

**The Melvyl Recommender Project
Full Text Extension
Supplementary Report**

California Digital Library

October 2006

Table of Contents

1 Executive summary	3
2 Extension project overview	4
2.1 Rationale	4
2.2 Goals	4
2.3 Activity	5
3 Data Sources	6
3.1 Scanned books (OCA)	7
3.2 Digital books (eScholarship Editions)	7
3.3 Archival Manuscripts (OAC)	8
3.4 Repository Materials (eScholarship Repository)	9
3.5 For Further Exploration	10
4 Indexing and Linking	11
4.1 Text Extraction	11
4.2 Indexing	12
4.3 Linking and Merging	13
5 FRBR (Functional Requirements for Bibliographic Records)	14
5.1 Index- time vs. dynamic grouping	14
5.2 Technical underpinnings of dynamic method	14
5.3 Matching algorithm	15
5.4 Results	17
5.5 For Further Exploration	17
6 Cross- search Ranking	19
6.1 Initial Evaluation	19
6.2 Scoring Improvement	20
6.3 Field Weighting	20
6.4 Results	21
7. Spelling Correction	22
7.1 Impact of Uncorrected Text on Spelling Correction	22
7.2 OCA Texts: OCR problems	22
7.3 Repository texts: PDF problems	24
7.4 Solution	24
7.5 For Further Exploration	25
8 Keyword Searching	26
8.1 Original Approach	26
8.2 True Multi- field Keyword Search	27
8.3 Results	28
9 Other Improvements	29
9.1 Personalization / SQL Interface	29
9.2 Date Range Searching	29
9.3 For Future Exploration	30
10 Appendix: Screenshots	31
10.1 Basic Search (multi- field keyword)	31
10.2 Advanced Search (including date range)	31
10.3 FRBR Results (unexpanded)	32
10.4 FRBR Results (expanded)	33

The Melvyl Recommender Project: Supplemental Report

<u>10.5 Score Explanation</u>	<u>34</u>
<u>10.6 Single Record with Recommendations</u>	<u>35</u>

1 Executive summary

The aim of this Extension to the Melvyl Recommender Project has been to carry out deeper explorations into the most interesting and promising avenues opened during the original project, and to add obvious missing pieces of functionality.

The largest research area was investigating the impact of adding thousands of full text objects from various sources into the metadata-only index from the original project. A host of questions were raised:

- Where would the data come from, and what form would it take?
- Could the system handle so much data in a single index?
- Would one type of object dominate the others in query result ranking?
- How could we adjust scoring to get a good mix of results?
- What unexpected issues would arise?

We gathered data from the Open Content Alliance, from internal CDL sources (eScholarship Editions, and the Online Archive of California), and from the eScholarship Repository. We then worked through a host of issues to get them indexed and easily searchable. Duplicate and near-duplicate records led to the need for dynamic FRBR grouping.

After adopting a new keyword searching strategy and adjusting document scoring, we were indeed able to achieve a good mix and balance of documents for typical search queries. This work validates the idea of creating a mixed index of full text and metadata- only objects.

Aside from the major work above, an unanticipated impact of adding full text was dilution of our index- based spelling correction dictionary. The causes had to be analyzed and a solution was found.

Finally, a few general enhancements rounded out the extension project activities, including support for granular date range searches and enhanced personalization.

2 Extension project overview

The Melvyl Recommender Project has always been experimental in nature, and in pursuing the basic objectives, new questions were raised. The aim of this Extension project has been to carry out deeper explorations into the most interesting and promising avenues opened during the original project. In addition, we wished to add some obviously lacking functionality which would be appreciated by users.

2.1 Rationale

The original Melvyl Recommender Project¹ focused on answering several major research questions, one of which was whether a full text indexing system (such as XTF) could be successfully adapted to serve a catalog containing millions of metadata records. The answer was a resounding “yes”, and a logical follow-on question presented itself: Since we have a full text system, what would happen if we added full texts to the millions of metadata records?

Libraries are gaining access to more and more full text objects, either as born- digital files or as scanned and OCR'd texts. They may come from publisher arrangements, internal projects, faculty repositories, inter-library sharing and consortia, and even targeted web crawls to collect knowledge about a particular domain.

Yet search facilities for these full text collections are often fragmented, obscure, and of widely varying levels of accessibility, features, and integration with other library services that patrons have come to rely upon.

It seems obvious then to combine full text documents into the main library catalog (which is typically metadata- only) to enable patrons to discover both digital and offline materials in a single, unified interface.

Assuming a successful melding of the two types of content, a unified system provides tremendous advantages. Development work on new features can benefit an entire user base instead of fragmented groups using small access portals.

2.2 Goals

The main focus then of this extension project has been to assess the technical landscape associated with integration of full text and metadata-only resources, exploring their intersection in our prototype OPAC

¹ See the California Digital Library page for the Melvyl Recommender Project: http://www.cdlib.org/inside/projects/melvyl_recommender/

The Melvyl Recommender Project: Supplemental Report

interface. We have also identified (and implemented to the extent that time allowed) applications enabled by this integrated dataset.

Here then are the goals the Recommender team initially set out to accomplish in the Extension project:

1. Data Sourcing and Integration
 - (a) Sources: where are large text repositories likely to come from, and what file formats or specifications are they likely to carry?
 - (b) Linking: what are the mechanics of linking with bibliographic records?
 - (c) Interface synergies: possible interface improvement through data mining: Table of Contents, Citation/Reference mining, Enhanced similarity- based recommendations.
 - (d) Integration: Address how to technically integrate likely full text resources into a catalog of indexed bibliographic data.
2. Cross- Search Tasks
 - (a) Ranking problems: identify scoring problems with result sets from mixed metadata and full text dataset.
 - (b) Optimize ranking: adjust the scoring formulas to achieve a good balance of document types in the results (so full text doesn't dominate simply because of the length of the content vectors).
 - (c) Recommendations: expand the original Recommender project's "more like this" query to work on full text.
3. Features for Merged Data
 - (a) Personalization: supporting user accounts and a persistent "book bag".
 - (b) Spelling correction: explore ways to improve results of index-based spelling correction in the case of multiple- word queries.
 - (c) FRBR: Improve integration of FRBR (Functional Requirements for Bibliographic Records) into the prototype interface, and explore improvements to the work set linking algorithm.
 - (d) Keyword searching: implement a more consistent single- box query that searches for all the terms in any of the specified fields.
 - (e) Date range searching: overcome known inefficiencies and limitations in the current date search capabilities of the prototype.
 - (f) Performance: resolve glaring speed and responsiveness problems that arise.

2.3 Activity

Over the past several months (July 2006 to the present) staff members of the Melvyl Recommender Project have been engaged in the three major strands of work outlined above, often switching from one to another as

The Melvyl Recommender Project: Supplemental Report

research in one area led to insights in another, or as blockage in one area forced the team to re- think its approach.

Inevitably in research work of this kind, some topics will consume more time than others. In the current work, due to time constraints, a few of the goals above were touched on only cursorily or not at all. We made decisions in order to maximize progress on the major goals.

3 Data Sources

The data we began with in this extension project was the set of bibliographic records from the original Recommender project, approximately 5 million MARC records from the catalog at UCLA. To this we added an additional 5 million records from UC Berkeley. While the resulting total of 10 million catalog records comes nowhere near the 28 million in the production Melvyl catalog, it was a significant increase for our system and significant test of our its capabilities.

The next step was to identify sources for full text documents that we could add to this bibliographic database. Sources we considered included the following:

- ◆ Open Content Alliance (OCA): texts hosted by the Internet Archive.
- ◆ eScholarship Editions: born- digital works already in- house at CDL.
- ◆ Online Archive of California (OAC): archival manuscripts.
- ◆ eScholarship Repository: preprints, journals, and working papers contributed by UC faculty and graduate students.
- ◆ Google Print/Google Scholar: print volumes.

