [Deerwester, 1990], and graph-based methods (which turn out to be basically equivalent to methods like LSI, see [Xu, 2003]). For our study, we employ a new method called Non-negative Matrix Factorization (NMF) [Lee, 1999]. NMF is a relatively new technique, applied to clustering in a similar manner to LSI, with NMF replacing the singular value decomposition (SVD) employed by LSI, and eliminating the need for a secondary cluster demarcation step. Recent work shows that NMF tends to match or outperform LSI and other methods for document clustering [Xu, 2003]. Thus, we selected NMF as the basis for our clustering system.

NMF, as we implemented it, basically works as follows. Given a document corpus encoded into the $m \times n$ term-document matrix $\mathbf{X}$, a determination of k topical clusters, and non-negative $m \times k$ matrix $\mathbf{U}$ and $k \times n$ matrix $\mathbf{V^t}$, we must minimize the objective function

$$J = \frac{1}{2}\left\| \mathbf{X} - \mathbf{U}\mathbf{V}^{\mathbf{t}} \right\|$$

By re-writing the objective function J and applying the Lagrange multiplier minimization method, one can arrive at the following iterative update conditions for $\mathbf{U}$ and $\mathbf{V}$:

$$u_{ij} \leftarrow u_{ij} \frac{(\mathbf{XV})_{ij}}{(\mathbf{UV^tV})_{ij}}$$

$$v_{ij} \leftarrow v_{ij} \frac{(\mathbf{X^tU})_{ij}}{(\mathbf{VU^tU})_{ij}}$$

It is these updating rules that make up the core of our implementation. In addition, $\mathbf{U}$ and $\mathbf{V}$ are normalized at each step, and we initialize $\mathbf{X}$ with the NC (Normalized Cut) weighting from [Xu, 2003]. The iteration typically converges after a small number of steps (10-20). The output, $\mathbf{U}$ and $\mathbf{V}$, can be thought of as weighted mappings of terms to clusters and documents to clusters, respectively. From the former, we can derive labels from the clusters. From the latter, we can derive classifications, by using the cluster with the maximum mapping weight for each document.

2.2.6 Experiment

Our goal was to compare the browse efficacy of a variety of AI-derived browse schemes to the trivial scheme (separation of resources into subcollections, the easiest to produce in the harvesting scenario). The four schemes we produced for evaluation were:

- Subcollections - Separation of resources into subcollections. These corresponded to their Open Archives site of origin.
- Top-level clusters - A one-level scheme consisting of clusters formed over the entire digital library corpus (the domain of the history and culture of the American South).
- ESC-classified - A one-level scheme consisting of pre-existing categories derived from the chapters of the *Encyclopedia of Southern Culture*. The articles of the encyclopedia were used as the training set for the classifier.
- Subcollection clusters - In order to test whether subcollections could be useful in combination with the implicit organization of subcollections, we performed a separate clustering within each subcollection. This resulted in a two-level scheme, with subcollections at the top, and specialized categories derived from clustering within each.

To test browse efficacy, we reasoned that it would make the most sense to simply have actual users engage in browse tasks using the four schemes above. To conduct the study, record results, and do it in a way which controlled for bias, we wrote an on-line experimental system accessible through the web.

The system worked as follows. Users entered through a front page, and logged-in with their email address. Internally, they were referenced by a unique identifier. Each user was first assigned a random browse scheme. Within this browse scheme, they were assigned a random resource for the current browse task, which was guaranteed to exist within the scheme. The screen was divided vertically into two panels, with the current resource was summarized in the left-hand panel. This summary consisted of selected portions of the metadata printed tabularly (title, description, subject descriptors, ID). In the right-hand panel was a hierarchical navigator for the current scheme. Categories were displayed as links, and clicking a category would yield either another page of category links, or a list of resources in the category if it was a leaf node (summarized as titles, descriptions, and IDs). Unique integer identifiers were used for each resource, replacing Open Archives string identifiers, to eliminate hints as to the collection of origin of each resource.

As the users navigated through the right-hand panel, their movements were recorded in a clickstream log. In addition to this, a record was kept of wall-clock time elapsed during each browse task. When they finally thought they found the resource for the current browse task, they could click on a “found it!" link. If it was not the correct resource, they would get an error message, and could go back to continue. Otherwise, they were congratulated on a successful find, the browse clock was stopped, and were assigned a new resource. They could also give up on the current browse task after a certain number of clicks spent searching, at which point they'd be assigned a new task (with a new resource). This process proceeded until four resources had been found within the current scheme, at which point the user would get a text box to enter in free-form comments about the scheme. After this, a new scheme would be randomly picked, and four browse tasks would need to be successfully completed within this scheme as well.

This continued until all four schemes were completed, at which point the first phase of the experiment was done. Phase 1 of the experiment is rendered as a state diagram in Figure 3.

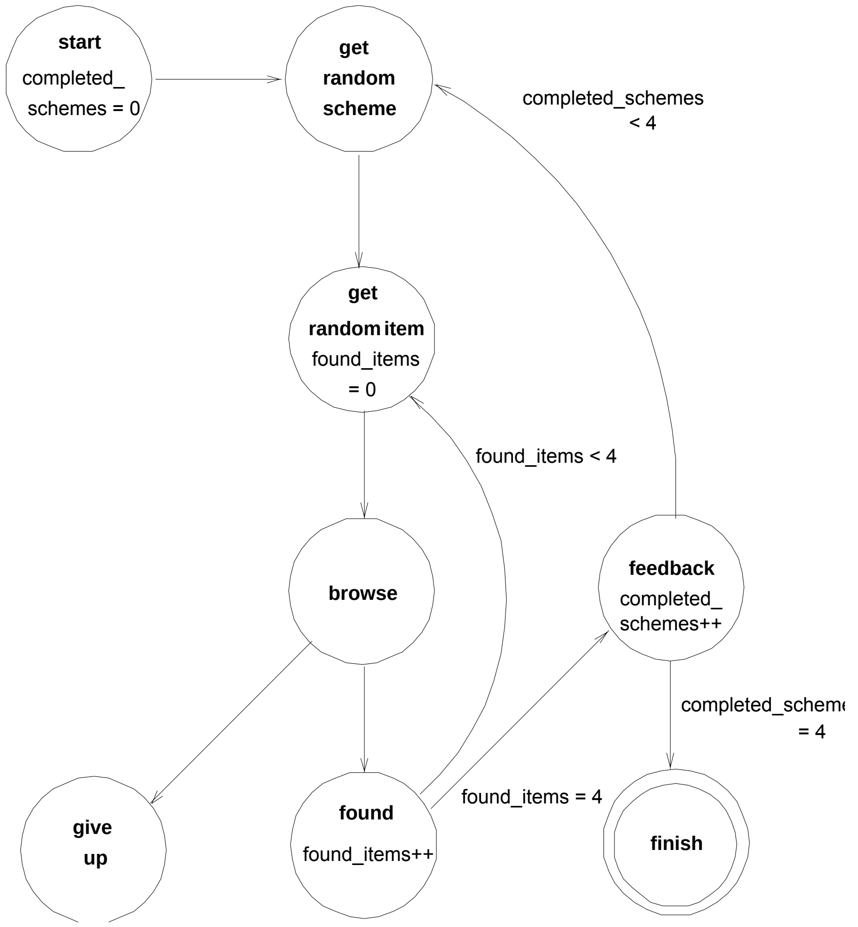


Figure 3 : "State diagram" for part one of the experiment (browse tasks).

In the second phase, the user was asked to review each step of their navigation through the browse schemes, in the context of which resource they were looking for. They were asked to select the reason for taking that step in the navigation, from a radio selection of five possible reasons (discussed in Section 2.2.7.6). Finally, they could enter in a few free-form comments about the study in general.

A few experimental design aspects are important to note. Firstly, the schemes were presented in a random order to each user, and resources were randomly selected within each scheme. This enabled us to capture learning effects *within* each scheme. At the same time, users did not have to finish the study. Because of the random order of the schemes, we did not have to worry about this resulting in a biased set of which schemes we had enough data for. In addition, the random selection of resources, ensured that no particular items from the collection received any emphasis. Further, this random selection was balanced across subcollections, so that there was no bias towards any one topic or particular style of metadata encoding. Finally, the ability to give up on tasks was counter-balanced by the requirement that the user still successfully complete four tasks within each scheme, ensuring we would have enough data from completed tasks, while also retaining participants of the study and reflecting an important aspect of the real-world setting.

For the study, we had 144 participants from Emory University. Two were faculty (1.4%), 16 were grad students (11.1%), 28 were staff (mostly library, 19.4%), 96 were undergrads (66.6%), and two did not specify their user class. Only 36 users completed all 16 browse tasks (25%). Only 54 completed at least 4 tasks (37.5%). However, due to the design of our experiment, we were able to glean useful data from all participants without bias, regardless of how much they did.

2.2.7 Results

In this section we present the results of our experiment. However, we believe that the real-world efficacy of a browse ontology cannot be summed up in a single metric, so we've identified a number of “dimensions" of performance and summarized the data for each.

2.2.7.1 Classification Accuracy

Classification accuracy is a well known kind of metric, used to measure how well an automated system places resources into a classification scheme, relative to some notion of their “correct" placement. This notion typically stems from expert or specialist judgments, as embodied in manual category labels applied to a test set of items. In our case, we lacked a substantial test set of this sort, as well as the expert time to create one. Although we had a training set for our classifier, none of the training items were actual resources from the digital library.

Instead, we formed our classification accuracy test set *post hoc*. As mentioned before, in the second phase of our experiment, we recorded user judgments of *why* each navigation click was performed. This became the basis of our evaluation set using the following methodology:

- Whenever a user selected the first reason (“I thought the category was about this resource"), this was interpreted as a category label for the (scheme, category, resource) triplet. *Note that there could be many of these labels for the same resource within a scheme, with the same or varying categories*.
- For each scheme, the set of resources included in such judgments was iterated over.
- The system's placement of the resource for that scheme was compared to these judgments. For metric $A^{(1)}$, the fraction of judgments matching the system classification was used. For metric $A^{(2)}$, we used a value of 1 if *any* category in a label specifying the same scheme and resource matched, and 0 otherwise (the motivation for this relaxed criterion is that we cannot objectively say which user's judgment is the correct one).
- The above values where averaged over the number of distinct resources specified in user judgments for that scheme.

To formalize this, let J_s represent the set of appropriateness judgments for resources in scheme s. J_s contains category-identifier pairs of the form (c, i). I is an operator that takes a set of these judgment tuples and returns the set of distinct identifiers over all of these tuples. Let $C_s(i)$ be a function which returns the set of unique categories from the judgments of J_s involving identifier i. Let $M_s(i)$ represent the machine-assigned category for identifier i in s. Finally, $H_s^{(2)}(C,i)$ is defined so that

$$H_s^{(2)}(C,i) = \begin{cases} 1 & \text{if } M_s(i) \in C \\ 0 & \text{otherwise} \end{cases}$$

and

$$H_s^{(1)}(C,i) = \frac{|\{M_s(i)\} \cap C|}{|C|}$$

Then $A^{(m)}(s)$, the accuracy of scheme s using metric m ($1 \le m \le 2$), is defined as

$$A^{(m)}(s) = \frac{\sum_{i \in I(J_s)} H_s^{(m)}(C_s(i), i)}{|I(J_s)|}$$

The classification accuracy using metric $A^{(2)}$ was quite decent for all of the browse schemes (see Figure 4), with subcollections coming in last by a statistically insignificant amount. Subcollection clusters broke out of a 3-way tie by a slight amount. The classification-based scheme performed the best by a significant margin of almost ten percent over the other schemes.

In the stricter metric $A^{(1)}$, nearly the same pattern holds, though the clustered schemes do slightly worse than the trivial scheme. Note that we can think of the divergence between $A^{(1)}$ and $A^{(2)}$ as a kind of “confusion" metric, which indicates how much disagreement there was in user intuitive judgments of classification. Proportionally speaking, this confusion is slightly greater for the AI schemes. It also seems to be the cause of the reduced performance of the clustered schemes in metric $A^{(1)}$.

These results show that only the ESC-classified scheme was significantly better than the trivial scheme in the classification sense. They also show that disagreement on subject is an issue which is present for the AI schemes, which we would expect from the fact that they are the only subject-centric schemes.

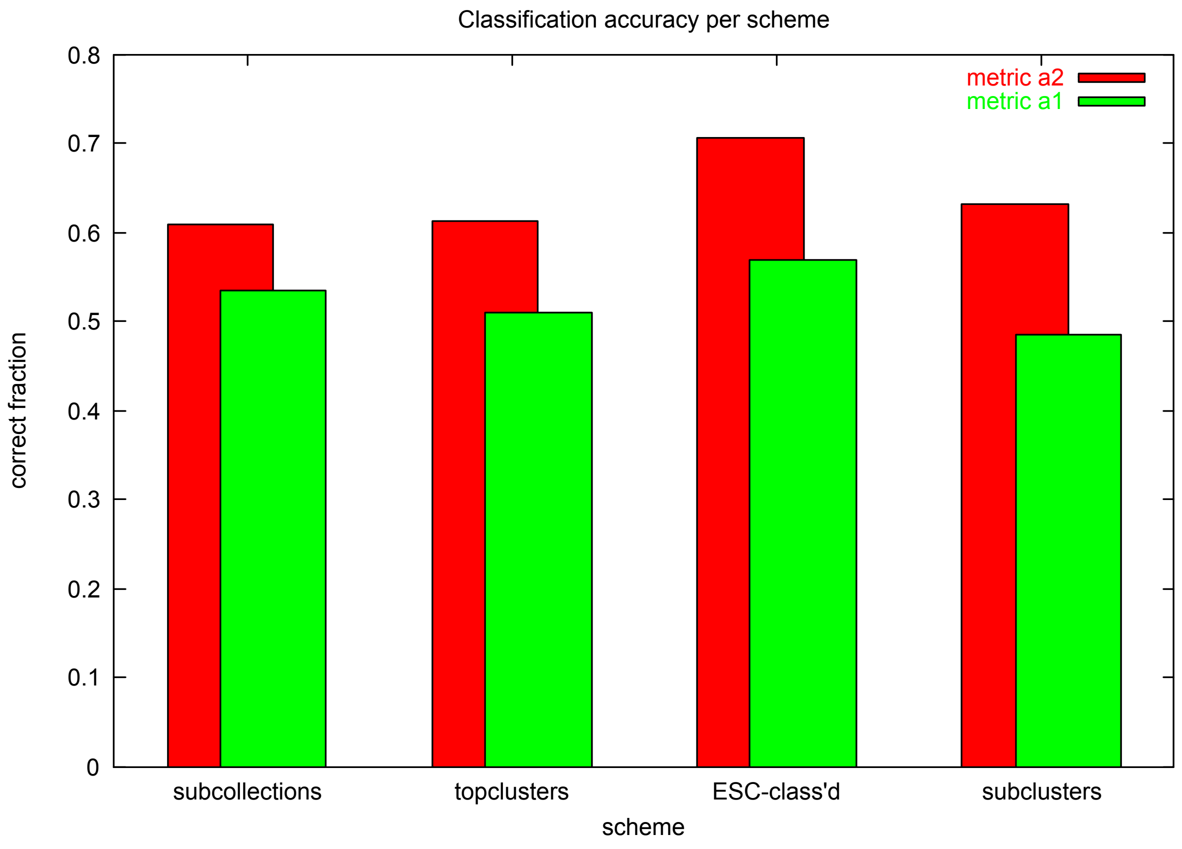


Figure 4 : Scheme classification accuracy.

However, classification accuracy is a metric which doesn't entirely capture the real-world aspects of elapsed time and total number of clicks, as well as other aspects we discuss below. In addition, the theoretical weakness of this metric is exposed by the simple observation that different users will select different categories for the same resources. There are many “right" answers in a specific instance; thus it probably makes less sense to look at *classification instances* than it does to look at the performance of *browse tasks* over the whole scheme.

2.2.7.2 Browse Failures

In this section we look at browse failures for each scheme, a simple metric which is based on the proportion of failed browse tasks compared to total browse tasks. A browse task was considered as “failed" when the user skipped it, or gave up trying to find the designated resource.

A chart of the of failed browse tasks is shown in Figure 5. Surprisingly, the trivial scheme suffers from the least browse failures, at about 10%. The clustering and classification schemes are significantly worse in this respect, though both come in under 20%. Finally, the two-level subcollection clusters scheme does quite poorly, with about a 30% failure rate.

However, there is more to overall quality and user satisfaction than the fraction of failed browse attempts. This ignores the aspect of how well the browse experience of how well the browse experience went for the majority of *successful* browse tasks. We look at this in a variety of ways in the following sections.

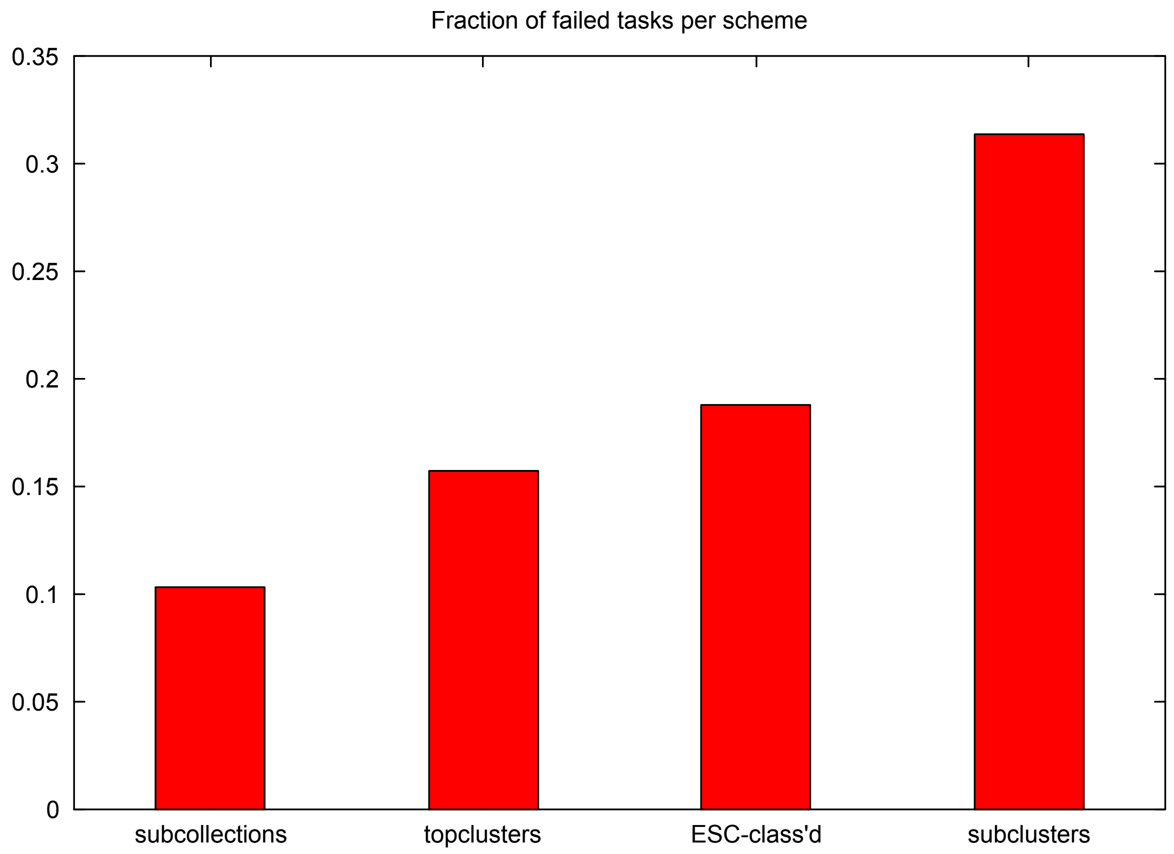


Figure 5 : Proportion of browse failures per scheme.

2.2.7.3 Elapsed Time

Elapsed time is simply the wall-clock time spanning the browse task: from when the user was assigned the resource to when they found it (or gave up on the task). In our view, elapsed time is one of the most important metrics, if not the most important performance metric, as it is corresponds to the kind of efficiency that often matters most (i.e., “time is money").

One general caveat to keep in mind with this elapsed-time analysis is that our investigation employed the browse schemes in a way that didn't test the free exploration mode of use of an ontology. In this mode, a user taking more time might actually indicate the scheme is doing a better job.

In Figure 6 we present a chart of average elapsed time for browse tasks, broken down by scheme. The chart is further broken down by all, completed, and skipped browse tasks. There are a few interesting things about this chart. Firstly, all of the automatically-derived schemes do worse than the trivial scheme, for all counts except skipped tasks. Here, subcollection clusters had a lower elapsed time, most likely because the two-levels of the scheme caused users to give up more quickly.

Interestingly, subcollection clusters did as good or better than both of the other automatically-derived schemes for finished and all browse tasks, indicating users found what they were looking for quicker, when they did not give up. Finally, top-level clusters basically tied ESC-classified browse tasks for elapsed time in skipped tasks, but beat it by a significant margin for all and finished tasks.

