Next-Generation Search Engines for Information Retrieval

Ranjeet Devarakonda devarakondar@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, 37831, USA

Les Hook hookla@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, 37831, USA

Giri Palanisamy palanisamyg@ornl.gov
Oak Ridge National Laboratory
Oak Ridge, 37831, USA

Jim Green jgreen@iiaweb.com
Information International Associates
Oak Ridge, 37831, USA

Abstract

In the recent years, there have been significant advancements in the areas of scientific data management and retrieval techniques, particularly in terms of standards and protocols for archiving data and metadata. Scientific data is rich, and spread across different places. In order to integrate these pieces together, a data archive and associated metadata should be generated. Data should be stored in a format that can be retrievable and more importantly it should be in a format that will continue to be accessible as technology changes, such as XML. While general-purpose search engines (such as Google or Bing) are useful for finding many things on the Internet, they are often of limited usefulness for locating Earth Science data relevant (for example) to a specific spatiotemporal extent. By contrast, tools that search repositories of structured metadata can locate relevant datasets with fairly high precision, but the search is limited to that particular repository. Federated searches (such as Z39.50) have been used, but can be slow and the comprehensiveness can be limited by downtime in any search partner.

An alternative approach to improve comprehensiveness is for a repository to harvest metadata from other repositories, possibly with limits based on subject matter or access permissions. Searches through harvested metadata can be extremely responsive, and the search tool can be customized with semantic augmentation appropriate to the community of practice being served. One such system, Mercury, a metadata harvesting, data discovery, and access system, built for researchers to search to, share and obtain spatiotemporal data used across a range of climate and ecological sciences. Mercury is open-source toolset, backend built on Java and search capability is supported by the some popular open source search libraries such as SOLR and LUCENE. Mercury harvests the structured metadata and key data from several data providing servers around the world and builds a centralized index. The harvested files are indexed against SOLR search API consistently, so that it can render search capabilities such as simple, fielded, spatial and temporal searches across a span of projects ranging from land, atmosphere, and ocean ecology. Mercury also provides data sharing capabilities using Open Archive Initiatives Protocol for Metadata Handling (OAI-PMH). In this paper we will discuss about the best practices for archiving data and metadata, new searching techniques, efficient ways of data retrieval and information display.

Keywords: Scientific Data Indexing, Mercury Search, ORNL DAAC, NASA, U.S. DOE

1. INTRODUCTION

Scientific data, in its most general context across multiple disciplines, includes measurements and observations of natural phenomena for the purpose of explaining the behavior of or testing hypotheses about the natural systems. This includes observational data captured in real-time by sensors, surveys, and imaging devices; experimental data from laboratory instruments, for example, gene sequences, chromatograms, and characterization of samples; simulation data generated from models where the model is equally important with the input and output data such as for climate and economic models; and derived or compiled data that are the result of text and data mining, and compiled and integrated databases from multiple sources.

Remote sensing and high throughput scientific instruments and high spatial and temporal resolution model simulations are creating vast data repositories and as a result of increased investments in engineering research, disease research , and environmental monitoring, the volume of scientific data is approximately doubling each year [1]. Data volume requires new scientific methods to analyze and organize the data for fully exploiting the value of the data and beyond the original collection purpose.

The data content can be collected in any format, most is digital by necessity and to be of greatest value for analyses. Instrument platform and science program specific formats are often integrated into the data processing stream. The data can be formatted as straightforward tabular ASCII files, spatially gridded files, Geotiff image formats or formatted to be consistent with programmatic repository needs such as Genomes-to-Life.

What is metadata and why is it so useful?

The digital data are not complete without descriptive information (meta) data to tell us what it means. Metadata are the descriptive information about data that explains the measured attributes, their names, units, precision, accuracy, data layout and ideally a great deal more. Most importantly, metadata includes the data lineage that describes how the data was measured, acquired, or computed [2]. Metadata in its simplest form, are those elements that concisely identify the "who, what, where, when, why, and how" of the data.

Metadata enables data sharing and access. Sharing data with colleagues the broader scientific community and public is highly desirable and will result in greater advancement of science [1].

We know data sharing is a good thing. Doesn't everyone else?

- Open science and new research
- Data longevity
- Data reusability
- Greater exposure to data
- Generation of Value added products
- Verification of published works
- Possibility for future research collaborations
- More value for the research investment

The volume of scientific data, and the interconnectedness of the systems under study, makes integration of data a necessity. For example, life scientists must integrate data from across biology and chemistry to comprehend disease and discover cures, and climate change scientists must integrate data from wildly diverse disciplines to understand our current state and predict the impact of new policies [3].

It is critical to begin to document your data at the very beginning of your research project, even before data collection begins; doing so will make data documentation easier and reduce the likelihood that you will forget aspects of your data later in the research project.

2. BEST PRACTICES FOR CREATING METADATA

2.1 Format Your Data

The motivation behind this memorandum is storing data in an easily retrievable format. Several metadata standards have been developed to ensure compatibility and interoperability across multiple databases and clearinghouses, including FGDC, NetCDF Attribute Convention for Dataset Discovery, Ecological Metadata Language (EML), and ISO-19115. The standard implemented by the archive for your data should be identified and values for any standard-specific fields should be provided.