Due to time constraints during negotiations with Google for permission to use their texts, we were unable to incorporate any of the Google Print/Google Scholar books.

The remaining sources (OCA, eScholarship Editions, OAC , and eScholarship Repository) provided 18,809 full text documents, with a good mix of born- digital works, scanned books, historical documents and repository materials.

Though the number of full- text documents might seem insignificant in comparison to 10 million bibliographic records, we felt that this ratio resembles the state of affairs in real- world collections. While it's clear that digital objects are now being created or scanned at a high rate, the sheer number of existing non- digital works overwhelms them, and we anticipate this situation will continue for some time. Thus, it made sense to investigate a hybrid discovery system based on a relatively small number of digital objects in a large index of bibliographic records.

The following sections describe each data source we used for the combined metadata and full text index.

3.1 Scanned books (OCA)

Description of collection (topic areas, origin)

The University of California is a major participant in the Open Content Alliance's² effort to build a permanent archive of digitized text. For the past year they have been scanning and OCRing books from their collections. The resulting digital objects are hosted by the Open- Access Text Archive³, part of the Internet Archive.

We selected one month's worth of scanning output, the topic areas for which happened to partly coincide with those used for assessment in the original Recommender project:

- ◆ American literature and poetry
- ◆ American history and biography
- ◆ Mathematical texts

Quantity and format of documents

It is difficult to judge the number of documents available as there appears to be a variable time lag between when a document is scanned and when it becomes incorporated into counts and searches.

Documents were made available as searchable PDF files, individual page image scans, plain OCR text, or XML text marked up with OCR data (line/word boundaries, etc.)

Using an in-house script, the team harvested 3,774 recently scanned texts from the Internet Archive web site. We chose to download the plain (un-annotated) OCR text, along with meta- data in MARCXML and Dublin Core (DC) formats.

Metadata

Metadata are acquired via z39.50 queries to the UC Berkeley catalog and associated with the scans. They are then made available for download in several formats, including MARC, MARCXML, and Dublin Core.

Acquisition and projected growth

Texts were being scanned at a rate of several thousand a month; in May we know that more than 3,700 were added to the archive.

Permissions/copyright status

² See the Open Content Alliance site: <http://www.opencontentalliance.org/>

³ See the Open- Access Text Archive site: <http://www.archive.org/details/texts>

The texts are presumed to be out of copyright; metadata contains notes on copyright status and proof of status for each item.

3.2 Digital books (eScholarship Editions)

Description of collection (topic areas, origin)

The eScholarship Editions collection includes almost 2000 books from the University of California Press on a range of topics, including art, science, history, music, religion, and fiction. As the collection is hosted directly by CDL, it was a very simple matter to acquire the texts and metadata for our prototype.

Quantity and format of documents

As of June 2006, the Editions hosted 1,846 monographs, all of which are encoded as XML according to a local modification of the TEI (Text Encoding Initiative) DTD.

Metadata

Originally dumped from a UC Press Filemaker database into METS, the metadata records are sampled and turned into Dublin Core records. There are also MODS records taken from Melvyl bibliographic data.

Acquisition and projected growth

Not currently acquiring new texts.

Permissions/copyright status

Access to the electronic books is open to all University of California faculty, staff, and students, while select books (518) are available to the public. Print versions of many of the electronic books can be purchased directly from the publishers.

3.3 Archival Manuscripts (OAC)

Description of collection (topic areas, origin)

A core component of the California Digital Library, the Online Archive of California (OAC) is a digital resource that facilitates and provides access to materials such as manuscripts, photographs, and works of art held in libraries, museums, archives, and other institutions across California.

The OAC includes a single, searchable database of "finding aids" to primary sources and their digital facsimiles. Primary sources include letters, diaries, manuscripts, legal and financial records, photographs and other pictorial items, maps, architectural and engineering records, artwork, scientific logbooks, electronic records, sound recordings, oral histories, artifacts and ephemera.

Since the collection is housed internally at CDL, it was simple to acquire the documents and metadata for our prototype.

Quantity and format of documents

For this project we selected a subset of 1,422 primary source documents from the OAC selection, these being all the documents that were encoded in TEI format.

Metadata

The metadata were available to us in METS and Dublin Core formats.

Acquisition and projected growth

The OAC is constantly growing, albeit at a inconsistent rate.

Permissions/copyright status

The contents of the OAC database are made publicly available by the collection- holding repositories for use in research, teaching, and private study. Whenever possible, the OAC and the collection- holding repository provide available information about copyright owners and other restrictions in the catalog records, collection finding aids, and other metadata associated with digital images, texts, audio and video recordings.

3.4 Repository Materials (eScholarship Repository)

Description of collection (topic areas, origin)

The eScholarship Repository⁴, sponsored by the California Digital Library, provides persistent access to scholarly work and makes the content easily discoverable. It is a project of the eScholarship initiative of the California Digital Library within the University of California Office of the President. The Repository is powered by tools from the Berkeley Electronic Press, which UC licenses on behalf of its faculty. This infrastructure enables the rapid and low- cost creation, management, and dissemination of journals, peer- reviewed series, working papers, discussion papers series, and other electronic forms of scholarship by UC researchers.

- 1,443 Postprints (previously or simultaneously published elsewhere)
- 503 Journal articles (i.e. original articles published within eSchol journals)
- 1,101 UCIAS Digital Collection (papers, articles, contents of edited volumes, etc)
- 11 UC Press Monographs (in series)

⁴ See the eScholarship Repository site. <http://repositories.cdlib.org/escholarship/>

The Melvyl Recommender Project: Supplemental Report

~80 Seminar papers (within series)

8,887 Working papers, tech reports, preprints, field notes, etc.

Quantity and format of documents

As of June 2006 there were 12,025 documents available, all in PDF format.

Metadata

Metadata are available in XML (Dublin Core DTD) and as Bib Export. Most are produced by the depositors. Postprints metadata are harvested from Web of Science and automatically uploaded.

Acquisition and projected growth

Postprints are deposited directly by authors or their proxies. Everything else is deposited by department and research unit administrators acting on behalf of their faculty.

The repository had 10,000 papers on Dec 8, 2005. Exactly six months later it had risen to 12,025. That annualizes to a rate of 40- 50% growth per year, but could be influenced by many variables that could swing the growth sharply upward or slightly downward.

Permissions/copyright status

UC regents have permission to store, display, preserve, etc. all objects. Authors retain copyright.

3.5 For Further Exploration

Crawled web archives were not represented in the prototype system, and these might be an interesting and probably challenging addition to any merged catalog.

4 Indexing and Linking

Having identified and gathered the diverse set of documents and bibliographic records we intended to use, we placed them all in a single file system hierarchy with consistent file naming.

Our next step was to index them all together in such a way that they could be queried as a single set, and the results displayed on an integrated result page. The resulting index formed the foundation of the remaining work of the project, enabling the team to explore methods and features, and to answer the various research questions.

Creating the index involved steps of extracting indexable text, indexing the data, and linking or merging duplicate records.

4.1 Text Extraction

In order to create a full text index, we needed text from each document. This was simple in the case of the TEI-encoded texts from eScholarship Editions, and TEI-encoded historical documents from OAC, since we already index these collections using XTF at CDL.

Only slightly more difficult were the scanned texts from the Open Content Alliance. XTF already had built-in support for text files, but we discovered its handling of unusual Unicode characters was insufficient. In addition, it was not properly escaping characters that aren't allowed directly in XML documents. Once these deficiencies were corrected, the OCA documents went smoothly (if slowly, due to their size) into the index.

