"Quality Metrics" Project Interim Report

Aaron Krowne, Martin Halbert, Urvashi Gadi (Emory University)
Edward A. Fox (Virginia Tech)

July, 2005

Contents

1 Introduction

1.1 Purpose of This Report

This document is the first interim report of the "Quality Metrics" IMLS NLG project by Virginia Tech and Emory University, formally titled *Study of User Quality Metrics for Metasearch Retrieval Ranking*. It serves as a complete update on the status of our activities with the project to date. These activities include both management of the project as well as research and development progress.

1.2 Project Background

Digital library search systems have not evolved to keep up with growing user expectations and the metadata-rich nature of digital library objects. These search systems, though an improvement from arcane, classical OPACs, behave clumsily in the face of heterogeneity and radically different values for important metadata attributes. The increasing prevalence of metasearch, a scenario whereby disparate and generally heterogeneous objects are searched together in one interface, has exacerbated the problem.

The reason for this situation is that search engines have come out of the field of information retrieval (IR), which has recently been focused on solving the web search problem. While digital libraries have benefitted from recent advances in information retrieval, largely spurred to solve the web search problem, the web search and digital library search scenarios are quite different. Aside from scale differences and different amounts and kinds of item interlinkage, the web largely lacks metadata.

This is not a minor distinction. As an example of how this manifests, digital libraries have the opportunity to distinguish between item-level and collection-level records in determining how to present retrieval results. Search engines developed for web-based materials do not address this issue of record type (which is also an issue of *granularity*). Some search engines adapted for digital libraries have attempted to differentiate results along such granularity boundaries, but their approaches have not been formally tested with users. A similar problem also manifests in treating results from various subcollections (e.g. separate library content databases, metadata records harvested from other digital

libraries) and items culled from the web vs. "native" digital library records.

Further, these and other attributes often also bear on notions of *quality*, which deeply influences the organization of retrieval results. In the standard web search situation (Brin and Page, 1998), hyperlink data is used to make inferences about quality which allow ranking to be vastly improved over purely content-based matching methods. In digital libraries, we could extend the gathering of such quality information to attributes pertaining to vettedness of records, rating, view popularity, logged activation from search results lists, aggregation in path or lesson plan objects, and much more. Thus, there is both a great need and opportunity to intelligently make use of digital library metadata in retrieval results presentation.

The goals of this project are (1) to discover the best way to present digital library retrieval results by digging down to the user expectations level, (2) apply these findings to a working prototype system, and then (3) evaluate the effectiveness of these systems relative to standard or typically available alternatives.

We will refer to the new search system we are developing to integrate and expose quality metrics as "QM-search."

2 Personnel Activities

In this section we briefly report on the hiring activities of the project since its initiation.

- **Rohit Chopra** has been hired at Emory to lead the design and execution of the focus groups. His position type is graduate research assistant.

  Rohit Chopra is a sixth year student in the Graduate Institute of Liberal Arts, Emory University. His dissertation addresses the historical relationship between technology and nationalism in Indian society from the mid-nineteenth century to the present, with a focus on current expressions of Hindu nationalism on the internet and world wide web.

  Prior to joining Emory University as a graduate student, Rohit worked in the Indian internet industry as an information architect in Mumbai at rediff.com, India's first and leading online portal and web solutions provider. His job involved conceptualizing, developing content for, and overseeing the production of corporate websites and in-house rediff.com web channels on education, health, and shopping. The job function also included liaising with corporate clients and with internal marketing and sales teams. Rohit has also worked as a copy editor in an academic publishing house, Sage India, and as a freelance journalist for various print and online publications.

  We believe that Rohit's unique combination of technology, communications, and "people" skills make him well-suited to coordinate the focus groups portion of this project.

- **Urvashi Gadi** has been hired at Emory to undertake the programming work of developing a prototype QM-search system. Her position type is full-time software developer. Urvashi is a recent Emory Computer Science Master's graduate, and has worked with us on the IMLS-funded Music of Social Change digital library sponsored project for the past year.[1] For this project, she developed the Metadata Migrator application, which dramatically lowers the barrier for the creation of Open Archives metadata repositories.[2] She also worked with OCKHAM and MetaCombine technologies

[1] `http://www.metascholar.org/mosc/`.
[2] `http://www.metascholar.org/sw/mm/`.

for her Master's thesis, which deals with the web service application of semantic clustering technologies.

- **Vikram Raj** has been hired at Virginia Tech to undertake the user studies for evaluation of QM-search. His position type is graduate research assistant. Vikram is a Master's student at Virginia Tech. He has broad academic and work experience with software engineering, including working with users and clients, and is particularly interested in Human-Computer Interaction (HCI).

The PIs and co-PIs began planning and theoretical work on this project in the Spring of 2005. The results of this work to date are laid out in this report. Urvashi Gadi has initiated the QM-search system development work. The research assistants, Rohit Chopra and Vikram Raj, will begin their work on the project for the fall 2005 semester.

3 Surveys

3.1 Survey of Systems That Recognize or Integrate Quality Metrics

In this section we discuss extant systems from which we "draw inspiration" for the activities of this project. These systems demonstrate particular aspects of certain things we think are necessary for an integrated quality-metrics DL search system, as well as certain things *not* to do. These extant systems are not in themselves complete solutions for a variety of reasons, such as their often-monolithic or closed nature and lack of generalization. In sum, while they demonstrate good which we have considered, they do not "completely solve" the retrieval organization problem for digital libraries.

3.1.1 Amazon A9

A9 is a search service for Amazon.com, which was created in 2004. It is a web metasearch system which is meant to contextualize search results within Amazon. Some of the domains A9 searches over are the web, books, images, yellow page directories, and reference sites (including Wikipedia). There are also "personal" domains, such as the current user's history and bookmarks.

A9 has a novel column-based interface (see Figure 3.1), where each column displays results from a particular domain in an appropriate way (for example, images are thumbnails, not text links). The user can also easily and fluidly select which domains are displayed using labelled buttons on the right-hand side of the page. Modifications to the display happen without a complete reloading of the page, adding to the fluidity of the experience.

