Apache Solr: Indexing and Searching
(Draft last modified 10/26/2010)

1. **Module name:** Apache Solr

2. **Scope**
   This module addresses the basic concepts of the open source Apache Solr platform that is specifically designed for indexing documents and executing searches.

3. **Learning Objectives**
   By the end of this lesson, students will be able to:
   a. Describe the basic workings of the Apache Solr search engine.
   b. Add/update/delete data to be searched using Apache Solr.
   c. Execute simple queries utilizing concepts of estimated relevance, sorting, facets, etc.
   d. Describe how queries are parsed by Apache Solr.

4. **5S Characteristics of the module**
   a. Space: The concept of space for this module includes physical space and web browser display. The data collections and the Solr application are stored in a physical space on servers (IBM cloud computers). Solr uses a web browser to retrieve the needed document/documents, which presents in a 2D space. In addition, the index uses a vector space representation.

   b. Stream: Text documents and queries are inputs that lead to result sets as well as documents as the output of an IR system. These all are sequences of characters.

   c. Structure: Data collections are stored in an XML format structure without deep nesting. Structures also are present in the index, screen organization, and database as well as data structures used in the system implementation.

   d. Scenario: These include where users/administrators and/or the system submit queries, build indexes, make updates, delete documents, sort lists, add highlights, or analyze data using a given data collection.

   e. Society: End user of the system such as journalists, librarians or developers who would like to use Solr to search relevant data, or help prepare for such searches.

5. **Level of effort required**
   a. Class time: 2 hours

   b. Student time outside class:
   Preparation /reading: 3 hours

6. **Relationships with other modules:**
   Related to module 7-a, which is indexing and searching.

7. **Prerequisite knowledge required:**
   Basic Linux commands

8. **Introductory remedial instruction**: None

9. **Body of knowledge**

- **Introduction to Solr**
  Solr is an open source enterprise search server. It is a search engine used by public sites like CNet, Zappos, and Netflix, as well as intranet sites. In addition to simple query searching, it offers extra features like highlighting, faceted search, and query spelling correction. Figure 1 depicts the flow of data through Solr.

Figure1:Common Solr Usage(Smiley D., Pugh E. (2009). Solr 1.4 Enterprise Search Server)

The core technology underlying Apache Solr is Lucene. Lucene was developed and open sourced by Doug Cutting in 2000 and has evolved and matured since then with a strong online community. In order to use Lucene directly, one writes code to store and query an index stored on a disk. Solr is considered as the server-ization of Lucene.

- **Layout of Solr**
  The example home directory of Solr is example/solr.
  Solr's home directory includes the following:
  - *bin*: Files for more advanced setup are placed here.
  - *conf*: Contains files which help set the Solr configurations.
  - *conf/schema.xml*: This is the schema for the index including field type definitions for given dataset.
  - *conf/solrconfig.xml*: This is the primary Solr configuration file.
  - *data*: It contains the actual Lucene index data in binary format.
  - *lib:* The additional Solr plug in jar files can be placed here.

- **Deploying and Running Solr**

  First go to the example directory, and then run Jetty's start.jar file by typing the following commands:

  cd example
  java -jar start.jar

  You'll see about a page of output including references to Solr. When it is finished, you should see this output at the very end:

  2008-08-07 14:10:50.516::INFO: Started SocketConnector @ 0.0.0.0:8983

  The 0.0.0.0 means it's listening to connections from any host (not just localhost, notwithstanding potential firewalls) and 8983 is the port.

  You can use the http://localhost:8983/solr/admin/ link to access the Solr admin page as shown in Figure 2.

