LucidWorks: Advanced Searching with cURL

(October 7, 2012)

1. Module name: LucidWorks: Advanced Searching with cURL.

2. Scope: This module focuses on advanced search techniques using Apache Solr through cURL.

3. Learning objectives

Successful completion of this module will enable students to employ advanced search techniques based on multi-values, multi-fields, phrase queries, query term proximity, boosting, etc. Also, students will be able to sort and display returned results in various ways.

4. 5S characteristics of the module

a) **Streams:** Different types of search queries that are translated into tokens for searching over an indexed document collection. These queries can be span, phrase or field-based queries.

b) **Structures:** Indexed document collections are stored and retrieved in various formats (e.g., XML and JSON). The input query also can be in a specific format.

c) **Spaces:** The indexed documents and queries are logically represented as vectors in a vector space. The document collections are physically stored on the server running LucidWorks.

d) **Scenarios:** Scenarios include users searching for a phrase, retrieval based on fields and ordering results based on sorting and scoring functions. These queries are applied to the document collection which is indexed and stored on LucidWorks.

e) **Society:** End users of the system such as researchers, librarians, software developers who wish to use LucidWorks for advanced searching.

5. **Level of effort required**

The required amount of time to complete this module should be about 5 hours.

a. **Out-of-class:** 4-5 hours
b. **In-class:** Students may ask questions and discuss exercises with their teammates.

6. **Relationships with other modules (flow between modules)**

Follows after module 1 – Searching with cURL.

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), Ch.3 (Searching)

b. Overview of LucidWorks Big Data Software

c. Basic cURL commands.

8. **Introductory remedial instruction (the body of knowledge to be taught for the prerequisite knowledge/skills required; completion optional)**

a. LucidWorks (creating an index)

9. **Body of knowledge:**

a. Using cURL from command line to fetch documents (Lucene: Advanced search techniques)

Note: Since basic cURL commands and their usage have already been explained in module 1, we assume that students are already familiar with them.

i) Phrase Search:

Command

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

Description

The above command tries to fetch documents that contain the phrase “think tank” in their ‘text’ field. It is important to notice how the search phrase is wrapped in double quotes and properly escaped: \”think tank\”.

ii) **Search with Query Term Proximity:**

Command

```
curl -u foo:bar -H 'Content-type:application/json' -d '{"query":{"q":"text:\"Center Information\"~3"}}' http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/documents/retrieval | python -mjson.tool
```

Description

The above command helps in finding the documents that contain the words “center” and “Information” with no more than 3 words appearing between them. These queries are also referred to as Sloppy Queries where “slop” is the amount of sloppiness allowed or the number of words that are allowed to appear between the query terms. Slop can be provided in the queries with a ‘~’, like ‘~3’ in the above query.

iii) Multiple Value Search:

Command

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

Description

The above command searches for documents that contain both ‘Javier’ and ‘Reagan’ in the ‘person’ field.

iv) **Search on Multiple Fields:**

Command

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

Description

The above command tries to fetch the documents based on two fields: person and text. The results returned would be documents that contain ‘Reagan’ in the ‘person’ field and ‘Oil’ in ‘text’.

v) Sorting Results:

Command

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

Description

Search results are typically returned in descending order of their relevance. However, should there be a need to sort the results on some other field/parameter and/or in a different sorting order, it could be achieved through the ‘sort’ query parameter. The ‘sort’ parameter must include a field name and sort order, ‘asc’ for ascending or ‘desc’ for descending order.

By default, sorting order is by score in descending order.

In the above command, the user has queried for documents that contain the term ‘oil’ in the ‘text’ field and are sorted in the ascending order of their ‘score’.

vi) Sorting Results on Multiple Fields:

Command

```
curl -u foo:bar -H 'Content-type:application/json' -d
'{"query":{"q":"text:Oil","sort":"score asc, id
desc"}}' http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/do
cuments/retrieval | python -mjson.tool
```

Description

Search results can also be sorted on multiple fields. The above query suggests that the results shall be sorted on ‘score’ in ascending order first and then on the ‘id’ in descending order.

vii) Limiting Fields Returned in Results:

Command

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

Description

The above command helps in controlling the fields that are returned in results. Typically, when search results are returned, all the fields in each document are returned. However, should there be a need to fetch only certain fields, an additional query parameter ‘fl’ may be used to specify the fields of interest. A delimiter ‘,’ can be used to specify multiple fields.

The above query returns only the ‘text’ and ‘person’ fields of the matching documents.

viii) Stopping a slow search:

Command

```
curl -u foo:bar -H 'Content-type:application/json' -d
'{"query":{"q":"text:Oil","fl":"text","timeAllowed":"100"}}' http://fetcher.dlib.vt.edu:
8341/sda/v1/client/collections/test_collection_vt/documents/retrieval | python -
mjson.tool
```

Description

In some situations it may be important to retrieve documents within a certain period of time irrespective of whether or not the search has terminated successfully. In such cases, the query parameter ‘timeAllowed’ can be used to specify a time limit, in milliseconds, beyond which the query shall not continue.