According to California Digital Library, the file format in which you keep your data is a primary factor in one's ability to use your data in the future. As technology continually changes, researchers should plan for both hardware and software obsolescence [4]. How will your data be read if the software used to produce them becomes unavailable?

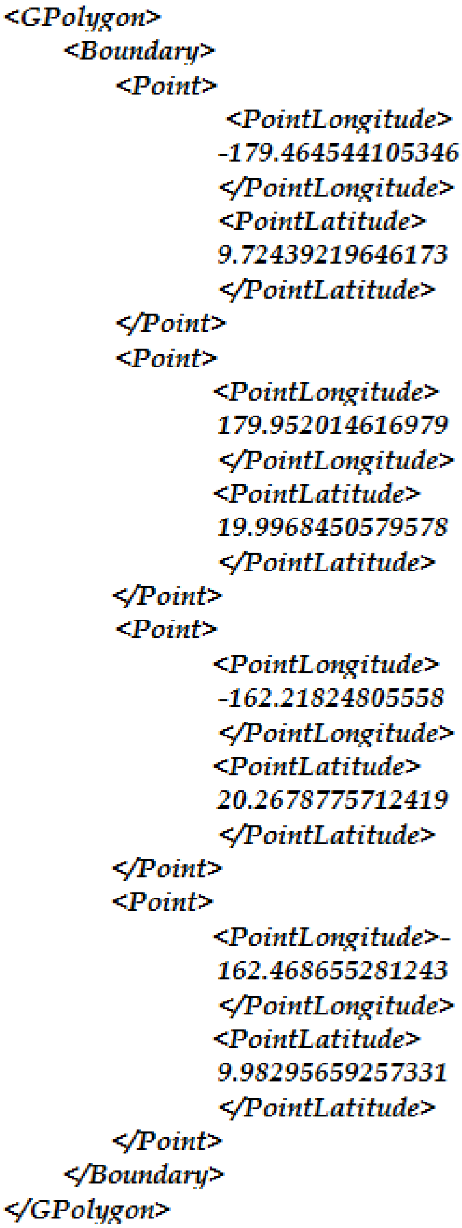
Use of Open Source tools, APIs, and formats is mandated in order to remove obstacles to usage and sharing of both data products and tools. This helps to ensure continuity of the work, lessening the opportunity for the project becoming “orphaned”. Some of the open source tools which the Mercury design makes use of are SOLR (index creation & maintenance), MYSQL(database), Eclipse(Integrated development environment) , Tomcat (Container), Apache(Web server), Spring(Java/J2EE application platform) and Hibernate (persistence framework). W3C standard XML is used as the open document standard.

Data exchange is standardized with the use of UTF8 in order to assure compatibility across modern applications [5]. The use of the older ISO-8859 is discouraged. Approaches to incorporating this include, but are not limited to, providing editors, parsers and indexers which are compliant.

Extensibility of the controlling schema is a design priority. SOLR uses a standard controlling API for specifying fields and processing instructions. Building on this methodology, XPATH is used to define data elements for extraction. As part of the indexing toolset, we create a set of parsing rules which allows for use of logical combinations, default values, lookups and transformations.
Design of the Mercury parser is based upon use of the core Java Transformer classes. Parser configuration is accomplished via reading an XML file which is used to inject spring style beans with a list of data “types” to be processed. Each type is a reference to either a bean containing a map of element names to XPATH definitions or a map associating a name to another bean definition.

Operationally, objects built during parser initialization are used to construct a singleton containing maps of maps to the compiled XPATHEXPRESSION(s) needed to programmatically extract values from the metadata files. The mapped fields correspond to ‘fields’ specified within the SOLR schema file. Values obtained are used to populate a SOLR document which is sent via http to the specified SOLR web application for inclusion in the index.

Flexibility and ease of expansion was a primary consideration in the choice of this technology. In the same way that XSLT can be expanded and enhanced, use of XPATH allows for expanding rules to accommodate new development without making changes to the compiled codebase. As an example, consider the need to extract information from an XML document which has arbitrary embedded business logic. In this case, consider the following XML snippet:



```
<GPolygon>
     <Boundary>
          <Point>
                    <PointLongitude>
                    -179.464544105346
                    </PointLongitude>
                    <PointLatitude>
                    9.72439219646173
                    </PointLatitude>
          </Point>
          <Point>
                    <PointLongitude>
                    179.952014616979
                    </PointLongitude>
                    <PointLatitude>
                    19.9968450579578
                    </PointLatitude>
          </Point>
          <Point>
                    <PointLongitude>
                    -162.21824805558
                    </PointLongitude>
                    <PointLatitude>
                    20.2678775712419
                    </PointLatitude>
          </Point>
          <Point>
                    <PointLongitude>-
                    162.468655281243
                    </PointLongitude>
                    <PointLatitude>
                    9.98295659257331
                    </PointLatitude>
          </Point>
     </Boundary>
</GPolygon>
```