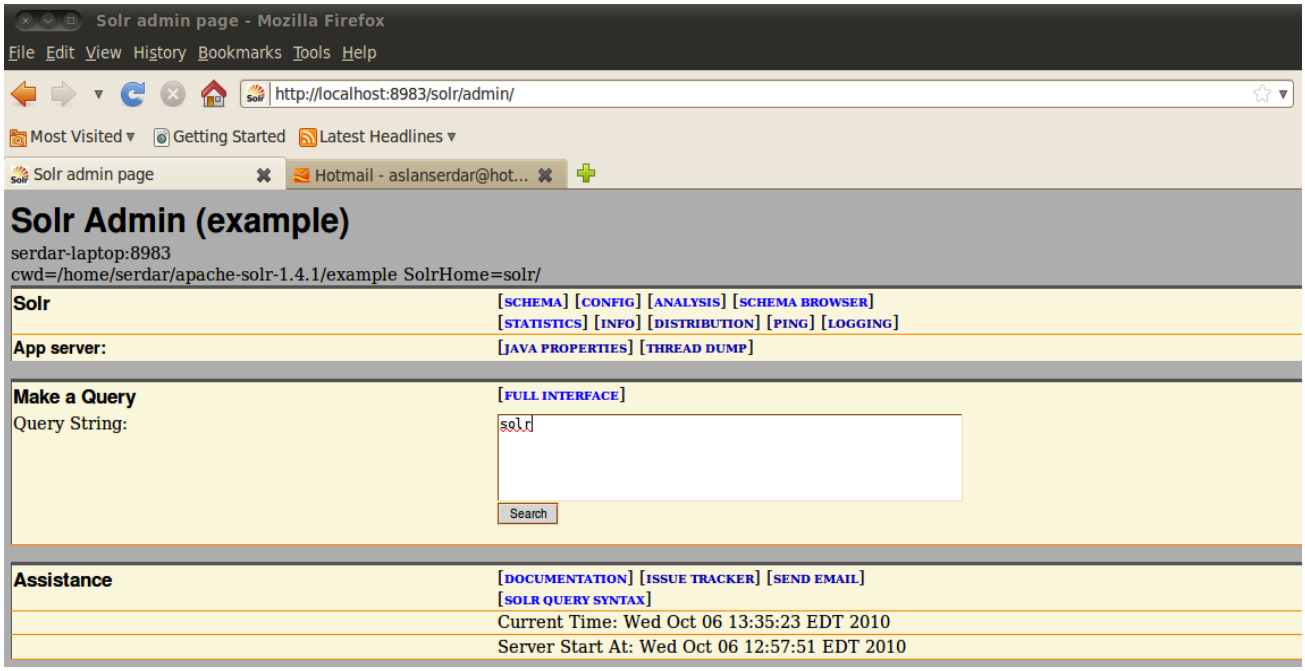

Figure 2: Solr Admin page

- **Adding/Updating/Deleting data**

  Adding

  Following is a sample input file:

  <add>
  <doc>
  <field name="id">10</field>

<field name="name">INFORMATION BOOKS ARE JUST THE BEGINNING. THE MICHIGAN WHITEHOUSE CONFERENCE ON LIBRARY AND INFORMATION SERVICES. A REPORT. </field>
<field name="content">DETAILS THE PROCEEDINGS OF THE MICHIGAN PRE-WHITE HOUSE CONFERENCE ON LIBRARIES AND INFORMATION SERVICES 26-28 MAR 79. DEMONSTRATIONS DISPLAYS DISCUSSIONS ELECTIONS AND PARTICIPANTS ARE DESCRIBED AND THE 68 RESOLUTIONS ARISING FROM THE DISCUSSIONS ARE LISTED ACCORDING TO THE FOLLOWING TOPICS (1) FINANCE (2) BARRIERS TO LIBRARY USE (3) OUTREACH (4) PUBLIC RELATIONS (5) NETWORKING AND (6) THE IMPACT OF TECHNOLOGY ON LIBRARY COLLECTIONS. REPORT NOT AVAILABLE FROM NTIS. </field>
</doc>
</add>

The *<add>* tag instructs Solr to add the document identified by the *<doc>* tag. The *<field>* tags indicate the different fields present in the document.

Deleting
Delete can be done by:

- Posting a delete command and specifying the value of a document's unique key field.
  java -Ddata=args -Dcommit=no -jar post.jar
  "*<delete><id>SP2514N</id></delete>*"
- Posting a delete command and a query that matches multiple documents.
  java -Ddata=args -jar post.jar
  "*<delete><query>name:DDR</query></delete>*"
- Don't forget to update data "java -jar post.jar"

Updating
User can edit the existing XML file to change data and then run the "java -jar post.jar" command.