3.1.2 Google

While Google is primarily a web search engine, in the past few years it has become more of a metasearch system, recognizing limited metadata and different types of content objects. For example, Google recognizes a distinction between "home pages" and subsidiary pages within the same site. This bears some similarity to a collection-level vs. item-level distinction. On search results displays, this distinction manifests as a grouping and indentation (see Figure 3.2).

Figure 3.1: A screenshot of a typical search with Amazon's A9 system.

Also noticeable in the figure is a book result for the query. Thus, Google is also searching commercial book catalogs for hits. In fact, if you follow this link, you will find that Google has provided a page-image digitization of the book which is browsable, and also which has been scanned, indexed, and is searchable. Further, Google has been indexing and making accessible library catalogs with its Google Scholar service.

All of this points to the reality that Google has become much more "digital library-like," and less of a pure web search engine. Google recognizes many different types of records, content objects, and metadata fields. In sum, Google deals with the following digital record attributes which are of interest to this project:

- different object types (web page, powerpoint presentation, pdf document, library holding, print book)
- granularity (home vs. lower-level pages)
- status of user access to holdings
- bindings to taxonomy (Open Directory)
- local results (detection of geographic proximity)

Digital libraries should be sensitive to all of these things. However, there is no evidence that Google's handling of these attributes is optimal for digital library purposes. Further, digital libraries can extend search engine capabilities in a number of ways, as we discuss in the theory section. Finally, all of Google's

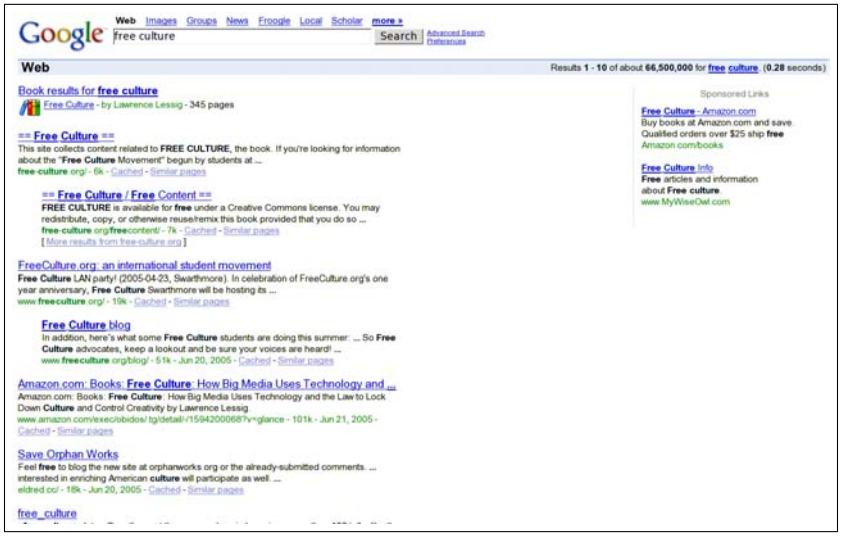

Figure 3.2: A typical Google web search result, showing the distinction between home pages and subsidiary pages, as well as results for matching print books.

functionality is hard-coded into the monolithic Google system, and can therefore not benefit DLs in redeployment.

3.1.3 OAISter

OAIster is a service of the University of Michigan that provides a unified search system for high-quality digital library resources exposed via OAI. There are over 500 such OAI repositories involved, and over 5 million records indexed by the service.

A shot of an OAIster search results screen is shown in Figure 3.3. As a metasearch system aggregating records from many different source repositories, OAIster has as its task to expose these sources to the end user. It does this by keeping a left panel breakdown of sources and counts of results for the current search within each source. In the main (right) panel, the user gets a linear list of records for all of the sources (this list can be sorted by various fields, including query-record score). To view just the results from one of the source repositories, the user can click on that repository's count on the left panel.

3.1.4 CIC Portal

UIUC's CIC portal is a union search and browse interface over resources collected from a consortium of major midwestern universities. The CIC portal software is not entirely original—it is a modification of the OAIster software. Among the enhancements one can detect from the screenshot in Figure 3.4 are

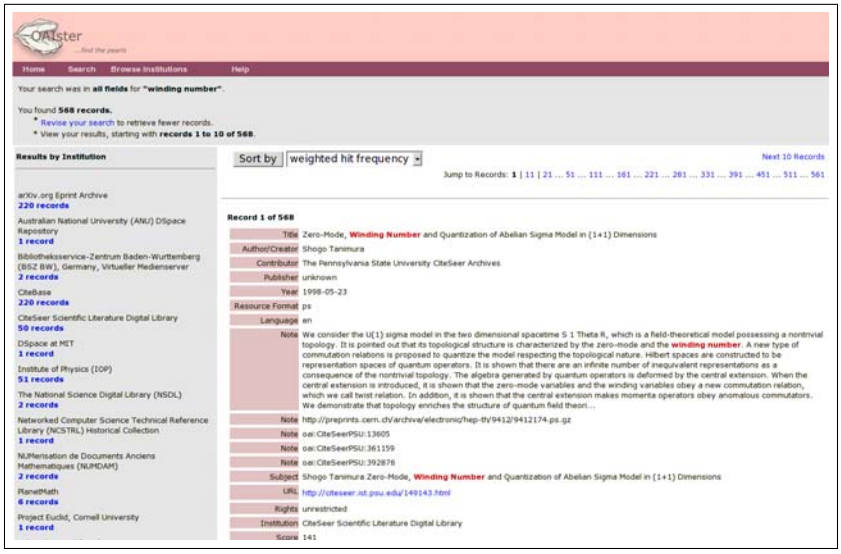

Figure 3.3: A screenshot of a search with the University of Michigan's "OAIster" OAI portal.

identification of collections *inside* archives (i.e., higher collection granularity) and the addition of *thumbnails* for records which describe images. These enhancements add greatly to the usefulness of the CIC portal.

3.1.5 Dogpile

Dogpile is a commercial internet meta-search engine. It chiefly dispatches queries to other large internet search engines and unifies the results, weighing-up resources which appear in multiple search engines.