FIGURE 1: Combination of Spatial Coordinates for indexing

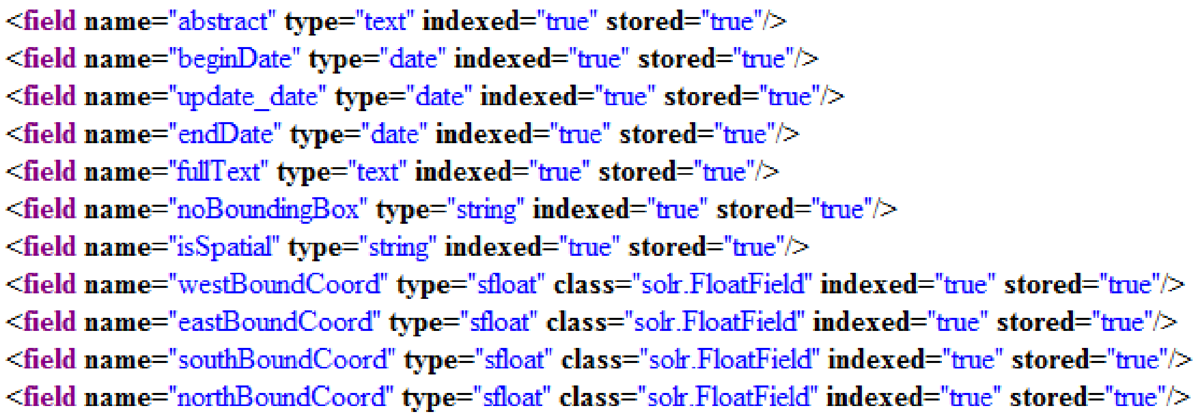
The compact logic available in XPATH allows us to construct the following, which would extract the minimum value from the embedded array of lattitude values:

<entry key="southbc" value = "//PointLatitude[not(. >= ../preceding-sibling::Point/PointLatitude) and not(. >=../following-sibling::Point/PointLatitude)]"/>
Assignment of this as an indexed (searchable) value is then possible, with the result that we can then provide exact or ranged searches within the collections. This is just one example of the type of operation possible with the combination of types defined in the compiled code and the use of injection to provide new behavior at run-time. This methodology allows for growth not just of the single use index, but extension of the tool to associated projects via use of multiple schema definitions allocated on a per source basis to index and serve up related content.

2.2 Indexing Metadata

It's not uncommon to have millions of pages to index in a corporate and government circles. Normally, search engine creates the index and the database of documents that it accesses when processing a user query. Indexer engine sorts every word on every page and stores the resulting index in a huge database [6]. Example: Google. Query processor then compares the search query against the index and returns the documents that it considers most relevant.

In Modern search indexers like Apache SOLR, Users can pass a number of optional Query parameters to the request handler to control what information is returned. Solrconfig.xml contains the default parameters values. Default parameters can be overridden during the request query-time. Schema.xml specifies all the fields that can be indexed from the metadata documents and how those fields should be dealt with when adding documents to the index or querying those fields (SEE figure below).



```
<field name="abstract" type="text" indexed="true" stored="true"/>
<field name="beginDate" type="date" indexed="true" stored="true"/>
<field name="update_date" type="date" indexed="true" stored="true"/>
<field name="endDate" type="date" indexed="true" stored="true"/>
<field name="fullText" type="text" indexed="true" stored="true"/>
<field name="noBoundingBox" type="string" indexed="true" stored="true"/>
<field name="isSpatial" type="string" indexed="true" stored="true"/>
<field name="westBoundCoord" type="sfloat" class="solr.FloatField" indexed="true" stored="true"/>
<field name="eastBoundCoord" type="sfloat" class="solr.FloatField" indexed="true" stored="true"/>
<field name="southBoundCoord" type="sfloat" class="solr.FloatField" indexed="true" stored="true"/>
<field name="northBoundCoord" type="sfloat" class="solr.FloatField" indexed="true" stored="true"/>
```

FIGURE 2: SOLR Fields

Common SOLR Fields:

- *field name* is the actual name of the file in the SOLR index
- *type* is the data structure type the field is stored in
- *indexed* is a Boolean type to enable the field for indexing, and hence for searching, sorting
- *stored* is a Boolean, used for specifying if the field should be retrievable during a search, Multivalve is used to specify if the field can accept multiple values per document.

Depending on the data and required search capability, there could be a number of factors to consider while indexing.

2.3 Factors Affecting Indexing

The Indexing process as implemented for Mercury is comprised of the steps necessary to first identify the content and then construct the necessary controlling structures which will allow this data to be processed for inclusion within a SOLR (LUCENE) index file. Disparate sources and misapprehensions regarding content are the major issues affecting this indexing process. Impacts of these issues on accuracy and responsiveness of the search application are minimized by proper design strategy.

An XML schema/DTD/XSD commonly is constructed for and used to define the structure of a document, with editors and parsers dependent upon its completeness and applicability. Using this construct, the indexer process is built according to the defined structures and types. But, there is no universal enforcement via a standard editor or data logging mechanism. Metadata files might originate in the lab, at a field site, or be harvested from the web. As a result, varying degrees of adherence to the structures and content rules are found, with the "standard" treated as a loose guide. These variances in the XML content can result in apparent errors and confusion with interpretations. There is a tradeoff to be made between use of strict standards for validation which can result in rules which result in a large rejection rate, and data volume processed.

