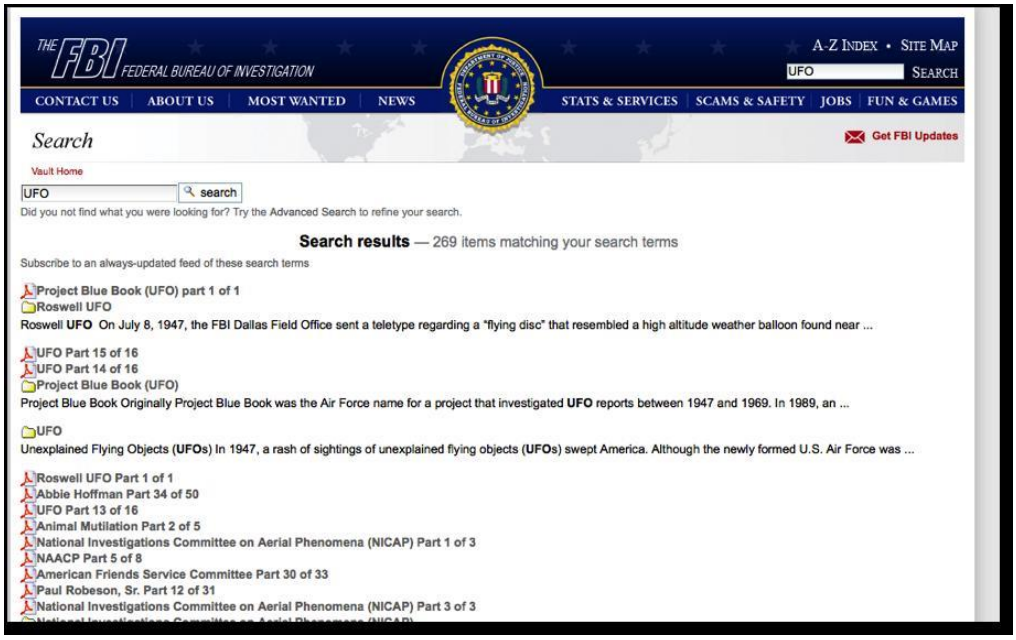
Content extraction and search using Apache Tika

Objective

In April 2011, the U.S. Federal Bureau of Investigation launched its *Vault* site (http://vault.fbi.gov). The site contains over 3000 declassified documents that have been scanned from paper and made available via a digital content management system.

One of the many documents on the FBI web site having to do with the subject of UFOs (Unidentified Flying Objects) We went to the search portion of the site and did a query for UFO:



269 hits come back from the search. Of the 3000 or more documents, 269 is not an insignificant number by any means (it amounts to about 9% of the total corpus), however, it seems like there would be a larger number of documents in the FBI's vault concerning the subject of UFOs. In this assignment, you are going to use Apache Tika to help us test our theory.

Task Description

The entire corpus of PDF files (13 GB) from the FBI's vault web site has already been collected and packaged into a tar file.
With the corpus of PDF files in hand, a Java program to be written as Hw3.javawill:

- Accept an input file containing any number of search keyword(s) and/or phrase(s)
- Iterate through the corpus of data on your local computer.
- Call Apache Tika to extract all of the text from each PDF file
- Scan the extracted text to search for the given search keyword(s)
- Count and output statistics about
    - the number of documents that contain the keywords
    - the total number occurrences of the keywords
    - list of files containing the keywords

The program analyze the results to help us decide whether or not the FBI's search is telling us the truth when it says only 9% of its vault corpus has to do with UFOs, or whether or not something more is hiding in that rich treasure trove of information.

Tika might not extract fully-formed text on each of the documents, depending on the quality of the OCR scanning that was used. To test the extracted text from each document a particular threshold by searching for the appearance of common words (usually called "stop words" in search engine terminology), such as "and", "the", "he", "she", etc. can be used.

Using a string edit-distance computation (such as Levensthein distance) to find words similar to UFO or saucer increase the possibility of detecting documents that truly contain information about UFOs.

Crawling the World Wide Web with Apache Nutch

Objective

We have to crawl the FBI's *Vault* site http://vault.fbi.gov/ and downloaded all of the relevant PDFs from that site, dropped them in a directory, and then packaged the whole directory into a tar file using Apache Nutch (http://nutch.apache.org/).

Getting all the PDF files would not be a snap as they were not named with URLs that ended with .pdf**.** And of course, provided that the PDFs were not all available from a similar directory structure, like http://vault.fbi.gov/pdfs/xxx.pdf.