The final collection of full text objects, PDF files from the eScholarship Repository, illuminated similar difficulties with Unicode and XML character handling. In addition, XTF had been using an outdated version of the open-source PDF conversion library PDFBox⁵, which proved to have several bugs and gross inefficiencies that had since been corrected in later versions. Upgrading PDFBox proved to be a simple matter.

Since PDF conversion was quite time-consuming, we elected to run it over all the repository documents in a pre-pass, storing the plain text files in our file system hierarchy. This made indexing runs faster by avoiding the PDF conversion step.

We encountered a few other interesting problems when converting the PDF files. Of the 11,771 PDF texts, 811 (or about 7%) didn't result in a significant amount of extracted text. Here's a breakdown of the causes:

⁵ See the PDFBox site: <http://www.pdfbox.org/>

- 681 Documents that actually consisted only of scanned (but not OCR'd) typewritten pages, most with a small extractable abstract at the beginning.
- 86 Various errors in PDF conversion, some apparently due to bugs in PDFBox, others of unknown origin.
- 44 Authors set a “text extraction protection” bit in these PDF files, signaling that readers are not allowed to extract text. Since PDFBox respects this flag, we were unable to index text from these documents.

It should be noted that automated conversion to text, both by the OCR process and by PDFBox, resulted in an explosion in the number of unique words, which in turn adversely affected spelling correction. This is addressed fully in section 7. Spelling Correction, below.

4.2 Indexing

Once the full text was made available for each document, getting all the documents to be compatible was mainly a process of metadata normalization, mapping fields from the various flavors of MARC, METS, and OCLC into the common format we had established for the original Recommender prototype. While this process was non-trivial, it was relatively straightforward and the details will not be covered here.

With a metadata mapping established, it was time to attempt the main index. We had never indexed so many records of such size before, and we encountered and overcame several problems in the process.

- ◆ RAM: Initial slowness and hanging were due to not allocating enough RAM to the indexing process. Since the prototype machine has 16 gigabytes of RAM, allocating 4 gigabytes instead of the original 1 gigabyte solved this problem.
- ◆ Time: A full run on a single processor took around 40-45 hours. To speed repeated index runs, we opted to try parallelizing the index process, running four different data sets into separate target directories, and then implementing an interface to Lucene's existing index merge code to enable merging the indexes together. With some XTF-specific modifications, this ended up working and enabled us to complete full runs more quickly.
- ◆ Disk space: A final problem encountered was the size of the final index, and the amount of overhead Lucene requires during the indexing process. Because of the way Lucene progressively adds

The Melvyl Recommender Project: Supplemental Report

segments and merges them together, it needs working space of about three times the final index size. Given the need to preserve a previously working index while experimenting with a new one, we had to juggle files among drives to reserve the 75 gigabytes required for a new index run.

To help future experimenters and implementors, tips on anticipating and addressing these and other common problems will be added to a new documentation section (“Tips & Tricks”) for XTF adopters.

The following table details final index composition:

Source	Text Format	Metadata Format	Record Count	Total File Size (GB)
UCLA Melvyl	n/a	MARC	4,931,338	4.46
UCB Melvyl	n/a	MARC	5,293,990	4.60
OCA	plain text	MARCXML + Dublin Core	3,774	1.66
eSchol Editions	XML (TEI)	METS	1,846	2.21
OAC	XML (TEI)	METS	1,422	0.27
eSchol Repository	PDF	OCLC (XML)	11,767	0.78 (13.89 before PDF to text extraction)
<i>Total</i>			10,244,137	13.98
<i>Final index</i>			10,244,137	25.26

4.3 Linking and Merging

A common problem faced by public access catalogs is that of linking or merging essentially duplicate records, and we faced the same problem. For instance, a given eScholarship Editions text might also be represented by several records in the UCLA catalog and other records in the UCB catalog.

However, our strategy early on was to try to skirt these issues as much as possible, seeking a “quick and dirty” linking solution so that we could get on with investigating our primary research goals. Sadly, our simplistic linking attempts failed in the face of the widely varying metadata in the collection.

For example, we hoped to link OCA records to those from Melvyl by using Gladys identifiers, since our understanding was that the OCA metadata had come from querying Gladys. This assumption was faulty, and only a small portion of the OCA records possessed Gladys identifiers; further,

The Melvyl Recommender Project: Supplemental Report

they were not clearly marked and could not always be positively identified as such.

Similarly, we hoped to link eScholarship Editions by ARK identifiers, since UCLA and UCB records often contain the ARKs. Again, this did not pan out as many records in both catalogs simply didn't fit the mold.

In the end, we had to give up on any kind of identifier linking, and our other simplistic linking ideas. Still, we needed to build a system with a unified, fairly easy to understand, merged display of query hits from all the data sources.

The solution? FRBR. Our original thought was to investigate FRBR late in the project, but we now realized that we could use the technique of “work sets” to address our linking problems, and so FRBR suddenly took the front seat. This work is covered in the next section.

5 FRBR (Functional Requirements for Bibliographic Records)

In our initial proposal for this extension project, we thought to further, if time permitted, the work we had done on dynamic FRBRization in the original Recommender project. It was thought that FRBR⁶ would be complimentary to our research goal of exploring a merged full text and bibliographic database.

However, as the data rolled in, we realized (as detailed in the section above) that FRBR would not only be complimentary, it would be crucial to the project. It turned out to be the only feasible way we found to achieve a merged query display, as metadata inconsistency simply didn't allow simple identifier linking schemes to work.

5.1 Index- time vs. dynamic grouping

Our previous approach had been to dynamically “FRBRize” a set of results, and this work was done entirely in the result formatter stylesheets. This approach had a large disadvantage of being fairly slow, and only merging the most “relevant” records (by score). For example, if one record scored highly and another similar record scored lowly, the low record wouldn't appear in the results at all. Other features were difficult to implement and thus were broken in the prototype, such as sorting by title or author, and paging through result sets.

For this extension project, we made FRBR more and more intrinsic to our approach, and thus had to address these deficiencies. One possible approach would have been to “statically” determine work- set groupings at index time, as is suggested by the FRBR Work Set Algorithm⁷ we originally started with.

But static grouping carries a significant disadvantage: it is difficult to experiment with variations on the algorithm, because one must wait for an entire index run to see the results. Especially given the size of our index and the time and space required to create it, we felt strongly that a more dynamic approach would be desirable. We needed an approach that allowed fast, iterative experimentation.

One disadvantage to dynamic grouping is that the groups generated are not “commutative.” Groupings are only formed for documents which match the user's query, and thus a slightly different query could produce different results. In particular, a group might be missing several documents that would naturally fall into it, simply because those

⁶ See the FRBR Report: <http://www.ifla.org/VII/s13/frbr/frbr.htm>

⁷ For more on the Work Set algorithm, see: <http://www.oclc.org/research/software/frbr>

documents didn't match the query. Static index-time grouping would prevent this and make stable groups, bringing in nearly identical records even if they didn't match the query. However, we felt that for our experimental system, the advantages of fast iterative turnaround far outweighed the disadvantage of unstable groups.

5.2 Technical underpinnings of dynamic method

We determined to try a more deeply integrated version of dynamic FRBRization. Instead of embedding it in the stylesheets, we would take it one level down and code it in Java, drawing on the raw Lucene index. The relative speed-up, it was hoped, would allow the entire result set to be considered in grouping (not just the top 200 hits) without overly slowing query responsiveness, and allow us to re-enable features that users are accustomed to (such as sorting results by title, author and date, and paging through result sets).

One might wonder how it could be possible to quickly group upwards of 50,000 record hits for a typical query like "china". The answer lies in the virtues of modern cheap hardware, and particularly in inexpensive RAM.