Users can also elect to see the results from the individual search engines side-by-side in a panel view much like A9 (see Figure 3.5). In this view, resources which are unique to each individual search engine are specially highlighted.

It is worth noting that the default view is simply a single list, as with a traditional web search engine, even though the list contents are aggregated from many sources. Thus, Dogpile attempts to derive value from the multiple sources represented by many search engines, while presenting the results to the user in a format which is by default no more complex than any of those individual search engines.

3.1.6 The European Library

The European Library (TEL) is a meta-search system which combines the resources of many prominent European national libraries. It keeps these source libraries very rigidly represented throughout the interface, and even in the search

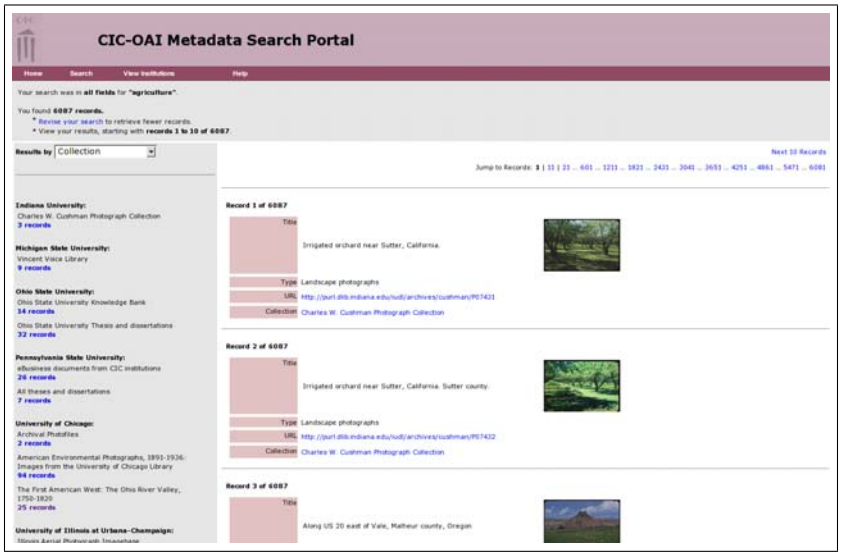

Figure 3.4: A search from UIUC's CIC portal.

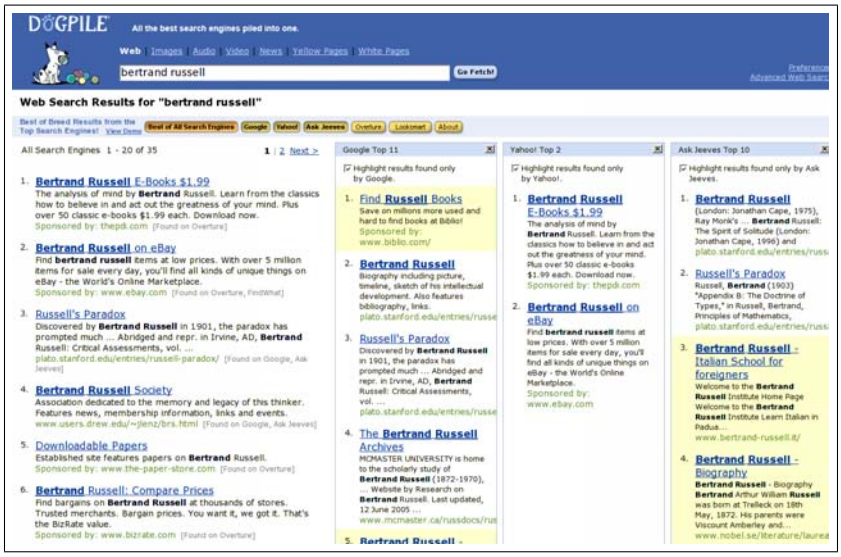

Figure 3.5: A screenshot of a search with Dogpile, a commercial web meta-search engine.

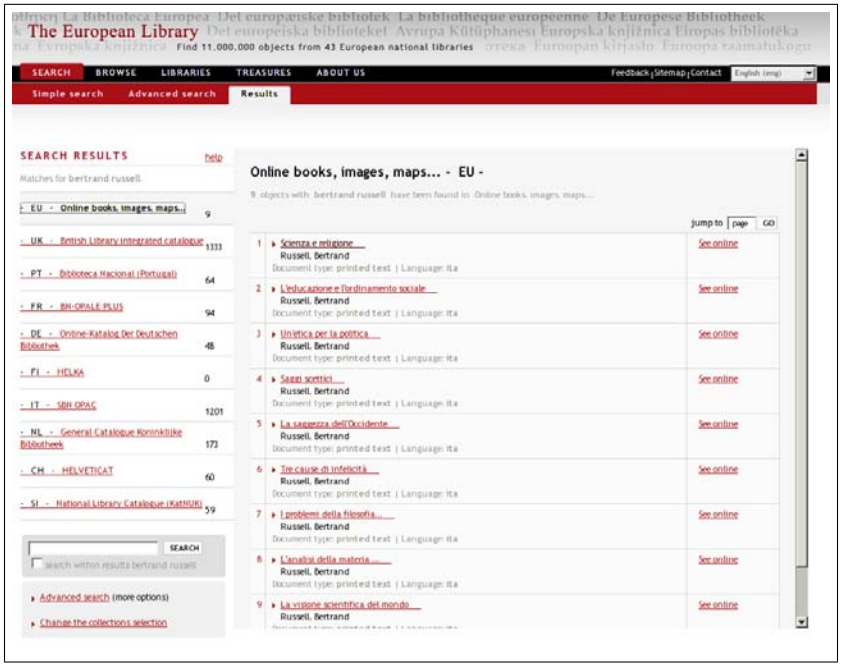

Figure 3.6: A screenshot of a search with The European Library's (TEL's) metasearch system.

results (see Figure 3.6). No results display will show items from the source libraries combined together; one must select which library to look in from the left-hand navigation panel. It is likely that this system could be improved by a greater level of integration.