International character sets, translation errors with http formatting, obsolete web standards and the general evolution of the art present challenges which are best addressed in the Java parsing code. This normalization is part of the design process for individual parser objects, and includes handling for special or illegal characters, Character set issues (ISO 8859 variants -vs- UTF8) and foreign or extended character sets. Additionally, dealing with date format issues is a recurring problem.

Given that the parsing operation (Extraction of the facet data) can be computationally intensive, it makes sense to offload this process to a scheduled operation (the "harvested" model). Such harvesting serves to increase the apparent speed of the application, which is a critical concern as regards usability.

Generation of a flexible schema for indexing and queries at design time allows translation to a common data language. Having a fixed set of elements which are then populated based upon logic customized per data source minimizes time to stand-up a new instance and enhances content navigation.

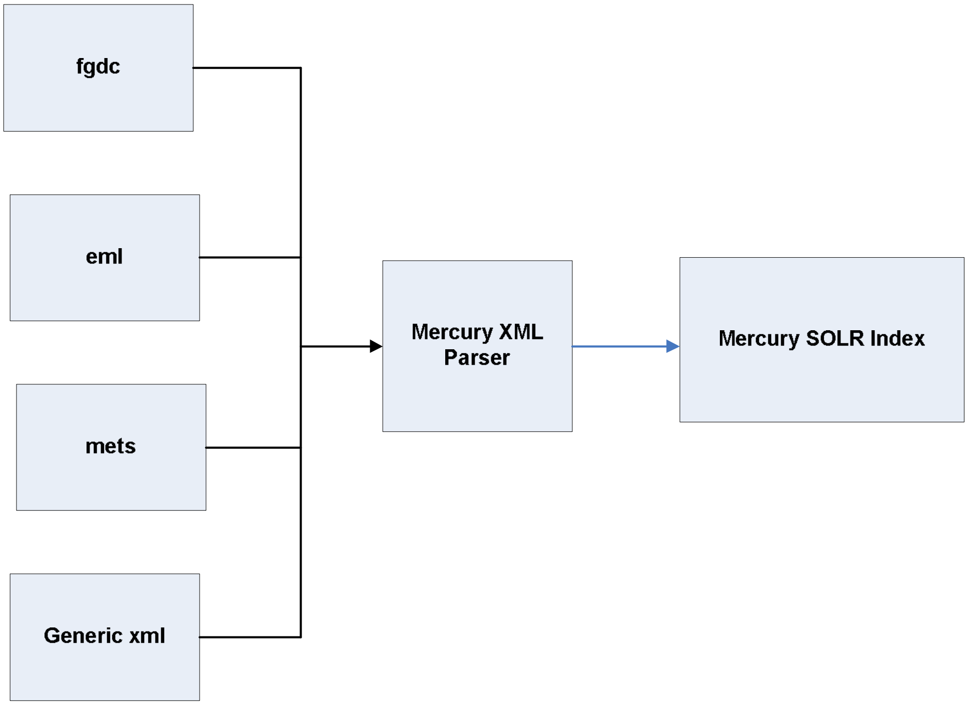


FIGURE 3: Metadata Indexing Flow

3. OPERATING THE METADATA

3.1 Requirements

The goal of the Metadata is to make it easy for the users to learn, locate, and retrieve the desired data. Metadata must also be compatible to add new fields to the metadata, provide user support and maintain the computer system, including future upgrade of software, hardware, storage media and network connectivity.

Why SOLR?