- **Indexing data**

Apache Solr can index data using four mechanisms namely:

- Solr's native XML
- CSV (Character Separated Value) which is a character separated value format (often a comma)
- Rich documents like PDF, XLS, DOC and PPT

- Solr-Binary is analogous to Solr-XML – it contains the same data in binary format.

We have stored our data to be searched in the example/exampledocs folder.
First of all ensure that the Solr server is running from the previous step, then type the following:

exampledocs$ java -jar post.jar *.xml

The response should be something like:
SimplePostTool: version 1.2
SimplePostTool: WARNING: Make sure your XML documents are encoded in UTF-8, other encodings are not currently supported
SimplePostTool: POSTing files to http://localhost:8983/solr/update
SimplePostTool: POSTing file xmllisa1.xml
SimplePostTool: COMMITting Solr index changes

- **Querying data**
  - **How it works**

Consider the query "Library". The response for it is as follows:

```
<response>
        <lst name="responseHeader">
        <int name="status">0</int>
        <int name="QTime">6</int>
        <str name="indent">on</str>
        <str name="start">0</str>
        <str name="q">Library</str>
        <str name="version">2.2</str>
        <str name="rows">10</str>
        </lst>
        </lst>
        <result name="response" numFound="1" start="0">
                <doc>
                        <str name="content">
                        CRITICALLY EXAMINES THE INDIAN LIBRARY
                        SCENE. THE MANAGEMENT, OPERATION, AND
                        SERVICES OF MOST LIBRARIES IS
                        INEFFICIENT AND INEFFECTIVE, RESULTING
                        IN RAVEMIS ALLOCATION AND UNDER
                        UTILISATION OF LIBRARY RESOURCES.
                        LITTLE MAJOR RESEARCH HAS BEEN
                        CARRIED OUT. UNIVERSITY LIBRARIANSHIP
                        COURSES ARE NOT MUCH HELP FOR
                        MANAGING AND OPERATING LIBRARIES
```

EFFICIENTLY. THE MAJOR NATIONAL INSTITUTIONS HAVE SERIOUS LIMITATIONS. TO IMPROVE THE SITUATION, PROPOSES THE ESTABLISHMENT OF AN INDIAN COUNCIL OF LIBRARY AND INFORMATION SERVICES RESEARCH AND TRAINING. THE COUNCIL WOULD BE SETUP BY THE GOVERNMENT AND HAVE WELL-DEFINED ROLES AND FUNCTIONS COVERING ALL ASPECTS OF INDIAN LIBRARIANSHIP.
</str>
<str name="id">1</str>
<str name="name">
THE INDIAN COUNCIL OF LIBRARY AND INFORMATION SERVICES RESEARCH AND TRAINING:A PROPOSAL FOR CONSIDERATION.
</str>
</doc>
</result>
</response>

The first section of the response, which precedes the <result> tag that is about to follow, indicates how long the query took (measured in milliseconds), as well as listing the parameters that define the query. Solr has some sophisticated caching. You will find that your queries will often complete in a millisecond or less, if you've run the query before. In the params list, q is the query. The numFound number is self explanatory. start is the index into the query results that are returned in the XML.

- **Simple Queries**
  Apache Solr uses HTTP Get in its URL to execute queries.

  You can use the "fl" parameter to control what stored fields are returned, and if the estimated relevance score is returned.

  q=library&fl=name,id (return only name and ID fields)

  q=library&fl=name,id,score (return estimated relevance score as well)

  q=library&fl=*,score (return all stored fields, as well as estimated relevance score)

  q=library&sort=id desc&fl=name (add sort specification: sort by ID descending)

  q=library&wt=json (return response in Java Script Object Notation (JSON) format)

- **Highlighting**
  As the name suggests, it highlights the corresponding field.
  Example query through web browser: ...&q=library
  &fl=name,id,score&hl=true&hl.fl=name

- **Faceting**
  - It's a dynamic clustering of search results into categories.
  - Allow users to refine their search result
  - Generates counts for various properties or categories
  - Also called faceted browsing, faceted navigation
  - The benefits:
    - Superior feedback
    - No surprises or dead ends
    - No selection hierarchy is imposed

- **Spell checking**
  To perform spell checking Solr uses the term dictionary as the collection of correctly known spelled words. If needed Solr can also be configured using a specified text file of words.