The basic idea was to create an in-memory random access table of just the essential elements from each record: title, author, date, and identifiers. The table would be loaded at start-up using the Lucene index as a source, and cached for subsequent requests.

Using such a table, we were able to easily change and experiment with ways of grouping records using the metadata, only incurring the table load time at start-up. This strategy proved very successful and easy to implement.

Some minor space optimizations were needed to fit the table in memory and avoid overloading the Java garbage collector. For instance, we used arrays of integer offsets instead of object references, and stored text strings in blocks of 8-bit bytes instead of 16-bit characters. For the merged data set, the in-memory table currently occupies 1.39 gigabytes of RAM, comfortably within the capabilities of our 16 gigabyte machine.

5.3 Matching algorithm

The dynamic FRBR code is an extension of XTF's existing support for faceted browsing. Whereas normally a facet is drawn directly from index data, the new code supplies the facet data dynamically, based on the particular records selected by the user's query. Some significant refactoring and generalization of the facet code was required to transparently support both modes of operation.

The Melvyl Recommender Project: Supplemental Report

The original FRBR Work Set algorithm depends on essentially “exact” matches, but we found this to be too restrictive, as it would fail to group documents that were clearly “the same”.

The Melvyl catalog employs a fairly complex score- based algorithm ⁸ to link duplicate records. We experimented with a simplistic variation of this approach and arrived at a solution that produced good results. Here then is the essence of the algorithm we currently use to group record “hits” into work groups.

doc- match algorithm:

```
begin
  For each hit document D1, look for matching hits to group with it:
    Look the D1.title up in the in- memory sorted title index
    Scan forward and backward for titles T while title- match(D1.title, T) is true
      For each document D2 with title T:
        If D2 is already in a group, skip it.
        Calculate score by matching D1 fields against D2 fields:
          st = score- title- match(D1.titles, D2.titles)
          sa = score- author- match(D1.authors, D2.authors)
          sd = score- date- match(D1.date, D2.date)
          si = score- id- match(D1.ids, D2.ids)
          total = st + sa + sd + si
        If total >= 150, add D2 to D1's FRBR group
end
```

title- match algorithm:

```
begin
  Remove any subtitles (strings beginning with “:”)
  If the results match exactly, return true, else return false
end
```

To determine if two candidate documents belong in a group together, a score is calculated by comparing their titles, authors, dates, and any available identifiers. Note that documents often have more than one title and more than one author, so these computations involve lists, not just a single item.

Title	All titles match exactly	+100
	All titles match after subtitles are removed as above	+80
	One list is a (nonempty) subset of the other	+80
	No match	- 100

⁸ For details on Melvyl's record merging algorithm, go to <http://melvyl.cdlib.org> and click on “Help”, then select “How Records are Merged”.

The Melvyl Recommender Project: Supplemental Report

Author	All authors match exactly	+100
	Keyword match (all words in shorter author match longer author), for all authors in list	+80
	One list is a (nonempty) subset of the other	+80
	No match	- 100
Date	Exact match	+50
	+/- two years	- 25
	No match	- 50
Identifier	All identifiers match exactly	+100
	One list is a (nonempty) subset of the other	+80
	No match	0
Total score	Minimum threshold for match	150

These values were arrived at by looking at a sampling of data records and forming some basic assumptions about how items should be linked, trying out some weights, and tweaking them to achieve good results. However, more “twiddling of knobs” would probably be required for a satisfactory production system, and indeed it's likely that a number of new “knobs” would be needed to handle corner cases.

Initially we used a broad definition of what the “title” of a document was. However, this ended up mixing main titles, series titles, subtitles, and sometimes other random bits of information in, depending on the particular MARC fields and how they were utilized in the data.

To circumvent this problem, the in-memory table now records the source of each data item (e.g. the particular MARC field it came from) and we only consider matches between values taken from the same field. For non- MARC metadata we simply filled in default MARC field numbers.

5.4 Results

The dynamic FRBR algorithm executes within acceptable time bounds (introducing a noticeable but short pause in query processing), and it produces good FRBR groupings. It is the default mode in the prototype

interface, and formed a good foundation for the remaining work in the project.

There are cases when a dozen or more very similar records fall into the same group. We felt that these additional records would add little information for the user who was skimming the result set, but would add value for items in which the user expressed additional interest. So we implemented a simple user interface trick to keep things clean: initially the display shows only the first (most highly ranked) item in a group, along with a link to see the remaining items. This is implemented in JavaScript in the client's browser, and thus clicking on the link results in instant expansion of the group.

With that important interface enhancement, we felt we had arrived at a useful, merged catalog containing full text and millions of bibliographic records.

5.5 For Further Exploration

The current algorithm attempts to form groups that represent a single FRBR “work”. It would be interesting to pursue a two level decomposition of the records into “work”, and “items” within each work. It is unknown whether the metadata would support such a decomposition.

Certainly further tuning of the matching algorithm would be necessary and desirable if it were to be used in a production system. Inevitably there will be many corner cases that result in poor groupings using the current simplistic algorithm. Additionally, the creators of Melvyl developed a separate, somewhat different algorithm for grouping serials (as opposed to monographs), and it seems likely we would discover the need for this as well.

Though our project was able to obtain Library of Congress authority files, which are generally considered a necessary step in FRBRization, we ran out of time to integrate them into our FRBR process. Certainly this should be considered as a likely way to improve grouping for that fraction of documents that match the authority files (and a baby step would be to quantify that fraction.)

6 Cross-search Ranking

Having gathered our data, merged and indexed it, and dealt with duplicates and overlaps by FRBRizing, the team was able to answer these questions:

- How well would our current ranking algorithm perform on the mixed data set?
- Would full text hits dominate metadata- only records? Would the opposite occur?
- What adjustments of the scoring algorithm itself would be necessary?
- Is the mixed index a promising idea in the long term?

6.1 Initial Evaluation

It became immediately apparent that the scoring algorithm we started with in XTF produced inadequate results. In general, documents for which full text was available dominated the search results.

We turned on the “Analysis mode” of the Recommender prototype interface to look at the raw score that XTF calculates for each matching record. This confirmed the source of the problem: scores for full text documents were typically 10 to 100 times larger than scores for metadata- only records.

To probe why this might be, we enabled the “explain scores” option in the interface. Unfortunately, the score explanations were difficult to use as they often included hundreds or thousands of different factors for each document.

But we observed that since the score computation is essentially a series of chained iterate- sum procedures, it breaks down into a natural hierarchy. A fairly simple modification to the user interface allowed us to show only the top level of the score explanation, with a link to expand each of the components that made up that score. Each of those in turn contained an expansion link. This made it possible to progressively explore and grasp how the score was computed for each document, without becoming overwhelmed by the details.

Below is a procedural view of the XTF/Lucene score computation we started with. For clarity this example has been simplified to consider a single- term query, and some miscellaneous score factors have been removed.

(Note that “idf” represents the standard “inverse document frequency” from Information Retrieval parlance.)

score(doc, term)

```
begin
  score = 0
  for each metadata field:
    freq = num_matches(doc.field, term) * idf(term, all-docs(field))
    score = score + sqrt( freq / sqrt(field_length) )
  for each matching chunk in full text:
    freq = num_matches(chunk, term) * idf(term, all-chunks)
    score = score + sqrt( freq / sqrt(chunk_length) )
  score = score * number_fields_matched / number_fields
end
```

6.2 Scoring Improvement

Two problems became apparent as we examined the foregoing computation.

The first problem (visible on the final line of the procedure) is the “coordination factor” as it's called in Lucene. This factor is meant to favor query results where all of the fields specified in a user's query were matched. However, our users enter a simple term and the complex multi-field query is constructed for them. It doesn't seem helpful to favor matches over many fields over better matches in fewer fields. To address this problem, we simply disabled the coordination factor for multi-field keyword queries.