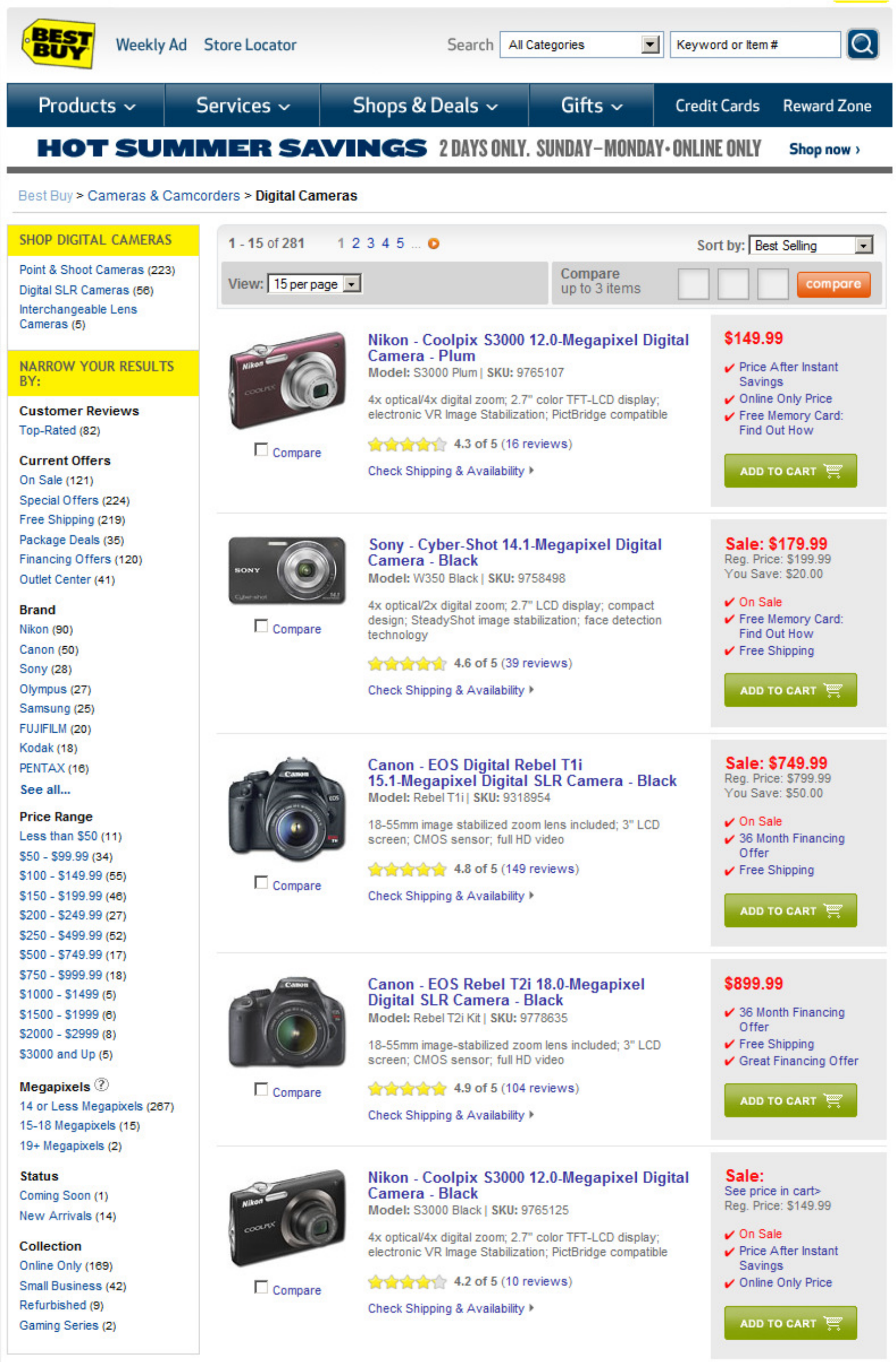
Apache SOLR is an enterprise search server based on Lucene. It was originally developed by CNET Networks as an in-house search platform. It's written in Java and runs on Application server [8]. See table below for its full list of features from Apache SOLR:

- Advanced Full-Text Search Capabilities
- Optimized for High Volume Web Traffic
- Standards Based Open Interfaces - XML,JSON and HTTP
- Comprehensive HTML Administration Interfaces
- Server statistics exposed over JMX for monitoring
- Flexible and Adaptable with XML configuration
- Extensible Plug-in Architecture

3.2 Data Retrieval

Information retrieval has changed drastically in recent years with the expansion of World Wide Web and the advert of modern and inexpensive graphical user interfaces and mass storage devices [7]. User Interface is the entry point of the information retrieval process. This gives the users flexibility in specifying the text to be retrieved from specific terms. Modern GUI's are largely based on the Cascading Style Sheets (CSS) and JavaScript (JS). CSS has been around for a long time, but how aggressively it has been utilized has been a factor of both browser compatibility and browser support. Modern Information retrieval revolves not only around simply retrieving the information, but also the presentation aspect. Faceted search is one of the new mechanisms for refining search results by filtering content using multiple taxonomy terms at the same. Faceted metadata representation has been found to be a better understandable data model for scientific search interfaces.

Facets allow the users to easily shift between filtering and expanding their original searches. It also gives a hint to the users but displaying the range, helping in looking for item of interest that user originally haven't thought about. A faceted search solution can be used to boost your business intelligence and data warehousing projects, giving your users a modern search engine based interface to your data. Many popular websites use facets to display search results.



BEST BUY Weekly Ad Store Locator Search All Categories Keyword or Item #

Products Services Shops & Deals Gifts Credit Cards Reward Zone

HOT SUMMER SAVINGS 2 DAYS ONLY. SUNDAY–MONDAY• ONLINE ONLY Shop now ›

Best Buy > Cameras & Camcorders > Digital Cameras

SHOP DIGITAL CAMERAS

Point & Shoot Cameras	(223)
Digital SLR Cameras	(56)
Interchangeable Lens Cameras	(5)

NARROW YOUR RESULTS BY:

Customer Reviews

Top-Rated	(82)

Current Offers

On Sale	(121)
Special Offers	(224)
Free Shipping	(219)
Package Deals	(35)
Financing Offers	(120)
Outlet Center	(41)

Brand

Nikon	(90)
Canon	(50)
Sony	(28)
Olympus	(27)
Samsung	(25)
FUJIFILM	(20)
Kodak	(18)
PENTAX	(16)

See all...

Price Range

Less than $50	(11)
$50 - $99.99	(34)
$100 - $149.99	(55)
$150 - $199.99	(46)
$200 - $249.99	(27)
$250 - $499.99	(52)
$500 - $749.99	(17)
$750 - $999.99	(18)
$1000 - $1499	(5)
$1500 - $1999	(6)
$2000 - $2999	(8)
$3000 and Up	(5)

Megapixels ?