- **Query elevation**
  Sometimes the solution to a user query may not match the user expectations. In such a situation it may be desired to make editorial modifications to the search results of a particular user query and hence query elevation is used.

- **Dismax query handler**
  The dismax handler has the following features over the standard handler:
    - Searches a cross multiple fields with different boosts
    - Automatic phrase boosting of the entire search query
    - Can specify the minimum number of word to match, depending of the number of the words in a query string

- **Wildcard queries**
  Wildcard query is a term that contains "*". It is used when the user uncertain of a query term or the user seeks documents containing variants of a particular term. In Apache Solr wildcard queries a*b are constant scoring (all matching documents get an equal score).

- **Text Analysis**
  Text analysis helps match user queries with phrases present in the documents. Figure 3 depicts how user query "power-shot sd500" is matched with the phrase "PowerShot SD 500" present in the document.

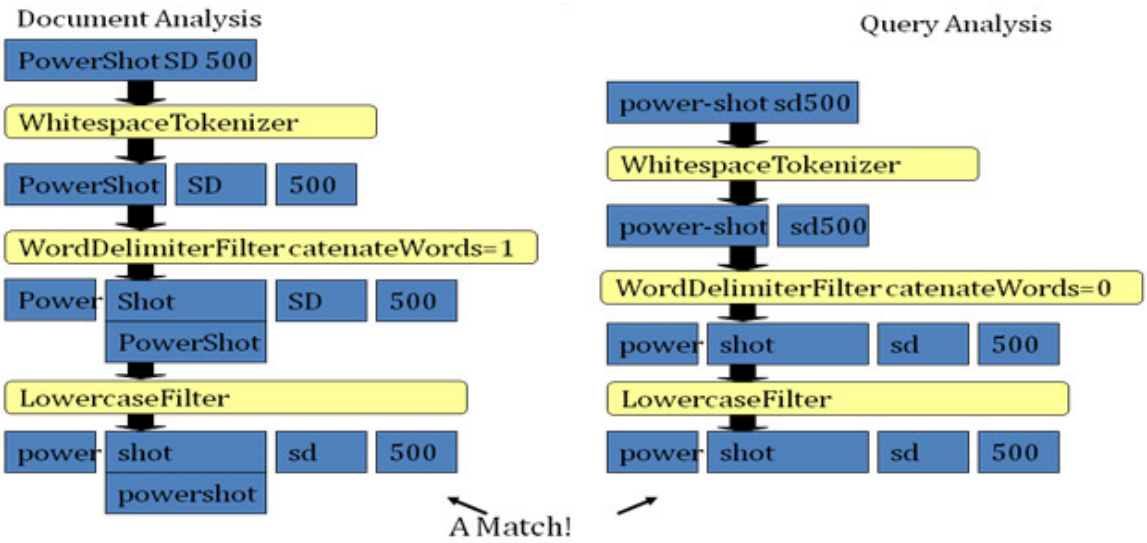

Figure 3: Document and Query Analysis (Seeley Y. 2006,"Apache Solr")

Text analysis is a topic that covers stop words, stemming, n-gram, synonyms and miscellaneous text processing methods.

- **Stemming**: Stemming is the process for reducing inflected words to their stem, base or root form.

- **Synonyms**: Synonym processing is used when someone searches using a word that was not in the original document but is synonymous with a word that is indexed and you want that document to match the query.

- **Stop words**: Apache Solr uses a simple filter called StopFilterFactory that filters out certain so-called stop words. For indexes containing lots of texts, common and interesting words like "the", "a " and so on make the index large and slow down phrase queries. To deal with this problem it's best to remove stop words from fields where they show up often.

- **N-grams**: N-gram analysis slices text into many smaller substrings ranging between a minimum and maximum configured size. N-gramming happens only index time. There is a high price to be paid for n-gramming because they translate to greater index sizes and thus take a longer time to index.

- **Other features**
  Apache Solr has other complex features like:
  - **Distributed search**: Distributed search is used for extremely large datasets, when an index becomes too large to fit in a single system or when a single query takes too long to execute. In such cases an index can

be split into multiple shards and Solr can query and merge results across these shards.