The second, more serious, problem has to do with field size normalization. For metadata fields, this was taken care of by $\sqrt{\text{field_length}}$, but length normalization wasn't working correctly for full text hits.

To understand why, one must grasp “chunking” in XTF. Full text documents are broken into hundreds or thousands of 200-word chunks; this makes Lucene searching and hit highlighting efficient. However, the score computation above only normalizes scores based on the size of each text chunk, not on the total size of the full text.

To address the length normalization problem, we took the term's frequency in each chunk and divided it by $\sqrt{\text{num_chunks_in}(\text{doc})}$. The result looks like this:

$$\begin{aligned} & \text{freq} / \sqrt{\text{chunk_length}} / \sqrt{\text{num_chunks_in}(\text{doc})} \\ = & \text{freq} / (\sqrt{\text{chunk_length}} * \sqrt{\text{num_chunks_in}(\text{doc})}) \\ = & \text{freq} / \sqrt{\text{chunk_length} * \text{num_chunks_in}(\text{doc})} \end{aligned}$$

This matches the length normalization for metadata fields, since $\text{chunk_length} * \text{num_chunks_in}(\text{doc})$ is approximately the length of the full-text “field”.

6.3 Field Weighting

With length normalization in place and the “coordination factor” disabled, scores started to make much more sense, and result pages began to show a healthy mix of full text and bibliographic records.

However, the relevancy of the returned documents still seemed to favor the full text items somewhat, and looking at the score breakdown revealed the reason: we went too far in equalizing the scores. In reality, hits in the title of a document should probably be considered more relevant than hits in the full text. So we needed a mechanism to adjust the weights of the various fields in determining the final score.

This was a simple addition, and with some experimentation we arrived at a set of weights that seem to give good, relevant results (for the limited set of queries run). The fields searched for a keyword query and their final weights were:

Field	Weight
text	0.5
title- main	1.0
author	1.0
subject	0.5
note	1.0

The process of adjusting the set of fields and their weights to achieve satisfactory results was remarkably easy, requiring only a few iterations.

6.4 Results

Our work on ranking analysis and scoring adjustment resulted in what we deem to be a usable and useful catalog system. We have shown that it is quite feasible to mix full text and metadata- only records, and to facilitate discovery of both kinds of resources in a single unified result page.

Further, we have validated the findings of the original Recommender project: that a text- based search engine such as Lucene can be used to perform meaningful large- scale experiments, and can form a solid basis for the next generation of Online Public Access Catalogs.

7. Spelling Correction

We had originally thought to extend the index-based spelling correction strategy developed during the first phase of the Recommender project by extending it to handle multiple word spelling correction. What we hadn't anticipated was the fact that full text content has important implications for index-based spelling correction. Because of this we ended up not having the time to pursue the multiple-word aspect, but still felt the time was well spent in adapting single-word correction to work well with indexes of full-text content.

7.1 Impact of Uncorrected Text on Spelling Correction

Our strategy of providing spelling corrections based on the data from a live index worked quite well in the initial recommender system, and we believed it would continue to work without significant changes.

What we hadn't anticipated, and what seems obvious in retrospect, is that index-based spelling correction depends heavily on the spelling quality of the source data. Our initial data set, MARC-formatted catalog records from UCLA, was of uniformly high accuracy. Certainly there were errors, but at a fairly low level such that the algorithm could use frequency data to filter out most of the outliers.

When we introduced large sets of unedited data, particularly the automatic OCR generated texts in the OCA archive, and the PDF converted text from eScholarship Repository, we immediately encountered problems.

First, the number of unique words increased by orders of magnitude, greatly increasing the time and space required to generate the spelling correction index. In fact, it was impractical to generate the index at all based on the full data set.

Second, OCR and PDF conversion are simply error-prone processes. Worse, the mistakes are often systematic, skewing frequency data and undermining the efficacy of our index-based spelling correction algorithm. Without good frequency data, that algorithm fails to choose the “correct” replacement for a misspelling, often suggesting nonsense or partial words.

The problems with each data set warrants closer examination.

7.2 OCA Texts: OCR problems

To identify the source of problems with the OCA texts, we ran a small index of 1041 texts taken roughly at random (identifiers starting with 'a'

The Melvyl Recommender Project: Supplemental Report

- 'd'). A dump of term frequencies from the resulting 'text' field produced 749,235 unique terms. Of these, 39,698 contain numbers and were automatically excluded from the spelling correction dictionary. But 700,000 terms seemed an uncomfortably high number.

Spot checking revealed a large number of invalid words, apparently attributable to OCR. The errors fell into two major classes: (1) basic OCR, and (2) hyphenation. In addition, many foreign words and archaic spellings of English words were present, but these couldn't be considered errors.

Here are some examples of basic OCR errors:

- ◆ In <http://www.archive.org/details/washingtonworks08washrich> each page begins with the chapter name in a decorative font which OCR was unable to read. For instance, "Bonneville's Adventures" was rendered as "JBonnepille s a&ventures" and "ffionneville s a&ventures".
- ◆ In <http://www.archive.org/details/deobligationecon00sanduoft> the word "SUMMARIUM" was sometimes rendered as "SUMMARITJM". This likely wasn't due to slightly unusual kerning.
- ◆ One text, <http://www.archive.org/details/anexplanatorydef00keatuoft>, contained large bodies of text in old Irish script. This completely useless OCR results such as "tlAon 1. ^\n - ojACAm 0.5 A mAicpToo A bpeACAit>c tx nb AUATO HIAICCO ACA".

A more troubling group of errors were related to hyphenated words. At least in the text files available at the time of our data gathering, if a word on a scanned page was broken between lines with a hyphen, it was not automatically reassembled, and in many cases the hyphen wasn't retained at all, preventing post-process reassembly. Here's one section of the word dump:

ricain	ricals	ricanante
ricaine	ricamente	ricane
ricaines	rican	ricanement
ricains	ricana	ricanements
rical	ricanai	ricanes
rically	ricanant	ricans

One can readily supply the missing prefixes for some of these, such as "amer- ", "af- ", and "numer- ".

Finally, there were many Latin words and English words spelled archaically. It's debatable whether these should be included in a spelling correction index, but there is a good arguments that they should.

In all, we estimated that 25- 50% of the terms would be considered by most people to be misspelled. While the frequencies of most of these words were low, so too were the frequencies of many correctly spelled but infrequent English words. It's difficult for a spelling correction algorithm to tell the difference, and we concluded that the OCR'd OCA texts weren't appropriate for our spelling correction dictionary.

7.3 Repository Texts: PDF problems

The bulk of the incorrect terms in the spelling correction index were due to incorrect conversion of text within the PDF files obtained from the eScholarship Repository. We had originally converted the PDF files using a free library called PDFBox, which has gone through several revisions and is considered fairly reliable.

While PDFBox did a creditable job with most texts, it performed very poorly on a small subset of the texts, estimated at 3- 5%. On these texts, the output is severely garbled, with long strings of nonsense words.

The problems encountered seemed to fall into two categories, the first due to PDFBox incorrectly breaking words, the second to strange encoding.

An example of the first type is the word "butwhatkindsofpeopleserveonthem" in the document <http://rec-proto.cdlib.org/data-esr/i/issr1071/>. In this case, PDFBox failed to detect the word separations properly. As a check, when we copied the same section of text using Adobe Acrobat Reader, the words were correctly separated.

An example of the second type can be found in <http://rec-proto.cdlib.org/data-esr/u/uclastat1195/>. Here PDFBox output long strings of mixed characters and numerals, such as "a197a105a104" and "a200a107a106a65a30a196a159a188a98a185". In this particular instance the words contained numbers and so would be filtered out in the normal process of building the spelling correction index, but this was not always the case. The relevant section of text looked fine on- screen in Acrobat, but Acrobat Reader also produced garbled strings when we copied the relevant section to a text editor. So there may have been some encoding

irregularity, either intentional to protect the data, or unintentional resulting from the process of producing the PDFs.