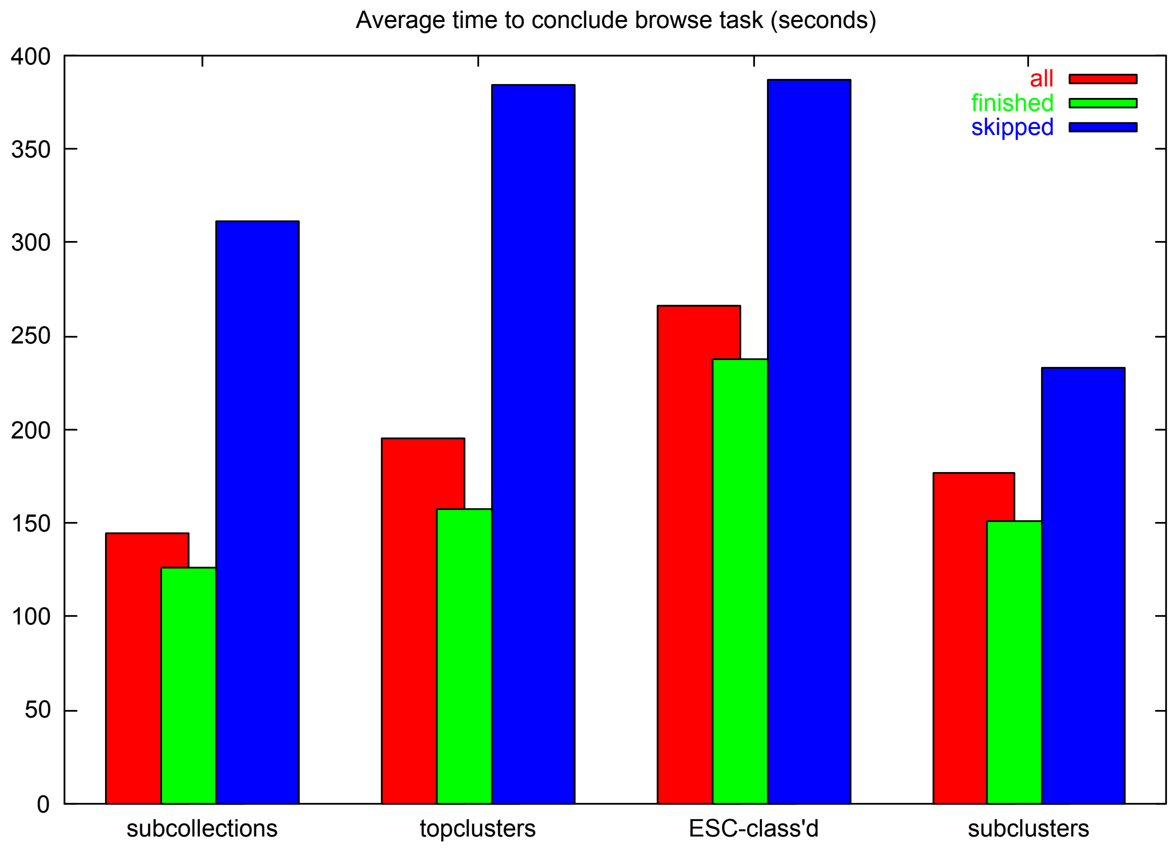


Figure 6: Average time (in seconds) to conclude browse tasks.

2.2.7.4 Learning Effects

A legitimate observation is that, while an ontology may be difficult to use and inefficient when it is first encountered, it may eventually become very efficient as the user learns it. In fact, this effect should be present to some extent in every situation where an ontology is employed. Thus, averages of efficiency metrics like elapsed time may not tell the whole story -- we also want to see how efficiency progresses as users get more familiar with a scheme.

Recall that we organized the experiment so that each user would receive a random sequence of schemes, but within each scheme, would have a series of consecutive browse tasks. This allowed for learning effects *within* each scheme, but prevented any bias towards learning effects *between* schemes. In Figure 7, we give a graph of average task completion elapsed time by task ordinal value. In other words, the first location on the horizontal axis is the first task, the second is the second task, and so forth.

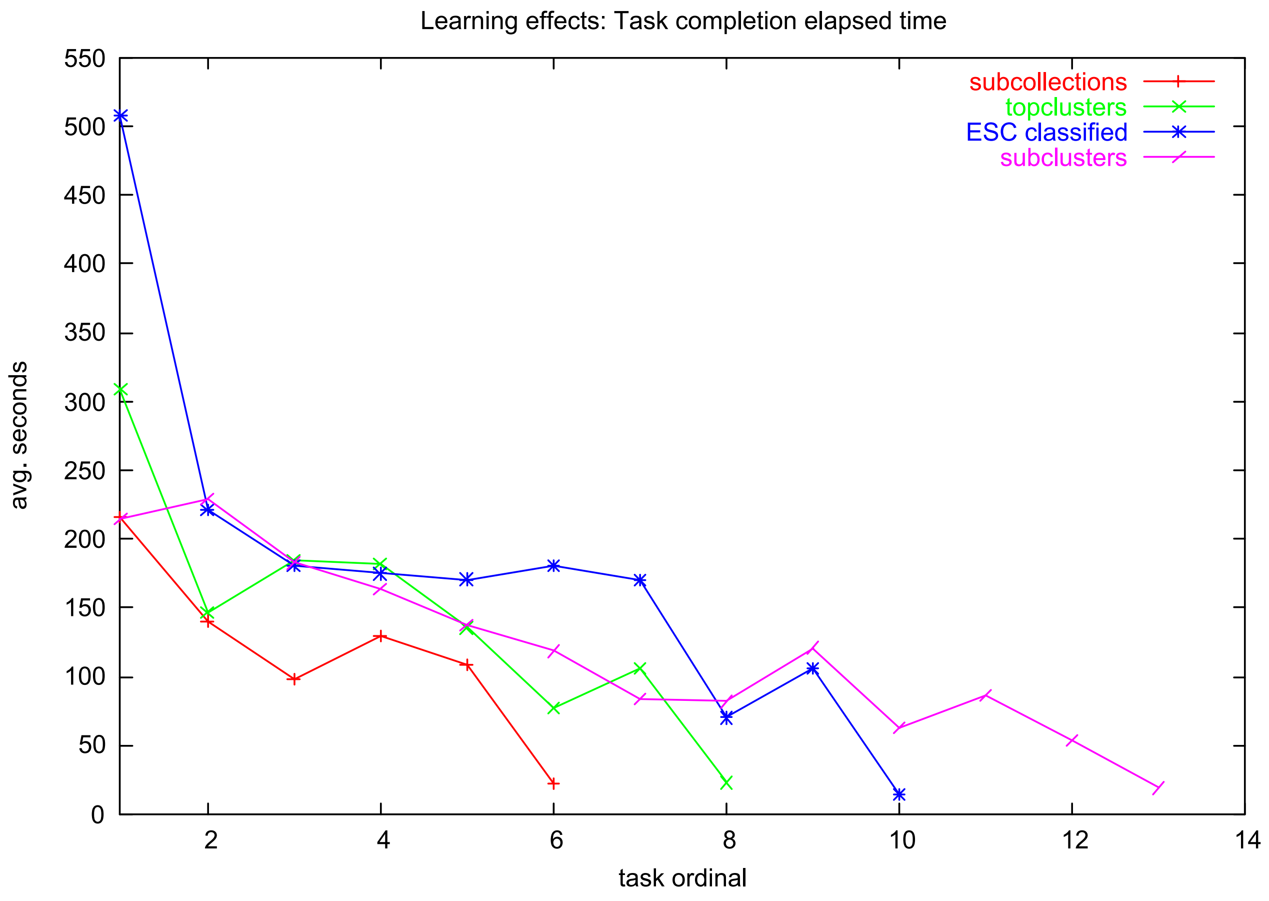


Figure 7 : Average time to complete browse tasks (seconds), by task ordinal value.

We can see from this chart that learning effects were indeed present. Some schemes, such as ESC-classified, started out quite poorly, but users learned to compensate fairly quickly. The other schemes had a more gradual decline. It could also be said that ESC-classified and subcollections had something of a “plateau" after initial learning, which could indicate a non-intuitive structure. Top-level clusters and subcollection clusters had the most steadily improving learning effects profile.

Note that these plots extend past four tasks. This is because skipped tasks are counted, and users received more browse tasks until they successfully completed four of them. Thus, schemes which had a higher prevalence of failed browse tasks will have plots that extend further toward the right. We count skipped tasks because the browsing that took place leading up to them still influences learning.

2.2.7.5 Clickstream Volume

Another interesting way to look at the economy of information retrieval through a browse scheme is to look at the number of clicks it took to find a resource (or give up). In

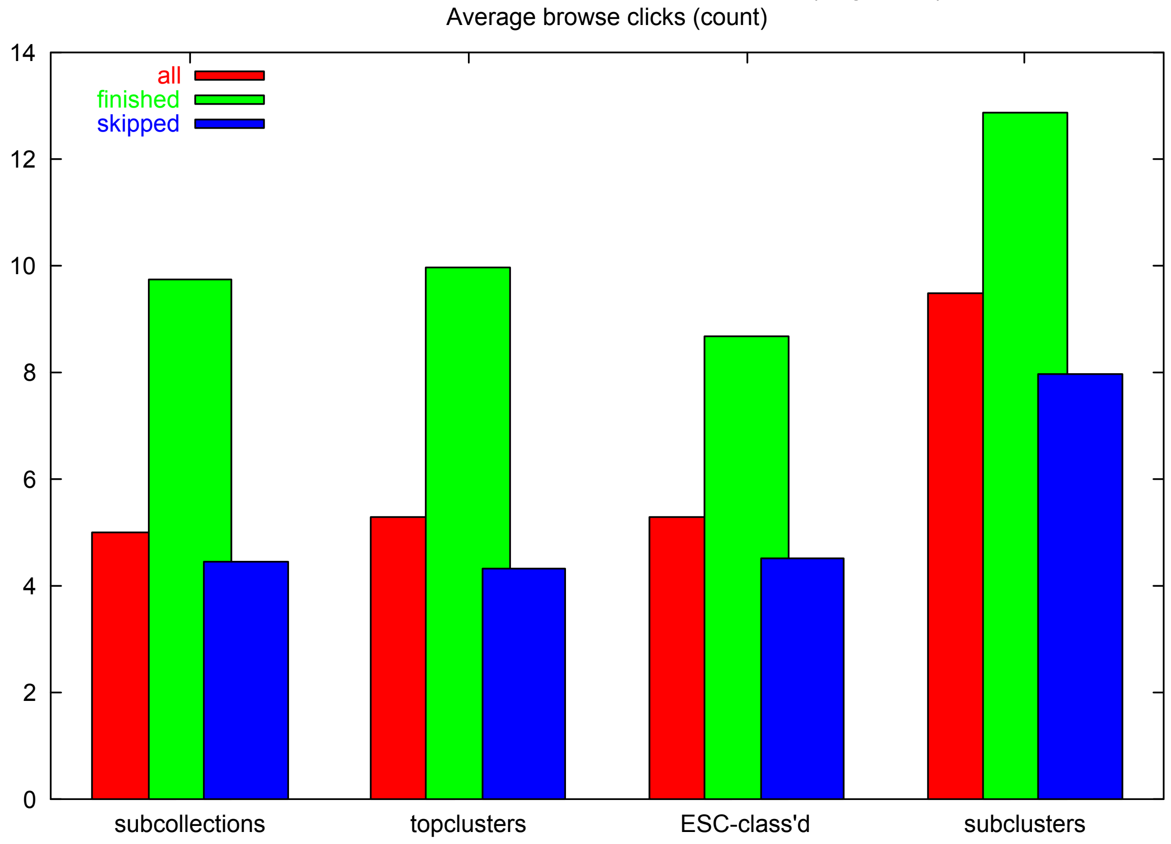


Figure 8, we give a chart of the average number of browse clicks per task. For each scheme, averages are shown for all tasks, completed tasks, and uncompleted (failed) tasks, allowing some insight to clickstream volume differences for successful and failure browsing.

This chart indicates that all of the one-level schemes seem to be running close to each other, with ESC-classified doing slightly better for finished tasks. The only two-level scheme, subcollection clusters, does predictably worse, as one must click a minimum of two categories to get to a resource. However, it does not do twice as bad, suggesting the two levels and additional detail are relatively helpful.

Note that in all cases, the overall clickstream volume most closely parallels the volume for failed tasks. The explanation for this is that by far, most clicks occur during browse tasks that the user eventually gives up on. This does not mean that most tasks fail to be completed successfully. As we saw earlier, this is not the case, as browse failures are in the minority for all schemes.

For comparison's sake, we also present *normalized* clickstream volume, which scales the same average by the ratio between the trivial scheme's size and the size of each other scheme. This can be interpreted as a kind of “relative efficiency". The intuitive justification for this is that, if one can add more categories to a scheme without requiring more clicks to find items, we have avoided frustrating retrieval and made it possible to more precisely find them.

To illustrate the normalization, the trivial scheme contains 21 categories and topclusters contains 25, so the topcluster averages were scaled by 21/25, or .84. These results are shown in Figure 9.

In this picture, we can see that all schemes beat the trivial scheme, most significantly in terms of finished tasks, with ESC performing the best of the one-level schemes. Our two-level scheme is by far the “most economical" in this sense, as the vast majority of categories in the scheme do not have to be explored to find the desired item.

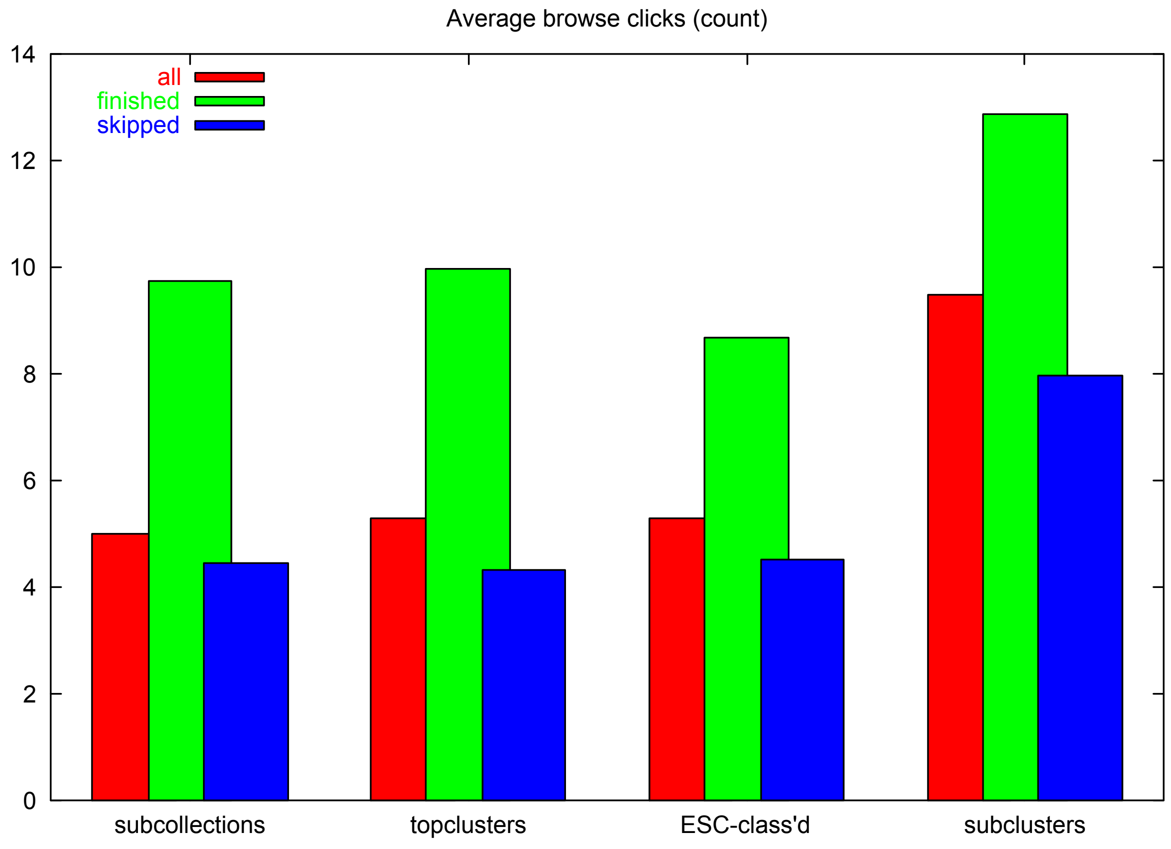


Figure 8 : Average number of navigation clicks to conclude browse tasks.

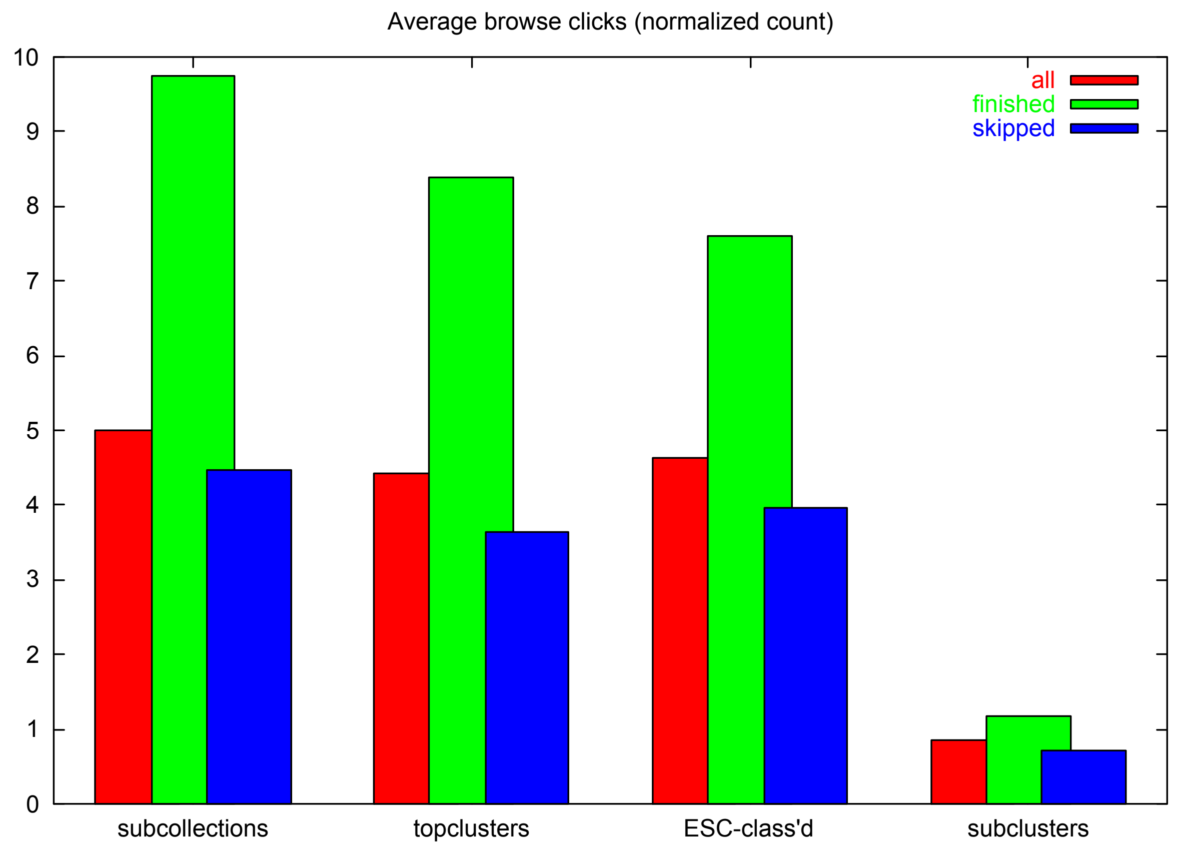


Figure 9 : Normalized Average number of navigation clicks to conclude browse tasks.

2.2.7.6 Navigation Reasons

Recall that our experiment was divided into two parts: browsing, and retrospective. In the retrospective portion, users provided feedback about *why* they performed each navigation event, or click. They were shown the record for the browse task, followed by a list of categories and a multiple-choice selector for each. The five choices available were:

1) I thought the resource was about that topic.
2) I guessed the computer would put the resource in that category.
3) The resource had words in common with category.
4) Process of elimination/random/didn't know where to look.
5) Misclicked/didn't mean to click on this category.

According to our preliminary investigations, these five reasons provide nearly complete coverage of clickstream motivations. The only vagary was that sometimes users thought multiple answers would be appropriate simultaneously; thus we instructed them to pick the *best* reason for simplicity.

With the exception of reason #5, these answers are not value-neutral. We can interpret the reason #1 as the most positive, as it indicates correct classification (both intuition and anticipation). Reason #2 indicates a lack of intuitive placement, but present was the ability to anticipate it. In other words, there was some understanding of the behavior of the classification system. If it was not intuitive, it was at least logical. Reason #3 is even weaker, indicating a guess by a simple word-occurrence heuristic, which is not necessarily the correct basis of automatic placement. Finally, reason #4 is quite negative, indicating a complete failing of intuition, no ability to anticipate placement, and not even the ability to apply a content-based heuristic to find the item.

In Figure 10, we plot the fraction of each of these types of answers for each scheme. A few patterns clearly emerge. The first is that guessing was clearly the dominant navigation type. We are unable to comment on this in an absolute sense. However, a clear sub-pattern of less guessing within the automatically-derived schemes is visible. In addition, both appropriate placement and anticipation of the system (reasons #1 and #2) rise for the automatic schemes. Finally, the simple heuristic navigation method (reason #3) falls for these schemes.

These results suggest that the automatically-derived ontologies are more useful as browsing schemes, due to an increased prevalence of positive modes of navigation, and decreased prevalence of more negative ones.