- **Numeric field statistics**: Numeric field statistics for Apache Solr returns simple statistics like min, max, sum, mean, stddev and sum of squares for the numeric fields specified within the document set.

- **Search result clustering**: Search result clustering component clusters both the search results as well as documents. This feature is currently experimental.

- **Function queries**: Function query allows us to use the actual value of a field and functions of those fields in an estimated relevance score. There are various ways in which function queries can be invoked –– embed a function query in a regular query, invoke the function query plugin or set function query as the default query type.

- **Boosting**: Boosting is used to give some fields more weight than others while searching. It can also be used to give some high quality documents more priority over other documents.

- **More-Like-This**: More-Like-This gives suggestions for a given document. This constructs a Lucene query based on terms within a document.

- **Relationship to Chapters from the textbook**
  Apache Solr uses some of the concepts described in Chapters 6 and 7 like:
  - Tokenization
  - Scoring tf-idf (Lucene Class Similarity)
  - Lucene Practical Scoring : Figure 4 depicts the formula of the Lucene practical Scoring.

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t \text{ in } q} \left(tf(t \text{ in } d) \cdot idf(t)^2 \cdot t.getBoost() \cdot norm(t,d) \right)$$

Figure 4: Lucene Conceptual Scoring Formula (http://lucene.apache.org)

  - Boosting – documents, queries
  - Wildcard queries (te?t,test*, te*t)
  - Clustering (result clustering via Carrot2)
  - Lucene's Conjunctive Search Algorithm uses skip pointers

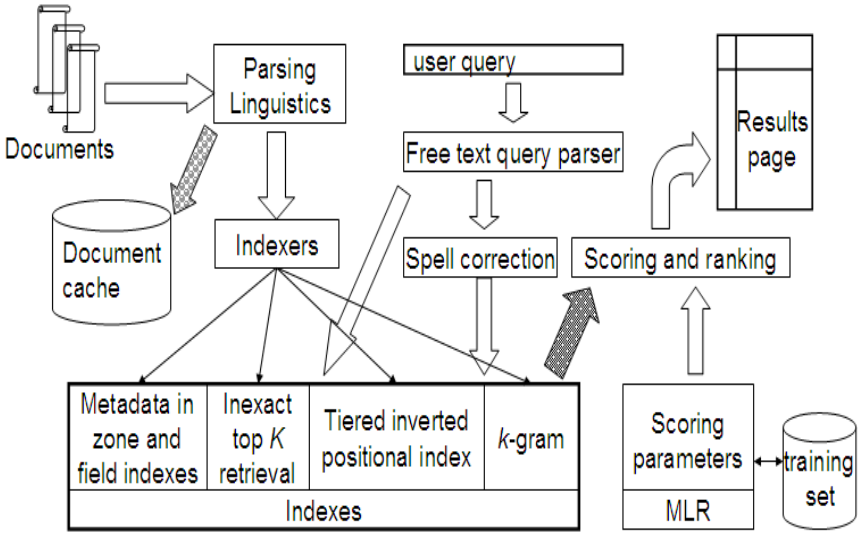

Figure 5: Chapter 7, Information Storage and Retrieval
(Christopher D. Manning, Prabhakar Raghavan and Hinrich Schutze)