Your goal is to download, install and leverage the Apache Nutch web crawling framework to obtain all of the PDFs from a subset of the FBI's *Vault* website that we have mirrored at http://fs.vsoeclassrooms.usc.edu/vault/, and then to package those PDFs and in order to create your own vault.tar.gzfile.

Extracting the PDFs from the Sequence File compressed format

After downloading and Installing Apache Nutch, configure Nutch to maintain Politeness. Nutch uses a compressed "SequenceFile" binary format to represent the data that it downloads. Data is stored in SequenceFiles which are part of "segments" on disk, some splittable portion of the crawl. Nutch performs this operation both for efficiency, but also to allow for distribution to allow multiple fetches to be distributed out on the cluster using the Apache Hadoop Map-Reduce Framework.

After successful crawling of FBI vault site you should have a crawl folder with a bunch of segments in it, and inside of those segments, SequenceFiles with the content stored inside.

Your goal in this portion of the assignment is to write a Java program that will extract the PDF files out of the SequenceFile format, and to write the PDFs to disk. Your program should be executed with arguments indicating the source (crawl data) directory and the output (PDF files) directory.
The result of this command will be to extract all PDF files from <crawl dir>, and to output them as named individual PDF files to the directory path identified by <output dir>.

Spatial Search using Apache Solr, SIS and Google Maps

Now it's time to look for another important and emerging property in the Search Engine research domain: spatial search and visualization. Wouldn't it be cool to visualize and determine where those PDF documents that you downloaded, and decimated, were located across the United States? Do you think that UFO related documents typically center on airports? Do you think that the JFK documents are centered on Texas, where he was shot and where he died? What about the documents that the FBI collected on the Branch Davidians and Waco, TX? Where were those located? Do you think that spatial location of the PDF data will generally make sense and coincide with the location of the event? Time to find out.

In this part, Goal is to:

Set up the Apache Solr search engine technology, which, along with what you learned in Homework 3, will allow you to load and tag your PDFs from vault.tar.gz.

Leverage the Geonames.org dataset, http://geonames.org/ which maps names and geographic locations to latitudes and longitudes, in order to geo-tag each of the PDF files with their approximate location of interest.

Load the geo-tagged data into Solr, making it available for spatial search.

Develop a web page that enables searching the PDF files for conspiracies. Your web page will need to integrate Google Maps to plot locations of the search results on a map.

Dump your Solr data via its GeoRSS response writer into an emerging Apache project called Apache SIS. SIS provides a quadtree index and web service to perform point-radius and bounding box searches of associated GeoRSS data. It also easily plugs into Google Maps, thus letting you visualize the results of your queries.

Implementation

For each conspiracy that you choose, come up with a list of search terms that you could use to find documents related to that conspiracy.

Geo-tagging the PDF documents in Solr

Next, we'll want to geo-tag all of the PDF documents from vault.tar.gz with a geographic location. To accomplish this, we're going to use Tika to grab out the most frequently occurring terms in the document. In a similar fashion to Homework 3, your job is to write a program that will use Tika to extract the text from the PDF document, and compute the list of unique words in the document, and sort them by their frequency of occurrence, from highest to lowest.

Armed with this list, we'll now try to use the geonames.org dataset to geo-locate the document based on the highest occurring term, as described in https://issues.apache.org/jira/browse/SOLR-2073. If geonames.org can't locate the term, move onto the next term. Repeat this process until you have a latitude and longitude for the document. Once you have the latitude and longitude, index the document in Solr, using one of the many available Solr client APIs.

Build a web page for searching PDF documents and visualizing the results on a map

Create a web page that provides a simple user interface for querying your Solr index. Your interface should accept multiple search terms, and be able to plot all documents in the search results on a map based on the geographic location that they were tagged with earlier, using the Google Maps API:

http://code.google.com/apis/maps/index.html

Make use of your search engine to query for each one of your three conspiracies, using the search terms you came up with earlier. For each conspiracy, you should either generate a map with markers for each document, or produce a heat map (akin to Nathan Yau's map) based on the location of the documents.

Figure out how to dump the Solr metadata into the Apache Spatial Information System (SIS) technology, which you can find here: http://incubator.apache.org/sis/. See the SIS README file here:

http://svn.apache.org/repos/asf/incubator/sis/tags/0.1-incubating/README.txt

Install the GeoRSS response writer and plug Solr into SIS. Once you have GeoRSS from Solr loaded into your SIS Location Service, perform various queries (bounding box and point/radius) and play around with the data. Look for some trends or patterns in the data set, and make a list of 3 such discoveries you make from exploring your data spatially with SIS. Be prepared to demonstrate with examples when presenting your assignment for grading.