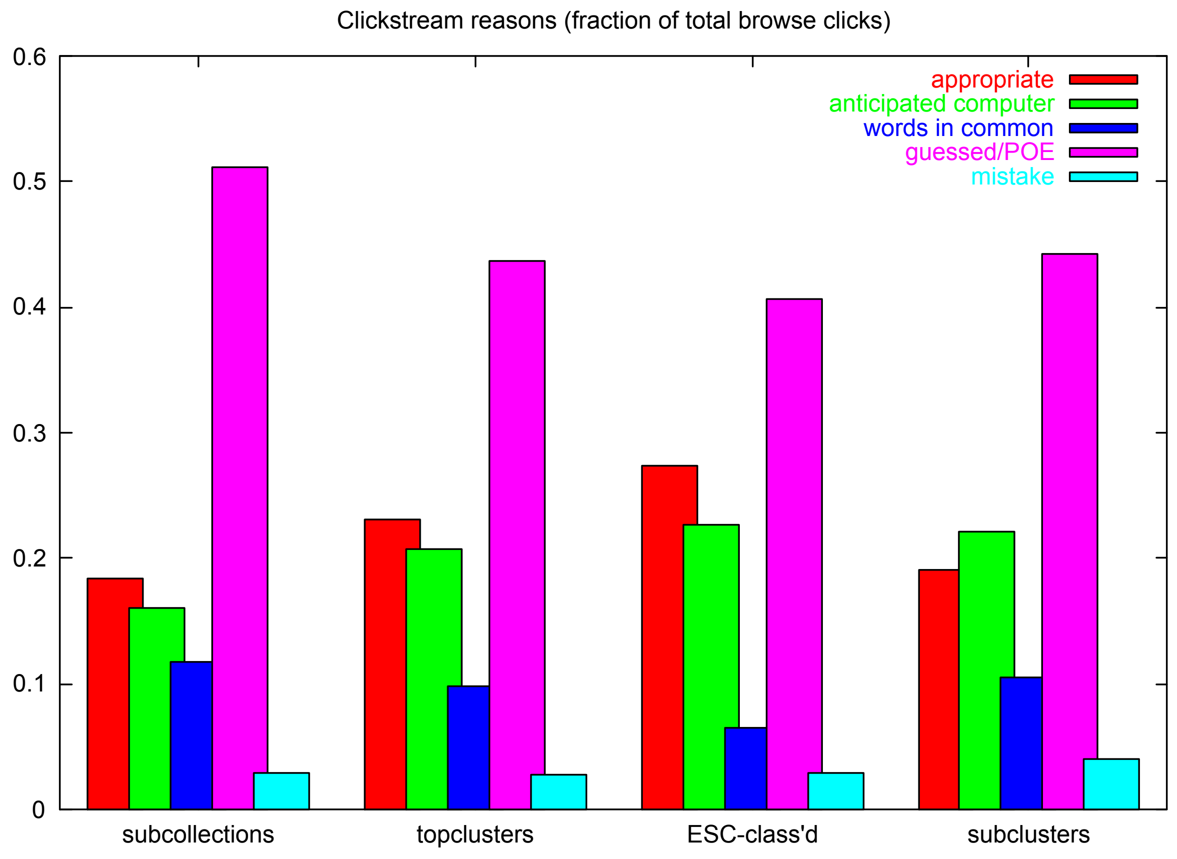


Figure 10 : Navigation clickstream reasons.

2.2.7.7. Free Response

At the end of each scheme, users were given a chance to enter in a free-form response, giving their impressions (both good and bad) of how easy the scheme was to use. This is exactly what many users did, and we received 30-40 non-null responses for each scheme. Some of the responses were purely emotional (such as “I liked it," “I hated it," “It was great," etc.), some were highly technical in the way they were complementary or critical, and some were basically irrelevant (commenting on the interface or aspects common to all schemes).

To get a picture of the overall gist of each scheme, we went through all of the comments and counted each one as either negative, positive, both, or neither. We kept a running tab of negative and positive comments for each scheme, with a single comment sometimes appearing in both or neither of the columns. In Figure 11, we illustrate these counts, normalized by the total number of non-null responses for each scheme.

From this, it is clear that the ESC-classification scheme was most-loved as a browse ontology; it contained both the highest number of positive comments and the lowest number of negative comments. The full ordering of browse schemes based on this metric, from best to least, is ESC-classified, subcollections, top-level clusters, and subcollection clusters. It is quite notable that subcollections does so well in this sense, better than the two clustering-based schemes.

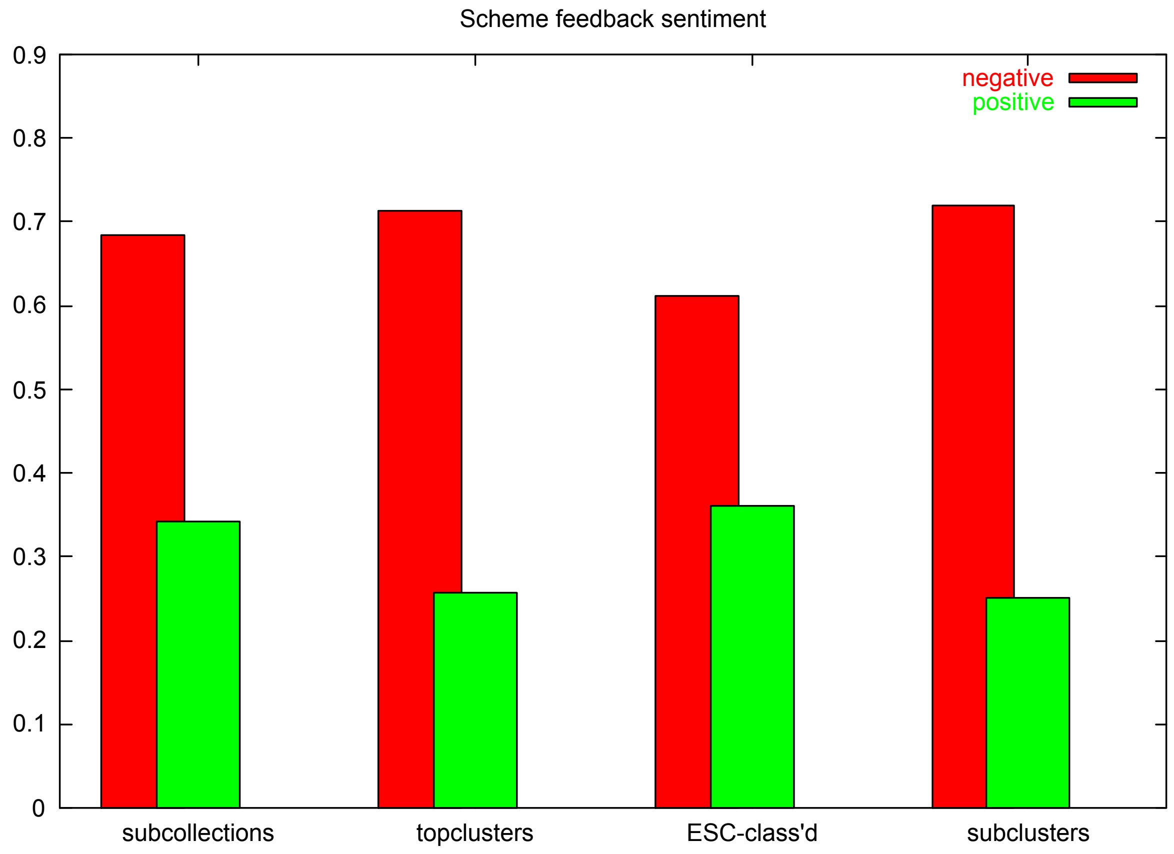


Figure 11 : Overall user sentiment for each scheme, as a proportion of non-null feedback responses for the scheme.

To show more in-depth taste of user reactions to the schemes, we present excerpts from selected comments in the following subsections. These comments highlight many important issues and problems with the schemes, which are important in determining how to improve their performance. Note that references to other schemes have been resolved to their specific nicknames. In some cases, we have made it clear where the user has seen all of the other schemes, to indicate how complete their comparison is.

2.2.7.8 Subcollections

Positive comments:

- “[Categories] were more precise [than ESC] but more overlapping. ..."
- “Browsing by collection was the most difficult and least intuitive method [of all of the schemes]."
- “Easier than [ESC], but still not ideal. Some of these collections, like Atlanta History Center, have lots of eclectic stuff that isn't hinted at by the collection name."

Negative comments:

- “The easiest set [of all of the schemes] (not counting mis-clicks), probably because of the inclusion of similar access points in both the description of the item to be found and in the various browsing categories."
- “It was easier [than the rest of the schemes], but only because the topics this time happened to have state names in them. If there is no reference to a state, it is almost impossible to find something unless you have other background info."
- “When assoc w/a location, easy."

The pattern which surfaces here seems to be that our subcollections, which had a high geographical bias, were quite easy to use for browsing when the geographical component of a record was known. However, in other cases, retrieval was much more difficult. The collections (like Atlanta History Center) which lacked a geographical bias were difficult to use through this ontology. The dominant positive impact seems to stem from the preponderance of geographically-slanted records in combination with geographically-segregated subcollections.

2.2.7.9 Top-Level Clusters

Positive comments:

- “Found most of the items in this section on the first try, but it was still slow-going. The categories were vague."
- “This [browse] scheme was much simpler than [ESC], as the subjects around which it was grouped were more specific to the material. There were still many subjects which could have been consolidated..."
- “Most of them were fairly easy to find, but documents that could be in multiple categories took a long time to find."

Negative comments:

- “It's still difficult to know where to start the search, because the category that seems most logical is often not the correct one. ... Sometimes the topic you want is buried under a category title that doesn't seem to have anything to do with it. And items that are similar seem to be split up into different categories for reasons that are not really clear."
- “Overall, the categories are not so appropriately named which makes it frustrating to find what you're looking for."
- “The trick seems to lie in dealing with overlapping subject categories."

There is evidence that the organization by inherent subject of the top-level clustering scheme, rather than source collection, is appreciated by users. However, a few problems are clear. One is that the cluster names only roughly correspond to their content; an effect which we propose is due more to the weak coupling of some records to the clusters than any weak coupling of descriptive terms to the clusters. In other words, classification confidence is relatively low for the vast majority of resources in a cluster, despite the fact that it is higher for that cluster than any of the others. This creates the dual effect of users feeling the cluster is both “too vague" and also poorly-named; two sides of the same coin.

Yet at the same time, users experience clusters as “too specific," focusing on extremely narrow topics and aspects of topics, such that many of them could be combined. For example, we had many top-level clusters related to the Civil War, yet many other topics remained “buried" throughout the other clusters.

The presence of both “too vague" and “too specific" complaints reflects the tug-of-war we experienced in selecting the optimal number of clusters for a non-hierarchical set. Since there was no way to automatically balance the detail of the clusters, in some places detail was lost, while in others, it was too refined.

2.2.7.10 ESC-Classified Ontology

Positive comments:

- “The subject [categories] are wonderful, and make so much more sense than the [subcollections]. At least at the initial search the information is so much more important than the originating institution. I had almost no trouble at all."
- “These category lists are much more straightforward than the [subcollections], though it's still not entirely intuitive which category to search for the item."
- “Was not intuitive, but went quicker than [subcollection clusters]."

Negative comments:

- “This final scheme was helpful when the placement of the item was evident, but generally, these categories were far too broad, and it was conceivable to place the item in several categories."
- “Some of these are VERY difficult to locate. I'm not sure it if is because I'm not very familiar with southern or civil war history, or if it is because I'm not making the correct connections with the descriptor phrases and categories."
- “Compared to [subcollections], this one was very difficult. Arranging the databases according to subject rather than according to collection was difficult because the articles for which one looks could plausibly be categorized under many of the subjects."

As we showed earlier, sentiment was the most positive for the ESC-classified scheme. The categories were considered much more clear, though still not entirely intuitive for the subject area. This is a result of the fact that the ESC “classification scheme" really was never intended to be such, it is instead an *a posteriori* grouping of articles received for an encyclopedia. Hence, much of the improvement over clusters can likely be attributed to its relatively uniform granularity, lessened duplication, and improved classification to within the categories. However, the negative comments show that the ESC scheme suffers from the same sort of problems as the clustered scheme.

2.2.7.11 Subcollection Clusters

Positive comments:

- “Overall ... I think the [sub-categories] provided this time around was more helpful than not."
- “This scheme, with its second tier of categories, was the most helpful search tool, [better than topclusters and subcollections]."
- “This was the easiest scheme [of all]. The descriptions on the items gave enough clues to find the right category."

Negative comments:

- “The browsing categories at the start are too scattered and unorganized."
- “This one was arranged more simply than [topclusters], though the fragmentation of information that could have been put together was frustrating..."
- “This was the worst [cluster-based] scheme by far. ... Who in the world would organize [this way]?"

This scheme seems to have “broken even", relative to both the clustered scheme and the trivial scheme. In some cases, it was an improvement, due to good clustering within subcollections. However, when the user did not know which subcollection (top-level category) to select, their retrieval task got harder by an order of magnitude. The main lesson to learn here seems to be that, in a hierarchical scheme, it is extremely important for the scheme's accuracy to be near-perfect at non-leaf levels, lest the effect of a failure become compounded.

2.2.8 Discussion

The various views presented above on the efficacy of the four browse schemes each tell a slightly different story. Each scheme seems to take its turn at being the “best" in one of the metrics, and the AI-derived schemes seem especially prone to swapping places with the trivial scheme for the title of “best scheme". Some of the differences in results likely stem from the qualities of the different resources assigned to users, in combination with the schemes they randomly fell under, further in combination with the character, mood, and experience of the user. Indeed, it is likely that there is no “one right answer" to the question of which scheme is “best". Rather, different schemes might be better for different modes of use which emphasis the different aspects we examined.

However, generally speaking, we can certainly say that the AI schemes do not clearly beat the trivial scheme, at least in our situation. The bias in our collection towards the geography of the American South ensured that subcollections remained unusually useful as an organizational tool. We do not, however, suspect that this will necessarily be the case for other subject portal digital libraries, in some analogous fashion.

Interestingly, our classification-based scheme did better than both clustering schemes in most situations. Therefore we feel that we were lucky to have this particular ontology and concomitant training set available, and suggest that other digital library architects try to find similar ontologies in their own domains. However, we feel that some of this relative performance of classification to a pre-existing ontology over clustering is due to to the nascent form of our clustering techniques. More could likely be done to improve our clustered ontologies greatly, such as noun-phrase parsing, the addition of “miscellaneous" clusters and thresholding into the others, and perhaps most importantly, hierarchical clustering.

We can also say with high confidence that, though they seem to be tied currently, AI schemes will likely surpass division into subcollections as the most effective browse ontology, even without major improvement. Ensuring this is the fact that, as subcollections integrated into a portal grow, the topics latent within them will reach a point of saturation, whereas the number of categories (collections) by definition grows without limit. This will lead to a high degree of topical scatter, with subcollections becoming less meaningful for organization.

There is also another way to improve the automatic ontologies: improved metadata. In the course of our study, we found that poorly encoded records were a major source of semantic confusion. Without clear field semantics, content-based organizers are left with an undermined basis. Although all of our records were Dublin Core, a major source of confusion lay in the interpretation of certain fields, especially description. We would like to see description used purely for the *topic* of an artifact, rather than meta-commentary about its presentation, location, provenance, or kind. For example, the description field for a portrait photograph should describe who is in it and perhaps where, while leaving photographic medium, method, and creator to other fields. Sometimes the “pollution" of the description field was unusually bad, containing boilerplate text from the originating institution or electronic access requirements. To even get to the level of quality presented above for our AI schemes, we had to perform a number of relatively labor-intensive clean-ups of content, such as lengthy stoplists and special text-cleaning routines to remove the most frequent boilerplate segments. Cleaner metadata would both speed up the preparation stage of forming and automatically utilizing ontologies, as well as improve their quality and the quality of resource placement.

Finally, we must emphasize that our study should be taken as a “stress-test" of various candidate browse ontologies, **not** a simulation of their usage in the real-world digital library setting. In an actual digital library, users would simply be able to switch to searching when they reach a point of frustration with the browse scheme, without harming overall satisfaction. In addition, search engines can be intertwined with browse ontologies, so that users can find their way through the ontology quicker. Finally, we did not test free-form browsing, which is a much more forgiving task. To illustrate, it may not matter as much if half of the resources on an arbitrary topic are in each of two categories; if the user finds one of them, they may find enough information for their exploration to succeed.

2.2.9 Conclusion

In conclusion, we have found that automatically-derived browse schemes at least match, and in many senses surpass the trivial organization of browse resources in portal-style digital libraries. Due in part to the variety of user characteristics, information needs, and collection characteristics, we believe there is already the ability to replace or supplement browse systems with one or more AI-enhanced schemes in the real-world digital library setting. We also think that the need for AI-based schemes may be even greater in some subject portal situations, and that the need for them across the board will increase as time goes on and a greater number of collections become federated. We also believe that it will be simple to enhance the quality of their information organization a modest amount over the results presented here.

2.3 Experiment A1: Revisitation With Scholars

[Section author: Aaron Krowne.]

2.3.1 Introduction

In this section we present a follow-up to the first A1 experiment in which the participants were the members of the scholarly design team. As there was no variation in the design of the experiment itself or its analysis, below we present just the results.

2.3.2 Results

Browse Failures

In this section we look at browse failures for each scheme, a simple metric which is based on the proportion of failed browse tasks compared to total browse tasks. A browse task was considered as “failed" when the user skipped it, giving up on trying to find the designated resource. A chart of the of failed browse tasks is shown in Figure 12. The results are precisely what we would expect; with topclusters coming in with the fewest failures, then ESC-classified, then subcollections, and then finally subcollection clusters. Our justification for this is that topclusters is a simple scheme that is “customized" for the content. The ESC scheme is also “simple," but it is not designed to fit our particular collection (both ESC and our collection have different and uneven coverage of the domain). Subcollections is a simple one-level scheme also, but does not particularly facilitate retrieval by content topic. Finally, subcollection clusters is a more complex, two-level scheme, so backtracking is more expensive and we would expect users to give up more often. These results contrast with the first run of this study, where subcollection clusters came in first (fewest failures). We are unsure how to explain this, except perhaps as some effect of the use exclusively of subject domain experts this time. The difference is strong enough that it cannot be ignored, and certainly proves that the AI-derived schemes have merit for some audiences. In the next sections, we take a more detailed look at the quality of the browse experience for *successful* browse tasks.

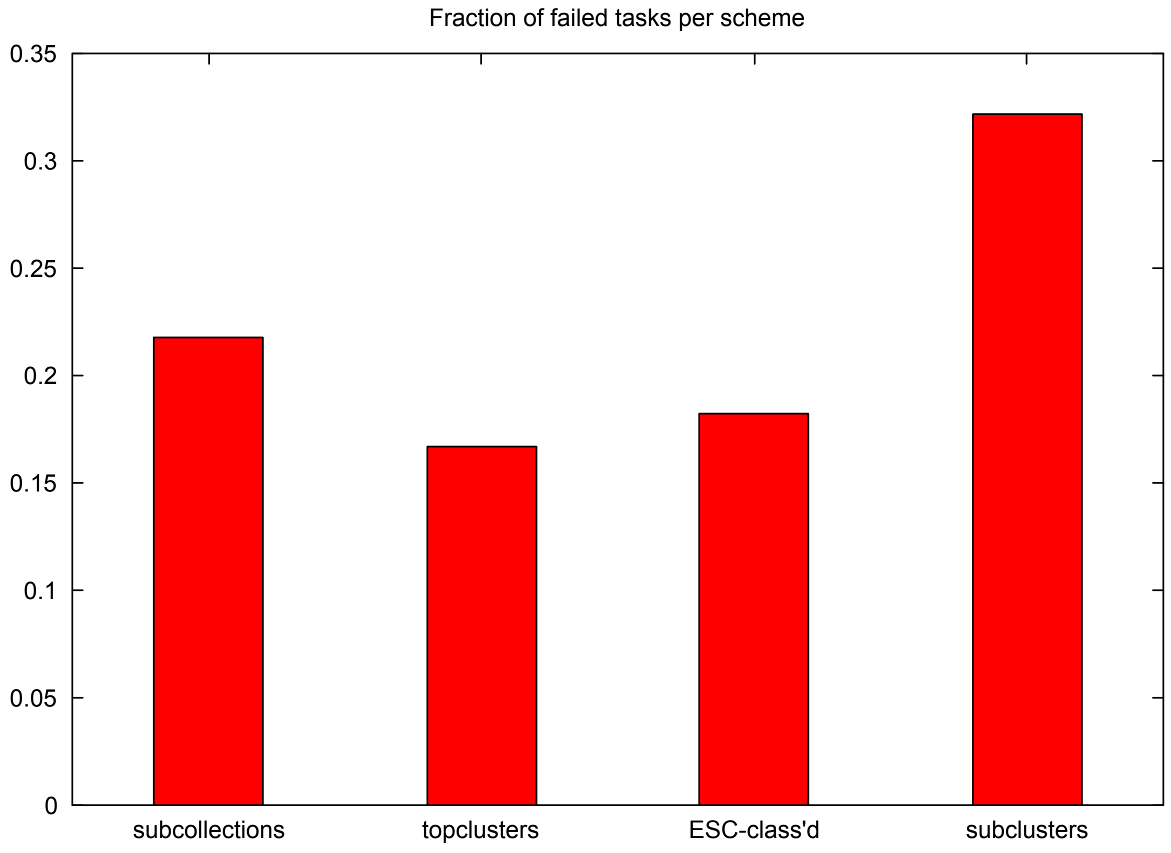


Figure 12 : Proportion of browse failures per scheme.

Elapsed Time

Elapsed time is simply the wall-clock time spanning the browse task: from when the user was assigned the resource to when they found it (or gave up on the task). In our view, elapsed time is one of the most important metrics, if not the most important performance metric, as it is corresponds to the kind of efficiency that often matters most (i.e., “time is money"). One general caveat to keep in mind with this elapsed-time analysis is that our investigation employed the browse schemes in a way that didn't test the free exploration mode of use of an ontology. In this mode, a user taking more time might actually indicate the scheme is doing a better job. In Figure 13, we present a chart of average elapsed time for browse tasks, broken down by scheme. The chart is further broken down by all, completed, and skipped browse tasks. We see that, for all conclusion statuses, the best-to-worst ordering is subcollections clusters, topclusters, subcollections, and ESC. The same ordering is mirrored, except even more pronounced, for tasks terminating with just successfully-found status. All schemes are about the same for failed tasks. This seems to indicate that a precise, two-level scheme provides significant advantages over one-level schemes, especially when it works successfully. The flat clustering scheme also does very well. Subcollections beats out ESC, likely due to more hints in the metadata as to which collection the resource came from than hints about which category of ESC it was likely to be placed into.

By average elapsed time, then, our clustering-based schemes win out. Classification is still too coarse and imprecise to beat the trivial organizational scheme, with our collection. These results conflict with the first run of this experiment, which was done on a more general end-user population.

