ISTA 456 Programming Project

**Custom Information Retrieval**

Homero Pawlowski

**Goal**

The goal of the project was to automate the retrieval and indexing of documents, pertaining to creative writing, for a custom web application. A web-based query tool would need to be created to allow users to easily search for indexed documents.

Currently the process for retrieving these documents is 100% manual. The owner of the website (http://www.michikokatsu.com) searches through manually selected websites, finds articles that are relevant, and then manually blogs or posts about them on their website. We wanted to expand this so that we could be more of a source and allow users to visit the website, search through articles, and let them decide which are relevant to their needs. The goal was to automate this whole process so minimal manual work needs to be performed.

**Research**

The first step of this project was to identify how we could automate retrieval of documents from these selected websites so that we could then index them.

I researched and spent a lot of time playing with Apache Nutch to do just this. Nutch was used to crawl a list of web domains and retrieve each page found on the domain as a document. This turned out to not be very effective for several reasons:

1. A lot of the pages retrieved from these domains weren't useful, Nutch was retrieving, home pages, contact pages, information pages, and in general lots of pages that couldn't be necessarily called creative writing documents.

2. Another issue was time, for each page that was to be indexed as a document, a HTTP request had to be made and this quickly added up to hundreds, then thousands, of potential requests for even a very small number of unique domains.

3. Once these documents were indexed, there was no way of updating them without running into the time issue once again.

4. Saving all these documents took up a lot more space then planned.

Looking for a better solution, I turned to looking at the RSS feeds for these websites, which turned out to be a much better idea. I decided to bypass Nutch completely and create a seed text file that contained one RSS feed URL per line. It was easy to create a BASH script that uses wget to download the XML file for each RSS feed, as well as timestamp the retrieved files. This saves a lot of HTTP requests (only one per domain now), space, and time.

Using Apache Solr, the XML files are then indexed and preprocessed with text analyzers that perform the tokenization and stemming of the documents. Each item in the RSS feed is treated as a separate document. Through a combination of timestamping the downloaded files and clever configuration of Solr, documents that are already indexed are updated on each subsequent Solr import. Documents that have not been seen before are added to the index. Documents are uniquely identified by the URL at which they are found.

Solr also acts a web server that serves as the backend for the web requests that the online search tool sends out. A custom HTML demo page was created to show that allowing users to create queries to search through indexed documents is easy. Solr also takes care of ranking the documents and all results returned are ranked using a vector space model.

**Command lines needed to run**

1. List the URLs of any RSS feed we want to monitor into seeds.txt

2. Setup a cron job to run bin/retrieve on a interval of choice, or manually run the bash script.

3. After the cron job is finished we need to index and import the data into Solr. Visit the URL http://localhost:8983/solr/dataimport?command=full-import&clean=false

4. Start the Apache server and visit the provided index.html

5. Using the search bar on index.html, the indexed documents can be searched through. The search uses the Extended DisMax query parser to search through the documents and the shown results are ranked in order of relevance.

**Description of code**

index.html – A demo web page that shows the capability to create a web front-end to query and retrieve documents that are indexed.

solr-4.10.2/example/solr/collection1/conf/schema.xml – The file that defines the schema that Solr uses to index the imported data. The query and index analyzers are defined within this file and were customized to this project.

solr-4.10.2/example/solr/collection1/conf/data-config.xml – This file defines the structure of our imported XML files. It acts as a data-import handler for Solr. It is customized to the structure of the retrieved RSS feeds. It is also customized to import all files in our download folder and to only index/import files that have not been already imported.

solr-4.10.2/example/solr/collection1/conf/solrconfig.xml – Settings to setup the data-import handler that ultimately imports the XML files.

seeds.txt – A custom file that designates what RSS feeds to import. Each line is a URL to a RSS feed. Commented lines designated with < – – > are ignored.

bin/retrieve – A custom bash script that looks at seeds.txt, uses wget to download all the URLs listed in it, and then dumps a separate XML file for each URL into the download folder. The files downloaded are timestamped by when they were downloaded, and not by when the files were last modified on the originating server.

**Results**

The results for this project are great. It will be very easy to implement this system on a VPS and into the existing Wordpress site that the original customer uses. Users will be able to visit the wordpress site and search though a large repository of information. Setting up a cron job on the VPS to run the retrieve bash script every night will be practical and will allow the archive of documents to consistently grow larger.

Just setting up efficient retrieval and indexing of documents from custom sources proved to be a challenge. This was thought to be initially trivial prior to beginning the project but ended up taking a large portion of time. Part of the reason was the unfamiliarity with software projects such as Apache Nutch and Apache Solr. Without ever using either product and not knowing their full capabilities, a lot of research had to be done before an efficient solution to retrieving and indexing could be engineered. Now that this step is done though, work can be focused on implementing algorithms to comb through the documents and present the information in a more useful manner to end-users, such as categorizing documents.

**Future work**

Though this project was completed for the ISTA456 final, it is also a ongoing project. The next steps to this project are:

1. Implement the web querying it into an existing Wordpress website.

2. Implement clustering algorithms to categorize documents into predefined fields such as “creative writing”, “fiction”, “non-fiction”, “drama”, etc.

3. Save the indexed documents into a database that will allow us to add functionality for users of the website to rank documents, mark them as relevant or unrelevant, and for us to monitor how popular documents are by clicks.