Regardless, together these two translation problems produced millions of incorrect words from the Repository texts, and filtering out numeric words only eased the problem slightly. Clearly, we could not use the output of either PDFBox or for that matter Acrobat Reader in creating spelling correction dictionary.

7.4 Solution

At this point it became obvious that the source for spelling correction data would have to be human- entered or human- verified textual data. Data generated automatically (either by OCR or via conversion from PDF files) simply isn't reliable enough to be the source for a spelling correction dictionary.

To apply this to the data at hand, we added an option in the XTF text indexer to mark arbitrary sections of text for exclusion from the spelling correction dictionary. Then we added this to the text portions of PDF and OCR documents (we still considered the meta- data to be reliable.) This resulted in a manageable spelling correction dictionary that makes quality suggestions (at least for the set of misspelled queries we entered.)

While this is hardly an ideal solution, it does preserve some of the benefits of index- based spelling correction (for instance, retaining author names and titles because meta- data still contributes to the dictionary). Unfortunately it omits thousands of valid, correctly spelled words present within documents from the dictionary. However, given no programmatic way to separate the translation errors from the valid words, this was the only avenue open to us given the content selected.

7.5 For Further Exploration

It should be noted that it's likely a small number of texts (such as the one written in old Irish script) probably contribute a large share of the misspelled words. One topic for research might be to try to identify these statistically and filter out these texts.

8 Keyword Searching

Due to the influence of the large Internet search engines users, even those in academia, have come to expect interfaces that present a single text box in which to enter their query. Of course an “advanced” search page is usually also supplied, but this is actually more straightforward to implement than the “keyword” search we have all become familiar with.

We have no way of knowing a priori which field(s) the user intends to search when they type text into a basic search box. We can only make the assumption that it's reasonable to search the principal metadata fields, such as title, author, subject, and note as well as the full text of the content, when appropriate.

Based on our long experience with academic users, we determined that they would expect this “keyword” query to be interpreted as a multiple-field Boolean AND of all the terms entered. In other words it would match only items containing all of the terms specified by the user, but in any of the preselected fields. Achieving this in the original recommender project was fairly easy, but the solution we developed was not really scalable; for the extension project we developed a more flexible and comprehensive approach.

8.1 Original Approach

For the original Recommender project which dealt only with bibliographic records, we adopted a minimal implementation to support our immediate research aims at the time, but this proved to be a maintenance problem as work progressed on the user interface.

For this approach, at index time our prefilter stylesheets created a special meta- data field called “newKeyword” which contained the concatenated values of the title, author, subject, and note fields. Keyword queries were then directed to search this concatenated field. However, because we still wanted to highlight matching terms in their original context, we had to add a messy second set of clauses, one per field, that didn't contribute to hit scoring but did activate term highlighting on each field.

Disadvantages of this method are:

- ◆ Producing the conjoined query was awkward, and the complexity in our query parser made it difficult to add functionality.
- ◆ The set of fields was of necessity fixed at query time. It seems possible that some users (or administrators) might want to change the field set dynamically.

- ◆ It was impossible to implement different boost factors for the different fields, since they were concatenated in a single value.
- ◆ When we add full text documents, it's not practical to concatenate their entire text onto the conjoined field.

8.2 True Multi- field Keyword Search

To make it simpler for both implementors and end- users to perform intuitive multi- field keyword searches, we extended the functionality of existing query operators.

When generating an `and` query, one can now simply specifies a list of fields, like this:

```
<and fields="title,author,text" slop="10">
  <term>africa</term>
  <term>apartheid</term>
</and>
```

Semantically this is interpreted as querying for documents that contain both of the terms in any of the listed fields (the terms need not both be present in the same field.) Internally this is implemented with a query like this:

```
<and>
  <or>
    <term field="title">africa</term>
    <term field="author">africa</term>
    <term field="text">africa</term>
  </or>
  <or>
    <term field="title">apartheid</term>
    <term field="author">apartheid</term>
    <term field="text">apartheid</term>
  </or>
</and>
```

Once matching documents are found, we don't want the scoring produced by the expanded query, because it doesn't take term proximity into account (if the terms do happen to appear in the same field). So scoring, ranking, and term highlighting are performed using a different query:

```
<or>
  <orNear field="title" slop="10">
    <term>africa</term>
    <term>apartheid</term>
  </orNear>
```

The Melvyl Recommender Project: Supplemental Report

```
<orNear field="author" slop="10">
  <term>africa</term>
  <term>apartheid</term>
</orNear>
<orNear field="text" slop="10">
  <term>africa</term>
  <term>apartheid</term>
</orNear>
</or>
```

We originally created the `orNear` operator to help implement “more like this” queries in the main recommender project. These queries needed to take term proximity into account to increase the relevancy of recommendations. Essentially the `orNear` operator looks at the `slop` argument, and checks the proximity of terms in each field. If the terms are separated by less than the `slop` factor, they are highlighted as a single joined unit, and scored higher than they would have been separately. In this way, terms that appear near each other cause a matching record to score and rank higher than a record where the terms appear farther apart or in separate fields.

The new multi- field query logic in XTF hides all of this complexity, allowing XTF implementors to quickly implement this commonly requested feature.

8.3 Results

Though we have not quantified the improvement, our informal evaluation of the new keyword query facilities has been very positive. Processing time does not seem to have been noticeably impacted, terms are being highlighted in context, and the hits seem to be relevant to the query. We were able to eliminate the cumbersome stylesheet rules that were required by the old method, resulting in an easier system to maintain and expand.

Finally, the new method allowed us to tweak per- field boosts, which in turn helped us to achieve a good mix of full text and bibliographic records.

9 Other Improvements

Two other areas received some attention during the extension project: personalization and date range searching, discussed in the following sections

9.1 Personalization / SQL Interface

Work in the original Recommender project included the development of support for session- level persistence; as an example application, users of the prototype can mark items in a result set and save them to a "book bag" to be used during a single session. This feature was used to support assessment activities, but is not well integrated into the prototype interface.

With this extension, we hoped to implement longer- term persistence, to allow for the development of personal and shared lists and persistent settings for personalization. Extension activities were to include: delineation of the architecture for storage and access, development of a permissions structure for sharing, and outline of basic presentation strategies for incorporating these features into the prototype.

Due to time constraints, most of these goals were not pursued. However, one significant enabling technology was added to XTF: the ability to connect to, query, and update an SQL database. Stylesheets anywhere within XTF can now connect to an SQL database, and perform insert, update, and delete operations.

This support was adapted from the SQL support originally present in Saxon, but has been greatly expanded to include not just insertion but also updating. We added more robust error checking and timeouts, and the ability to control server- specific connection parameters.

A basic user account creation and login/logout mechanism was created and tested in XTF. However, there was insufficient time to integrate it into the main prototype interface, nor was there time to implement a full persistent bookbag and to explore avenues for users to tag items and to share tags/queries/items. This could be a promising avenue of future research.

9.2 Date Range Searching

From its inception, XTF has had very limited date range searching. Based as it is on expansion using wildcards in Lucene, it cannot gracefully handle granularities down to the day level (the current system had only year- level searching), and completely breaks down at granularities below a day.

However, some user activities require a granular date range search, especially in an academic, research-oriented environment. For instance, a user might wish to research co-publishing of documents within a month of each other. As another example, a user might want to check the system periodically for new additions to the catalog. Both of these tasks would be difficult to implement without a true date range search.