14 or Less Megapixels	(267)
15-18 Megapixels	(15)
19+ Megapixels	(2)

Status

Coming Soon	(1)
New Arrivals	(14)

Collection

Online Only	(169)
Small Business	(42)
Refurbished	(9)
Gaming Series	(2)

1 - 15 of 281 1 2 3 4 5 ... Sort by: Best Selling

View: 15 per page Compare up to 3 items compare

Nikon - Coolpix S3000 12.0-Megapixel Digital Camera - Plum
Model: S3000 Plum | **SKU:** 9765107
4x optical/4x digital zoom; 2.7" color TFT-LCD display; electronic VR Image Stabilization; PictBridge compatible
4.3 of 5 (16 reviews)
Check Shipping & Availability
□ Compare
$149.99
✔ Price After Instant Savings
✔ Online Only Price
✔ Free Memory Card: Find Out How
ADD TO CART

Sony - Cyber-Shot 14.1-Megapixel Digital Camera - Black
Model: W350 Black | **SKU:** 9758498
4x optical/2x digital zoom; 2.7" LCD display; compact design; SteadyShot image stabilization; face detection technology
4.6 of 5 (39 reviews)
Check Shipping & Availability
□ Compare
Sale: $179.99
Reg. Price: $199.99
You Save: $20.00
✔ On Sale
✔ Free Memory Card: Find Out How
✔ Free Shipping
ADD TO CART

Canon - EOS Digital Rebel T1i 15.1-Megapixel Digital SLR Camera - Black
Model: Rebel T1i | **SKU:** 9318954
18-55mm image stabilized zoom lens included; 3" LCD screen; CMOS sensor; full HD video
4.8 of 5 (149 reviews)
Check Shipping & Availability
□ Compare
Sale: $749.99
Reg. Price: $799.99
You Save: $50.00
✔ On Sale
✔ 36 Month Financing Offer
✔ Free Shipping
ADD TO CART

Canon - EOS Rebel T2i 18.0-Megapixel Digital SLR Camera - Black
Model: Rebel T2i Kit | **SKU:** 9778635
18-55mm image-stabilized zoom lens included; 3" LCD screen; CMOS sensor; full HD video
4.9 of 5 (104 reviews)
Check Shipping & Availability
□ Compare
$899.99
✔ 36 Month Financing Offer
✔ Free Shipping
✔ Great Financing Offer
ADD TO CART

Nikon - Coolpix S3000 12.0-Megapixel Digital Camera - Black
Model: S3000 Black | **SKU:** 9765125
4x optical/4x digital zoom; 2.7" color TFT-LCD display; electronic VR Image Stabilization; PictBridge compatible
4.2 of 5 (10 reviews)
Check Shipping & Availability
□ Compare
Sale:
See price in cart>
Reg. Price: $149.99
✔ On Sale
✔ Price After Instant Savings
✔ Online Only Price
ADD TO CART

[1]

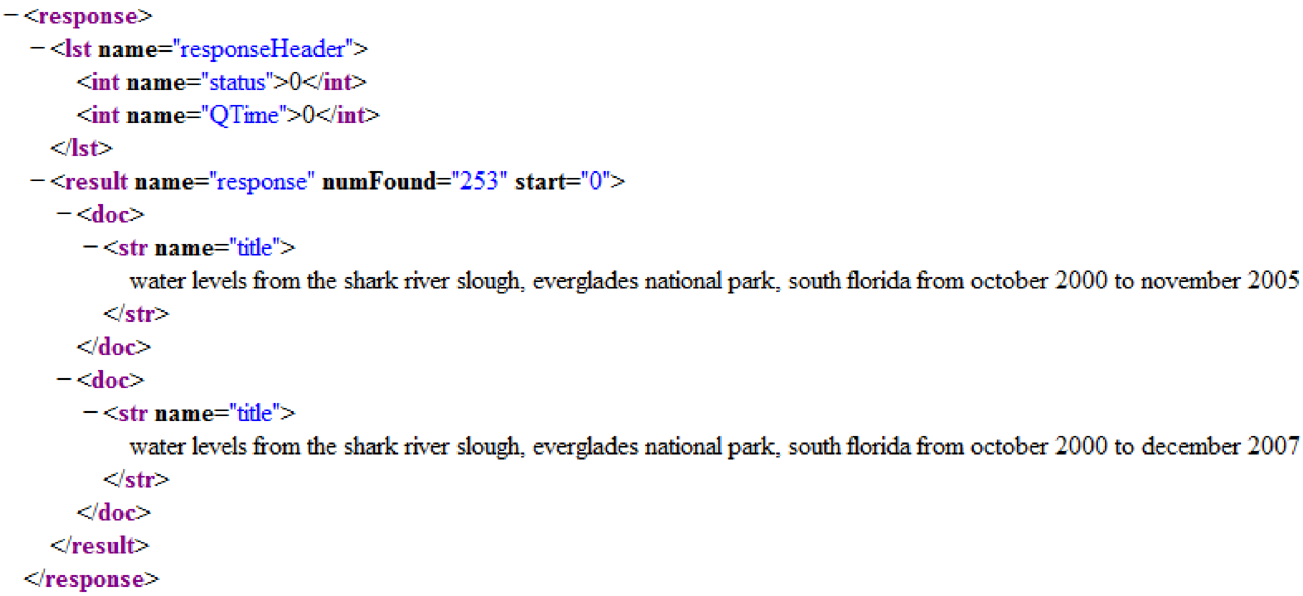
FIGURE 2: BestBuy using facetted results