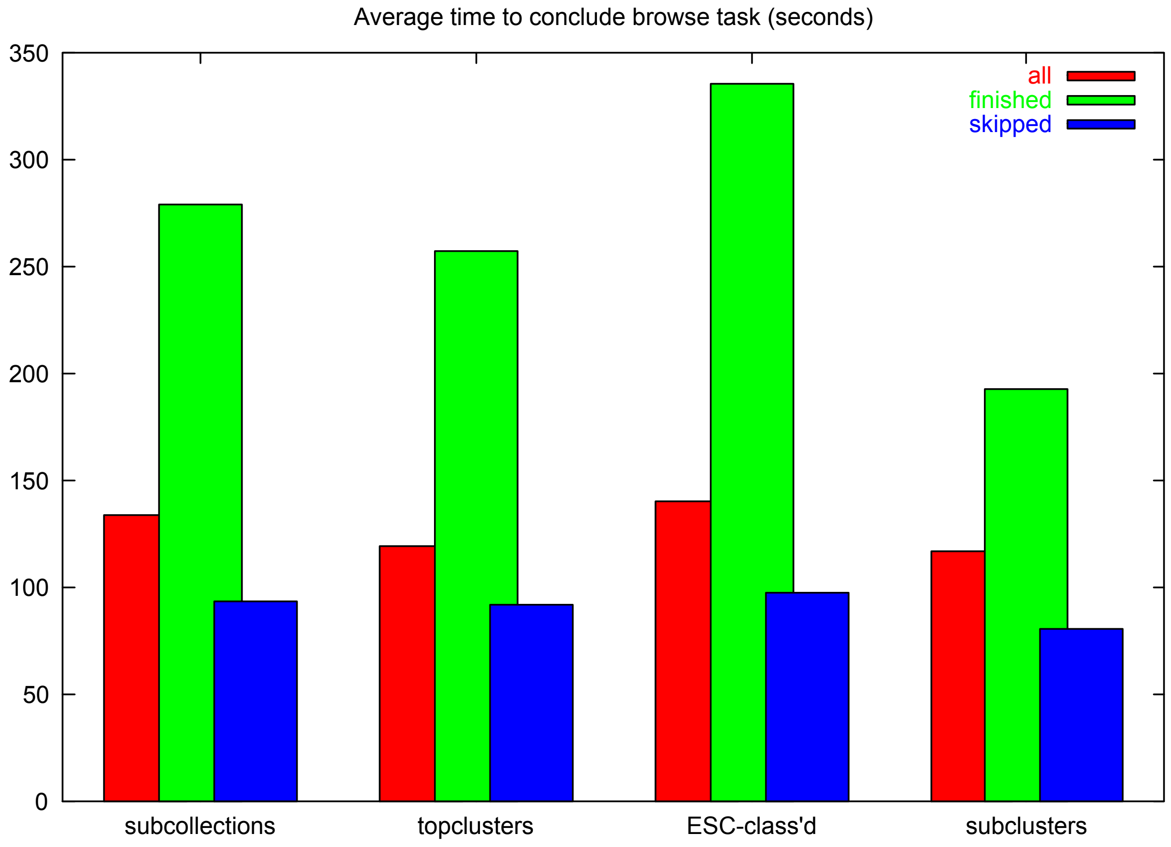


Figure 13 : Average time (in seconds) to conclude browse tasks.

Learning Effects

A legitimate observation is that, while an ontology may be difficult to use and inefficient when it is first encountered, it may eventually become very efficient as the user learns it. In fact, this effect should be present to some extent in every situation where an ontology is employed. Thus, averages of efficiency metrics like elapsed time may not tell the whole story—we also want to see how efficiency progresses as users get more familiar with a scheme. Recall that we organized the experiment so that each user would receive a random sequence of schemes, but within each scheme, would have a series of consecutive browse tasks. This allowed for learning effects *within* each scheme, but prevented any bias towards learning effects *between* schemes. In Figure 14, we give a graph of average task completion elapsed time by task ordinal value. In other words, the first location on the horizontal axis is the first task, the second is the second task, and so forth.

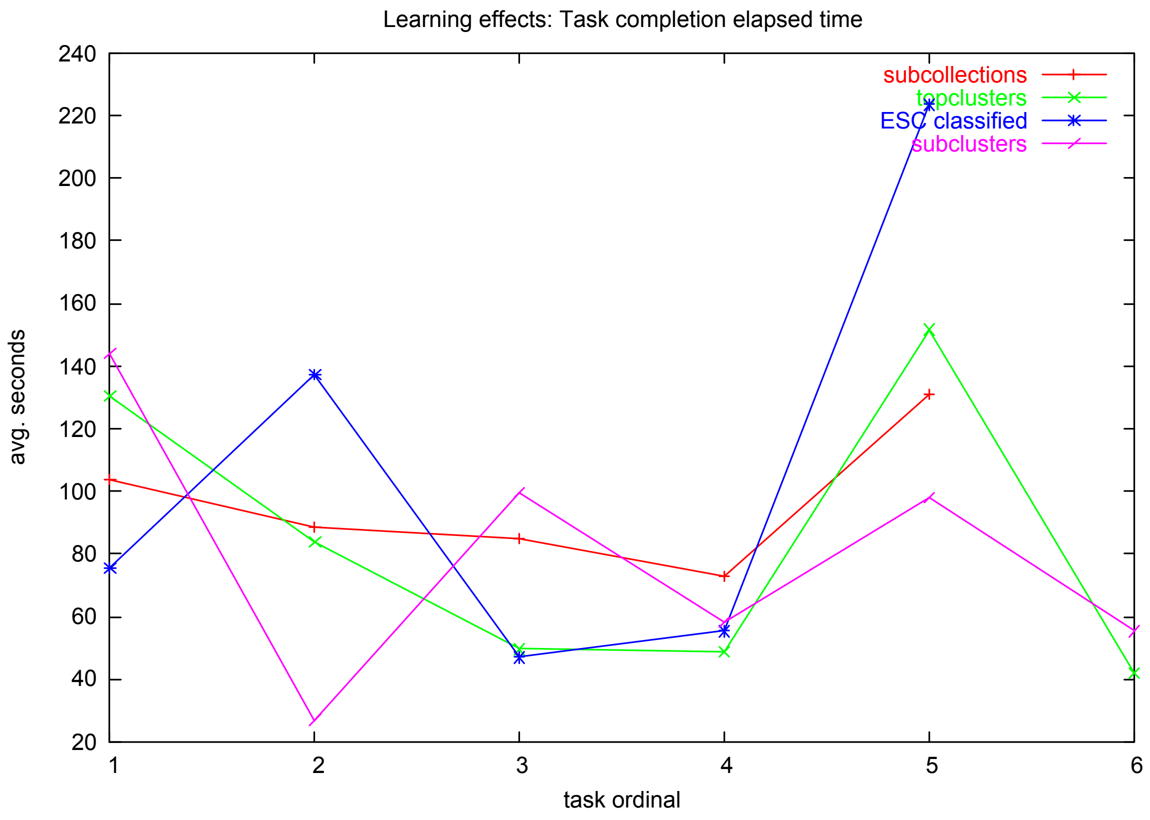


Figure 14 : Average time to complete browse tasks (seconds), by task ordinal value.

We can see from this chart that learning effects were indeed present. All of the schemes exhibit an overall, slight tendency towards decreasing browse time. However, the variance in the data is much too high to make any claims about relative differences in learning effects or overall time taken. Interestingly, near the end, all of the schemes spike in time taken. We are not sure why this takes place, especially for all of the schemes. It is not the case that the later tasks consist of more failed tasks; failures can occur at any task ordinal value, and we would expect more failures near the *beginning*, due again to learning effects. We can only explain the datum as some sort of experimental fatigue. We plan on eliminating much of the tedium and time-taken in future browse experiments and revisions of this system, which will hopefully eliminate this anomalous effect. In conclusion, this trial does at least tell us that all schemes are somewhat learnable. In other words, they are all consistent and predictable to some degree. This conclusion agrees with the results of the first run of this experiment. In addition, we can learn something useful from the aforementioned “fatigue effect": if it is taking too long to use a browse system to fulfill an information need, the user's ability to browse effectively may drop precipitously, amplifying flaws in the browse scheme. Designers of digital library services may do well to interpret this as a reason to provide “escape hatches" to other methods of meeting the information need.

Clickstream Volume

Another interesting way to look at the economy of information retrieval through a browse scheme is to look at the number of clicks it took to find a resource (or give up). In Figure 15, we give a chart of the average number of browse clicks per task. For each scheme, averages are shown for all tasks, completed tasks, and uncompleted (failed) tasks, allowing some insight to clickstream volume differences for successful and failure browsing. In this measure, all of the schemes run pretty close to each other. There is a slight ordering corresponding to an ascending ordering in the number of categories in each of the schemes. This is what one would expect statistically; more categories mean proportionally more opportunities for missteps and semantic confusion. Notably, while subcollection clusters comes in last, it does not require many more clicks than the flat schemes, on average. This suggests that little additional navigation work is generated by adding the second level of the scheme. Combined with the time-elapsed results for each scheme, there is strong evidence for the intuitive claim that more clicks are not necessarily a bad thing, and sometimes a little bit more is a good thing. We might conclude from this that, if missteps are not expensive, they are a normal process of browsing and should be expected. Multi-level schemes will incur additional clicks for each level, but overall may be the most efficient time-wise, due to their ability to facilitate elimination of unrelated resources in general-to-specific stages.

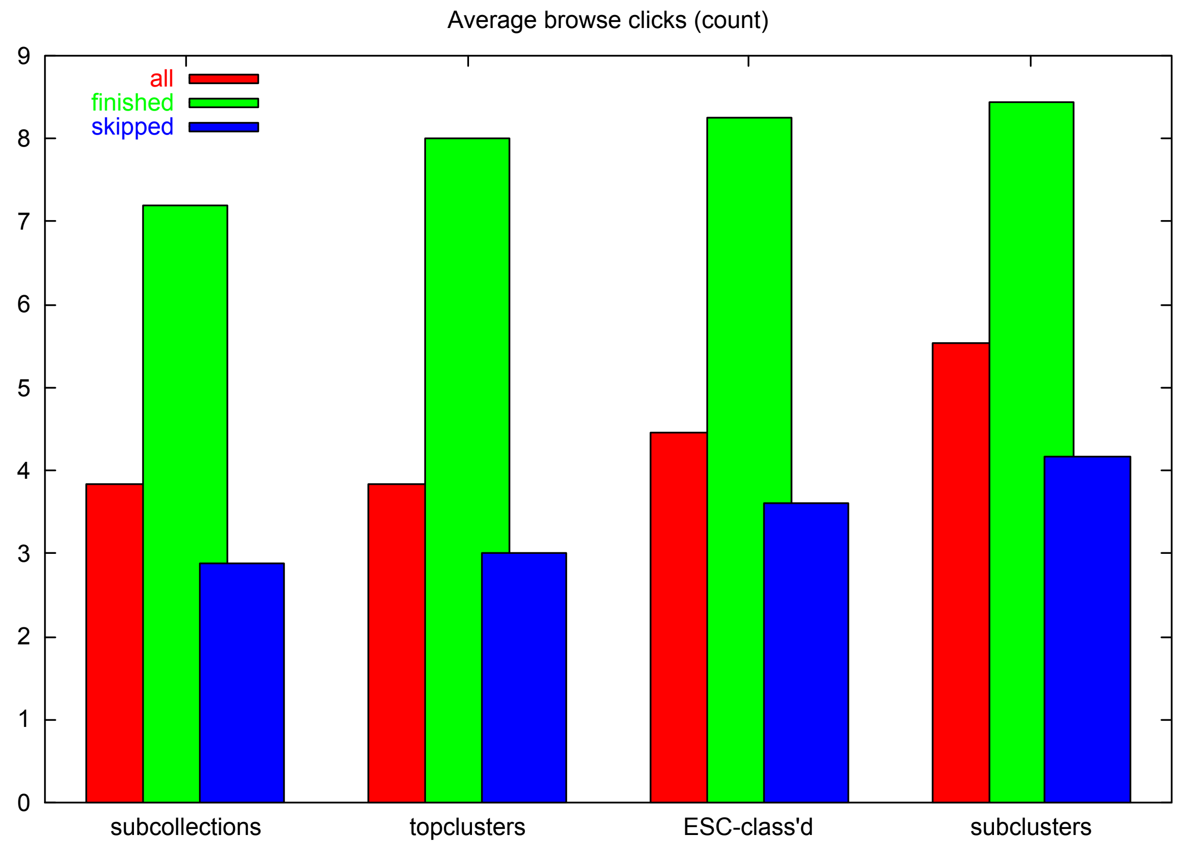


Figure 15 : Average number of navigation clicks to conclude browse tasks.

Clickstream Reasons

Due to time constraints, we were not able to have our scholarly participants undertake the second part of the experiment. Thus, we lack clickstream reason feedback for this run of the experiment, and are unable to give summary statistics on explanations for browse clicks. In addition, we cannot infer a classification accuracy metric for the schemes. In future re-runs of the experiment, we will be able to update these statistics.

Free Response

At the end of each scheme, users were given a chance to enter in a free-form response, giving their impressions (both good and bad) of how easy the scheme was to use. Below we provide a selection of some of these comments, both negative and positive, to give an idea of the pros and cons of the various schemes. We have not done a count of negative and positive comments as we did in the first iteration of this study, as the data set is simply too small to be meaningful. From this, it is clear that the ESC-classification scheme was most-loved as a browse ontology; it contained both the highest number of positive comments and the lowest number of negative comments. The full ordering of browse schemes based on this metric, from best to least, is ESC-classified, subcollections, top-level clusters, and subcollection clusters. It is quite notable that subcollections does so well in this sense, better than the two clustering-based schemes.

Subcollections

Comments for this scheme:

- “Again, finding the correct collection is key. The more you know about the collection, the more likely you are to find it. For example, one item in this scheme was a Civil War document. I knew that the Valley of the Shadow is a Civil War collection but not all users would know that."

- “Finding the right category is extremely difficult. when the item is only the most general of titles (and is in French), it is very difficult. when I know the contents of a site such as Valley of the Shadow, that makes it much easier."

  Clearly, figuring out what the subcollections mean was critical to retrieval through this scheme. Depending on whether the users were very familiar with the subcollections or not, usage of the scheme was either simple or very difficult.

Top-Level Clusters

Comments for this scheme:

- “In this group, each item was findable because a word from the item also appeared on the browser list."

- “Terrible browse categories-e.g., civil rights activist from Montgomery, AL, involved mostly at local and state level listed under "Washington, D.C." apparently because attended March on Wash. And, civil rights organization in AL (1961-1995) listed under "Civil War, civil rights aspects." No categories clearly on literature and fiction, but some entries on this. Never did find papers of former Congressman even though there are multiple "government" and "congressional" categories and I think I browsed them all. "

Here we see the double-edged nature of content-based categorization. While NMF can approximate an understanding of high-level semantics by modelling word combinations, humans still work with much higher-level semantics. Thus, one can most reliably anticipate placement generated by the NMF clustering system by looking for matching words. When common words aren't present (or don't narrow down to a cluster very well) and very high level semantics are relied upon, the algorithm can do quite poorly.

ESC-Classified Ontology

Comments for this scheme:

- “History is a huge category."

- “Subject categories lack geographic refs but entries often emphasize them. Entries lack content parallel to subject categories-e.g., last entry was a letter re: writing to FBI but without other info on content."

The comments here refer to two issues with the scheme. The first is unique to ESC, in that an *a priori* ontology is being used to organize an ad hoc collection. The mapping of our content into the ESC ontology produces an unbalanced organization; a disproportionately large fraction of the records end up in the ESC “History" category. Users quickly learn to check this category more frequently, but once in it, it is tedious to sift through. The other comment refers to a common problem with all of our AI-based schemes: any text within the original metadata is considered during the organizational process, even when that metadata doesn't describe the subject matter. Some of these effects can be reduced by better stoplisting and text cleaning during preprocessing; however, the best fix would be to separate topical descriptions from other metadata fields at the time of encoding/cataloging.

Subcollection Clusters

Comments for this scheme:

- “Helpful that clicking on a category revealed subcategory. Still not helpful to have close resemblance such as "home life" and family. On one search list the word "Biographies appeared 3 times - different items in each. Must avoid that repetition."

- “I had to give up on two searches, as the identified collections the items might have been in were too numerous to search without wearying."

- Here we see that it is useful to be able to narrow down using the two-level hierarchy. However, the ambiguity sometimes present in the generated subclusters is still very noticeable. In addition, when the user cannot figure out which top-level category to start with, the burden of searching through a number of subclusters is overwhelming and causes abandonment of the task.

2.3.3 Discussion

This second evaluation of our browse schemes provided even stronger support for the usefulness of the AI-based schemes. This time, the trivial scheme is generally beaten by at least one AI scheme. These results are especially improtant to consider, seeing as our subjects were of the important user population of domain experts. The commentary provided by the domain experts was very revealing, and highlighted the shortcomings of each scheme. Even though the AI schemes were generally very efficient overall, there were still problems in individual cases due to underlying disadvantages of the methods. We will seek to remedy these shortcomings of the AI-based schemes in the second half of this project.

2.3.4 Conclusion

This retrial of the digital library browse experiment provided us with a bit of a surprise. In this case, the results were more unambiguously in favor of our semantic clustering-based schemes. While this may not prove the schemes are objectively better, it does clearly show that one client community (domain experts) tends to do better with them. Our prior claim that the user characteristics are a factor in the efficacy of the browse system still holds, and now has even more supporting evidence. Thus, we can confidently say that automatically-derived schemes have a bright future and hold great potential in digital libraries, though they will likely be most effective for the most users if many alternatives are given.

2.4 Experiment A2: AmericanSouth Web Clustering

[Section author: Aaron Krowne.]

2.4.1 Introduction

The goal of the web clustering portion of MetaCombine is to acquire and make browsable a collection of web resources related to the digtial library's subject domain. This goal has three components:

- Acquiring the web resources: Though we already have a scholar-vetted set of weblinks, there is much more content out there on the web which should be part of any subject portal on the American South. However, we need to acquire this vast array of resources in as automated a fashion as possible.

- Metatagging the web resources to produce metadata: The web resources cannot be a part of a digital library and be handled by digital library services without metadata.

- Making the web resources browsable: A browse ontology must be provided for structured, intuitive browsing of the collection.

How these objectives are being met by the project's efforts is discussed in more detail below.

2.4.2 Resource Discovery

The resource discovery process entails sifting through web pages to find pages which are relevant to the subject domain—the history and culture of the American South. This is a massive task, given the billions of web pages in existence. An approach which extracts a small, domain-specific subset of the web to deal with this problem has already been formulated. This approach is called "focused crawling". The concept is a modification of the standard web crawler (spider) concept. The key difference is that, rather than visiting all links from each page blindly, the crawler only visits links to pages that are classified as "relevant" by a text classifier. The end result is a selective crawl of the web, hopefully "focused" on the subset of web pages specific to the subject domain.

In section 3.1.1, we go into detail on our efforts with focused crawling. We discuss how we implemented it, what the challenges were, and what the quality of the latest output is. For this section, it suffices to say that we have successfully acquired a web collection via focused crawling, as of this writing, on the order of 10,000 items. We hope to expand this in the future, up to the "natural" size of our subject domain on the web, as we improve the focused crawling software.

2.4.3 Metatagging

A focused crawler's output is essentially just a list of URLs. However, we require more than this in a digital library: we need metadata. Metatagging is the process of producing metadata tags for a bare object like a URL (web page). At this stage we use a very simple procedure for generating metadata through metatagging:

- download page contents (HTML).
- store URL in metadata as "identifier".
- check for <title> tag in page. If present and nonempty, store title in metadata as "title".
- extract the plain text from the body of the page (keep link anchors and other visible text, discard all other HTML).
- grab the first 256 characters of the plain text (subject to word boundaries). store this in the "description" field of the metadata.
- if the "title" field is still blank, store the first 64 characters of the "description" field as the title.

This is a very simple-minded, intuitive approach to generating some basic metadata tags (we also note that classification produces a "subject" field or equivalent). The results are quite good, and certainly workable for our later manipulation of the records.

We could potentially populate some other fields. For example, date could be parsed out from the document's "last modified" date. Author could be parsed out as the first (or last) email address, and nearby name.

In the future (as time permits), we would like to experiment with more advanced forms of metatagging, employing their own machine learning, as in [Han, 2003]. However, we are skeptical of the applicability of this approach for our problem, seeing as it has so far only been applied to more structured document types (research papers).

2.4.4 Making the Web Collection Browsable

To enable browsing of the web collection, we employed both the NMF clustering algorithm to generate a cluster hierarchy and a set of flat clusters, and the BOW classifier to place the web resources within the *Encyclopedia of Southern Culture* (*ESC*) subject hierarchy. We discuss both our clustering and classification efforts below.

2.4.5 Hierarchical Clustering

For clustering, the cluster hierarchy can be interpreted as a class hierarchy and browse scheme. This enables either text-based, directory-style browsing of the resources, or visual browsing (discussed in detail in the section on A3). Either way, the structure of the hierarchy must be generated based on the underlying content.

As mentioned earlier, we employed NMF for clustering on the A1 browsing portion of the project. However, our NMF clustering system only produced flat clusters at the time of the first and second evaluations of the output. By the time of our first experiments in clustering for A2 (and A4), we had developed hierarchical clustering based on NMF. Further details on our development of NMF hierarchical clustering appear in a later section. For comparison, we still included a flat cluster scheme for this content set.

On the MetaCombine web site, we have accessible an A2 demo with a sample class hierarchy for A2 produced by the hierarchical NMF clustering algorithm, which allows browsing of the web collection via this hierarchy (or more recent version of it) and others. From this interface, the web resources can be viewed as metadata, or as the original web pages.

We note that we formed document vectors from the extracted plaintext of the web pages for the web resources and did not rely solely on the metadata produced by our metatagging. This makes sense, as the metadata produced through metatagging is simply a subset of the full text of the web page anyway.