3.1.7 Envision

Envision is different than all the previous systems yet is still very relevant; it is not a metasearch system, but is rather a search results visualization system. It allows the user to view results based on up to six attributes, mapping them to display facets such as color, shape, intensity, and two dimensions of position (see Figure 3.7).

While Envision provides an advanced interface for understanding digital library search results, it has a number of drawbacks that prevent it from being the *de facto* digital library search interface. In fact, the very attributes of being advanced and displaying many facets of the data make it too cumbersome for most users and most digital library applications. Besides this, as a stand-alone desktop application, it requires a particular local computing environment and installation work.

Much of what we hope to achieve in this project is to maintain Envison's

Figure 3.7: Results being displayed in the Envision digital library visualization system.

ability to handle many resource facets that are related to quality, but to do it in a way that does as much work for the user as possible and simplifies the interface and display. We also hope to generalize many of the capabilities of Envision, so that they can be applied more easily and flexibly.

3.2 Survey of Open Source Search Systems for Use as an Experimental Basis, and Selection of Lucene

In this section we discuss open source engine systems which we have considered as a basis for a prototype QM-search system for this project. The open source nature of such systems means they are all potentially extensible by us. However, radically different amounts of work may be required for each due, to their differing goals and application environments and correspondingly different architectures. Finally, as open source software, the support of the community for a particular system is an important factor in its consideration. We conclude this section with our discussion of Lucene, the system we will most likely utilize on as a basis for our work.

3.2.1 Swish-E

Swish-E was built as a web site indexing and search system. It contains a crawler, indexer, and search engine, all implemented in Perl. The general functionality of the system is to crawl a site or sites from a list of seed URLs, parse and index each page, and then make this page searchable through a CGI interface. The scope of the crawl is configurable (though it does not contain advanced functionality like focused crawling), and the system recognizes META tags if available. While intended as a site search engine, Swish-E contains all of the essential elements to be used as a digital library search engine. If digital library records are exposed as HTML pages with META tags, nothing is lacking in terms of fielded search functionality. Indeed, we studied this approach in (Krowne and Halbert, 2004). We set up a Swish-E based demo system that searches a snapshot of AmericanSouth.org web and digital library content, all exposed as web pages.[1]

However, Swish-E is intended as more of a search system "in-a-box," and is thus not easily configurable in terms of core functionality (its look-and-feel are readily altered). Also, while it is possible to adapt Swish-E to integration with extant digital library application systems, we found that Swish-E's opaque, monolithic nature does not foster maximal digital libary search capabilities.

[1] `http://metacluster.library.emory.edu/~akrowne/cgi-bin/swish.cgi`.

3.2.2 Greenstone

Greenstone is an open source digital library system from Waikato University. Its construction is supported by UNESCO, so it excels at multilingual and collection-in-a-box capabilities.

Greenstone can behave as a metasearch system, as it can harvest and index remote OAI repositories, and uses the MG system for its indexing and search engine capabilities. MG is the search engine described in the *Managing Gigabytes* digital library book, by the authors of Greenstone. We believe that systems like Lucene are more modern and modular, but are investigating MG for suitability as a standalone search engine component.

3.2.3 PKP

The Public Knowledge Project (PKP) provides a number of free, open source "research management systems." Among them is an OAI "harvester," which in fact harvests, indexes, and exposes (via search and browse) the resources harvested. Thus, the harvester subsumes an entire metasearch system. This system has extremely basic capabilities in terms of search, as its search engine is based on the built-in mysql text search. Thus, any implementation of QM-search over this basis would in fact have to be done to mysql, which is at the level of text search of table rows as opposed to searching of metadata fields. There would also likely be issues getting QM-search enhancements to mysql included in subsequent mysql distributions, as QM-search really is not logically a type of database functionality. For situations where the rest of mysql is not desired or another DBMS is in use, we would prefer not to have QM-search coupled to mysql.

3.2.4 ESSEX

ESSEX is a digital library search engine system developed by Aaron Krowne for the CITIDEL project. It is designed to be a fast, modular search system, which handles fielded digital library metadata. Its speed comes from its C++, in-memory-only index implementation. The modular nature is implemented through its sockets communication API.

ESSEX excels at its goals of speed, modularity, and handling of digital library information. Unfortunately, it needs a considerable amount of development work to improve stability and scalability. Lack of a community of users and developers has meant that the project is basically dormant. Thus, ESSEX would be a poor choice for use in a production setting.

Further, ESSEX is not built to handle customizeable ranking methods, and it contains no provisions for results organization other than a linear, ranked list. This means that deep hacking would have to be done to implement QM-search capabilities.

3.2.5 Lucene

Lucene is perhaps the most prominent of the open source search engine projects. Lucene is a Java-based search engine system, which is part of the Apache project. It is built upon a document abstraction that allows it to handle fielded information, and is thus well suited to digital libraries.

Lucene is well-architected and implemented cleanly. In fact, Lucene is the basis for higher-level crawl-and-search systems, such as Nutch.[2] Thus it is easy to modify and extend. Its class hierarchy and general functionality caters to the use of different ranking algorithms.

Perhaps most importantly, Lucene has a vibrant community of users and developers surrounding it. This has resulted in its elegant architecture, scalability, and lack of bugs. In essence, it is the model of an open source, industrial-strength production system.

Due to all of this, we feel Lucene is the best candidate as a development platform for our QM-search.

[2] `http://lucene.apache.org/nutch/`.

4 Theory and Architecture

In this section we discuss the theory underlying the desired QM-search system, specific algorithms and techniques potentially helpful for its implementation, and our plan for their implementation leveraging an extant software system (Lucene).

4.1 Theoretical Model for Quality Metrics

Fundamentally, the goal of QM-search is to elegantly solve the problems of effectively discovering and utilizing quality information in digital libraries. The hazards are that this information may be heterogeneous in nature, sparse, and overwhelming to end users.

In this section we sketch the beginnings of a theoretical model which meticulously and rigorously defines all of the conceptual structures and functionalities necessary to solve these problems, and better meet the needs of digital library users.