3.3 Querying Data

Querying is the third part of the search engine. In this, user specified terms are searched and ranked against the millions of pages that are recorded in the index. SOLR provides many functionalities, including keyword searching, search terms highlighting, parsing of the results,

[1] http://www.bestbuy.com

filtering and facet-based browsing. Mercury’s GUI interacts with SOLR’s functions through HTTP, and the responses are retrieved in its own generic XML representation.



```
- <response>
  - <lst name="responseHeader">
      <int name="status">0</int>
      <int name="QTime">0</int>
    </lst>
  - <result name="response" numFound="253" start="0">
    - <doc>
      - <str name="title">
          water levels from the shark river slough, everglades national park, south florida from october 2000 to november 2005
        </str>
      </doc>
    - <doc>
      - <str name="title">
          water levels from the shark river slough, everglades national park, south florida from october 2000 to december 2007
        </str>
      </doc>
    </result>
  </response>
```

FIGURE 3: SOLR search response

In the above SOLR response example, Element *<str name=”title”>* represent the ‘Title’ of the metadata, which is of type ’String’. Element ‘Status’, Integer type, defines the success of the query, 0 being the successful. ‘QTime’ is specified in milliseconds, it is the time taken to process the entire query [8].

Main search responses are captured in the result element. It specifies the total number of documents matched the search parameters by numFound. Offset of the returned results is specified by start.

Parameters are passed to SOLR web application just like a simple HTTP GET form submission. User search terms are embed into the URL with some key required and other essential fields [8].

Example:

http://localhost:8080/project_solr/select/?q=soil+temperature&version=2.2&start=0&rows=10&indent=on&fl=*

- /solr/ is the web application running under
- apache Tomcat or other Application Server
- “q” is the actual query string. In the above example Soil temperature is search term. Operands such as and/or/andnot can be specified as q.op
- As mentioned earlier start and rows defines the offset of elements to be displayed on the resulting page. Default values start: 0 and row: 10, meaning return first ten documents from the index, starting from zero position
- “fl” is the field list, separated by commas and spaces. [*] refers to all the fields to be returned. Field list are the list of fields to be returned in the response

3.4 Fielded Searches

Many search engines categories their searches into: Simple or quick search, advanced search, and command search. Simple search is like a free text search. In this, search terms are not limited to any specific field; entire document is scanned for its occurrence. Example: Google's simple search[2].

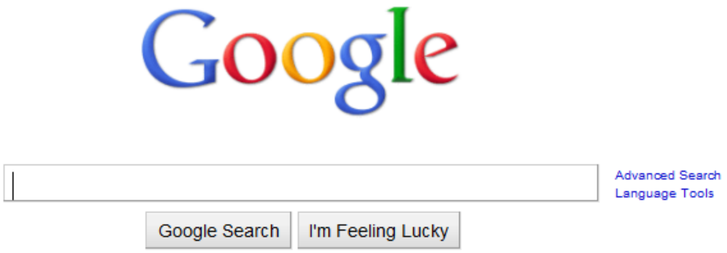


FIGURE 4: Google's simple search

Advanced search, also called as fielded search, allows the user to limit the search to certain fields or elements in document. This gives the users more flexibility to narrow their results. Especially in scientific data, fielded searches can be useful as the data is extensively parameterized. Example: Users can search for 'Carbon Emissions' only limiting its occurrence in the' Title' field rather than an 'Abstract'.

Mercury's advanced search[3] allows users can do a search by Keyword, Spatial-coordinates, temporal data-range or by data providers [9]. In keywords search, searchable fields vary by project to project, but some common fields include: Data set Title, Project, Site, Parameter, Sensor, Source, Term, Topic, Keyword, Investigator, and Abstract. Mercury indexes all these fields into SOLR Index.