2.4.6 Classification

As discussed in A1, we interpreted the sections of the *Encyclopedia of Southern Culture* as subject categories, resulting in a flat classification scheme. We loaded this scheme, and used the full text content of the encyclopedia articles as a training set with the BOW classifier. We ran BOW as a query server, in on-line linear, tfidf/Rochio mode, and sent document vectors over a socket to be classified. We captured the categories returned, and applied basic linear multiple thresholding with a factor of .90 to multiclassify each item, storing the categories with weights and identifiers in an output file. A separate program read this classifier output and generated classifications in our database, for use in browsing and testing.

The only deviation from what might be expected in this process was that we had to limit the size of the document vectors to 32k. The reason was that, for larger inputs, sometimes BOW crashes. We feel this is not a major limitation, especially when one considers that even Google only reads the first 8k of content of web pages.

2.4.7 Evaluation

The design for evaluation of A2 is almost identical to that of A1; we simply have users browse for items from the collection through the generated scheme, and record both the hard data of their browsing and their informal reactions. At this point, we have not executed the evaluation. However, we have built the web-based evaluation system, besides a few modifications intended to make the experiment go much faster for participants. We expect to complete the evaluation in the next couple of months.

2.5 Experiment A3: Multidimensional Visualization

[Section author: Stephen Ingram.]

2.5.1 Introduction

A major difficulty of exploring digital libraries is understanding the nature of their content. What kinds of resources exist in a library? How do they all relate to each other? Because of the serial nature of language, text is a difficult medium to express answers to these questions. A list of text may require many tedious readings to convey these kinds of relationships. Worse yet, the reader may come away with an incomplete or even wrong idea of a library's content. Using pictures is a much more natural method of expressing this complex information because the human visual system can more rapidly deconstruct images than read lists. A short glance at an image tells a viewer what visual elements are in an image and the spatial relationships between them. More briefly, "a picture is worth a thousand words."

The goal of the A3 experiment or Multidimensional Visualization (hereafter **MDV**) is to graphically explore ways of conveying useful information about digital library content to a user without confusing or overwhelming them. It seeks to present a user with an interactive graph of numeric and textual information gleaned from semantic clustering experiments. We experiment with several kinds of graphs, each type conveying a different kind of information about the digital library collection.

Ultimately, the graphs will provide users with a system for efficient navigation through a document corpus. By graphically moving through a hierarchy of clusters, casual users will browse through large collections of documents, understanding the arrangement of records at multiple levels of specificity. It is important that the browse interface be simple and natural to use for potential researchers with no technical background. Through user feedback and focus groups, the system will continue to be tuned to researchers' needs and expectations.

2.5.2 MDV Graph Elements

A typical graph is composed of different elements that convey information about the content of the underlying data. Each graph element represents some kind of data object and the appearance of the object represents its attributes. Element relationships can be represented by placing the elements in conjunction with each other. For example, in a typical bar graph the bar elements represent quantities, while the color, size, and labeling of the bars represent attributes like identity, quantity, and name respectively. Placing the bars next to each other conveys relationships between the quantities such as which quantity is greater or lesser.

The following is a list of objects, their attributes, and relationships among objects that we are interested in conveying:

OBJECTS

- **Collection** – This is the highest-level object in the data representing an entire corpus of records. Currently, it is implied in MDV that you are browsing within a given collection. If browsing extends to multiple collections, then it becomes important to include collection information in the graph. In this case, multiple options exist. Collections can become top tiers in a hierarchical browsing scheme, an object attribute represented by label or color, or a navigation option on the graph where a user selects from a list of collections through which to browse.

- **Clusters** – Clusters represent a group of records in the collection that share semantic features. They can be represented as objects themselves or as attributes of records. When representing the clusters as objects we've considered two ways of displaying them: bubbles and clouds. Bubbles are just circles whose center is the cluster centroid. Clouds try to depict clusters as a set of distributed points. When displaying clusters as objects it is important to consider "un-cluttering" or differentiating schemes so they remain distinguishable. As record attributes, clusters can be displayed by adding some color or shape to a record representation.

- **Records** – It is important that MDV users have some access to information about records. This can be performed by adding records to the graph as objects or providing access to a listing of records represented by a cluster. Due to the large number of records in a typical document corpus, difficulty arises for each of these respective approaches. Large numbers of objects on a graph create clutter and long lists of records can cause reader disinterest: this can be solved by constraining the set of records viewed as objects or in a list with a hierarchical browsing scheme.

ATTRIBUTES

- **Size** – We are interested in the size of a cluster object, meaning the number of records within it. As bubbles we can control the radius of a bubble drawn. Because clouds plot the actual points of records, size is implicit in its representation. We can also provide access to a cluster information page containing specific details on the number of records in a cluster.

- **Coherence** – This is a measure of the average of how strongly records match a given cluster. A typical method for displaying intensity is to vary an object's opacity. We can vary the overall opacity of a cluster or individual records.

- **Labels** – A picture alone doesn't allow a reader to understand what a graph's objects represent. Text labels help inform readers about the content of individual clusters. To retain readability, it is important to avoid overlapping object labels. We use a technique called maximum independent rectangle set [Agarwal, 1998] to avoid rectangle overlap.

- **Metadata** – Each record has some corresponding metadata text. Typically, this information is too large to place in a label. Therefore we do not place this information on the graph. Instead we can give a user access to it on a separate webpage.

- **Hierarchy** – While all clusters represent a collection of records, some clusters contain sub-clusters of records. It is helpful to a user to be able to know which do and don't. We manage this by changing the appearance of a cluster object, such as adding a plus sign.

RELATIONSHIPS

- **Distance** – In the clustering experiments, clusters and records are represented by vectors. The relationship between records in this mathematical space is roughly analogous to a semantic relationship. Because of this we can use spatial distance to depict semantic "distance." How close we place objects on the graph gives a picture of this distance. Color can also be used to depict a third spatial dimension. It is important that graph users realize that the spatial distance doesn't truly correspond to anything particularly meaningful but is simply a rough measure of relationships between semantic objects.

- **Membership/Overlap** – Semantic graph objects exhibit member relationships. For example, a sub-cluster or record is a member of a cluster. Presenting objects as part of a navigable hierarchy disambiguates most of these member relationships. However, when viewing a given tier it is important to actually visualize these relationships especially the situation of an object's dual-membership.

An important factor in information visualization is not to overwhelm a viewer with too many features on the graph. An object with too many varying attributes can be too confusing to be helpful. It is best to show only a few attributes at a time and allow a user to choose between which attributes they want visualized. As an example, color was mentioned several times in the above listing. A user could choose which attribute color represents from a list, keeping the graph from becoming a pastiche of cryptic symbols.

Software Design

There is an abundance of information visualization (**infovis**) software, free and commercial, available for use in scientific projects. Typically, each library is tailored for a specific visualization task that depends on the nature of the underlying information and the type of visualization you would like to produce. For example, the Inxight Star Tree expects tree-structured graph data and renders a hyperbolic tree with which a user can interact. Because the nature of clustered data lends itself to node-and-edge representation (also called graphs), a visualization library tailored for work with graph data would be the wisest base upon which to build the MDV component.

Among free, open-source software projects dedicated to the visualization of graph data, we chose the prefuse toolkit. Prefuse is a powerful set of open-source java libraries designed to make the development of graph-dependent infovis applications as rapid as possible. It comes pre-packaged with a variety of graph layout and animation routines, and because prefuse is java-based, it is easy to write external modules that plug in to the prefuse toolkit.

For more information, the authors of prefuse have compiled a paper detailing the features and design of the toolkit [Heer, 2004].

Dimensionality Reduction

In a mathematical sense, semantic clusters are points in extremely high-dimensional space. The dimensions in this space correspond to text features like word stems and noun phrases. Because of the need to represent all the text features in a document corpus, the space can contain over 30,000 dimensions. To display the clusters on the screen, they must be transformed into a low-dimensional representation. In this process some information must be lost. Dimensionality reduction seeks to minimize lost information and preserve the spatial relationships of the data as much as possible. It is an active field of scientific research and there are a wide variety of techniques to try. We have explored the following techniques:

- **PCA** (Principal Components Analysis) – See Figure A3.1. A classic dimensionality reduction technique.
- **NMF** (Nonnegative Matrix Factorization) – See Figure A3.2. A modified form of the cluster-finding algorithm [Tenenbaum, 2000]. We compile the centroids of semantic clusters into a matrix $\mathbf{X}$ similar to the document matrix used to do the initial clustering (see section 2.2). We then perform NMF on $\mathbf{X}$, specifying that the number of clusters equal the number of dimensions we want to visualize. This gives us $\mathbf{UV}$ as a factorization of $\mathbf{X}$. We then project $\mathbf{X}$ on to $\mathbf{U}$ (more precisely $\mathbf{U}^{t}\mathbf{X}$) to get a matrix $\mathbf{P}$. The columns of $\mathbf{P}$ contain the clusters projected into a lower dimensionality.
- **Euclidean tree** – See figure A3.3. An interactive dimensionality reduction technique where a graph tree is constructed from a user-specified root node. Rather than presenting a picture of the global structure of the collection, this technique arranges the clusters while only considering their relationship to the root node.
- **Isomap** – A dimensionality reduction technique that searches for nonlinear manifolds in the data [Roweis, 2000]. A graph is constructed by placing edges between all nodes' k-nearest neighbors. Then, a distance matrix is constructed between nodes by using Dijkstra's shortest-path in the constructed graph to compute distance instead of true linear distance. Finally, Multidimensional Scaling is performed on to compute the final position of the nodes.
- **LLE** (Local Linear Embedding) – Similar to isomap [Heer, 2004]. Uses a different edge-selection technique to construct the graph.
- **Physical Simulation** – a graph layout based on n-body physics.

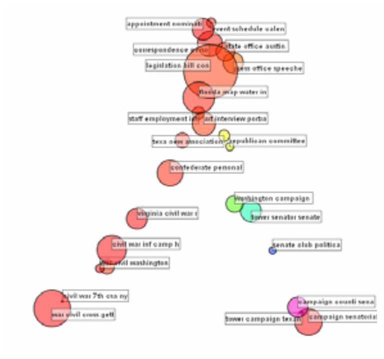


Figure A3.1: PCA

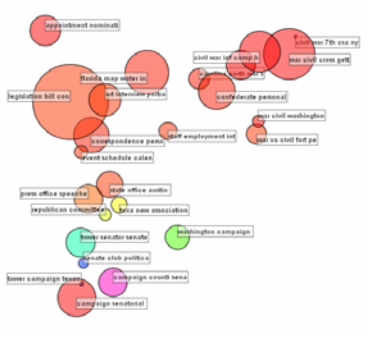


Figure A3.2: NMF

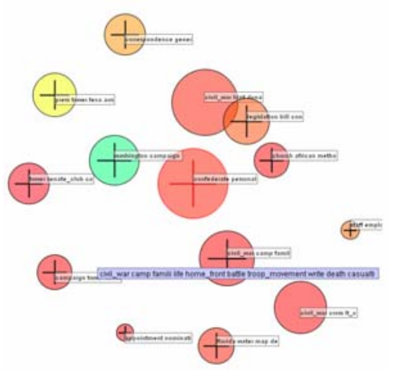


Figure A3.3: Euclidean Tree

It is important to remember that the precise layout of the clusters in any of the above graphs has no exact meaning. Likewise, the placement of a single cluster in the plane by any the above techniques is meaningless. The importance is the placement of the clusters as a whole.

Input Format

The MDV reads semantic cluster information in an XML file format (see abbreviated example in listing A3.1). The XML file is output from the NMF clustering system, and contains enough information to perform dimensionality reduction on the clusters. It also contains semantic information about the clusters with which MDV can label the clusters. The format is still being developed to suit the needs of the project.

```
<clustering infile="amsouth" clusters="14">
    <cluster id="0">
    <centroid>0.0002 0.0000 … </centroid>
    <words significant="182" threshold="0.01">
                    <word rank="0.719826" shade="47">confederate</word>
                    <word rank="0.286621" shade="b5">personal</word>
                    <word rank="0.280671" shade="b7">civil_war</word>
…
    </words>
    <docs total="7889" showing="20">
            <doc id="14923" rank="0.732061" ord="1">
                            <identifier>oai:docsouth.unc.edu:212</identifier>
                            <title>The  Experience  of  Thomas  H.  Jones,  who  was  a
Slave for Forty-Three Years</title>
...
    </cluster>
...
</clustering>
```

Listing A3.1: Abbreviated example of XML input file

Overview Page and Prototypes

There is a requirements-based overview page (www.metacombine.org/mdv) detailing the individual features of the MDV project. Each individual feature of the project is available for review. From the overview page, a user has access to several prototypes. These include:

- Scatter Plot – traditional plotting of points on a Cartesian plane.
- Radial – interactive layout based on graph trees.
- Physics – interactive layout based on n-body physics.
- 3-D scatter – plotting of points in a Cartesian volume.
- Image-maps – Prefuse is a java-based library, however some internet browsers are not equipped with the java plugin. A static, image-map based version of the 2D prototypes is provided for these users.

MDV Evaluation

An important part of the development of MDV is tuning its performance based on user experiments with focus groups and user evaluation. There are several main reasons to do this. First, developers of a software system become acclimated to their software's features and cannot see areas where their target audience may become confused. Secondly, experts in a domain different from the software developer, such as history or culture, are likely to suggest features or behavior the developer would be unable to conceive. Lastly, it is important from a diagnostic perspective to compare the performance of MDV with other schemes to determine if user browsing experience has improved at all. More concisely, these evaluations correct, improve, and test the efficacy of the software.

We plan to do two kinds of evaluations. The first is similar to the experiments in section 2.2. These experiments examine a test subject's click-stream, evaluating the choices a user makes when navigating through the hierarchy and how they felt about said choices. The second evaluations are focus groups with experts. In these the developers walk non-technical experts through the features of the system and collect their comments and evaluate their reactions and conclusions regarding the software.

We have already performed a focus-group evaluation of the software. In it we walked several liberal-arts oriented experts including a library scientist and a historian through a software demo. We learned about the type of explanatory information necessary for non-experts to properly understand the graph. For example it was important to gently introduce the idea of what clusters are and how the classification experiments use text features to derive clusters. We also learned how non-experts can become confused about the exact location of items on a graph, overestimating the importance of cluster placement in a plot. Knowledge like this helps reduce the learning curve of the software and ensure that experts have all the necessary information to properly interpret the visualization.

Conclusion and Planned Work

Using a variety of methods, MDV reduces the dimension of semantic clusters and provides its users with a variety of interactive and visual services for a digital library collection. MDV is written on top of the infovis toolkit prefuse, and reads graph data XML-based file format. Through evaluations we will iteratively improve the efficacy of the software as a research and navigation tool.

MDV is categorized as *in mid-development*. We are currently working on the following tasks:

- **MDV as browsing tool** (as opposed to text hierarchy) – As previously mentioned, MDV provides a powerful method for conveying information about a document collection. We will therefore experiment with providing MDV-based browsing services through a collection. Ultimately, MDV will allow a user to navigate from clusters down to actual documents.

- **Integrate MDV with web site** – MDV now resides in its own demonstration web-space. Part of making MDV a browse tool involves integrating it with a website. This involves simplifying its interface and ensuring it works with all browser-types.

- **Make visualizations hierarchical** – Currently MDV only visualizes a single tier of semantic cluster data. A preliminary version exists that allows navigation of the results of hierarchical clustering experiments. We will work to incorporate tiered browsing into all the features of MDV.

- **Continue with labeling/summarizing work** – MDV now labels a cluster with the ten most relevant text features of that cluster. Work must be done to make cluster labels more readable and informative. We will experiment with automatic and manual labeling methods.

- **Investigate use of MDV as part of iterative clustering process w/experts** – Besides providing user-services, MDV provides researchers with a picture of the efficacy of clustering experiments. We plan integrating MDV with clustering research. By giving quick visual feedback about the results of a given clustering experiment, researchers may use MDV-generated graphs to tune parameters and iteratively improve the quality of results.

2.6 Experiment A4: AmericanSouth Combined Clustering

[Section author: Aaron Krowne.]

The goal of the combined clustering phase of the project is to determine how well clustering and the subsequent browse system work when performed over two very different kinds of resources (web pages and native DL). Structurally, there is very little that is different in the development and evaluation of this phase of the project, as compared to A1 and A2 (DL and web clustering).

Basically, we took the individual databases from the DL harvest and the cached web crawl results (with generated metadata) and combined them into a single database. Web-derived resources were "special," as they had what amounts to a full-text field which was called "indextext" in our database and was derived from the HTML-stripped version of the whole web page. Clustering and classification vectors were derived from title + subject + description for the DL resources, and indextext for the web resources. These vectors were unified and sent to the clustering and classification systems.

A sample cluster hierarchy for the combined collection, as well as a flat cluster scheme and the ESC-classified scheme are accessible from the MetaCombine browse demo page (this is the same system as for A1 and A2). Discussion of the quality of the hierarchical clustered schemes takes place in the section on NMF hierarchical clustering.

Once again, the same evaluation system built for A2 will be re-used for testing A4. We plan to undertake this testing shortly.

Section 3: Experiments with Combined OAI / Web Search

This portion of the report describes the various project development efforts, experiments, and findings to date concerned with combined OAI and Web searhing (Part B of the MetaCombine Project).

3.1 Enabling Research on Focused Crawling

[Section author: Aaron Krowne.]

We conducted an evaluation of focused crawling, which we deployed to autonomously discover resources to augment to our collection of records pertaining to the American South. We here discuss the accuracy of the resulting collection and compare with other results reported in the literature. In addition, we discuss various techniques we used to improve the crawl efficiency and describe the process of deploying a focused crawler from scratch. We find that focused crawlers are very effective in autonomously discovering web resources and building domain specific web collections, but they are lacking in deployability. Keeping in mind the rate at which the web is growing, we feel there is great potential for digital libraries to make widespread use of focused crawlers to augment their collections and serve as more complete subject portals.

3.1.1 Introduction

In the past few years, library and digital library professionals have taken note of the trend that much scholarly research begins with a web search. This community naturally desires to stay relevant, focusing on providing real value over the web in the form of aggregation, organization, facilitation, and vetting of intellectual content, in a way the web does not even pretend to do. Thus, the digital library community has sought to somehow make use of the wealth of scholarly resources available on the web, bringing them into the fold of digital libraries and adding value through digital library services. However, this desire collides with the continual growth in documents on the web and the scale of the web relative to the size of a typical digital library project.

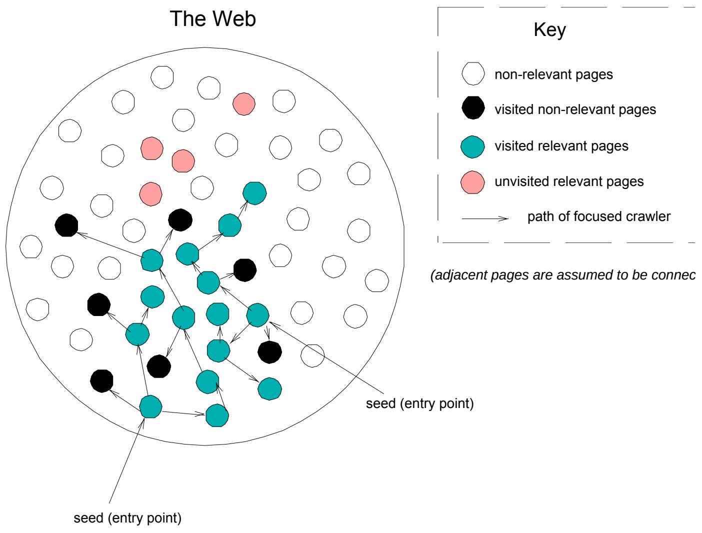


Figure 16 : The benefit of focused crawling.

Focused crawling has emerged as a method to deal with this problematic situation [Chakrabarti, 1999]. The goal of focused crawling is to extract from the web just the subset of web pages pertaining to some specific, relatively-narrow subject domain. Focused crawling solves the scalability problems of both space (elimination of the need to store the whole web) and computation (elimination of the need to analyze the whole web's content). The end result is that most attention is paid to the relevant subset of the web, which can then be included in the digital library (either as a focused web search, or as fully-qualified digital library records via meta-tagging).

Focused crawlers are still in their formative stages, and many researchers are working to improve their efficiency. There are several ways to increase the efficiency of a focused crawler, which we will discuss briefly below. Then, we will describe the experiment we conducted. Finally, we will present our results and discussion.

3.1.2 Background

To understand the effectiveness of focused crawlers, we need to understand the difference between a focused crawler and an ordinary, so-called "unfocused" crawler. But what is a crawler?

A *crawler*, a.k.a. spider (bots, robot, wanderer, intelligent agent...), is a program which scans through the whole web or some narrow set of web hosts and collects data to assist in creating indexes for search engine. A crawler starts with a *seed set* of URLs (or a single starting point) and follows the links contained inside web pages it scans. A crawler in the web-wide, unrestricted sense (such as those used by the big web search engines) traverses every page connected to the seed set before terminating. This approach serves the purpose of indexing the whole web just fine, but if one is trying to build a targeted, domain-specific collection, then it will not scale. If the crawler follows the links blindly, irrespective of the fact that a link could lead to a document not relevant to the topic of concern, most of the pages retrieved will be irrelevant. Thus, most space, time, and computation will be wasted on the task, even if some filtering is applied to the initially retrieved collection.

One way of dealing with this is to simply narrow the crawl to hosts linked to in the seed set. This has a number of problems: the first and most serious of which is that this ignores potentially valuable content elsewhere on the web (the seed set is only supposed to consist of *starting* points). Another problem is that it is not necessarily the case that all pages on a domain will be relevant just because one page (or even the home page) is.