The theoretical model of QM-search can be broken into two connected but relatively independent sub-components:

- **The presentation model** - Determines how objects should be organized logically, given their attributes and valuations delivered by the scoring model.
- **The scoring model** - Determines how metadata attributes (either explicit or latent and inferred) are fused into scalar score values upon which the presentation model is built.

The names of these two models are only approximations. The presentation model in fact is closely-tied to much of what is typically described as "visualization" as well as "presentation" or "reporting." Fundamentally, it deals with establishing the informational basis for all of these activities, all of which might be done as the end of the process of digital library searching. Thus the presentation model helps solve both the "dealing with overload" and "dealing with heterogeneity" problems alluded to above.

The scoring model, in turn, extends beyond scoring to the gathering of information which is necessary to perform scoring. Therefore it addresses both

the "heterogeneity" and "sparsity" problems, as it deals with combining and fusing (potentially latent) quality information.

Our goal is to produce a model which will always allow us to present records in a way that is based solidly upon intuitive notions of their quality, despite missing or unreliable underlying quality attributes. This is a guiding principle of *resilience*, and it is a property existing systems tend to lack.

Below we list some quality attributes (i.e., record or metadata attributes or aspects) which typically contain a high level of "quality impact." The idea of our model is to achieve the kind of resilience described above by integrating as many of these attributes as possible in an appropriate manner (i.e. without confusing or overwhelming the end user):

- vettedness
- rating
- circulation/activation/views
- collection vs. item level type
- collection association
- reference/citation linkage
- use in aggregations/sequences
- taxonomic binding (classification)

One can imagine how some of these attributes might not be present for some individual records, subcollections, entire digital libraries, and so forth. Nevertheless, information in the form of other attributes *is* likely to be present, and our framework would handle this situation elegantly.

4.1.1 QM-search and 5S

We couch these two models in the language of 5S; an extant digital library modeling framework which breaks DLs down into *structures, streams, scenarios, spaces, and societies*. 5S represents these entities, and their relationships to each other, in a formal manner. For example, the formalism of *spaces* can be expressed in terms of familiar mathematical concepts like probability spaces, metric spaces, Cartesian spaces, vector spaces, and so forth. Typical 5S elements for each of the "S's" are shown in Table 4.1.

The formalism of 5S helps us to clearly conceptualize our digital library systems and services, model them, articulate them, and construct them. We sketch the presentation and scoring models of QM-search, including their interactions with each other and all of the 5S elements, below:

- *Streams* - Search results can be modeled as record identifiers delivered in streams.

Models	Primitives	Formalisms	Objectives
Stream Model	Text; video; audio; software program	Sequences; types	Describes properties of the DL content such as encoding and language for textual material or particular forms of multimedia data
Structural Model	Collection, catalog; hypertext; document; metadata; organizational tools	Graphs; nodes; links; labels; hierarchies	Specifies organizational aspects of the DL content
Spatial Model	User interface; index; retrieval model	Sets; operations; vector space; measure space; probability space	Defines logical and presentational views of several DL components
Scenarios Model	Service; event; condition; action	Sequence diagrams; collaboration diagrams;	Details the behavior of DL services
Societies Model	Community; managers; actors; classes; relationships; attributes; operations	Object-oriented modeling constructs; design patterns	Defines managers; responsible for running DL services; actors, that use those services; and relationships among them

Table 4.1: 5S sub-models and valid elements.

- *Structures* - Search result streams have structure (linear ranked ordering, separation into logical "bins," graph-theoretic interlinkages, linkages to taxonomies, etc.).

- *Spaces* - Spaces are used to render the structure onto search result streams. For example, in traditional IR, you have a stream of search results structured (ordered) linearly. This is a one-dimensional rank space, which is discretized from a continuous, one-dimensional scoring/weighting space. These scorings are generated using another space, such as a vector space or probability space. We can imagine in a QM-enhanced search system that we are extending the results organization space into two or more dimensions. The first dimension would be rank as before, but the second dimension could be viewed (when discretized) as organization into "bins". This binning could be based on object type (web page, presentation, book, etc.), rating, source collection, etc. We can imagine also that the attributes used to map objects to different dimensions and positions along these dimensions should be configurable, so for instance, it would be simple to integrate rating into first dimension rank *or* use it instead to create five second-dimension "bins" of rating values equal to one through five.

- *Scenarios* - In different scenarios (serving different information needs) user notions of value and interest change. Also, the expected mappings of attributes and values to the results organization spaces dramatically changes.

- *Societies* - Different societies have different notions of value, which attach to the various content and collection attributes we consider. As above, different societies may have different typical notions of how the attributes should effect organization of results into spaces.

A useful conceptualization of the "API" to the QM-search framework, as modeled above, is that of a structured XML stream of records (as identifiers), which is then transformed by XSL into human-comprehensible XML artifacts. Some basic primitives of the XML language would be that of record sets corresponding to logical bins, which have some ordering (including rank and numeric weight), references to taxonomic elements and other records, and type information that gives the presentation layer hints on how to display the record set.

4.1.2 The Presentation Model

In this section we give a more detailed sketch of the presentation model. At this point the theory may change considerably, but we feel that the following is a solid beginning.

In Figure 4.1.2, we give a diagrammatic illustration of the presentation model. Shown are two presentation "views" incorporating the *same three un-*

derlying metadata attributes—vettedness, domain (source OAI repository in this case), and query-content similarity.

The two views render these attributes in two different "dimensionalities"—the first contains two dimensions, the second three. Each dimension is also called an "axis." Note that in view 2, each utilized attribute corresponds to one axis. However, in view 1, vettedness and query-content similarity are "squeezed" into a single axis. This is done with the help of a *combination function* which takes the two attributes and yields a single score.

When noting the "presentational rendering" portion of the diagram, it becomes apparent why one might want to select a different number of axes for the same set of attributes: because these different dimensionalities yield radically different presentations. View 1 has a natural rendering as a familiar "tabbed" display, with the "domain" axis corresponding to the tabs, and the vettedness + content-query similarity axis corresponding to vertical rank. Essentially, this is the same presentation model as Google, which fuses content-query similarity with inferred PageRank values for each object, yielding the vertical organization, while simultaneously allowing the user to switch between various "horizontal" domains (web, images, library holdings, books).