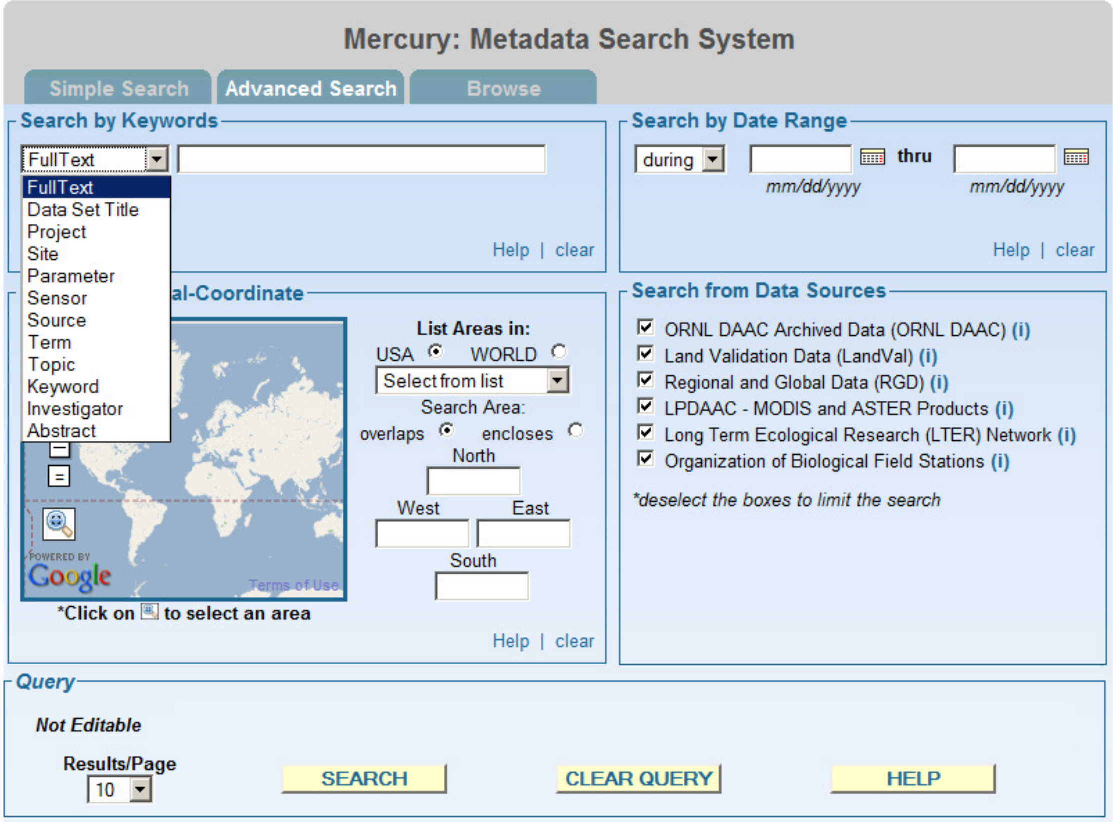
Search Option	**Comments**
Keyword Search	Allows to search for words or phrases within a specific metadata field, restricting the results to only a handful
Spatial Coordinate Search	Restricts data to certain location or region. A area can be defined by selecting a bounding box, this area will be compared with coordinates found in the metadata records, and information for all datasets that cover, intersect, or are covered by the defined area will be included in the results
Temporal Search	Allows to restrict data only from certain time periods

Since Indexed Fields are searchable and sortable. You also can run a SOLR query on indexed Fields, which can alter the content to improve or change results. Mercury also provides data

[2] http://www.google.com

[3] http://mercury.ornl.gov/ornldaac

sharing capabilities across data providers using Open Archive Initiatives Protocol for Metadata Handling (OAI-PMH) [10].



USA ● WORLD ○

overlaps ● encloses ○

*Click on to select an area

☑ ORNL DAAC Archived Data (ORNL DAAC) (i)
☑ Land Validation Data (LandVal) (i)
☑ Regional and Global Data (RGD) (i)
☑ LPDAAC - MODIS and ASTER Products (i)
☑ Long Term Ecological Research (LTER) Network (i)
☑ Organization of Biological Field Stations (i)

*deselect the boxes to limit the search

Not Editable

Results/Page: 10

FIGURE 5: Mercury's advanced search

4. CONCLUSION

The volume of scientific data, and the interconnectedness of the systems under study, makes integration of data a necessity. Also, the file format in which we keep the data is a primary factor for data reusability. As technology continually changes, researchers should plan for both hardware and software obsolescence. SOLR is one of the popular OpenSource libraries for scientific data management, and is gaining recognition for its scalability and extendibility in handling varied combinations in XML.

5. ACKNOLEDGEMENTS

Oak Ridge National Laboratory is managed by the UT-Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR22725.

6. REFERENCES

[1] Gray J. et al, "*Scientific Data Management in the Coming Decade*", Technical Report, Microsoft Research, MSR-TR-2005-10, 2005

[2] Mayernik, M.S., "*Metadata Realities for Cyberinfrastructure: Data Authors as Metadata Creators*". In Proceedings of the iConference 2010

[3] Creative Commons (last accessed February 2011) "*Protocol for implementing open access*

data". http://sciencecommons.org/projects/publishing/open-access-data-protocol/

[4] UC3: Data Management Guidelines, "*Organizing your data*" http://www.cdlib.org/services/uc3/datamanagement/organizing.html (last accessed February 2011)

[5] Markus Scherer, "*UTF-16 FOR PROCESSING*" Technical Note #12 Unicode, Inc., 2004

[6] Blachman, N. and Peek, J. "*How Google Works*" Retrieved Feb 2011 from Googleguide Web site: http://www.googleguide.com/google_works.html

[7] Ricardo Baeza- Yates and Bethier Ribeiro- Neto,"*Modern Information Retrieval*" ACM pp. 73 – 108, 1999

[8] David Smiley and Eric Pugh, "*Solr 1.4 Enterprise Search Server*" PACKT Publishing Ltd., Birmigham, UK, ISBN 978-1-847195-88-3, pp. 89 – 156, 2009

[9] Devarakonda R, Palanisamy G, Wison B, Green J "*Mercury: reusable metadata management, data discovery and access system*", Earth Science Informatics, 3(1):87-94, 2010

[10] Devarakonda R, Palanisamy G, Green J, Wison B, (2010) "*Data sharing and retrieval using OAI-PMH*", Earth Science Informatics, 4(1):1-5, 2010