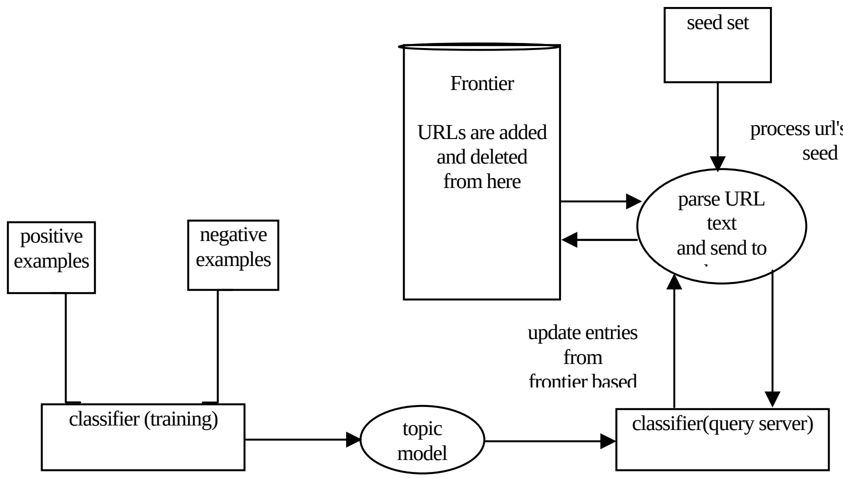


Figure 17 : Block diagram for a focused crawler.

This is where focused crawling comes to the rescue. A focused crawler, in the process of crawling, will only follow the links in a page if the page is judged to be a relevant document. Otherwise, it will discard the page and its links. Thus, it will filter out the portions of web not relevant to topic of crawl, not only not avoiding downloading them, but also not even visiting wide swaths of the web beyond the boundaries of the subject domain. Hence the name "focused crawler".

In Figure 16, we illustrate the benefit of focused crawling. By focusing, the crawler maximizes the visitation of relevant pages (blue nodes) while minimizing the visitation of non-relevant pages (black nodes). Most of the non-relevant web is ignored (white nodes). In addition, it is desirable to minimize the number of relevant pages which are missed (pink nodes). This is chiefly done by intelligent and thorough seeding.

In the next section we briefly describe various components we used to perform our experiment.

3.1.3 Building Blocks

3.1.3.1 Nalanda iVia Focused Crawler

Nalanda-iVia Focused Crawler (NIFC) is a program by Dr. Soumen Chakrabarti (Indian Institute of Technology, Bombay) and developed with Roger Menezes (IIT Bombay) with the support of IIT Bombay, the INFOMINE Team and the U.S. Institute of Museum and Library Services. This software can be downloaded from the site http://infomine.ucr.edu/iVia/ .

The focused crawler starts with a seed set which is a list of highly domain-specific URLs. This list can be hand crafted or automatically generated (we discuss some methods below) and its purpose is to bootstrap the crawler.

As the crawling progresses, an inter-linkage graph is developed of which resources link to one another (i.e., cite and co-cite). Good resources focused around a common topic often cite one another. Highly inter-linked resources are evaluated, differentiated and rated as to the degree to which they are linked to/from, as well as for their capacities as authoritative resources (e.g., a primary resource such as a database which receives many in-links to it from other resources) or hubs (e.g., secondary sources such as virtual library collections which provide out-links to other, authoritative resources). After such assessments have been made, a second automated process is then put into play which rates resources, as a second indirect measure of quality, by comparing for similarity of content (e.g., similarities in keywords and vocabulary) between the potential new resources and resources already in the collection. The most linked to/from authorities and hubs, with terminology most similar to that in other high-quality collections, thus become prime candidates for either adding to the collection as automatically created records or for targeted expert vetting.

We modified NIFC slightly to suit our purposes. The training set we used initially was separated into 24 subcategories pertaining to the American South (the *Encyclopedia of Southern Culture* subject headings). As NIFC is designed to crawl the web for a specific topic or category, we modified it so that it took into consideration of all the sub-categories we have. Our motivation was that we do not believe the subcategories in our subject domain are well-defined enough to do a good job focusing a crawl on their own. Thus, instead of selectively taking documents pertaining to one category, our modified crawler looked for documents matching any of the 24 sub-categories we have in our training set.

But later, when our training set grew, we merged all the documents into one category for simplicity (however the crawler still contains our modifications to be able to use many subcategories). We did this to strip the task down to the “relevant” and “non-relevant” set basics which are the minimum needed to identify a domain-specific collection.

The other modification is that we are not using the second automated process as described above. Instead, we filter the documents in the first process based on a classification threshold value. In subsequent sections, any reference to NIFC will refer to our modified version of it.

3.1.3.2 BOW

BOW is a text mining and retrieval library produced at Carnegie Mellon. It consists of C libraries and a number of programs to use the algorithms in the libraries. One of these programs, rainbow, is a classifier. Rainbow supports all of the algorithms in BOW: naive Bayes, SVM, on-line linear, and many more. Rainbow has a convenient interface: the training set is read in as directories (one per category) containing text files which serve as examples for those categories. Rainbow reads in this directory and builds a statistical model of the corpus, which is stored on disk. This model is used by all classifiers. NIFC works in conjunction with the rainbow classifier. Before the crawling begins, rainbow is set up as a query server on a specified port. Once crawling starts, NIFC then sends document text over a socket to rainbow. For each document, rainbow returns a sorted, weighted list of categories to NIFC.

In the next section we describe the basic preparation which we undertook to set up the dependencies for our experiment.

3.1.4 Preparation

Before conducting our focused crawl accuracy evaluation experiment, we had to build the web collection for our topic (history and culture of the American South) using the focused crawler. Below, we relate the process we followed to refine our crawl efficiency and build the web collection.

We started off with downloading and installing the tools (described in above section) to initiate the focused crawl. After that, we set out to train the rainbow classifier. Now, as the classifier's judgment is key to the overall crawl accuracy, we used many techniques together to enhance the training process. These techniques are discussed below.

3.1.4.1 Bootstrapping

We created a set of both positive and negative examples to better train the classifier. Positive examples assist the classifier during the learning process to identify key salient features of the subject domain, while negative examples help it recognize the features that are not correlated (and are instead correlated with other irrelevant topics).

The positive examples we used were derived from following sources:

- We used the articles and chapters from the *Encyclopedia of Southern Culture*, harvested records from AmericanSouth.org collection, and the content of scholarly articles from the SouthernSpaces.org online journal.

- We tapped the documents related to our topic from the online web directory created by the Open Directory Project (ODP). We did this by methodically fetching URLs of the web sites in the ODP under geographic categories for the American South. To do this, we first downloaded the ODP RDF dump and then parsed it and stored all the urls in a database. Then we manually crafted a list of categories from ODP which could potentially fetch us relevant documents. For instance, here is one of the many categories we selected:

  > "Top/Regional/North_America/United_States/Regions/South_and_Southeast/ Society_and_Culture/History"
  >
  > After creating this list, we extracted the URLs corresponding to subcategories and ultimately leaf records from our database. We then extracted their content and produced a set of text files to augment the positive training set.

- We used keyphrases - most of which were generated by domain experts, but also some of which were automatically generated by heuristics. In specific, we combined names of southern states with *Encyclopedia of Southern Culture* headings to get keyphrases like "Georgia Black Life". To use the keyphrases, we sent them to Google as queries and took the first few results for the positive set. To query Google we used the Google API.

The negative examples were derived from following sources:

- Again we used ODP as described above, but this time we extracted URLs from topics **not** belonging to the previously-determined American South-related subset (i.e. the compliment of this set in the ODP). Once we generated this set of all "negative" URLs, we took a random sample from it, which was roughly the same size as our set of positive examples.

- We added some of the negative examples manually based on initial crawl results. This way we also got rid of some of crawler traps which we were encountering while crawling.

Once we obtained the set of positive and negative examples, we trained the classifier on them. We used the default classifier provided by rainbow (naïve Bayes). Once this was done, we set up the classifier as a query server on a specific port.

After this, we embarked on the creation of our seed set. The seed set was initialized with URLs which were manually provided by domain experts. This list of URLs was highly relevant to our crawl topic and contained high-quality items. Much of it overlapped with the "Weblinks" section content from AmericanSouth.org. This initial seed set was augmented automatically by using the URLs from the positive examples from the ODP as well as the results of the keyphrase searches on Google and other search engines (both discussed above). We recommend others adopt a similar combination of all the above techniques to generate seed and training sets.

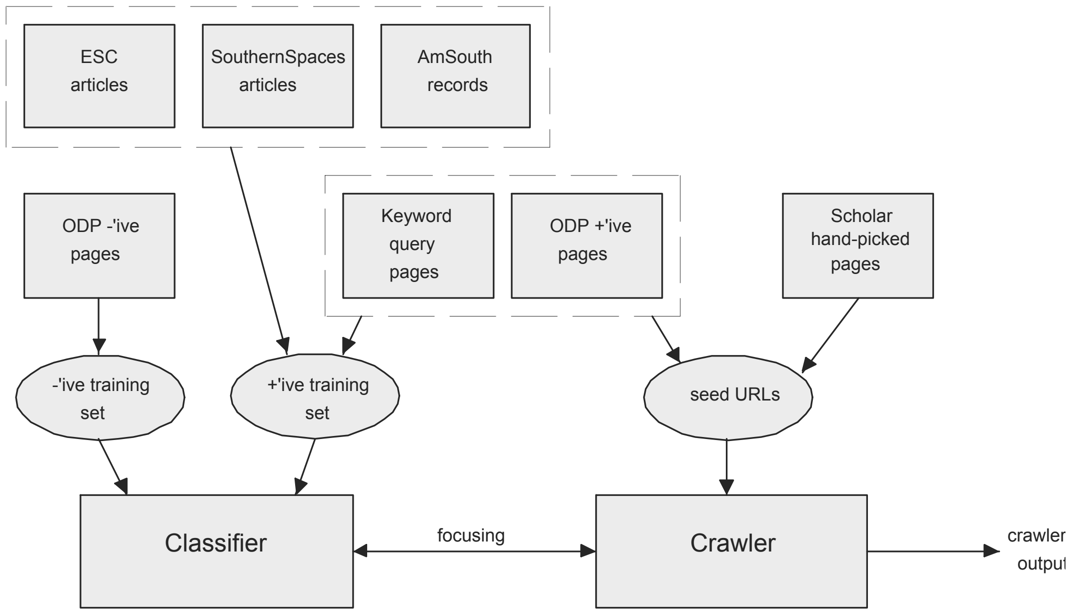


Figure 18 : Bootstrapping the crawler.

In Figure 18, we illustrate our bootstrapping arrangement, including both classifier training and seed set development.

3.1.4.2 Collection Building

Once we had the complete seed, set we ran the crawler and obtained a set of result URLs. We then processed these URLs to obtain the text corresponding to these web pages. After this, we cached the text in a database along with some generated metadata. We created a basic *metatagger* to serve this purpose.

The metatagger first downloads the Google cached HTML page for the URLs. Though Google-cached entries do not always reflect the latest version of document, using Google's cache gave us advantages like faster and more consistent download speed and increased reliability (the page will be available unless Google is down). Once the metatagger gets the HTML page, it strips out the Google cache header and then parses the document to get its title (from the <title> tag). After this, it uses Lynx (a console web client) to get the plain text corresponding to the HTML page[1], and then further parses and cleans the text. Finally, it creates a summary for the text using just the first few lines, and stores all of the metadata in a database.

3.1.5 Evaluation

To evaluate the accuracy of the crawl collection, we needed a search system running over an index of the crawled content. For this we employed Swish-e, an open source (unfocused) crawler and search engine. We created the search system by indexing the crawl URL list with Swish-e. We were able to use Swish-e's

[1]We used Lynx because we found obscure HTML code (such as for scripting) often left in the other plaintext-extraction parsers we tried.

command-line interface to submit arbitrary queries over this index, and parsed the output to capture the results.

Finally we created a web-based front-end for our experiment which, in brief, sends each user query to Google as well as the Swish-e back-end. It then presents the unified, scrambled list of results back to user and then gets their feedback in the form of binary relevance judgments.

In the following sections we describe the experiment we conducted in detail.

3.1.5.1 Experiment

Our goal with the experiment was to determine the *accuracy* of the focused crawl. For focused crawling, "accuracy" means simply "precision," or the fraction of retrieved pages which are actually relevant to the focusing topic.

To evaluate the efficacy of the focused crawl in an objective sense, we needed something to compare it to. At minimum, we think it is important to compare the accuracy of searches in a focused collection with those of the web in general. This is because, if the focused crawl cannot beat the generic web in accuracy, the case for inclusion of web resources in a digital library is very diminished (much of the intended value has failed to be added). We chose Google for the web search engine, due to its quality and the availability of an API.

To guide subjects through the experiment, record results, and organize it in a way which controlled for bias, we wrote an on-line experimental system accessible through the web. The system works as follows. Users log in to the system through a front page by entering their email address. Internally, users are referenced by a function which maps a unique id to every email address. Each user is then asked to enter a query related to the American South subject area. Also at this point, users can browse through AmericanSouth.org to look for inspiration, by clicking a link to a popup window which we provided.

Once the user enters a query, we first feed it to Swish-e locally, which performs a search in our crawl collection. We fetch the top 10 results URLs from Swish-e and cache them. We then feed the same query to Google by using the Google API (we used the Net:Google module of Perl to make API calls), retrieving the top 10 result URLs. We then check if the content and metadata for each URL is already in our cache, otherwise we fetch the page using Google's cache and parse it and store in our database.

Once we have all the results from both search engines, we combine the two lists into a single list of distinct items. We then randomize the list of distinct items, based on a key we generate for every user and query id. This way, we remove any potential bias based on the user inferring how the search engines ranked the results. At the same time, the consistent seeding of the randomization means that if the user's browser crashed or was closed accidentally, the items will be in the same order when they come back. Thus, state is preserved and bias is removed.

Once we have the final results list, we generate some synopsis information about the items, including title and a highlighted summary of the content of the page. To generate a highlighted summary, we wrote a script which takes document text (the visible portions of the HTML page, extracted) and the query (stoplisted) as input. It then looks for query keywords in the document and extracts the portions from text which contain query terms, highlighting the terms in bold wherever they occur. The output looks much like Google's content synopsis, but without making numerous (and limited) calls to the Google API.

The results list is presented to the user with the synopsis data. Every item has a checkbox associated with it. The user is directed to look at the synopsis information associated with each entry and decide whether the result pertains to the subject domain (the history and culture of the American South). Users can also follow a link to the site and browse it directly if they feel that the synopsis information is not sufficient to make a judgment.

When the user has determined the relevance of all of the items, they submit the page and we store their judgments in a database. After this, the user is asked to enter another query, and the whole of the above process is repeated. In all, users have to enter 5 queries and make judgments for each. At most, this requires inputting 5 queries and making 100 relevance judgments. As often there is overlap in the live web search and focused crawl results, significantly fewer relevance judgments are typically required (with 50 being the theoretical minimum).

While the experiment is conducted, the system records the user's state at any given point of time, so that for some reason if they restart the experiment we can resume at the same point they stopped. This way users don't have to worry about browser and system crashes and they can elect to stop the experiment at any point in time and resume it later. As discussed above, bias removal is done in a way that does not conflict with this statefulness.

3.1.5.2 Evolution of the Experimental System

In the section above, we described the final experimental system. However, this was not our initial implementation. We feel it is worth discussing the problems we faced in our initial system design as they were very significant and not terribly predictable, due to external factors.

Initially our system relied on Google and the Google API extensively. After getting the result URLs from Swish-e and a Google search, we were making a second round of Google API calls to generate synopsis information for all of the URLs (even though we had metadata and pages from the focused crawl collection stored locally). By synopsis information, we mean a title and the content summary ("snippet," in Google parlance).

Once we tested this system, we found some major limitations. First of all, the Google API cannot be used to make more than 1000 calls a day for non-paying users. This restricted our experiment in many ways, as each query and results item was a separately-counted call to the Google API. Thus, we had to limit the number of users in the experiment, the total number of queries each user issued, and the number of results to retrieve for each query. We actually ran out of calls during the first live experiment, and had to use a secondary Google account.

Another major problem we faced was that the Google API was crashing for some of the queries. This not only left us with no results for those queries, but also crashed our whole system, as we could not handle blocking errors from the Google API (explicit exceptions were **not** returned).

To address these problems, the most important modification of our system was to limit our use of the Google API to just one call: getting the result URLs for each query. As a consequence, we had to build our own synopsis generator, consisting of a title extractor and a description generator (context-highlighted, as described above). We continued to retrieve page content through Google's cache via the CGI front-end, without using the API. This afforded higher speed and greater stability, without depleting our finite account of API calls. We also modified our caching system for page data and metadata so that both the Google-only and focused crawl items are opportunistically cached, which allowed us to avoid a page download if the data was already in the cache. The end result of these changes was a 10- to 20-fold reduction in Google API calls to start with, and gradually more due to the fact that the cache will grow over time. The new system is also much faster due to reduced page downloads and standardization on Google's servers.

After our first live run, we also identified some other significant problems. For instance, our initial system was not designed to handle various things like browser failures, system crashes, multiple clicks, and page refresh/reload by user. We remedied all of these pitfalls in the second iteration of our system. From all of this, we learned that it is not enough for just the developers to test an experimental system - "outsiders" should be enlisted. If possible, a dry run should be done which is on the same size scale as the live run.

3.1.5.3 Results

We are still in a process of conducting our experiment. And after analyzing the feedback we have obtained as yet we can conclude that our results are at par with current research and indicate that focused crawler are the best way to generate a domain specific web collection.

3.1.6 Future work

In the future we plan to further work on finding more innovative ways to improve the efficiency of the focused crawler. As we mentioned before, focused crawlers are still in their formative stages, and so our goal is to contribute as much as we can to their development, particularly for digital libraries. We would like to improve the performance, stability, and initialization portions of the focused crawler, as well as making them easier to install and deploy, and more available to interested institutions.

3.1.7 Conclusion

Based on the above study, we have determined that focused crawling for autonomous web collection development and is indeed feasible and helpful in extending the scope and service provision of digital libraries. We are surprised that, even for a subject domain as difficult to define as ours, useful results were attained. However, the deployment of the focused crawler was a labor-intensive task. We encountered bugs and had to employ many extensions and innovations to develop the training set. We hope to help remedy this situation in the future, with more complete and easily-deployable focused crawler software, designed with the digital librarian in mind.

3.2 Experiment B1: Combined Search via Web Crawling

[Section authors: Aaron Krowne and Martin Halbert.] Note: Section 3.2 was published in the JCDL 2004 conference proceedings as a paper entitled “Combined Searching of Web and OAI Digital Library Resources”; see Appendix C: References for full citation.

3.2.1 Introduction

Digital libraries are increasingly aiming to integrate valuable web resources, identified by human expert or automatic means, together with “native” DL records. Currently there are means to acquire web resources (lists of URLs, manually constructed or crawled) and DL records (harvested via the Open Archives Protocol for Metadata Harvesting), but these two remain separate both in representation and service functionality.

The MetaScholar Initiative [3] found that users do not understand why DLs must keep web and native resources separate, and do not find this method of organization particularly easy to use. There is a need, then, for DL services to more completely and meaningfully integrate resources from these two worlds.

In this article we discuss an experiment with such a web and native DL integrated search system, built using “off-the-shelf” open source software. We selected searching as it is a major type of DL service, and the results of MetaScholar were in the context of DL searching. For this experiment we take the approach of unifying both DL and web resources via a web crawl, then building a search system on top of this union. In future work we plan to explore the “opposite” approach; unifying all resources into an Open Archives framework as the basis for the search system.

3.2.2. Experimental System

The general strategy for constructing our experimental combined search system (with software used) was:

1) Expose digital library records via an OAI provider (ARC).
2) Expose Open Archives version of DL records to web crawls via a gatewaying service (DP9).
3) Build an index via crawling the union set of {WWW resource URLs, the gatewaying service entry point URL} (Swish-e).
4) Build a search service over the union index (Swish-e).

DP9 is a gateway service which is part of the ARC digital library software suite from ODU[2]. It allows any type of web agent to view the resources of an OAI repository through standard HTTP requests. Swish-e is a free,

[2] See http://dlib.cs.odu.edu/dp9/

open source search engine which can build its index via web crawling[3]. DP9 is a gateway service which is part of the ARC digital library software suite from ODU[4]. It allows any type of web agent to view the resources of an OAI repository through standard HTTP requests. If the entry point to a DP9 gateway is accessible from the home page of the digital library, its native resources can be indexed by web search engines. In our system, we used DP9 to a similar end, albeit with our own search engine.

Swish-e is a free, open-source search engine which can build its index via web crawling. After examining many alternatives, we selected Swish-e, due to its ability to support searching by META tags and other reasons. Because DP-9 exposes metadata fields as META tags, Swishe's META tag support means that native DL metadata need not be lost when building a combined web search system via crawling.

Our test setting was the AmericanSouth.org digital library. The WWW resources of our experiment were the Web Links from AmericanSouth, which were identified by subject scholars as valuable resources.

3.2.3. Analytic Methodology

We probed the quality of our system by repeated focused searches for specific web or DL resources, using subject scholars in a real-world setting. We analyzed the results by looking at the metrics of rank of the desired item in the results list and the number of queries issued to fulfill the information need. We did not use precision and recall, as these metrics are meant for sets of documents. With the time limitations of our study, we found it more tractable for users to search for a specific document and look for this document in the results list, then average scores over many instances.

Due to this, our experiment more closely resembles the "real world" situation. We built five alternative search engine indexes, and compared the same searches issued on all of them to determine the quality of our combined system. As we only had a single user session, we used simulations to approximate the effect of having users repeat their searches in all of the search systems.

To do this, we recorded their queries and the positions of results, and constructed a system to "play them back" on any search system and record the new results. The combined search systems were DL for "digital library", containing only the OAI-exposed digital library records; SW for "shallow web", containing only top-level pages of the web links; DW for "deep web", containing all pages up to two links from the weblinks top-level pages, inclusive; DL+SW, containing both DL and SW items, and DL+DW, containing both DL and DW items. In addition to looking at the objective quality of a combined system like DL+SW, this setup enables us to measure any negative effect on results due solely to combining items from the two different domains.

3.2.4 Experiment

Our subject scholars were stationed at a computer with two web browser windows, one opened to the AmericanSouth.org web site and one opened to our test Swish-e search engine. They were given instructions to pick and memorize any three AmericanSouth.org "Research Collection" items (our native DL resources) and any three AmericanSouth "Web Links" items, recording the URL or identifier of each. A brief reminder sufficient to get the "gist" of each resource was recorded/ memorized.