View 2 allows us to construct an "A9-like" display of the results. Here, three dimensions are all available, despite the fact that the display is two-dimensional. This is achieved by mapping domain to columns, vettedness to vertical panels, then query-content similarity to vertical organization within vettedness panels of the same value.

Note that there is some difficulty in "projecting" these three dimensions into two display dimensions. It may not be immediately apparent to the end user that there are *two* types of vertical organization. In essence, two axes are "stacked" on top of each other (vettedness and content-query similarity). Also, such a display would be impossible without *binning* of the objects into equivalence classes of vettedness value. Thus, if the underlying vettedness attribute is decimal or has more discrete values than the three-star display, these values will have to be "packed" into just three bins. This entails a loss of fidelity and thus a trade-off in the overall display rendering.

In general, perhaps the most important insight of the presentation model is the packing of many (but arbitrary) attributes into single axes. This means the system deployer (or even the end user) has the potential ability to select all and any attributes important for retrieval, and display them in as few or as many dimensions as necessary for clarity.

Interestingly, many aspects of our presentation model closely resemble existing work done at Microsoft Research on the "Data Cube;" a relatively recent relational operator for understanding multidimensional data objects (Gray et al., 1997). We extend the data cube model by delving into the aggregation functions (our *combination functions*) and rendering the selected objects into a presentation. Thus it is important to note that there is conceptual support for our model of the comprehension and presentation of multifaceted objects.

Each axis of the presentation model needs the following elements for this scheme to be complete:

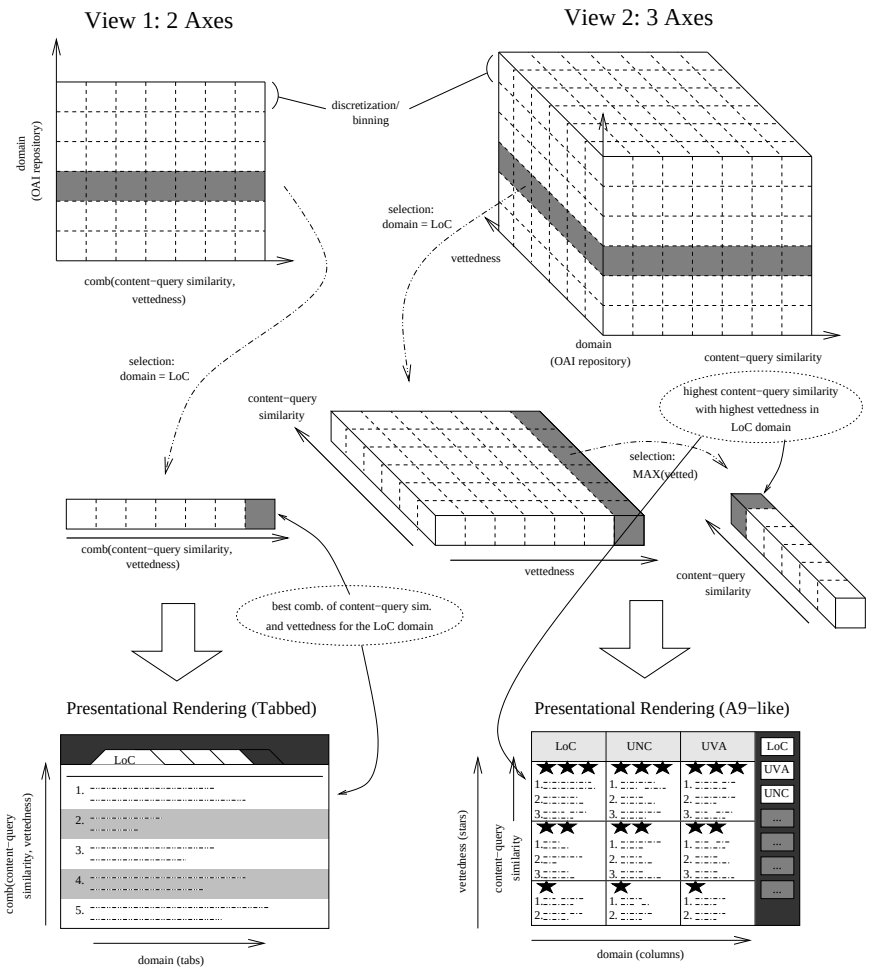

Figure 4.1: A diagram of the presentation model aspect of QM-search, by example. Two views are shown, both built upon the same three metadata attributes (vettedness, content-query similarity, and OAI repository domain). The Cartesian top portion should be thought of as a "field" which organizes retrieval results. Due to ranking, discretization, and binning, the field is "solid." Thus, each square (or cube) should be thought of as a single object (or a set of objects with the same equivalence class modulo scoring, ranking, and binning). The axes point in the direction of increasing score for various attributes. The top portion of the diagram shows a "slicing" process, by which we can conceptually imagine zooming-in on the "best" object. The bottom portion, with "mock-up" screen shots, shows a potential way to render the corresponding presentation model in a standard 2-D web browser interface. This illustrates the correspondence between model axes and implicit "screen axes," as well as the way metadata attributes play into this relationship.

1. An underlying *scoring function* which generates a scalar value based *on one or more* metadata attributes. When two or more attributes are the inputs, we call this a *combination function*.

2. A function which sorts the scoring values along the axis, given the type of the value (integer, real, character, etc.). Such a function is generally obvious and natural given the score and underlying attributes.

3. A function that discretizes and bins the values. This function is responsible for giving us the "solid block" model, as opposed to a scatter-plot field of points corresponding to each object. As shown in the presentation model diagram, such a function is critical for enabling presentation elements such as tabs or panels.

Note that the scoring function is the nexus between the scoring model and the presentation model. Given this model and the above functions, we can thus say the digital library deployer must specify the following items in preparing a QM-search system:

1. Number of axes (dimensions).

2. Grouping of attributes to axes.

3. Scoring/combination function for each axis.

4. Sorting function for each axis.

5. Discretization/binning function for each axis.

6. Presentational rendering.

Eventually we can envision the use of a tool such as 5SGraph (Zhu, 2002) to select values for these elements, with an attached generator to output a working QM-search system.

