Homework: Building an Apache-Solr based Search Engine for DARPA XDATA Employment Data
Due: November 10th, 12pm PT

---

1. Overview

This assignment picks up where the last one left off. You will take your JSON dataset constructed from upstream TSV files representing job postings collected and crawled from the upstream http://www.computrabajo.com affiliate sites that primarily serve Mexico and South American countries.

In the first assignment you built a simple crawler and deduplication process using Apache Tika (http://tika.apache.org/) and ETLLib (https://github.com/chrismattmann/etllib/) to go from TSV files to the JSON files that comprised the collected metadata about the upstream employment jobs.

In this assignment, you will take those JSON files and you will build an indexing system to send the data into Apache Solr and to create an Employment job search engine. You will develop a content based, and a link based ranking algorithm to return back the appropriate sets of employment jobs based on user queries. The indexing system will be constructed again using the Apache OODT (http://oodt.apache.org/) large-scale data processing system. Apache OODT is particularly effective in dealing with file-based retrieval, processing and management. You will combine OODT with ETLlib and its tooling to get the JSON data files loaded into Solr, and then extend Solr to support your new ranking algorithms.

You should still have a copy of the JSON dataset, the 4 GB TSV dataset, and (for those of you that were given access) the full dataset that consists of ~119 million jobs and is currently ~40GB in size.

2. Objective

The objective of this assignment is to build both an indexing system (inverted index), and a set of ranking algorithms for the XDATA employment dataset that you have been working with. Once developed, you will use your newly generated search index and ranking algorithms to answer a suite of queries or "Challenge Questions" which will illustrate your correctly constructed search engine, and the efficacy and effectiveness of your content-based and your link-analysis based ranking algorithms. The queries that your system will answer range from per job record (and large scale) information, to per multiple job record comparisons and rankings.

Your indexing system will be scalable, and robust, and will be an Apache OODT "workflow" pipeline that prepares the JSON data to be indexed using several "stages" in the workflow pipeline. These stages will be provided by the ETLlib package that you are already familiar with from assignment #1. The ETLLib library provides tools for working with JSON data and for sending it to Solr. You will be responsible for configuring Solr accept your employment JSON data, and you will also be responsible for ensuring that your ranking algorithms are integrated into Solr so that it can use your algorithms to answer user queries.

You will develop two ranking algorithms, both styles of which we discussed in class. The first ranking algorithm will be a "content based" ranking algorithm, and it will use the metadata information present inside of the JSON files to develop a "score" or "rank" for each JSON document. Your content-based algorithm should be tailorable, and easy to adapt based on what types of user queries and challenges it is faced with (e.g., when searching for jobs based on short duration temporal characteristics it should score documents with a startDate and endDate appropriately within that range higher than those not). The other ranking algorithm you will develop is a link analysis algorithm. The jobs have several types of "relationships" or "links" between them. Some of the more obvious ones are: (1) geolocation area; (2) job type, or family; (3) company, etc. Your link analysis ranking algorithm should take these relationships into account and (without searching or ranking based on the internal content inside of the file, it should provide a method for ranking and citation similar to the algorithms we have discussed in class.

Finally, your system should leverage the indexed data in Solr and the associated ranking algorithms that you implement there to provide answers to challenge questions, which will be specified in Section 3.

3. Tasks

1. Develop an indexing system using Apache OODT and ETLlib
    a. Your system should take, as input, the original JSON dataset you created from assignment #1 (the smaller dataset; and if you have it, the larger one too) and then call ETLlib programs to index the JSON documents in Solr.

   b. Capture metrics about the ingestion into Solr using OODT. How long did each job take on average? What were the bottlenecks? What was the overall wall clock time for the full indexing?
2. Develop two ranking algorithms for your JSON documents
   a. A "content based" ranking algorithm. This should inspect the JSON document's "content" and do specific text-based matches by numerically computing a "relevancy" based on the textual content provided, along with the query provided.
   b. A "link analysis" based ranking algorithm. Links should be identified and derived based on "relationships" between the documents (e.g., same geolocation; same job type, etc.) and ranking should be independent of the actual query text provided.
   c. Your ranking algorithms should be integrated into Solr as "scoring" approaches. You should take a look at the Solr "function query" syntax (as a hint): http://wiki.apache.org/solr/FunctionQuery
3. Develop a suite of queries that demonstrate answers to the following challenge questions using your indexed data and your ranking algorithms.
   a. Predict which geospatial areas will have which job types in the future.
   b. Compare jobs in terms of quickly they're filled specifically in regards to region.
   c. Can you classify and zone cities based on the jobs data (E.G. commercial shopping region, industrial, residential, business offices, medical, etc)?
   d. What are the trends as it relates to full time vs part time employment in South America?
4. Develop a program that can either in batch mode or interactively demonstrate answers to the questions from #3. Your program can be a command line program, written in Java and/or Python or Bash, etc.

4. Assignment Setup

4.1 Group Formation

You can work on this assignment in groups sized 2-4. You may keep the same groups you had during the previous assignment. If you have any changes to your group or any questions, e-mail the graders.

Gouthami Kondakindi kondakin@usc.edu
Preethi Ramesh pramesh@usc.edu;

Use subject: CS 572: Team Details

Groups for your 2nd assignment must be confirmed and finalized by Friday, October 17, 2014.

4.2 Dataset

The initial dataset is (still) available at:
http://baron.pagemewhen.com/~chris/employment/

You should also have a copy of your JSON dataset produced in assignment #1. Groups will either have the original (reduced) dataset, or the full dataset. For the purposes of this assignment you can begin from your de-duplicated data.

4.3 Downloading Apache Solr

Download Apache Solr from:

https://lucene.apache.org/solr/downloads.html

The latest version is 4.10.1.

Apache Solr comes with a web application server (Jetty), or you can also deploy and configure Solr with Apache Tomcat. Either way will work fine for this assignment and the instructions are provided here:

https://cwiki.apache.org/confluence/display/solr/Running+Solr+on+Jetty
https://cwiki.apache.org/confluence/display/solr/Running+Solr+on+Tomcat

You should also review the basic installation instructions:

http://wiki.apache.org/solr/SolrInstall

Once installed, you will need to configure Solr to accept your JSON employment data model. You will also be responsible for integrating your ranking algorithms into Solr and for using Solr to query and answer your challenge questions.

Please review Solr function query documentation:
http://wiki.apache.org/solr/FunctionQuery

Your ranking algorithms can either be interactive (per query), or also on document index-time ranking. This is something you will need to figure out as a group and based on the type of ranking algorithm you are developing.

4.4 Downloading ETLlib

ETLlib (https://github.com/chrismattmann/etllib) is a Python based toolkit for extract, transform and load operations around file based data. There are command line tools and APIs for manipulating data in TSV and in JSON.

You can install and download ETLLib per the instructions, here:

https://github.com/chrismattmann/etllib/blob/master/README.md

You will be specifically using ETLlib tools to get JSON documents into Solr.

4.5 Downloading Apache OODT

Apache OODT (http://oodt.apache.org/) is a data processing and management system. You will use OODT in this assignment to "pipeline" together ETLlib steps to index your JSON based employment data into Apache Solr.

Getting started with OODT:
Use the Apache OODT RADIX installer:
https://cwiki.apache.org/confluence/display/OODT/RADiX+Powered+By+OODT

Also please read information about the OODT file manager:

https://cwiki.apache.org/confluence/display/OODT/OODT+Filemgr+User+Guide

You will be responsible for creating:

1. Product type information for your JSON file
2. Capturing basic file metadata about your JSON files that you ingest into Solr using ETLLib

Please also read the OODT workflow manager user guide:

https://cwiki.apache.org/confluence/display/OODT/Catalog+and+Archive
(everything under the Workflow and Resource Manager pages)

You will create a Workflow that will call ETLlib programs to index the JSON documents into Solr.

4.6 Submitting patches and contributing to open source

Please note that if you contribute patches to ETLlib on Github (via pull requests), and if you contribute to OODT or Tika or Solr from this assignment, you will have the opportunity to earn extra credit in a similar fashion to the prior assignment.

5. Report

Write a short 4 page report describing your observations. For example, how effective was the link-based algorithm, compared to the content based ranking algorithm? What challenge questions were more appropriate for the link based algorithm compared to the content one? What do you think were the strengths of OODT? What did you think were the weaknesses of OODT? Do NOT simply provide advantages and disadvantages from a quick Google search. You are required to think critically about this portion of the report and sincerely provide your feedback.

Describe in detail and formally both of your ranking algorithms. You should describe the input, what your algorithms do to compute a rank, how to test them (and prove that they are working as expected).

Describe the indexing process – what does your OODT workflow do? How does it interact with ETLlib?

Also include your thoughts about ETLlib – what was easy about using it? What wasn't? Again a quick Google search will not suffice here.

Please also note that the graders will be given great flexibility to add/subtract from various areas of the report based on your submitted work.

6. Submission Guidelines

This assignment is to be submitted ***electronically, by 12pm PT*** on the specified due date, via Gmail csci572fall2014@gmail.com. Use the subject line: CSCI 572: Mattmann: Fall 2014: Solr Homework: <Your Lastname>: <Your Firstname>. So if your name was Lord Voldemort, you would submit an email to csci572fall2014@gmail.com with the subject "CSCI 572: Mattmann: Fall 2014: Solr Homework: Voldemort: Lord" (no quotes).

- All source code is expected to be commented, to compile, and to run. You should have (at least) one Java source file and should also include other java source files that you added, if any. Do **not** submit `*.class` files. We will compile your program from submitted source.
- Also prepare a `readme.txt` containing any notes you'd like to submit.
- Do **not** include `OODT radix and/or Solr's deployment` in your submission. We already have these.
- Do include your Solr `schema.xml` file for your employment data collection.
- Do include your OODT workflow and file manager policy and state how to install them and overlay them onto RADIX to replicate your indexing workflow.
- However, if you have used any external libraries other than OODT and Solr, you should include develop a Maven and/or Ant/Ivy or mechanism for dynamically downloading these dependencies. Same goes for Python code (use PyPI, setuptools, distribute, etc.). Note in your `readme.txt` a detailed explanation of how to use these libraries when compiling and executing your program.
- Save your report as a PDF file (`Lastname_Firstname_SOLR.pdf`) and include it in your submission.
- Compress all of the above into a single zip archive and name it according to the following filename convention:

  <lastname>_<firstname>_CSCI572_HW_SOLR.zip

  Use only standard zip format. Do **not** use other formats such as zipx, rar, ace, etc.

Important Note:

- Make sure that you have attached the file the when submitting. Failure to do so will be treated as non-submission.
- Successful submission will be indicated in the assignment's submission history. We advise that you check to verify the timestamp, download and double check your zip file for good measure.

6.1 Late Assignment Policy

- -10% if submitted within the first 24 hours
- -15% for each additional 24 hours or part thereof