LucidWorks: Searching with cURL
October 1, 2012

1. **Module name**: LucidWorks: Searching with cURL
2. **Scope**: Utilizing cURL and the Query admin to search documents
3. **Learning objectives**
   Students will be capable of:
   a. Querying an index, working with results, and describing query parsing
4. **5S characteristics of the module**
   a. Streams: Queries are tokenized and parsed for searching over an indexed document repository. These queries can be free form text or field based.
   b. Structures: Indexed documents are stored and retrieved in various formats (e.g., XML and JSON). The input query also can be in a specific format. Documents are indexed based on Lucene-index methods.
   c. Spaces: The documents and queries fit into a vector space and are shared in the data space on the server running LucidWorks.
   d. Scenarios: Scenarios include users submitting search queries that are arbitrary text or based on specific field names and values. These queries are applied to the document index stored on LucidWorks.
   e. Societies: End users of the system, as well as researchers and software engineers.
5. **Level of effort required**
   This module should take at least 4 hours to complete.
   a. **Out-of-class:** each student is expected to work at least 4 hours to complete the module and the exercises. Time should be spent studying the documentation for LucidWorks Document Retrieval. Time should be spent reading the material from the Textbook and Lucene chapters [12e, 12h].
   b. **In-class:** students will have the opportunity to ask and discuss exercises with their teammates.
6. **Relationships with other modules** (flow between modules)
   a. Apache Solr Module
7. **Prerequisite knowledge/skills required** (what the students need to know prior to beginning the module; completion optional; complete only if prerequisite knowledge/skills are *not* included in other modules)
   a. Lucene Ch. 2 Indexing
   b. Overview of LucidWorks Big Data Software

c. Lucene Ch. 5 Advanced Search Techniques

d. Lucene Ch. 6 Extending Search

8. Introductory remedial instruction

a. None

9. Body of knowledge

The student should know and study (a) below. The student should study section (b) with respect to LucidWorks. Additionally, if the student wishes for more advanced work, see sections (c) and (d).

a. Understanding tf-idf weighting and scoring (Textbook: Chapter 6)

i. Term Frequency and Inverse Document Frequency is a weighting scheme that calculates the weight for each term that appears (or does not appear) in a given document.

ii. Conceptually, the following occurrences hold true for tf-idf calculations:

1. Higher weight when a term occurs many times in a small number of documents.

2. Lower weight when a term occurs less in a single document, or it occurs in many documents.

3. Lowest weight occurs when a term is found in all (or almost all) documents.

iii. The tf-idf weight is determined by the following equation:

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

b. Basic cURL Fetch (Lucene: IndexSearch)

i. cURL is a command line tool and library (libcurl) that is used to transfer data to or from a URL. cURL supports a large number of protocols (HTTP, HTTPS, FTP, SCP, SMTP, etc.). A simple cURL command is the following.

curl http://www.vt.edu

This command simply uses HTTP to fetch the HTML document at www.vt.edu and prints the result to stdout. If a protocol is not explicitly provided (e.g., the "http://" part above), cURL will oftentimes default to HTTP. The cURL command has a large number of command line arguments that may be specified, but the ones relevant to this module will be covered here. The rest may be reviewed in cURL's man page.

The -u and --user flags may be used for server authentication. Consider the following semantically identical examples.

curl -u user:pw http://example.com/
curl --user user:pw http://example.com/

Where "user" and "pw" must be substituted with the user's username and password, respectively. The -X and --request flags may be used with an HTTP server to specify a specific HTTP request method. If this is not explicitly supplied, cURL will default to the GET request. The following examples use POST rather than GET.

```
curl -X POST http://example.com/
curl --request POST http://example.com/
```

If you are trying to do a POST request, you need data to POST. The arguments -d and --data may be used to specify what data to POST to the HTTP server.

```
curl -d "example data" http://example.com/
curl --data "example data" http://example.com/
```

It should be noted that if the data argument is supplied, cURL will default to a POST request; one need not supply both arguments. One can specify any number of extra HTTP headers using the -H or --header argument.

```
curl -X POST -H 'Content-type: application/json' http://example.com/
curl -X POST --header 'Content-type: application/json' http://example.com/
```

These two commands state that we will be POSTing JSON data. The following command will query the LucidWorks server for documents containing the text "example".

```
curl -u foo:bar -H 'Content-type:application/json' -d
'{"query":{"q":"text:example"}}'
http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents/retrieval
```

This command attempts to authenticate the user "foo" with the password "bar", adds a custom header to the HTTP request specifying that we'll be supplying JSON data, specifies the data to send to the server (note that we're implicitly doing a POST request by specifying this data), and specifies the URL of the LucidWorks document collection to query.