4.1.3 The Scoring Model

The scoring model is critical for mapping metadata attributes to presentation axes. The problem of scoring encompasses both the translation of explicit metadata fields into scalar scores, as well as inferring metadata attributes which are subsequently translated into scores. A great part of what the scoring subsystem does is to gather sparse information and make it dense.

In Table 4.2 we give some examples of object attributes, whether they are explicit (metadata fields) or implicit, where their information is derived from or originates, and the kind of scoring function one might expect to be built upon them.

Whether a scoring function is based on explicit or implicit attributes actually depends on the situation, and is not universal. For example, in Table 4.2,

attribute	type	data source	scoring function
rating	explicit	numeric ratings	AVG(ratings)
vettedness	explicit	peer review data / publication venue	count(reviewers) / trust(publisher)
citedness	implicit	citation links	Amsler, etc.
popularity	implicit	activation records	%age of views
granularity	explicit	containment data	1/0 (collection / item)
topical sim.	implicit	co-classification, activation / selection by users in same affinity group	sim(topic(query), topic(doc))

Table 4.2: Digital library object attributes and potential scoring functions, with attention to whether the attributes are implicit or explicit, and what the data source is. Note that all of this is merely illustrative, and would likely vary considerably from situation to situation.

"popularity" was classified as an "implicit" attribute. However, one could impliment this attribute as a simple count of views of a record, thus making it quite explicit. In this case, one would be losing some fidelity, as one could not distinguish between kinds of activation. An a posteriori estimation of popularity might instead be based on sophisticated analysis of log data, which would be more of an implicit rendering of the attribute.

Note that scoring based on implicit attributes *may* require offline computation, but that scoring based on *explicit* attributes never does. Thus, a vettedness-based score requires no offline computation if peer-review data is stored in a relational database as links between review records, people, and objects, but offline computation would be required if this information incorporated some latent element; e.g., if an impact factor of publication journals was used to estimate trustworthiness.

The scoring model goes to the core of why this project is called "quality metrics": it is because the scoring model has the ability to take latent and explicit digital library information and turn it into scores which represent object *quality*. In the next section we examine some techniques for performing scoring which we hope to exploit for building the most general, flexible, powerful scoring component of a search system to date.

4.2 Techniques for Quality Metrics Scoring

In this section we discuss scoring techniques which are good candidates for integration into the QM-search scoring model portion. Note that the first two, link fusion and implicit ratings, basically deal with inferring quality metrics

information from raw data. This must be done to address the common underlying problem of *low density of quality information*, or *high sparsity*. These frameworks address this problem by performing significant pre-processing and synthesis to produce *dense* quality information, which is more useful to digital library applications and their end users.

4.2.1 Link Fusion

Link fusion (Xi et al., 2004) is a generalized framework which casts object relationships as *links*. This simple but powerful concept is a generalization of the familiar web-and-hyperlinks setting, where Google has exemplified the benefit of deriving quality information from interlinkages between web pages. Thus in this view, web pages are the objects, and the relationships are hyperlinks, which are considered an "endorsement" judgment.

Link fusion extends this concept to different relationship types and other objects. For example, we could consider selection or activation of digital library records (culled from log data) as links between users and objects which make a value judgment. Further, we could consider ratings to be a weighted link between objects and users. Another common type of information we could consider are citation linkages between records, thus accounting for the probable higher quality of highly-cited items.

Essentially, any relationship between objects in the digital library which contains some latent information about quality can be integrated with the link fusion framework to yield some definite, inferred, scalar score value for objects.

Link fusion is implemented with an iterative, convergent augmented matrix method. This is similar to how PageRank (Brin and Page, 1998) or HITS (Kleinberg, 1999) are calculated, but with the integration of a larger number of adjacency matrices into the computation. We plan to explore the feasibility of utilizing the link fusion framework to implement the scoring subsystem of QM-search.

4.2.2 Implicit Ratings

A major problem in recommender systems and other information systems which rely upon upon value judgments to perform quality filtering is that explicit ratings feedback is sparse. This makes it difficult to cover all objects with valuation information, and further reduces the confidence considerably even for objects which are covered.

Digital libraries are only now beginning to incorporate explicit value judgment feedback, such as numeric ratings and vettedness values (moderation from experts). However, there is still relatively little information of this nature available, and what's more, it is not appropriate to capture this kind of information in many digital library and similar information systems. The workflows of many digital library scenarios simply do not allow it.

While we should hope to incorporate any such explicit value judgments if they are available, there is actually a greater body of *latent* valuation informa-

tion which could potentially be used to perform the same collaborative filtering tasks. This is the focus of (Kim et al., 2005), which demonstrates the feasibility of using *implicit* "rating information" (their general term for value judgments) in performing collaborative filtering.

In the study, when a user skipped, expanded, or selected an object, this was considered a rating of the object by that user (which was positive or negative, depending on the activity). This makes intuitive sense, as when the user is presented with a set of objects, clearly they will select objects of interest to them, and skip objects which are not. Herein lies the valuable latent quality-related judgment information.

The study statistically proves that this latent information is in fact valuable for performing collaborative filtering tasks, and provides a framework for capturing and making use of this information.

We are interested in integrating this sort of information in the scoring model portion of the QM-search framework, as it seems highly likely to bolster the quality-based filtering ability of the system. Further, the approach seems compatible with the link fusion computational framework, which adeptly handles the required entities of DL objects, users, queries, and more.

One challenge will be making this sort of personalized recommender system approach compatible with a more global collaborative filtering goal. While we would be happy to include a personalization aspect in the functionality of QM-search, such a thing may not be scalable to large production systems, with high volumes of objects, queries, and users. The specific connections between implicit quality information, topics, personalized information, and the link fusion framework will have to be explored.

4.2.3 Genetic-Algorithm Scoring Combinations

As observed in earlier previous sections, often times it is necessary in our model to produce combined scoring functions that are based on more than one metadata attribute. Such a function allows the mapping of many attributes to a single presentation axis. However, discovering the optimal way to combine many sub-scores into a single combined score is a nontrivial task, which in the past in similar situations has taken much guesswork, tweaking, trial-and error, and experimentation.