The above query will return the documents that are found in the first 100 milliseconds and also contain the term ‘Oil’ in their ‘text’ field.

ix) Limiting no. of returned results:

Command

```
curl -u foo:bar-H 'Content-type:application/json' -d
'{"query":{"q":"text:Oil","fl":"text","rows":"5"}}' http://fetcher.dlib.vt.edu:8341/sda/
v1/client/collections/test_collection_vt/documents/retrieval | python -mjson.tool
```

Description

The above command limits the number of results returned using a query parameter called ‘rows’. In this case, the first five documents that contain the term ‘Oil’ in their ‘text’ field will be returned.

x) Boosting scoring using function queries:

Command

```
curl -u foo:bar -H 'Content-type:application/json' -d
'{"query":{"q":"person:Reagan^10
text:Oil"}}' http://fetcher.dlib.vt.edu:8341/sda/v1/client/collections/test_collection_vt/
documents/retrieval | python -mjson.tool
```

Description

At times, it may be desired to boost certain documents based on a field value.

The above command tries to fetch all the documents that contain the term ‘Oil’ in their ‘text’ field and also boost the documents that contain the term ‘Reagan’ in the ‘person’ field. The character ‘^’ suggests boosting the field that it is attached to by a factor specified after ‘^’.

In the above query, a boost factor of 10 has been added to the field “person”.

10. Exercises / Learning activities

[a] Fetching documents using cURL commands: (Please submit cURL commands for all the exercises in this section.)

i. Explain how to fetch the documents that contain the phrase “Defense Information”

ii. Give the command to fetch all the documents that contain the term ‘United’ in the ‘location’ field. Also, results should be sorted on two field values, say person and rank. The first field shall be sorted in ascending order and the second in descending order.

iii. Give the command to retrieve all the documents that contain "shipping" and "United" in their "text" field.

iv. Suppose you have four documents with their respective contents as follows:

   i. Doc: D1, Content: "Back square"

   ii. Doc: D2, Content: "Back to square one"

   iii. Doc: D3, Content: "Back to being square"

   iv. Doc: D4, Content: "Back with a big square"

   How would you go about writing a query that would fetch D1, D2 & D3 but not D4?

v. You are given a task to search all the blogs on the internet and retrieve a set of one hundred blogs that are authored by 'Obama' and talk about 'Romney'. And, the blogs are required to be ordered in reverse chronological order. How would you go about achieving this? (You are free to assume the existence of any fields that may make your life simpler.)

vi. How would you fetch the documents with the term 'manager' in the 'text' field and 'Larry' in the 'person' field? Also, you have to give more importance to the 'person' field. Suggest a way to do this?

vii. How would you go about searching a repository for documents that contain the terms "Beamer" and "Football" with no more than 25 words between them. However, you cannot afford to take longer than five seconds to return the results even at the cost of not being to return all the results. How would you go about achieving this?

[b] Consider module1- exercise(c), let the static quality scores g(d) of Doc1, Doc2 and Doc3 be 9, 8.5, and 7.5 respectively on a 10 point scale. Show your work and answer to use the computed tf-idf scores and static quality score g(d) to calculate the net-score using the equation:

net-score = .25*g(d) + 0.75*tf-idf

(Hint: The value of g (d) should be between 0 to1. Refer to textbook section 7.1.4: Static quality scores and ordering.)

11. Evaluation of learning objective achievement

The evaluation of learning object shall be done based on the correctness of cURL commands submitted by students.

12. Resources.

[a] cURL command documentation, URL: **http://curl.haxx.se/docs/httpscripting.html**

[b] LucidWorks documentation, URL: **http://lucidworks.lucidimagination.com/display/bigdata/Document+Retrieval**

[c] Document collection, URL: **http://fetcher.dlib.vt.edu:8888/solr/#/test_collection_vt/**

[d] 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.

[e] McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Chapter 5: Advanced search techniques. In Lucene in Action (2nd Ed.). Stamford: Manning Publications Co.

[f] McCandless, M., Hatcher, E., and Gospodnetic, O. (2010). Chapter 6: Extending Search. In Lucene in Action (2nd Ed.). Stamford: Manning Publications Co.

[g] 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.

13. Glossary

14. Additional useful links

[a] Rafal Kuc, Apache Solr Cook Book, Chapter 5: Quering Solr, 2011, URL:**Apache Solr cook book**

15. Contributors

Authors: Hemanth Makkapati (makka@vt.edu), Rajesh Subbiah(csrajesh@vt.edu) and Rushi Kaw(rushik@vt.edu)

Reviewers: Dr. Edward A. Fox, Kiran Chitturi, Tarek Kanan
Class: - CS 5604: Information Retrieval and Storage. Virginia Polytechnic Institute and State University