After the selection phase, the searchers switched to the browser window containing the combined search system and attempted to find each resource, issuing up to three queries to do so. They were told to search as they normally would, recording the result list position of the resource when found. At the end, they were also given a short 3-question survey asking their perceptions about the quality of the combined search system.

The results below are based on this recorded information, and focus on whether the combined search system successfully gives satisfactory results in a real-world sense.

[3] See http://www.swish-e.org/
[4] See http://oaiarc.sourceforge.net/

3.2.5 Results

Our five scholars each performed six search tasks, so there were thirty search tasks total. Two (both research collection tasks) had to be thrown out because the subject did not follow instructions. This leaves a sample set of 28 search tasks. There were 36 queries issued in attempt to fulfill these search tasks.

The sizes of the five search systems were as follows: DL+SW, 29,695; DL+DW, 35,420; DL, 29,611; SW, 84; DW, 5809. Thus, the reader should note that the shallow-web (SW) collection is very small compared to all of the other collections.

Table 2: Baseline results for uncombined search systems.

Statistic	**DL**	**SW**	**DW**
% of items found	100	100	53
Avg. # of queries needed	1.3	1.0	1.0
% of found in 1st or 2nd pos.	70	100	25
Avg. pos of found items	2.5	1.1	5.3
% found on 1st query	77	100	53

Table 3 : Results for combined search systems.

statistic	**DL+SW**			**DL+DW**		
	All	DL	Web	All	DL	Web
% of items found	100	100	100	60	69	53
Avg. # of queries needed	1.3	1.3	1.0	1.1	1.3	1.0
% found in 1st or 2nd pos	86	70	100	35	44	25
Avg. pos of found items.	1.6	1.3	2.1	4.2	3.4	5.0
% found on 1st query	89	77	100	50	46	53

Table 2 shows some key data for the “uncombined” systems (systems which only had records from either the web or DL domain). This serves as a baseline for comparison with Table 3, which has the same statistics for the search systems that combined the digital library collection with either the shallow or deep web crawl.

Note that the stats in Table 3 are also separated out into “dl” and “web”, showing just how the digital library or web content subsets performed. These can be compared directly with the corresponding baselines.

Comparing DL+SW to DL and SW, we can see that this combined search system seems to be nearly lossless. All items were found in both the combined and uncombined systems. In fact the only change is that results positions in the combined system were slightly different, with DL dropping from 1.3 to 2.5, and web items rising from 2.1 to 1.1 (surprisingly, given the relative sizes of the two collections, this is the opposite of what one would expect).

The results for DL+DW appear to be not as good, with the largest effect being the addition of the ˜6,000 deep web resources hurting the results for searches within the ˜30,000 DL resources. Search results for the web pages themselves did not change much. The survey results were also quite good, but are omitted for space reasons.

3.2.6 Discussion

The above results seem to indicate that combining toplevel web items, i.e. “home pages”, with DL resources in a combined search system works well. This is doable currently with freely available open source software. However, we uncovered problems in combining large web crawls with DL resources, even when the crawls were limited to two links deep.

A few qualifications are in order. First, one must remember that the results shown above for DL+DW, DL, SW, and DW are all simulations. These are based on “replays” of user queries which were actually only issued for DL+SW. During simulation, a failed query leads to issuing the next recorded query, which had been issued by the user for the same search task. However, sometimes such a query is not available, and even if it is, it is no longer being constructed in response to the current situation. Thus, our simulations provide what we believe is a very conservative lower bound on results.

Secondly, our “deep web” crawl was limited only by two criteria: (1) distance from top-level page ≤ 2, and (2) page is at or below the same level as top-level page in URL hierarchy. These are the limitations of what we could achieve with Swish-e without considerable modification. Optimally, it would be more fitting to use a focused crawler to protect from off-topic pages on the same web site, and even allow traversing the web in general, with good precision. We plan to explore this in the future.

Finally, we feel that deep-web results could have been much better had there been a way to “tweak” the search engine weightings for home/seed pages, particularly considering that these are the only types of pages our users searched for. This knowledge would be easy to pass to the search engine, but it is not supported in the search engine core (of Swish-e or any other search engine we know of). This is indicative of a general search object granularity/attribute weighting problem that we plan to explore in the future.

3.2.7 Conclusions

We believe the results above show that a functional and useful combined web and DL resource search service can be built currently with free, Open Source software. The results are particularly good when chiefly “home pages” make up the web portion of the collection, and users mainly search for DL records and home pages. However, we think more work is needed to improve search results for combined search systems built on general collections.

3.3 Experiment B2: Combined Search via OAI-PMH

[Section author: Aaron Krowne.]

In this experiment we attempted combined searching of web pages and digital library resources, wherein the web pages were crawled and converted to Open Archives repository digital library records. We utilized only free/open source software components for our investigation, in order to demonstrate feasibility of deployment for all institutions. This work complements the previous B1 study, in which we demonstrated the success of a combined search system via conversion of all objects to web pages.

3.3.1 Introduction

Digital libraries are increasingly aiming to integrate valuable web resources, identified by human expert or automatic means, together with “native" DL records. Currently there are means to acquire web resources (lists of URLs, manually constructed or crawled) and DL records (harvested via the Open Archives Protocol for Metadata Harvesting), but these two remain separate both in representation and service functionality.

The MetaScholar Initiative found that users do not understand why DLs must keep web and native resources separate, and do not find this method of organization particularly easy to use. There is a need, then, for DL services to more completely and meaningfully integrate resources from these two worlds.

In this article we discuss an experiment with such a web and native DL integrated search system, built using “off-the-shelf" open source software, conducted as a part of the MetaCombine project. We selected searching as it is a central type of DL service. For this experiment we take the approach of unifying both DL and web resources via an Open Archive, then building a search system on top of this union. Previously, we built a combined search system by unifying a web collection and digital library records as HTML pages (see Section 3.2). In a sense, then, we have explored the “opposite" approach here.

3.3.2 Experimental System

The general strategy for constructing our experimental combined search system (with software used) was:

1) Expose digital library records via an OAI provider (ARC).
2) Build a web collection from seed URLs via focused crawling (Nalanda).
3) Produce Dublin Core metadata records for web collection via metatagging (unsophisticated custom software).
4) Build an Open Archive of web metadata records (OAI-XMLFile).
5) Index the union of the DL and Web OAI providers (Lucene).
6) Build a search service over the union index (unsophisticated PHP scripting interfacing with Lucene).

Lucene is a java search engine library, from the Apache Jakarta project. It is full-featured enough to handle metadata (i.e., it can index distinct metadata fields and allow querying by them). Lucene was also purported to be quite fast, which we confirmed in our deployment. As Lucene is just a library and not a front-end, we built a simple web-based front end for the search engine using PHP.

OAI-XMLFile is an Open Archives data provider system built by Hussein Suleman at Virginia Tech. It allows the construction of an Open Archives provider based on directories of XML files, each file containing Dublin Core metadata (other formats can be added). This provider meshed well with our system, as we were working with metadata for the collection records in the form of XML files.

Our test setting was the AmericanSouth.org digital library. The WWW resources of our experiment were autonomously discovered via a *focused crawler* [Chakrabarti, 1999], which was seeded and trained via web links content from AmericanSouth.org, content from the *Encyclopedia of Southern Culture*, and portions of the Open Directory.

3.3.3 Analytic Methodology

We used the same general experimental strategy as in Section 3.2. We probed the quality of our system by repeated targeted searches for *specific* web or DL resources, using subject scholars in a simulated real-world setting. We analyzed the results by looking at the metrics of *rank of the desired item* in the results list and the *number of queries* issued to fulfill the information need (and variants of these).

We built three alternative search engine indexes and compared the same searches issued on all of them to determine the quality of our combined system. As we only had a single user session, we used *a posteriori* simulations to approximate the effect of having users repeat their searches in all of the search systems. To do this, we recorded their queries and the positions of results, and constructed an analysis system which could “play them back" on any of the indices, recording the new results.

The search systems were **DL**, for “digital library", containing only the OAI-exposed digital library records; **WWW**, for the web collection; and **Combined** for the union of these two. The latter was the full-fledged combined search system. In addition to looking at the objective quality of the combined system, this setup enables us to measure any negative effect on results due solely to combining items from the two different domains.

3.3.4 Experiment

We built a web-based system to guide participants through the experiment and record results. We utilized domain experts as our participants, each of whom accessed the system from a standard computer with a web browser. They were instructed first to log in, then were presented with a summary of a randomly selected resource from the collection. They could also click on a link to get the full metadata. They were told to phrase a search, in their own words (i.e., without intentionally copying verbatim from the metadata). The goal was to write a query that they thought would retrieve the resource.

The system would then present them with a list of results from the combined search engine. If their item was in the list, it would be highlighted specially, and would be clickable to finish the search task. If the item was not in the list, the user was prompted to try another query (they could try up to three queries). If the item was not found after three queries, the search task ended and a new resource was presented.

This process continued through five web resources and five digital library collection resources for each user. This resulted in a balanced test set for each kind of resource. The results below are based on the information recorded by the search system, and focus on whether the combined search system successfully gives satisfactory results in a real-world sense.

3.3.5 Results

We had five scholars who each performed ten search tasks, so there were fifty search tasks total. However, many queries had to be thrown out due to bugs in our initial search system. Thus, the results below consider only queries that worked.

The sizes of the three search systems were as follows: Combined: ~42,000, DL: ~35,000, Web: ~7,000.

Table 4 shows some key data for the “uncombined" systems (search systems built on the DL and web collections separate). This serves as a baseline for comparison with Table 5, which has the same statistics for the combined search system, and the performance of the DL and web subsets within it.

When comparing the number of attempts that each search task took, the results do not show a dramatic advantage for either the combined or separate index engines. The separate indices did provide slightly better results for finding items on the first attempt, but only for two of the queries. On average, the combined search found each item in 1.24 attempts, while the separate searches averaged 1.14 queries. Also, for one anomalous query, the combined engine actually outperformed the separate index one.

Table 4 : Retrieval results for Web and DL records individually.

statistic	Average	DL	Web
% items found	64.3	60.7	67.9
avg. # of queries needed	1.14	1.06	1.21
% of found in 1st or 2nd pos	51.8	60.7	40.0
avg. pos of found items	2.14	1.06	2.31
% found on 1st query	57.1	57.1	57.1

Table 5 : Results for combined search system.

statistic	Combined	DL	Web
% items found	66.1	60.7	71.4
avg. # of queries needed	1.24	1.06	1.40
% of found in 1st or 2nd pos	51.8	57.1	41.9
avg. pos of found items	1.72	1.12	3
% found on 1st query	53.6	57.1	50

When the number of successful queries is compared to the total number of queries, the advantage of using separate indices also becomes less apparent; percentage of successful queries differs by less than half of a percent.

When comparing the average position of the correct answers, the separate index model once again had a slight performance increase. On the combined search engine, the correct answer had an average position of 2.14. The average of correct answers in the separate engines was 1.72. Once again, however, the difference was not dramatic, as both are close to the second result returned.

Also noteworthy is the fact that the difference between the two engines was most predominantly noticeable for the web collection. The only improvement to the digital library collection was in the form of an increase in rank on one search task. This changed the average position of the correct answers from 1.117 to 1.059 for the digital library collection. The web collection showed a slight change in performance. A few of the queries yielded the correct answer in less attempts, but the average ultimately remained unchanged.

3.3.6 Discussion

The above results seem to indicate that combining DL with OAI-exposed Web resources in a single search system works well. We've demonstrated that this is doable currently with freely available, open source software.

As a disclaimer, one must remember that the results shown above for the Web and DL separate search engines are all based on *simulations*. These are “replays" of user queries which were originally only issued for the combined system. During simulation, a failed query leads to issuing the next recorded query, which had been issued by the user for the same search task. However, sometimes such a query is not available, and even if it is, it is no longer being constructed in response to the current situation. Thus, our simulations provide what we believe is a very conservative *lower bound* on results. Conversely, the live system (the combined search system) potentially gained a boost from the ability of users to respond directly to the results. Thus, the combined system may be slightly boosted relative to the separate systems. We do not expect that this effect is major, however.

3.3.7 Conclusions

We believe the results above show that a functional and useful combined web and DL resource search via OAI-PHM service is feasible currently. The portions of the deployment where we did the most original work were in the metatagger, which derived Dublin Core metadata for the web records, and in the construction of a CGI search front-end. However, these are minor tasks compared to building a search engine, acquiring a web collection, and interfacing collections and components. All of these parts have been taken care of by existing, free/open source software, making the overall construction of a combined search via OAI-PMH system relatively simple and effective.

Section 4: Federated Digital Library Frameworks

[Section authors: Aaron Krowne and Martin Halbert.]

The goal in Part C of the MetaCombine project is to create a federated digital library services framework and build for this framework services integrating the technologies developed in the other phases of MetaCombine. The impact of such a framework, along with its services, is expected to manifest in terms of increased prevalence of semantic clustering-based services at many libraries and digital libraries, particularly those that do not desire to or cannot install the software that powers these services.

4.1 Semantic Clustering Federated Services

Below, we briefly describe the most likely semantic-clustering related federated services for us to start with:

1) **Classification** – the assignment of category labels to unlabeled resource sets.
2) **Ab initio clustering** – the generation of an ontology from “scratch” from a set of resources, as well as mappings of resources to points within that ontology.
3) **Focused crawling** – the generation of a web collection of resource based on a domain-specific training set.
4) **Metatagging** – the generation of metadata records from various types of media, utilizing assumptions about the content type, structure, and advanced heuristics and machine learning methods (we could start with content-types such as web pages and research papers).
5) **Combined search** – collections of combined digital library and web resources, searchable through the federated framework.

To give a more concrete idea of how these services would work, consider a clustering service. A client digital library should be able to submit a set of resources (as metadata) to a clustering service provided by another peer on the MetaCombine federated network, and receive in response metadata correlating these resources with novel categories (clusters).

Similarly, for classification, a client should be able to submit a labeled training set along with an un-labeled document set to a classification service, and receive back labels for the unlabeled set.

4.2 Architecture and OCKHAM Connections

Emory is currently a participant institution of the OCKHAM project, which seeks to build a federated network of digital library services and provide some initial test-bed services for the network. Key in this network is the idea of connecting together standard (often pre-existing) digital library services such as Open Archives via a registry layer, using Web Services and open standards like XML and WSDL.

Because this is almost exactly what we wanted from a MetaCombine federated framework, we propose that instead of re-inventing the wheel in terms of this component of the project, we simply build upon the framework produced by the OCKHAM project. A framework has already been mostly-designed, and initial versions of many services have already been produced for OCKHAM. In the coming months, much more actual OCKHAM software will appear, through efforts at Emory and other participant institutions.

4.3 Implementation Plan

We plan to implement and/or demonstrate most of part C within the next half-year. Keep in mind that the framework itself is mostly provided for us by the OCKHAM project, saving us considerable labor and allowing us to focus on the MetaCombine-specific services outlined above (and potentially others).

Implementation will be coordinated and development led by Aaron Krowne. We will also utilize the development efforts of project research assistant Urvashi Gadi, who will do a master's thesis related to this part of the project. In addition, it is likely that Saurabh Pathak and Stephen Ingram (MetaCombine developer) will be involved with “repackaging” some of their MetaCombine efforts for the federated framework.

4.4 Evaluation

Investigations should be done into the scalability, deployability, usability, and effectiveness of these federated services. To test scalability and effectiveness, our principle developers can attempt to deploy the services based on the AmericanSouth.org content we have been working with. We could potentially utilize other collections that are available to us, for example, NSDL, or any subset of it.

To test deployability and usability, we need to involve users other than the principle developers. For this, we have the opportunity to leverage the community of OCKHAM developers. In addition, we can recruit library systems professionals from our environs.

We plan on developing informal and formal metrics for all of these facets of evaluation of the MetaCombine federated framework and services.

4.5 Next Steps

We believe that we will be able to rapidly implement many MetaCombine technologies for usage on a federated framework, utilizing the work the OCKHAM project has already done and is continuing to do. Implementation will allow us to focus on evaluating many aspects of the effectiveness of these services. In addition, we will be able to leverage the momentum, interest, and community surrounding the OCKHAM project to increase the scope and depth of our work on this phase of MetaCombine.

Section 5: Advisory Board Work and *Southern Spaces*

[Section author: Katherine Skinner.]

The MetaCombine project design includes a group of scholarly consultants who serve as an advisory board to the project. These consultants have been involved with the technologists throughout each stage of the project's design and implementation. This group has contributed a significant perspective to the project, one that highlights the needs of the communities (scholars, students, lifetime learners, librarians) that will eventually use these search and discovery systems.

To date, seven subject scholar consultants (hereafter referred to simply as *advisors*) have been recruited to accomplish two interrelated goals within this project: 1) to provide guidance and feedback on the SVM systems for semantic clustering during their design and implementation, and 2) to serve as the editorial board for Southern Spaces, a rapidly evolving online scholarly resource that has developed out of the AmericanSouth.Org project (2002-2004).

We are pleased to report that all five members of the *Scholarly Design Team* (SDT) from the previous AmericanSouth.Org project unanimously elected to continue working with the MetaCombine project. Their continued participation has allowed the MetaCombine project to quickly integrate the expert advice and suggestions of an unusually knowledgeable advisory board.

We are also excited over the further addition of two nationally recognized humanities faculty members as advisors, Dr. Tom Rankin of Duke University and Dr. Barbara Ellen Smith of the University of Memphis. Their participation has provided the Metacombine project and *Southern Spaces* with fresh insights and has broadened the subject focus of the group.

The Advisory Board is comprised of the following seven scholars. (The new additions are listed last.)

Dr. Charles Reagan Wilson is Professor of History and Southern Studies and Director of the Center for the Study of Southern Culture at the University of Mississippi, where he has taught since 1981. He is the coeditor of the *Encyclopedia of Southern Culture*, author of several books, and general editor of a new book series, "New Directions in Southern Studies," published by the University of North Carolina Press.. He has directed numerous symposia on topics ranging from the Caribbean and the South to Religion and the American Civil War.

Dr. Allen Tullos is Associate Professor of American Studies at Emory University. From 1982 until 2004 he was editor of the journal *Southern Changes*. He has worked as producer, co-producer, and sound recordist for numerous documentary films and has served as the website coordinator for americanroutes.com. Tullos has published numerous articles and book chapters on popular music, southern film and visual culture, the politics of space, and contemporary southern politics, including the Sydnor Award winning book *Habits of Industry*.

Dr. Lucinda MacKethan is Associate Professor of Literature at North Carolina State University, and a specialist in southern studies. She has published two books centrally concerned with the distinctive regional tradition of southern literature, co-edited *The Companion to Southern Literature*, and has helped to reissue three new editions of nineteenth century plantation novels. She is a senior consultant for the Scribbling Women website, scribblingwomen.org and has served as Chair of the NC Humanities Council for 2002-2004.

Dr. Carole Merritt has been the director of The Herndon Home, an Atlanta house museum, since 1983. She is the author of two books, has directed major exhibitions on the significance of race and class among African Americans, and has also served on state and national review boards in historic preservation and museum development. Currently a member of the African American Collections Advisory Committee of the Emory University Library and the editorial board of *Atlanta History*, the journal of the Atlanta History Center, she seeks to preserve, interpret and make more accessible for the general public, the history of African American families and communities in the South.

Dr. William G. Thomas, III is the Director of the Virginia Center for Digital History and Associate Professor of History in the Corcoran Department of History at the University of Virginia. He is the author of one book, co-author and assistant producer of a history of Virginia series for public television, and co-author with Edward L. Ayers of a fully electronic scholarly article published in the *American Historical Review* (December 2003). Thomas shared the Lincoln Prize in 2001 from the Civil War Institute at Gettysburg College for his involvement with the "Valley of the Shadow" project at UVA.

Dr. Tom Rankin is Director of the Center for Documentary Studies and Associate Professor of the Practice of Art and Documentary Studies at Duke University. A photographer, filmmaker, and folklorist, Tom Rankin has been documenting and interpreting American culture for nearly twenty years. His books include "Sacred Space: Photographs from the Mississippi Delta" (1993), which received the Mississippi Institute of Arts and Letters Award for Photography, "'Deaf Maggie Lee Sayre': Photographs of a River Life" (1995), "Faulkner's World: The Photographs of Martin J. Dain" (1997), and "Local Heroes Changing America: Indivisible" (2000).

Dr. Barbara Ellen Smith is Associate Professor of Sociology and Director of the Center for Research on Women at The University of Memphis. For the past thirty years, she has been an activist-scholar in Appalachia and the U.S. South. She is the author or editor of three books and numerous articles, and has recently completed a community-based research and education project, carried out in collaboration with the Highlander Research and Education Center and the Southern Regional Council, on Latino immigration to the U.S. South.

5.1 Advisory Board Contributions to Evaluation of SVM Systems

The advisors are charged with overseeing and cultivating the user interfaces and categorization schemes of the MetaArchive project. As previously mentioned, the advisors are uniquely situated to provide expert opinions in this area, in large part due to their long relationship with and deep understanding of the issues we have been studying across the projects of the MetaScholar Initiative.

5.1.1 December 2003 On-Site Meeting

In December of 2003, the advisors convened for their first annual meeting at Emory University. Prior to this meeting, each advisor had been instructed in the MetaCombine project goals, including the development of new text clustering and classification methods for the improvement of digital library browsing interfaces across unstructured and non-normalized metadata (or metadata that lacks a unified ontological structure).

During the December 2003 meeting, the advisor members evaluated the initial implementation of the DP9 and Swish-e combined web and OAI search through an experimental session. They offered feedback to the technologists of the project regarding the functionality offered in this initial DP9/Swish-e combined web and OAI search engine and suggested improvements to the way that the model is presented to the user. They also debated various options for structuring the tools and production systems that our Head of Digital Research is developing for focused searching and integrated web/OAI crawling.