For the extension project we replaced XTF's old wildcard-based facility with a robust range operator that could be used at an acceptable level of performance against broad and highly granular ranges (to the level of the day, second, or even millisecond).

The basic method was to create an in-memory table of all the object dates, compact and ready for quick numeric comparison. This necessitated indexing the date/time field without tokenization, so that the data values are available in compact, sorted order in the Lucene index. The new query operator transparently loads the table, then filters out documents that don't match the range specified in the query.

The new code is actually applicable for any numeric data with consistent formatting (i.e. same number of digits, punctuated identically in each instance.) While dates and timestamps are the most obvious application, one could imagine other potential applications, such as geospatial querying on latitude/longitude coordinates.

9.3 For Future Exploration

Though the team didn't have time to pursue them, a few miscellaneous topics may deserve further work in the future.

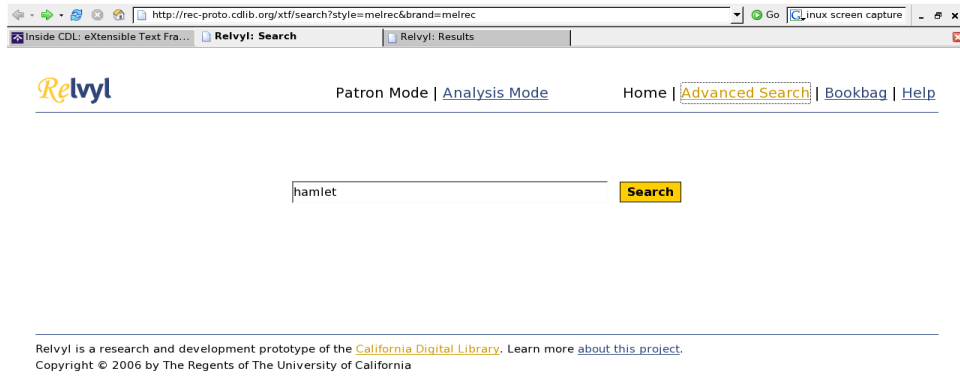
- Interface synergies: it might improve the user's experience to present structured data from full text hits within the query result page. For instance, rolling the mouse over a document might pop up a table of contents.
- Enhanced similarity-based recommendations: perhaps the “more like this” feature could be enhanced to use citation/reference mining to compare works which are marked up with sufficient precision.
- Currently the “more like this” recommendations are based strictly on comparing item metadata. It might be fruitful to expand this to consider the full text of an object, when that is available. For efficiency, it might be necessary to create (at index time) some sort of short automated summary of the most “interesting” words in each document.

The Melvyl Recommender Project: Supplemental Report

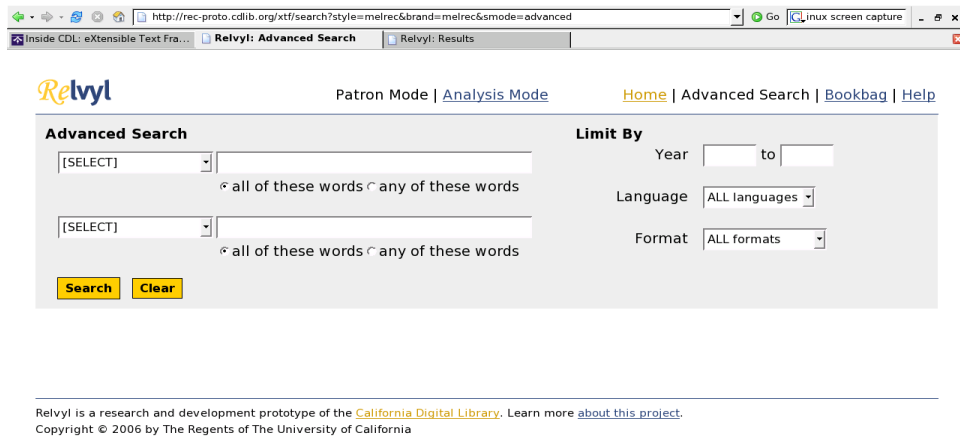
- Performance analysis: While the team has performed some basic load testing on the prototype system, coverage was not comprehensive. Additionally, there seem to be some queries that are quite slow, and it would be useful to identify the causes of that slowness and develop faster algorithms and/or other ways to improve responsiveness.

10 Appendix: Screenshots

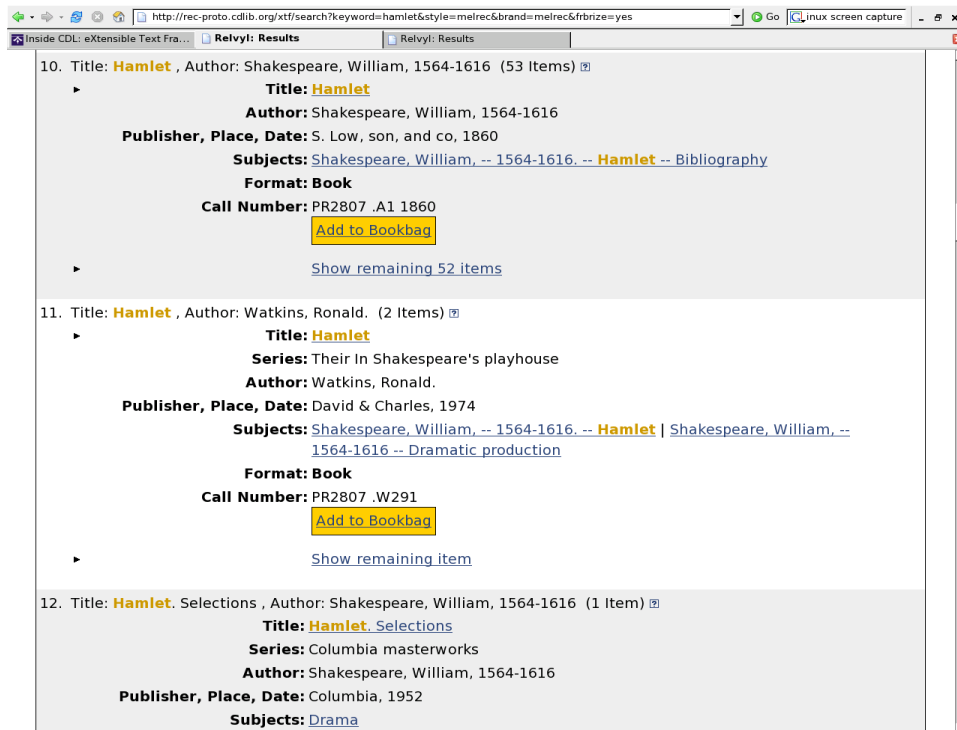
10.1 Basic Search (multi- field keyword)






10.2 Advanced Search (including date range)



10.3 FRBR Results (unexpanded)



The screenshot shows a web browser window with the URL <http://rec-proto.cdlib.org/xtf/search?keyword=hamlet&style=melrec&brand=melrec&frbrize=yes>. The browser tabs include "Inside CDL: eXtensible Text Fra...", "Relvyl: Results", and "Relvyl: Results". The search results are as follows:

- 10. Title: **Hamlet**, Author: Shakespeare, William, 1564-1616 (53 Items) 
 - ▶ **Title:** [Hamlet](#)
 - Author:** Shakespeare, William, 1564-1616
 - Publisher, Place, Date:** S. Low, son, and co, 1860
 - Subjects:** [Shakespeare, William, -- 1564-1616](#), -- [Hamlet](#) -- Bibliography
 - Format:** Book
 - Call Number:** PR2807 .A1 1860
 - [Add to Bookbag](#)
 - ▶ [Show remaining 52 items](#)
- 11. Title: **Hamlet**, Author: Watkins, Ronald. (2 Items) 
 - ▶ **Title:** [Hamlet](#)
 - Series:** Their In Shakespeare's playhouse
 - Author:** Watkins, Ronald.
 - Publisher, Place, Date:** David & Charles, 1974
 - Subjects:** [Shakespeare, William, -- 1564-1616](#), -- [Hamlet](#) | [Shakespeare, William, -- 1564-1616 -- Dramatic production](#)
 - Format:** Book
 - Call Number:** PR2807 .W291
 - [Add to Bookbag](#)
 - ▶ [Show remaining item](#)
- 12. Title: **Hamlet**. Selections, Author: Shakespeare, William, 1564-1616 (1 Item) 
 - Title:** [Hamlet. Selections](#)
 - Series:** Columbia masterworks
 - Author:** Shakespeare, William, 1564-1616
 - Publisher, Place, Date:** Columbia, 1952
 - Subjects:** [Drama](#)

10.4 FRBR Results (expanded)

10. Title: **Hamlet** , Author: Shakespeare, William, 1564-1616 (53 Items)

- ▶ **Title:** **Hamlet**
Author: Shakespeare, William, 1564-1616
Publisher, Place, Date: S. Low, son, and co, 1860
Subjects: [Shakespeare, William, -- 1564-1616](#), -- [Hamlet -- Bibliography](#)
Format: Book
Call Number: PR2807 .A1 1860
[Add to Bookbag](#)
- ▶ **Title:** **Hamlet**
Author: Shakespeare, William, 1564-1616
Publisher, Place, Date: S. Low, son, and co, 1860
Subjects: [Shakespeare, William, -- 1564-1616](#), -- [Hamlet -- Bibliography](#)
Format: Book
Call Number: PR2807 .A1 1860
[Add to Bookbag](#)
- ▶ **Title:** **Hamlet**
Series: Booth's series of acting plays, no. 9
Author: Shakespeare, William, 1564-1616
Publisher, Place, Date: French, 1866
Format: Book
Call Number: PR2807 .A1 1866
[Add to Bookbag](#)
- ▶ **Title:** **Hamlet**
Series: The works of Shakespeare, edited for the syndics of the Cambridge University Press by John Dover Wilson
Author: Shakespeare, William, 1564-1616
Publisher, Place, Date: University Press, 1934

10.5 Score Explanation



10. Title: **Hamlet**, Author: Shakespeare, William, 1564-1616 (53 Items) 

▶

Title: **Hamlet**

Author: Shakespeare, William, 1564-1616

Publisher, Place, Date: S. Low, son, and co, 1860

Subjects: [Shakespeare, William, -- 1564-1616](#). -- [Hamlet -- Bibliography](#)

Format: Book

Call Number: PR2807 .A1 1860

Score: 2.511

Explanation: 2.5108914 = sum of:

- 1.3530288 = weight(title-main:hamlet in 2691726), product of:
 - 0.5345225 = queryWeight(title-main:hamlet), product of:
 - 2.531285 = tf(spanFreq=6.4074035)
- 0.34749797 = weight(subject:hamlet^0.5 in 2691726), product of:
 - 0.26726124 = queryWeight(subject:hamlet^0.5), product of:
 - 1.3002183 = tf(spanFreq=1.6905677)
- 0.81036484 = weight(note:hamlet in 2691726), product of:
 - 0.5345225 = queryWeight(note:hamlet), product of:
 - 1.5160538 = tf(spanFreq=2.2984192)

[Add to Bookbag](#)

▶ [Show remaining 52 items](#)

11. Title: **Hamlet**, Author: Watkins, Ronald. (2 Items) 

▶

Title: **Hamlet**

Series: Their In Shakespeare's playhouse

Author: Watkins, Ronald.

Publisher, Place, Date: David & Charles, 1974

Subjects: [Shakespeare, William, -- 1564-1616](#). -- [Hamlet](#) | [Shakespeare, William, -- 1564-1616 -- Dramatic production](#)

Format: Book

10.6 Single Record with Recommendations

http://rec-proto.cdlib.org/xtf/search?object=002839217&keyword=hamlet&style=melrec&brand=melrec&mode=ot

Inside CDL: eXtensible Text Fra... Record: Hamlet, William S... Relvyl: Results

Relvyl Patron Mode | Analysis Mode Home | Advanced Search | Bookbag | Help

[Return to search results](#)

Title: Hamlet, William Shakespeare
Series: New casebooks
Author: Coyle, Martin.
Publisher: St. Martin's Press, 1992
Subjects: [Shakespeare, William -- 1564-1616 -- Hamlet](#)
Format: Book
Call Number: PR2807 .H269 1992
Language: English
Note(s): Includes bibliographical references (p. 193-198) and index.
 Tragic balance in [Hamlet](#) / Philip Edwards -- The comedy of [Hamlet](#) / Peter Davison -- Poison, play, and duel / Nigel Alexander -- On the value of [Hamlet](#) / Stephen Booth -- Verbal presence : conceptual absence / James L. Calderwood -- A heart cleft in Twain : the dilemma of Shakespeare's Gertrude / Rebecca Smith -- Chaste constancy in [Hamlet](#) / Marilyn French -- Representing Ophelia : women, madness, and the ref[s]possibilities of feminist criticism / Elaine Showalter -- The woman in [Hamlet](#) : an interpersonal view / David Leverenz -- Revenge in [Hamlet](#) / Catherine Belsey -- Power in [Hamlet](#) / Leonard Tennenhouse -- A thing of nothing : the catastrophic body in [Hamlet](#) / John Hunt.
ISBN: 0312075650

Recommendations:

Similar Records <small>(Experiment with the similarity settings)</small>	Patrons who borrowed this item also borrowed	Amazon Recommendations
<ol style="list-style-type: none"> Shakespeare's workshop / Lawrence, William John, 1862-1940 H. Mifflin, 1928. Subjects: Shakespeare, William -- 1564-1616 Shakespeare, William -- 1564-1616 -- Hamlet Shakespeare, William, 1564-1616 Shakespeare, William, 1564-1616 -- Hamlet Shakespeare, William, 1564-1616 Shakespeare, William, 1564-1616 -- Hamlet William Shakespeare: Hamlet, Modern critical interpretations / Bloom, Harold. Chelsea House Publishers, 1986. Subjects: Shakespeare, William -- 1564-1616 -- Hamlet Shakespeare, William, 1564-1616 -- Hamlet Briefe über Shaksperes Hamlet / Friesen, Hermann Freiherr von, 1802-1882 B. G. Teubner, 1864. Subjects: Shakespeare, William 	<ol style="list-style-type: none"> What happens in Hamlet / Wilson, John Dover, 1881-1969 University Press, 1956. Subjects: Shakespeare, William -- 1564-1616 -- Hamlet Shakespeare, Hamlet, Landmarks of world literature / Cantor, Paul A. (Paul Arthur), 1945- Cambridge University Press, 1989. Subjects: Shakespeare, William -- 1564-1616 -- Hamlet Ab urbe condita, English & Latin Livy, with an English translation, Loeb classical library [114, 133, 172, 191, 233, 355, 367, 381, 295, 301, 313, 332, 396, 404] / Livy. Harvard University Press, 1919. 	<ol style="list-style-type: none"> Macbeth : Modern English Version Side-By-Side With Full Original Text (Shakespeare Made Easy) Rosenkrantz & Guildenstern Are Dead (An Evergreen Book) The tempest, with new and updated critical essays and a revised bibliography / Shakespeare, William, 1564-1616 Signet Classic. Subjects: Shakespeare, William -- 1564-1616 -- Tempest Shakespeare, William -- 1564-1616 -- Criticism and interpretation Survival after airplane accidents, shipwrecks, etc. -- Drama Fathers and daughters -- Drama Castaways -- Drama Magicians -- Drama