In order to understand the above query, one must first understand JSON format. Put simply, a JSON object is simply a listing of fields and their values.

```
{
   "stringField" : "foo",
   "numField" : 0,
   "boolField" : true
}
```

Additionally, JSON objects may contain inner objects. The above query is passing the following JSON object to the LucidWorks server.

```
{
   "query" : {
      "q":"text:example"
   }
}
```

The server will also return a JSON object. The output of cURL may be piped into python -mjson.tool to provide proper JSON formatting (for ease of reading). For example:

```
curl -u foo:bar -H 'Content-type:application/json'-d
'{"query":{"q":"text:example"}}'
http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents/retrieval | python -mjson.tool
```

c. Performing arbitrary text-based searches (Lucene: QueryParser)
   i. Text-based searches can be performed on the full-text index of a document using the “text” keyword as part of the query.
   ii. All queries interpreted by Lucene are split using a colon (“:”). For example, to search some arbitrary text one might use the following query: “text:foo”
   iii. Query strings can contain Boolean operations like AND and OR
   iv. All special characters must be escaped in the string (i.e., ^ or ?)
   v. Lucene allows query terms to weighted (or boosted) depending on the user’s requirements
d. Performing field-based searches (Lucene: FieldQuery)
   i. Field-based queries can be performed on specific content types and terms that are indexed
   ii. Field-based queries can be ranged values like dates and numbers
e. Advanced searching techniques (Lucene: Chapter 5)
   i. Lucene allows for multiple field searching which the student will get to experience in Exercise 10.b.IV
   ii. Multi-phrase queries are used quite extensively in particular industries and Lucene provides support (if needed)
   iii. Lucene provides searching over multiple indexes. This mainly used by large applications that require multiple indexes
   iv. Term vectors can be created on the fly from particular queries to be used on a collection. These vectors are the equivalent of an inverted indexed.

10. Exercises / Learning activities

a. **Using cURL on the command line to fetch documents.** For this exercise, please write out your solutions and submit them to the instructors.

   i. Explain how one might fetch the HTML document located at www.vt.edu using HTTP.

   ii. Sometimes, web servers may refuse to respond to requests not sent from a popular web browser. Explain how one might use the User-Agent HTTP header to spoof the appearance of a proper web browser in cURL.

   iii. The web server at http://domain.com/ requires authentication in order to POST data. Given the username "foo" and the password "123456", give an example of how one would authenticate them when POSTing data to this server.

   iv. You've discovered that http://domain.com/json returns data in poorly formatted JSON. Give an example of how you might properly format this from the command line.

b. **Fetching Reuters Articles from LucidWorks.** For this exercise, please work with the Reuters News collection on the LucidWorks system that we are using for the class. Please document the step-by-step process you executed, and submit it with your answers.

   You were recently hired by an independent news outlet, and part of your training requires you to catalogue and search for archived news bulletins. Since you are an accomplished computer scientist, you decide to use your favorite command line tool to search for documents. Furthermore, your boss wishes you to focus on articles relating to Oil.

   i. Using a cURL command, fetch all documents that contain “Oil” and show the results.

   ii. Now that we have all the documents, our boss wants to focus on the Oil Price. How do we change the query to search for these documents? Perform the search, and show the results.

   iii. How would you sort these results by the date they were added to the system showing the most recent first? What field did you use?

   iv. Your boss instructs you to find all documents that were indexed by the system at a certain date and time: 2012-10-01T20:03:02.507Z. What field would you search on? What would the query look like? What are the document ID’s that match your query?

   v. How would you create a query that matches on two different fields? What would the syntax of a query look like if we were to use text and the date/time string from above?

c. **Textbook Exercise 6.10, Pg. 110**
Consider the table of term frequencies for 3 documents denoted Doc1, Doc2, and Doc3 in Figure 1. Compute the tf-idf *scores* for the terms car, auto, insurance, and best, for each document, using the idf values from Figure 2.

	Doc1	Doc2	Doc3
car	27	4	24
auto	3	33	0
insurance	0	33	29
best	14	0	17

Figure 1: Table of tf values

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Figure 2: Example idf values.

11. Evaluation of learning objective achievement

a. Understanding of the conceptual use of cURL
b. Accurate fetching of documents
c. Correct computation of tf-idf values

12. Resources

a. http://fetcher.dlib.vt.edu:8888/solr/#/test_collection_vt/query
   i. The Administration panel for Apache Solr and our Reuters News Collection
b. http://lucidworks.lucidimagination.com/display/bigdata/Document+Retrieval
   i. Documentation on LucidWorks
c. http://curl.haxx.se/docs/manpage.html
   i. cURL Documentation
d. http://lucidworks.lucidimagination.com/display/bigdata/Document+Indexing
   i. Information on Document Indexing
e. McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Chapter 3: Adding search to your application. In *Lucene in Action* (2nd Ed.). Stamford: Manning Publications Co.
   i. Identifying and using specific query syntax for Lucene

f. McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Chapter 5: Advanced search techniques. In *Lucene in Action* (2nd Ed.). Stamford: Manning Publications Co.
    i. Discussion on advanced topics for Lucene
g. McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Chapter 6: Extending search. In *Lucene in Action* (2nd Ed.). Stamford: Manning Publications Co.
    i. Discussion of creating custom search and index function in Lucene
h. Manning, C., Raghavan, P., and Schutze, H. (2008). Chapter 6: Scoring, term weighting, and the vector space model. In *Introduction to Information Retrieval.* Cambridge: Cambridge University Press.
    i. Information and discussion on term weights and the tf-idf weighting scheme

13. Glossary

a. cURL – command line tool for transferring data with URLs
b. Query – a search string for particular items
c. LucidWorks – a secure, enterprise-level, searching system built on Apache Solr
d. Apache Lucene – a text search engine library built with Java

14. Contributors

Authors: Kyle Schutt (kschutt@vt.edu), Kyle Morgan (knmorgan@vt.edu)
Reviewers: Dr. Edward A. Fox, Kiran Chitturi, Tarek Kanan
Class: CS 5604: Information Retrieval and Storage. Virginia Polytechnic Institute and State University