LucidWorks: Vectorize Workflow

(May 07, 2013)

1. Module name: LucidWorks: Vectorize Workflow

2. Scope: This module focuses on starting, analyzing, and interpreting the Vectorize workflow supported by LucidWorks through their Big Data Software.

3. Learning objectives: Completion of this module will enable users to generate a vectorize workflow in LucidWorks, send queries to LucidWorks for progress of the workflow, and formulate the data into vectors so that the users can use it for searching, machine learning, etc.

Users will gain experience by following the steps in this module to start a workflow that will produce vectorized documents. The exercises are designed to assess whether the students have learned the underlying architecture of LucidWorks and their knowledge of what proper parameters to pass into a vectorize workflow.

4. 5S characteristics of the module

a) Streams: The input stream of data into the vectorize workflow has documents that have not been tokenized or parsed, and are not in a usable format for machine learning and searching.

b) Structure: The collections created and managed by Apache Hadoop's Distributed File System, that are used to store the documents before and after the vectorize workflow. The input and output files can also be in working directories in HDFS.

c) Spaces: The data used in searching and machine learning are represented in vectors spaces, as explained with the vector space model.

d) Scenarios: Users who need vectorized documents for machine learning and searching, can run the text through the vectorize workflow in order to parse their documents into a usable format.

e) Society: The vectorize workflow is for people who wish to use a large amount of data they have gathered for machine learning and searching, in which they can utilize the vectorize workflow to vectorize their documents.

5. Level of effort required

The required amount of time to complete this module should be about **4** hours.

a. **Out of class:** 3 hours to read and learn about the model
b. **In class:** 1 hour in class to do exercises

6. Relationships with other modules:

This module follows after the LucidWorks: Overview module, and can be done in parallel with the Clustering and Classification modules, since the vectorize workflow also deals with the analysis and construction of the results produced by LucidWorks.

7. Prerequisite knowledge/skills required

- Familiarity with UNIX environment
- Overview of LucidWorks Big Data Software
- Basic cURL commands

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

Complete Module 7-i(1): LucidWorks: Overview.

9. Body of knowledge (Theory + Practice)

9.1 Vector Space Model

The vector space model or the term vector model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing, and relevance rankings.

Advantages of the vector space model:

- Simple model based on linear algebra.
- Term weights not binary.
- Allows computing a continuous degree of similarity between queries and documents.
- Allows ranking documents according to their possible relevance.
- Allows partial matching.

Limitations of the vector space model:

- Long documents are poorly represented. They have poor similarity values due to the quantity of information.
- Search keywords must precisely match document terms; substrings may result in a "false positive match".
- Semantic sensitivity; documents with similar context but different term vocabulary won't be associated, resulting in a "false negative match".
- The order in which the terms appear in the document is lost in the vector space representation.
- Assumes terms are statistically independent.
- Weighting is intuitive but not very formal.

The vectorize sub-workflow prepares ingested and extracted content for tasks like document similarity or statistically interesting phrases. In the vectorize workflow,

documents are transformed into vectors suitable for machine learning and searching. This workflow is run before any of the collocations, annotations, k-means, or similar document workflows can be run.

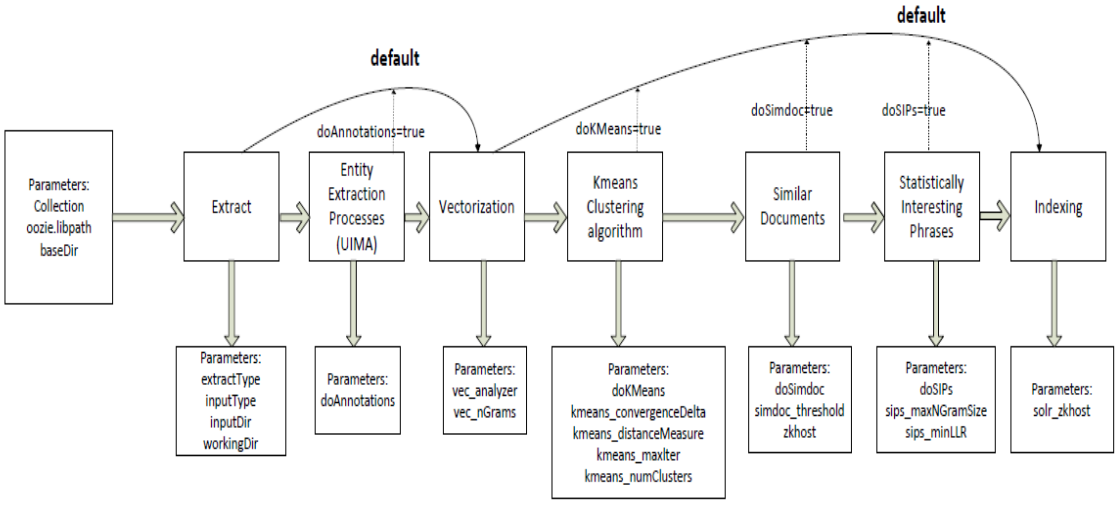

Figure 1. LucidWorks workflow diagram

9.2 Different ways of generating vectors

a) Installing LucidWorks/obtaining access

The latest version of LucidWorks big data software is installed on a virtual machine on the Digital Library Research Laboratory. In order to obtain access to the software, users can contact Kiran Chitturi who manages the virtual machine or visit http://www.dlib.vt.edu/ and contact Dr. Edward A. Fox who is in charge of the lab.

Once users have access to the lab, they are able to SSH into the virtual machine and send cURL commands to LucidWorks in order to generate the vectors desired. Also the virtual machine that is hosted by the DLRL contains example documents that users can use to generate a test collection for the vectorize workflow.

If LucidWorks is not set up on any virtual machine, then users can refer to the documentation at http://docs.lucidworks.com/display/bigdata/All-in-One+Virtual+Machine+Installation

This will give students step-by-step instructions in order to install a virtual instance of LucidWorks and be able to use it.

b) Using Apache Mahout's KMeans

In order to generate vectors, we must first create a collection in LucidWorks. In this module we will demonstrate how LucidWorks uses Mahout to

create a collection using the test data that is already on the system.

```
curl -verbose -X POST -H 'Content-type:
application/json' -d
'{"collection":"collection456"}'http://localhost:8341
/sda/v1/data/collections
```

This command creates a collection named collection456 using Mahout. You may also be interested in piping the commands to `python -m json.tool` in order to make the output from curl more user friendly. For the purpose of our tutorial, we are going to exclude it, but feel free to always pipe it through python.

The first command shown creates the collection. After the creation of the collection we need to add documents and files into this collection.

```
curl -u administrator:foo -X POST -H
'Content-type:application/json' -d
'{"inputDir":"hdfs://localhost:50001/input/reuters/*.
txt", "inputType":"text/plain",
"collection":"collection456", "doKMeans":"true"}'
http://localhost:8341/sda/v1/client/workflows/_etl
```

Output: in JSON format

```
{
        "children": [],
        "createTime": 1365965245000,
        "id": "0000017-130407181220424-oozie-hado-W",
        "status": "RUNNING",
        "throwable": null,
        "workflowId": "_etl"
}
```

This command accesses the Hadoop file system, and looks in the Reuter's folder to retrieve the documents to send through the initial ETL (Extract, Transform and Load) workflow that LucidWorks runs before it sends it to the sub-workflows.

Now that we've created a collection with data inside of it, we can use the Hadoop file management system in order to access and view the documents.

```
hadoop dfs -ls
/data/collections/collection456/tmp/0000000-130407181
220424-oozie-hado-W/document-vectors
```

The -ls command allows you browse Hadoop's distributed file system, then if you want to view the files contained within, you would change the flag to -text. The exact name: `0000000-130407181220424-oozie-hado-W` in this directory is obtained from the output from the creation of collections. If you are not sure what the name of the file is exactly, you can browse Hadoop's file system using the -ls flag and look for the directory where you created your collection.

c) Vectorize Parameters

For vectorizing documents there are a couple of important parameters and arguments that you can change or modify, many of which if unspecified LucidWorks will define as null and continue with the default action for that parameter. We will explain the key parameters below.

-workingDir: takes the collection directory where the documents that you are working with are stored and should be saved. Most of the time this is a collection generated on Hadoop's Distributed File System. This is the general working directory, where the collection is located, as opposed to the specific directories requested below.

-vec_analyzer: takes the type of analyzer that is used to check and generate the tokens for the workflow, in our example we used the doKMeans analyzer instead of Hadoop's analyzer.

-vec_nGrams: specifies the type of n-grams to use when constructing the vectorize workflow, which in simpler terms is just the number of consecutive letters to parse, usually recommend 1 as the parameter. The workflow generates the tokens based off of the type of n-grams specified here. For an example, if the parameter was 3, then trigrams would be used.

-documentsAsText: is where you provide the specific directory with the original documents, that the user wishes to parse/tokenize.

-documentsAsVectors: is where the user specifies the output directory, where the users want the vectorized documents to be.

10. Concept map

Students should create a concept map about the topics contained in this module.

11. Exercises / Learning activities

Have the students break up into groups of 4-5, then have them discuss the following:

1. What are vectors?

2. What is the vector space model useful for?
3. What is the vector space model not useful for?
4. Why are the types of n-grams important for the vectorize workflow?
5. How do you check the status of a workflow?

12. Evaluation of learning objective achievement

The completion of the learning objectives shall be evaluated by the correctness and level of understanding student's response to the exercises. Students will gain the ability to be able to properly submit workflows, check the status of workflows and be able to access and manage the collections in Hadoop.

13. Resources

LucidWorks Big Data Vectorize documentation: (2012)
http://docs.lucidworks.com/display/bigdata/Vectorize

Digital Libraries, LucidWorks Modules: (2013)
http://en.wikiversity.org/wiki/Curriculum_on_Digital_Libraries#Table_3._Modules_for_LucidWorks_Big_Data_Software

Vector Space Model: (2013)
http://en.wikipedia.org/wiki/Vector_space_model

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. New York: Cambridge University Press.

14. Contributors

Authors: David Kniphuisen (david21@vt.edu) and Alan Tran (alan2438@vt.edu)
Reviewers: Dr. Edward A. Fox, Kiran Chitturi