We plan to explore the use of genetic algorithm combined functions for the QM-search scoring subsystem. Genetic algorithms provide an automated search framework, which searches a parameter space for the optimal set of values for a list of attributes (in this case, weightings of scoring elements). Some promising work has been done in this area which we plan to investigate further (Fan et al., 2004).

Genetic algorithms applied in this scenario could afford a much higher degree of automation than would otherwise be available, and hence greatly simplify deployment of the QM-search system. There are precedents for the inclusion of genetic algorithms into large-scale information systems for the purposes of

finding optimal parameters, for example, the genetic algorithm query execution planner in the PostgreSQL system (Utesch, 1997).

4.3 Implementation Plan for Lucene

We have begun analyzing the Lucene codebase to determine how we would build QM-search onto it. Urvashi Gadi is the primary developer on this work.

Lucene is architected in a modular fashion so that different classes can be added to perform ranking in different ways. Lucene is based on an offline indexing process, so updating of nontrivial ranking support information which must be handled in batch mode should be easy to integrate.

The most difficult part of building QM-search onto Lucene seems to be the presentation layer. Lucene is basically hardcoded with a one-dimensional presentation model, based on scores and ranks in a linear list. We will have to study how to build more capable presentation semantics into Lucene's reporting subsystem, as well how to express these syntactically within its configuration engine.

5 User Studies

5.1 Focus Group Planning

The basic plan for conducting the focus groups is to introduce the broad problem we are trying to solve (as in the introduction to this report), expose the users to some of the metasearch systems surveyed above, and then collect their thoughts on how to improve the presentation of search results. Users will naturally have a variety of backgrounds, so that their responses will illuminate the needs of different scenarios and societies.

We plan to demo the various systems surveyed as well as let the participants play with them directly. We will also supply a number of questions to the group to structure the discussions, in addition to capturing ad hoc commentary. The sessions will be recorded and we will dynamically adjust our follow-up questions, as well as the plan for subsequent focus groups, based on the interactions.

Detailed plans for the focus groups will be developed by Rohit Chopra during September 2005.

5.2 Search Experiments Planning

The objective of the experiments is to test the efficacy of the QM-search system. In reality, this is a test of our model for quality metrics and retrieval presentation, as well as our implementation of the model.

The general structure of these experiments will be to compare our prototype QM-search system to available alternatives of the same nature by subjecting both to the same queries. By "alternatives of the same nature" we mean that the systems must be free and open source metasearch systems (or more general digital library systems containing a metasearch component), of the sort that digital librarians could conceivably deploy for their own low-cost projects. Many such systems (e.g. Greenstone, PKP, OAIster, unmodified Lucene) have been discussed above.

We will use a variety of metrics to determine the relative performance of the systems. In addition to metrics like precision and recall, we will asses qualitative satisfaction feedback from users. New metrics may have to be developed to more fully and naturally describe the performance of retrieval in a setting of presentations rich with quality information.

The detailed planning and execution of these experiments will be led by Vikram Raj during the fall of 2005.

6 Work Plan

The high-level tasks we must undertake to completion of this project are given below, by quarter.

6.0.1 Q3 2005

- Finalize selection of search engine upon which to build QM-search.
- Continue research into scoring model techniques.
- Continue to develop the scoring and presentation models.
- Detail how to map these models into the specific search system.
- Begin implementation of search system.

6.0.2 Q4 2005

- Complete search system implementation (initial version).
- Detail focus groups (gather exemplary system, define "seed" questions).
- Design experimental evaluation of QM-search system.
- Execute focus groups.

6.0.3 Q1 2006

- Execute focus groups (multiple rounds).
- Execute quantitative experiments.
- Refine search system.

6.0.4 Q2 2006

- Analyze and report on results of focus groups; work into theoretical models.
- Analysis of quantitative experiments.
- Refine search system.

6.0.5 Q3 2006

- Conclude analysis.
- Reporting (peer and project).

Bibliography

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998. URL http://citeseer.ist.psu.edu/brin98anatomy.html.

Weiguo Fan, Michael D. Gordon, Praveen Pathak, Wensi Xi, and Edward A. Fox. Ranking function optimization for effective web search by genetic programming: An empirical study. In *HICSS*, 2004. URL http://csdl2.computer.org/persagen/DLAbsToc.jsp?resourcePath=/dl/procee%dings/\&toc=comp/proceedings/hicss/2004/2056/04/2056toc.xml\&DOI=10.11%09/HICSS.2004.1265279.

Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997. ISSN 1384-5810. URL http://dx.doi.org/10.1023/A:1009726021843.

Seonho Kim, Uma Murthy, Kapil Ahuja, Sandi Vasile, and Edward A. Fox. Effectiveness of implicit rating data on characterizing users in complex information systems. In *European Conference on Digital Libraries*, 2005.

Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999. URL http://citeseer.ist.psu.edu/kleinberg99authoritative.html.

Aaron Krowne and Martin Halbert. Combined searching of web and oai digital library resources. In *Joint Conference on Digital Libraries*, June 2004. URL http://br.endernet.org/~akrowne/my_papers/metacombine_b1/p238-krowne.pd%f.

Martin Utesch. Genetic query optimization in database systems. In *Postgresql 6.3 Documentation*. 1997. URL http://www.postgresql.org/docs/6.3/static/c49.htm.

Wensi Xi, Benyu Zhang, Yizhou Lu, Zheng Chen, Shuicheng Yan, Huajun Zeng, Wei-Ying Ma, and Edward A. Fox. Link fusion: A unified link analysis framework for multi-type interrelated data objects. In *The Thirteenth World Wide*

Web conference, 2004. URL http://research.microsoft.com/research/pubs/view.aspx?pubid=1261.

Qinwei Zhu. 5sgraph: A modeling tool for digital libraries. Master's thesis, 2002. URL http://scholar.lib.vt.edu/theses/available/etd-11272002-210531/unrestri%cted/Thesis_5SGraph.pdf.