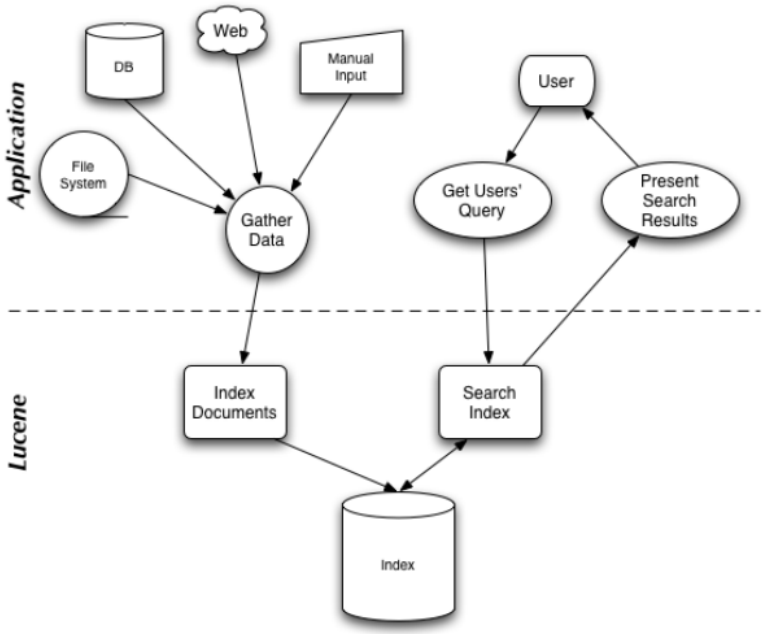

Figure 6 : Chapter 1, Lucene In Action
(Otis Gospodnetic and Erik Hatcher)

Figure 5 and Figure 6 can be compared to illustrate the correlation between how searching is described in the textbook by Manning C. D. et al. (2009), *"In Introduction to Information Retrieval",* and how Lucene performs searching.

10. Resources

Required reading for students

- Smiley D., Pugh E. (2009). *Solr 1.4 Enterprise Search Server,* Packt Publishing,

- Manning C. D. et al. (2009). *Chapter 6. Scoring, term weighting and the vector space model.* In Introduction to Information Retrieval. Cambridge University Press.

- Manning C. D. et al. (2009). *Chapter 7. Computing scores in a complete search system.* In Introduction to Information Retrieval. Cambridge University Press.

Recommended reading for students

- Solr web page http://lucene.apache.org/solr/
- Solr wiki page http://wiki.apache.org/solr/FrontPage
- Wikipedia webpage http://en.wikipedia.org/wiki/Apache_Solr
- Getting started with Apache Solr http://www.youtube.com/watch?v=eRQeYiuPgMA

11. Concept map

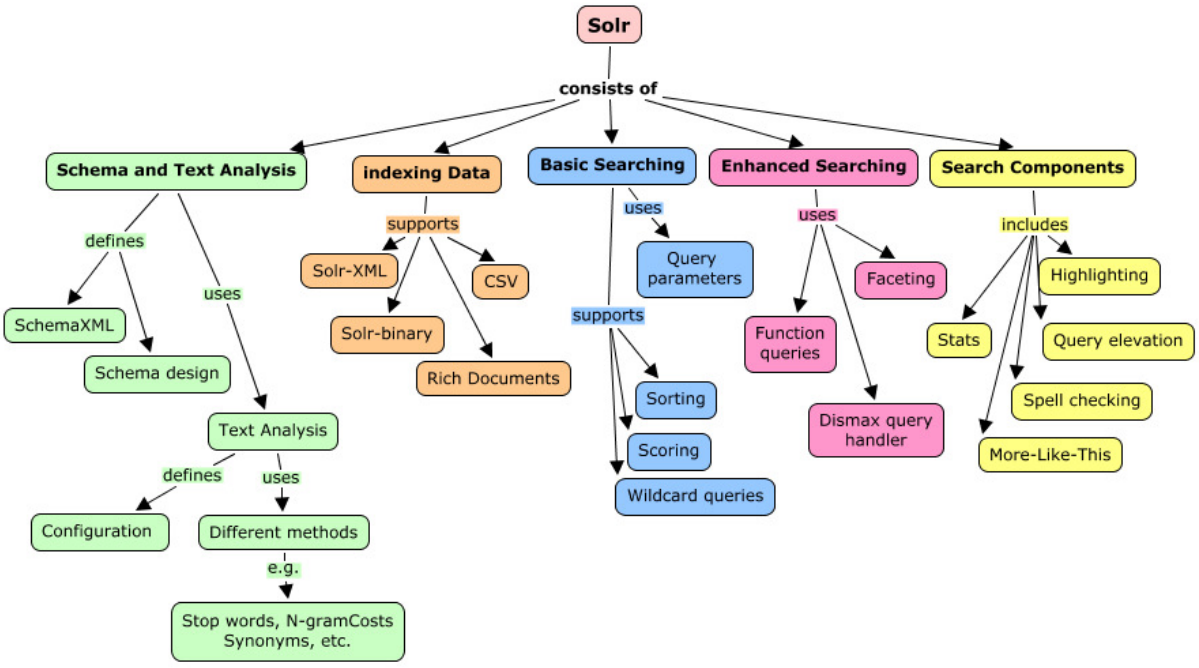

Figure 6: Solr concept map

12. Exercises/Learning activities

General Instructions:

- A folder for each team has been created on the IBM cloud instance. The folders are named team1, team2, team3, team4 and team5.
- Each team folder has a dataset folder which contains the LISA and UMW courses dataset. The data has been converted to XML format and is ready for use. You required to only copy it to the corresponding folder apache-solr/examples/example-docs/ and then run your computations.

i) Data Set-1 Courses

(1) John, an undergraduate at UWM, has just entered his senior year. Being an all-rounder he has to find a timetable that strikes the perfect balance between the courses. The wide variety of courses offered by UMW just adds to his confusion.

Your duty is to help him choose the right courses by following the steps given below. The data related to his courses is present in team1/dataset/uwm/:

(a) Index the data related to the queries so that it can be used to process queries. (Provide a screenshot.)

(b) Find the available classes that John can take as an undergraduate student**.** (Provide the number of undergraduate classes and enclose a screen shot.)
Note: "U" indicates an undergraduate level course.

(c) John has an interest for Calculus classes. List the titles of available undergrad calculus classes. (Write the number of classes and provide the corresponding screenshot.)

(d) Help him to find most relevant calculus class. (Provide the title of the class and the instructor name. Also provide the corresponding screenshot.)

ii) Data Set-2 LISA

(1) The LISA data set has been converted to XML format and stored in folder team1/data/lisa1.
   (a) Add the data to Apache-Solr and index the added data. Attach screen shots of how Solr indexes the files.

(2)
   (a) Find the articles (name + body) that have the word "Library" occurring in them and display their titles in descending order of their score.

(b) Highlight the occurrences of the query term in content. Write the query and enclose screenshots.
(c) Find the time taken to process the query.

(3) What should you do to delete documents that contain “Library” in their title? Perform the deletion operation. Give a screenshot showing that the deletion was successful.

(4)
(a) Now search for the articles that have the word “library”. Attach the screenshot.
(b) Query for articles that have “library” in their title. Attach a screen shot.

(5) What are the different ways in which Apache Solr will process the query "FEBRUARY 10-MARCH 31, 1979"?

13. Evaluation of learning achievement

At the end of this module, students should understand the workings of Apache Solr. They must be able to formulate queries and execute searches.

Online quiz:

- http://ir.ims.uni-stuttgart.de/mod/quiz/view.php?id=35
- http://ir.ims.uni-stuttgart.de/mod/quiz/view.php?id=36

Question from the textbook (Manning C. D. et al. (2009). *Chapter 6. Scoring, term weighting and the vector space model.* In *“Introduction to Information Retrieval”*):

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital			10,000						
video			100,000						
cameras			50,000						

Table 1

Compute the vector space similarity between the query “digital cameras” and the document “digital cameras and video cameras” by filling out the empty columns in the above Table 1. Assume N = 10,000,000, logarithmic term weighting (wf columns) for query and document, idf weighting for the query only and cosine normalization for the document only. Treat ‘and’ as a stop word. Enter term counts in the tf columns. What is the final similarity score?

14. Glossary

- *Collection*: A group of items, often documents
- *Feature:* Information extracted from an object and used during query

processing

- *Index:* A data structure built for the documents to speed up searching

- *Information Retrieval:* Part of Computer Science that studies retrieval of information (not data) from a collection of written documents or of other types of multimedia content

- *Metadata:* Attributes of a data item or a document or other digital object

- *Query:* The expression of the user information need in the input language supported by the information system

- *Relevance feedback:* An interactive process of obtaining information from the user about the relevance and the non-relevance of retrieved documents, automatically constructing revised queries using that information, and retrieving new documents

15. Contributors

a. Iccha Sethi
b. Serdar Aslan
c. Dr. Edward Fox

Information Storage and Retrieval CS 5604
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA