Creating an Elasticsearch index of Congress Bills using Pig

Recently Mortar worked with Pig and CPython to have it committed into the Apache Pig trunk. This now allows to take advantage of Hadoop with real Python. Users get to focus just on the logic you need, and streaming Python takes care of all the plumbing.

Shortly thereafter, Elasticsearch announced integration with Hadoop. “Using Elasticsearch in Hadoop has never been easier. Thanks to the deep API integration, interacting with Elasticsearch is similar to that of HDFS resources. And since Hadoop is more then just vanilla Map/Reduce, in elasticsearch-hadoop one will find support for Apache Hive, Apache Pig and Cascading in addition to plain Map/Reduce.”

Elasticsearch published the first milestone (1.3.0.M1) based on the new code-base that has been in the works for the last few months.

The intial attempt at testing out Mortar and Elasticsearch didn't work. Working with the great team at Mortar and costinl at Elasticsearch, Mortar was able to update their platform to allow Mortar to write out to Elasticsearch at scale.

Test Case
To test this out, I decided to process congressional bill data from the past several congresses. The process will be to read in the json files, process the file using Pig, use NTLK to find the top 5 bigrams and then write the data out to an Elasticsearch index.

The Data
GovTrack.us, a tool by Civic Impulse, LLC, is one of the world's most visited government transparency websites. The site helps ordinary citizens find and track bills in the U.S. Congress and understand their representatives’ legislative record.

The bulk data is a deep directory structure of flat XML and JSON files. The directory layout is described below.

Our files are in three main directories:

- /data/congress-legislators/
  Information on Members of Congress from 1789-present, presidents and vice presidents, Congressional committees, and current committee assignments. This data is a mirror of the files in github:unitedstates/congress-legislators.
- /data/congress/ (i.e. http://www.govtrack.us/data/congress/)
  Bill status and other legislative data from 2013 (113th Congress) and forward. This data is the output of the scrapers developed by the github:unitedstates/congress project.

Getting the Data

To fetch the data we support `rsync`, a common Unix/Mac tool for efficiently fetching files and keeping them updated as they change. The root of our rsync tree is `govtrack.us::govtrackdata`, and this corresponds exactly to what you see at http://www.govtrack.us/data/.

To download bill data for the 113th Congress into a local directory named bills, run:

```
rsync -avz --delete --delete-excluded --exclude **/text-
versions/ \    govtrack.us::govtrackdata/congress/113/bills .
```

(Note the double colons in the middle and the period at the end. This is a long command. I've indicated the line continuation with a backslash.)

Directories

- `/data/congress/113/bills/[bill_type]/[bill_type][bill_number]/data.`
  Bill and resolution status for bills in the 113th Congress. See the github:unitedstates/congress project documentation for details of the JSON format. The XML format is backwards-compatible with our legacy bill XML files (documentation).

The following code loops through a directory of bills and converts all of the .json files into single line .json files.

The following pig code reads all of the single line .json files, pulls out some of the fields, calls a Python UDF to find the top 5 bigrams and then writes the data into an elasticsearch index.

The important steps are to:

a) register the jar file
b) define the storage to the elasticsearch index
c) write out the data using the defined storage\

If you are using the mortar framework, nltk isn't installed by default. Here's how you can install it:

For the bi-grams, I re-used some sample Mortar code from Doug Daniels shown below:

Results
The pig job loaded 58,624 files, processed them and created the elasticsearch index in 53 seconds. The NLTK python UDF finished in another 34 seconds resulting in a total time of 87 seconds.

You can see the working elasticsearch in the following screen shot:

One thing of note

The elasticsearch hadoop connector doesn't handle geo-coordinates quite yet so you can't create an index with latitude/longitude. That should be coming soon.