5.1.2 August 2004 On-Site Meeting

In August, the advisors returned to Emory for their second annual meeting. At that time, they completed three sessions of experimental testing of the MetaCombine Project, including:

- an evaluation of focused crawling accuracy
- an evaluation of the B2 combined search via OAI-PMH system
- a second evaluation of native digital library resource browsing using semantic clustering

They also discussed the benefits of using an expert team of subject specialists to manually tweak the classification schemes generated by machines. They concluded, in concurrence with our technological staff, that the strongest and most cost effective system would begin with automated mechanisms that are trained to generate a classification ontology for a collection, and then should work with subject experts to help to train the SVM algorithms and to edit and clean up the results of automated constructions.

During the August meeting, the Scholarly Communications Analyst (SCA) proposed that the main production system for the MetaCombine project be structured as a sister site to the internet journal that the advisors have created (*Southern Spaces*). The proposed (and now approved) site, *Southern Searches* will include a sophisticated search platform and a wide variety of browsing schemes for research collections (harvested through the OAI protocol) and for web-based resources (collected through focused crawling). The advisors suggested that this site contain both computer generated and manually generated browsing schemes. The advisor is committed to helping the SCA and the project staff to create the manually generated ontologies.

5.1.3 September 2004 Remote Testing

Due to technical problems with the evaluations conducted at the August 2004 meeting, we are re-conducting two of the experiments (focused crawling and combined searching) at this time. These technical issues and, and where possible, the new results, are discussed in the corresponding sections of this report.

The re-evaluations are being conducted remotely in the interest of lowering costs. This is enabled by their format as web-based systems. We used feedback from the initial run to improve the systems for this run, and expect a gentler learning curve due to the fact that the SSC has already been introduced to the systems in person.

5.2 Advisory Board Contributions to Developing *Southern Spaces*

Southern Spaces is a peer-reviewed internet journal and scholarly forum that intends to provide a working model for online scholarship. This freely available publication site distributes research findings and scholarly analysis concerned with spatiality and the U. S. South. More specifically, *Southern Spaces* presents work that is concerned with representation of spaces and places in the South, as well as work which addresses the interrelationships of southern regions and places with other places and spaces in the broader world. The *Southern Spaces* audience includes researchers and teachers, students in and out of classrooms, independent scholars and writers, library patrons, and the general public.

Southern Spaces was created by the MetaScholar Initiative in response to findings from the recently concluded AmericanSouth.Org project (as summarized below).

5.2.1 Findings from AmericanSouth.Org

The AmericanSouth.Org portal was intended to provide a model of online scholarship, one that fused a scholarly, on-line publishing apparatus with a new OAI-based search and discovery system for research collections held by various cultural repositories.

Our work with the *Scholarly Design Team* (SDT) during this earlier project demonstrated that such a model needs to be constructed, not as a portal, but as a set of modular pieces. More specifically, we found that constructing a non-modular portal like AmericanSouth.Org yields a publication forum that is too complicated and cumbersome to serve current scholarly needs. The multiple options offered by the portal proved confusing to users (as demonstrated in our usability study). They also proved difficult for a technical team to build simultaneously and maintain in a cost-effective manner. Moreover, legitimating the publication side of such a system to important institutional groups, including tenure-review boards, would be difficult to achieve at this early stage of web-based academic publishing, especially in the humanities.

At the end of the AmericanSouth.Org project, the SDT determined that on-line scholarship would be most likely to earn credibility if it had its own domain and was structured more closely around the idea of an on-line journal. This journal, then, could be linked with “sister sites,” or other modular pieces (including, for example, the search and discovery system currently under development in the MetaCombine project).

There were particular advantages to web publishing that we all felt we could capitalize on in this new journal, including: 1) the ability of new media to present scholarship in an exciting and timely fashion; 2) regular updates of content, not based on issues, but published on a rolling basis; 3) globally, freely, and instantly accessible publication and archiving of materials; 4) new means of organizing and presenting research that complement the interdisciplinary research underway in many fields of scholarship; and 5) the ease and accuracy of tracking submissions with web journal software (we are using the Open Journal Software (OJS) system to govern the submission process).

5.2.2 December 2003 Meeting

The advisors determined at their first meeting at Emory in December of 2003 that *Southern Spaces* should aim to provide a pioneering model for web-based academic publishing in the humanities. Such a model, they felt, is currently lacking in scholarly publications. Even such exclusively on-line journals as *ECHO* (http://echo.ucla.edu) and *Postmodern Culture* (http://www.iath.virginia.edu/pmc/) have thus far tended to publish text-heavy pieces that look and feel very much like those found in traditional print journals.

The advisors recommended that *Southern Spaces* should distinguish itself by seeking to make full use of the internet medium. This publication encourages (and already exemplifies) the multifarious uses of streaming audio and video, high-quality photography, maps, and hyperlinks, as well as text. *Southern Spaces* accepts publications in four formats: essays, gateways, interviews and performances, and events and conferences. These latter categories, in particular, utilize the multimedia capacities of the internet to bring excerpts from public talks and conference proceedings, as well as performance pieces and interviews, to a broad public of users.

The advisors also decided during this meeting to build upon the theoretical emphasis they had established for the scholarship produced for AmericanSouth.Org. To that end, *Southern Spaces* is concerned with representation of spaces and places in the South.

Southern Spaces went live in April of 2004. Between April and August, when the advisor/Editorial Board reconvened at Emory, the technological team and the Scholarly Communications Analyst worked to standardize procedures for submissions and publications, including setting permanent locations for all *Southern Spaces* URLs, including those that lead to video and audio clips on the site, for citation purposes.

5.2.3 August 2004 Meeting

During this meeting, the advisors designed a plan for publicly launching *Southern Spaces*. They are currently soliciting contributions to the site, both nationally and internationally, from seventy leading scholars in Southern Studies. They are also helping to design the bridge between *Southern Spaces* and Southern Searches, thinking about the ways that the browsing ontologies designed by the MetaCombine project team for the production site may be able to connect with the browsing ontologies used in the *Southern Spaces* journal.

To date, we feel that the work of the advisors is a distinguishing feature of the MetaCombine project. This group holds a vested interest in producing search and discovery tools and scholarly communications mechanisms that will provide scholars and researchers with new resources and publishing outlets. The advisors have contributed greatly to the oversight and cultivation of MetaCombine and *Southern Spaces*.

Section 6: MetaCombine Project Outputs

[Section Author: Aaron Krowne]

In this section we discuss "outputs" of MetaCombine that take the form of some artifact: either a web site and specific systems, or software packages. The common feature of these artifacts is that they are accessible or usable by the MetaCombine target user community: digital librarians or interested parties. We focus on items that we have produced, all or in part, as of the present time.

6.1 MetaCombine Web Site

The MetaCombine web site (http://www.metacombine.org/) exists as a central locus of MetaCombine activities on the web. From here, we provide information about the project, demonstration systems, and packaged software that can be used elsewhere. The sections of the web site are:

- **Overview** – Introduction to the project at a high level, along with more detailed content about the phases of the project.
- **Demos** – Web-based systems which allow the user to get some sense of what we have developed in MetaCombine, and how it would be applied to a real-world digital library system. Thus far, we have demos for A1-A4 and B1-B2 available.
- **Software** - "Release" packages of software. Installation steps and at least basic usage instructions must be provided for software to be considered a release.
- **Reports** – Public MetaCombine documentation. Here we release all or part of deliverables to Mellon, as well as portions published or submitted for publication to digital library (and related fields) journals and conferences.
- **Links** – Hyperlinks to external sites, related to MetaCombine. This includes MetaCombine "products," current and future production systems derived from MetaCombine efforts.
- **People** – All who have participated in some capacity on the MetaCombine project, with mini-bios and a synopsis of their within MetaCombine.

The web site should be considered the starting point to finding the most up-to-date information about MetaCombine. Things that have changed between formal project reports will appear here first.

6.2 MetaCombine Software Releases in Development

We believe that releasing free, open source software is a very important part of meaningful work in the library and digital libraries fields. The impact of funded research and development is severely limited if it is not easy for others to programmatically re-used the efforts of the project. Flashy systems or interesting findings will likely elicit a "so what" response if others cannot inexpensively produce similar systems or take advantage of the findings. Thus, we feel it is due diligence on MetaCombine to follow up our research with actual F/OSS software products. Below we discuss software packages (released or intended for release) that we have worked on thus far, as well as some we plan to release.

6.2.1 Clustering System

We have created a complete, initial release of the NMF clustering system described in section 2.2(?). This consists of a main archive, with installation and usage instructions, containing the clustering program, a common MetaCombine text processing library, and a patch meant to be applied to SparseLib++ to fix some critical bugs.

These packages provide all that is needed for another effort to perform clustering in a stand-alone fashion, or integrate the clustering procedure into a larger system. The software is currently being documented and we aim to release it next year.

6.2.2 Greenstone Clustering Plugin

To further increase the impact of our clustering efforts upon the digital library community, we realized it would be helpful to integrate our clustering system with an existing digitial library system. We chose one such popular system, Greenstone, as the basis for the plugin.

Greenstone allows the modular addition of arbitrary organizational schemes via the use of external programmatic calls and something called *classifiers*. A Greenstone classifier is simply a program that places a record into a bin. The bins are then displayed through the Greenstone interface. For example, an alphabetic-by-title classifier might create a bin for each letter of the alphabet, and then use the title field of the metadata records to determine which bin to place each resource into. Greenstone would then use the output of this classifier to display the records linked under their corresponding letter “categories.”

For clustering, the mapping to a bin (cluster) is more sophisticated. However, as far as Greenstone is concerned, it is just another means of defining bins and mapping resources to them. This is the basis of our integration with the Greenstone system.

Jia Liu, a summer intern and Emory math PhD candidate, has been the main developer of the NMF Greenstone clustering plugin, which is still in a testing and development stage. We expect to release it during the second half of the project.

6.2.3 Other Potential MetaCombine Software Releases

We would like to release software corresponding to other parts of the project, including:

- a combined search via web crawling system (B2)
- a combined search via OAI-PMH system (B2)
- a cluster/classification visual browsing system (A3)
- a focused crawling tool

Note that the clustering tool released above corresponds especially to A1, A2, and A4.

One of our project assistants, Saurabh Pathak (Emory CS, M.S. Candidate), has done most of the development work on our focused crawling capabilities, and plans to investigate the production of a simple and powerful focused crawling tool for his Master's thesis. Many of the training and bootstrapping methods we discussed previously in this report are planned for inclusion.

We also would like to release federated services versions of many of these and other MetaCombine programs and technologies. This is discussed further in previous sections of this report.

Appendices

Appendix A: Project Staff

Martin Halbert, *Principal Investigator*: Martin provided the overall definition of MetaCombine and other projects in the MetaScholar Initiative and continues to oversee its progress. He is Director for Library Systems at Emory University. His efforts to the project are contributed to the project by Emory University.

Aaron Krowne, *R& D Lead*: Aaron leads research and development to achieve MetaCombine's objectives. He oversees the technical progress of the students and developers on MetaCombine, as well as personally conducts as much design, implementation, and results reporting as possible. He draws on his experience at Virginia Tech's Digital Library Research Lab and his work on the CITIDEL and PlanetMath digital libraries. Aaron is employed full time on the project, his salary provided with project funds.

Katherine Skinner, *Scholarly Communications Analyst:* Katherine is the lead and coordinator of MetaCombine's interaction with prominent scholars in the field of American South studies. These scholars serve the role of content experts in our semantic clustering experiments and as members of the editorial board of the Southern Spaces forum. This team carries over from the Scholarly Design Team (SDT) of MetaScholar. Katherine will be employed full time on the project during its second year, her salary provided by project funding.

Stephen Ingram, *Developer:* Steve is a full-time professional developer working on MetaCombine, focusing on the visualization aspects of the project. Much of Steve's work is novel and crosses into active digital library and visualization research. Steve is a recent Georgia Tech grad, with an interest in graphics. He is employed full time on the project, his salary provided by project funds.

Saurabh Pathak, *Project Research Assistant:* Saurabh is MetaCombine's focused crawling expert. He has worked with the iVia software and Nalanda crawler to produce useful autonomously generated web collections for MetaCombine. Saurabh has also worked extensively on classification with the WEKA classifier. Saurabh is a CS master's student at Emory. He is employed part-time on the project with project funding.

Johnny Healey, *Project Research Assistant:* Johnny administers MetaCombine's server cluster (MetaCluster) and is the current lead on the cluster infrastructure development. He also evaluates, customizes, and deploys various digital library components (such as search engines). In addition, Johnny is in participating in the design, construction, and evaluation of MetaCombine experimental systems. Johnny is a master's CS student at Emory. He is employed part-time on the project as a contribution by Emory University.

Liz Milewicz, *Project Research Assistant:* Liz brings her training as a librarian and her theoretical work on library culture to bear on the project. Liz works closely with the R&D Lead to ensure that the search and discovery systems designed by the technologists comply with library standards. She is employed part-time on the project as a contribution by Emory University.

Paul O'Grady, *Project Research Assistant:* Paul is the head of copyediting for *Southern Spaces* and is helping the Scholarly Communications Analyst and the Advisory Board to establish and implement standards for the publication site. He received a Woodruff Fellowship to work on the project this year as a contribution by Emory University.

Sarah Toton, *Project Research Assistant:* Sarah serves as the digital image coordinator for *Southern Spaces*. She works with photographers to ensure archive-quality standards for submissions, designed the standard page display for large photographic images, and assists with the coding of the publication site. She is employed part-time on the project as a contribution by Emory University.

Steve Bransford, *Project Research Assistant:* Steve is the Digital Media coordinator for *Southern Spaces.* A trained documentary videographer, Steve has helped to set standards for high-quality encoding of sound and video files on the site. He manages the sound and video submissions, and has contributed some of his own video footage to *Southern Spaces*. Steve is employed part-time on the project as a contribution by Emory University.

Emily Satterwhite, *Project Research Assistant:* Emily assists with the copyediting process for *Southern Spaces*. She is currently writing a dissertation concerned with parallels between late 19th and late 20th century local color texts and their reception. She is employed part-time on the project as a contribution by Emory University.

Urvashi Gadi *Project Research Assistant:* Urvashi is doing a thesis on semantic clustering software for federated frameworks, and will help with Part C of the MetaCombine project. Urvashi is a master's CS student at Emory. She is employed part-time on the project as a contribution by Emory University.

Jia Liu, *Project Research Assistant:* Jia worked on evaluation and deployment of the Greenstone digital library software. She also piloted implementation of an NMF clustering plugin for Greenstone. Jia is a math PhD student at Emory. She was employed part-time on the project as a contribution by Emory University.

Appendix B: Glossary

Classification	Classification is the placement of an item within a classification scheme. This is considered a process of organization. Classification can be done by human experts (as has been the case classically) or by automated programs. Programmatic classification, as on MetaCombine employs a family of methods called machine learning. The actual classification process, or placement, is not literal: a classification is simply the association of a class (or "category") label with an item. Once such a link exists, it can be used to enable browsing, searching, or other forms of co-navigation between items of the same class.
Clustering	Clustering is the process of grouping similar items based on their attributes. These groups are called "clusters". For text records, the attributes typically have to do with word content. The goal with clustering is for the clusters to reflect the "latent topics" of the content. Thus, clustering actually identifies the latent topics, and can be used as a tool to explore a collection. This can be seen as ontology discovery, and eliminates the need to provide an ontology prior to the organization of the objects.
Focused Crawling	A modification of web crawling whereby the visitation of a page via a link is selective. A classifier is used to determine whether a page to be visited is relevant or non-relevant. The page is only further recurred into if it is deemed relevant. Thus, focused crawling goes hand-in-hand with classification, and classifier must be trained before focused crawling can be done. The training set is typically from a specific subject domain (such as American south history and culture), as the relevant set must be distinct from the non-relevant set (items on the web in general). The point of focused crawling is to automatically discover as many pages as possible on the web for a particular subject domain, without having to crawl much of the web outside that subject domain. This allows efficient extraction of a relatively small subset of the web in an automated fashion.
LSI	*Latent Semantic Indexing.* A text indexing method developed to address the observation that meaning in text is carried not only by individual words, but by groups of words and classes of synonymous words. LSI (which is based on PCA) is classically applied as a preprocessing method to a text collection. The output consists of documents expressed not as words, but as pseudo-words, which are linear combinations of the original words in the dictionary. Queries over the collection are similarly translated into pseudo-queries. The goal is to be able to retrieve documents which contain words and phrases that may be different from the query words and phrases, but contain the same meaning. When the number of pseudo-features/words is smaller than the true number of words in the corpus, dimensionality reduction is performed by LSI. For a very small number of output features, LSI can be thought of as producing clusters (each feature is a topic).
NMF	*Non-Negative Matrix Factorization.* A relatively new method which works much the same as PCA, but relaxes mathematical constraints to allow more natural output. NMF can be used as the basis for clustering in any situation where LSI can be used. However, NMF has been shown to produce better output, due to its relaxed internal constraints which allow it to better model real-world data sets.
PCA	*Principle Components Analysis.* A method of delineating and discovering the most important informational components of objects represented in a matrix. PCA is based on SVD (Singular Value Decomposition) and is a classical method in numerical analysis. PCA is the core of LSI (Latent Semantic Indexing), which can be used for clustering.

Ontology	A framework for understanding or organizing a conceptual domain. In the very widest sense, we all hold complex ontologies in our heads for understand the world around us. Every field of expertise has a shared ontology, the acquisition of which is the key distinction between workers in that field (experts) and those outside it. For MetaCombine, we concentrate mostly on the organizational sense of "ontology", and thus use the term interchangeably with "classification scheme".
Subcollections	Collections within a collection. On MetaCombine, we consider all resources originating from the same Open Archives repository to be a "subcollection".
Topclusters	Abbreviated form of “top-level clusters.” Used to refer to a flat or one-level clustering scheme.

Appendix C: References

Agarwal, Kreveld and Suri, “Label placement by maximum independent set in rectangles.” In *Computational Geometry: Theory and Applications* v. 11. pp. 209-218. (1998)

Baker, L. and A. McCallum, “Distribution Clustering of Words for Text Classification,” in *Proceedings of ACM SIGIR*, 1998.

Chakrabarti, Soumen, et. al, “Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery,” in *Computer Networks* 31: 11-16 (Amsterdam, Netherlands), 1999.

Deerwester, S.C., et. al, “Indexing by Latent Semantic Analysis,” in *Journal of the American Society of Information Science*, 41(6):391-407, 1990.

Garner, S. R., “WEKA: The Waikato Environment for Knowledge Analysis.” In *Proceedings of the New Zealand Computer Science Research Students Conference*, 57-64, 1995.

Han, H. et. al “Automatic Document Metadata Extraction Using Support Vector Machines. In *Proceedings of JCDL 2003*: 37-48 (Houston, TX) 2003.

Heer, Card and Landay. “Prefuse: A Toolkit for Interactive Information Visualization.” Submitted paper draft, April 2004.

Joachims, T., “Text Categorization With Support Vector Machines: Learning With Many Relevant Features.” In *Proceedings of ECML-98, 10th European Conference on Machine Learning*, 137-142, 1998.

Koller, D. And M. Sahami, “Hierarchically Classifying Documents Using Very Few Words,” in *Proceedings of the International Conference on Machine Learning*, 170-178, 1997.

Krowne, Aaron and Martin Halbert, “An Evaluation of Clustering and Automatic Classification for Digital Library Browse Ontologies,” in *MetaCombine Technical Reports.* http://metacombine.org/reports/research/.

Krowne, Aaron and Martin Halbert, “Combined Searching of Web and OAI Digital Library Resources,” in *JCDL 2004* (Tucson, Arizona).

Krowne, Aaron and Saurabh Pathak. “An Evaluation of The Effectiveness of Text Classifiers on Digital Library Metadata,” in *MetaCombine Technical Reports*. http://metacombine.org/reports/research/.

Lee, D. and H. S. Seung, “Learning the Parts of Objects by Non-Negative Matrix Factorization" in *Nature* 401 (October 1999).

Liu, A. and Y. Gong, “Document Clustering With Cluster Refinement and Model Selection Capabilities,” in *Proceedings of ACM SIGIR 2002*, (Tampere, Finland) 2002.

Liu, Xaioming, et. al., “DP9: An OAI Gateway Service for Web Crawlers,” in *JCDL 2002*: 283-284 (Portland, Oregon, June 14-18).

Maslowska, I., Phrase-Based Hierarchical Clustering of Web Search Results," in *Proceedings of ECIR 2003*, pp. 555-562 (Pisa, Italy).

McCallum, A. K., “Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and clustering." http://www.cs.cmu.edu/~mccallum/bow. 1996.

Rochio, J. J., “Relevance Feedback in Information Retrieval.” In *The SMART Retrieval System -- Experiments in Automatic Document Processing*, ed. G. Salton. (Englewood Cliffs, NJ: Prentice Hall, Inc.,) 1971.

Roweis and Saul. “Nonlinear dimensionality reduction by locally linear embedding.” Science, v. 290 no. 5500. pp. 2323-2326. (Dec. 2000)

Salton, G. and M.J. McGill, *An Introduction to Modern Information Retrieval*, (McGraw-Hill), 1989.

Sebastiani, Fabrizio, et. al., “Boosting Algorithms for Automated Text Categorization,’’ in CIKM 2000, (McLean, VA).

Sebastiani, Fabrizio, “Machine Learning in Automated Text Categorization.” In *ACM Computing Surveys (CSUR),* 34:1, 2002.

Tenenbaum, Silva and Langford. “A Global Geometric Framework for Nonlinear Dimensionality Reduction.” Science 290 (5500). pp. 2319-2323. (Dec. 2000)

Wensi Xi, et. al, “Link Fusion: A Unified Link Analysis Framework for Multi-type Interrelated Data Objects,” in *Proceedings of the 13th International Conference on World Wide Web*, 319 – 327, (New York, NY) 2004.

Willett, P., “Document Clustering Using an Inverted File Approach.” In *Journal of Information Science*, 2:223-231, 1990.

Xu, Wei, et al, “Document Clustering Based on Non-Negative Matrix Factorization," in *SIGIR 2003*, 267-273.