**Team #32**

Sanjay Jagadeesh

Manohar Thagadur Nataraju

Soumya Gowda Basvana

Pramod P. Setlur

CSCI–572 Assignment 2

*An Apache-Solr based Search Engine, Ranking Algorithms and NER for Weapons Datasets*

We indexed all the data of our first assignment into Apache-Solr. We implemented *Content-Based and Link-Based* for retrieval of results from the index.

CSCI–572 Assignment 2

## Question 1: Develop an indexing system using Apache Solr and its ExtractingRequestHandler (“SolrCell”)

1. We installed Apache Solr from Lecene’s 4.10 branch.
2. N/A
3. We built Geotopic parser, OCR and ctakes as per the instructions provided to us.
4. We first executed bin/nutch dump to create the dump files. We then use this dump file in *dump\_index.py* to index the data to Solr.
5. N/A

## Question 2: Leverage the Nutch indexing system to build up an Apache Solr Index

1. The metadata extracted from NUTCH using TIKA included the following: id (unique id), title (name of document), segment (which segment it was uploaded from), boost (value to determine relevancy), digest, tstamp, type (metadata about file), date, contentLength, url and version.

The metadata extracted from SOLRCELL using TIKA included additional metadata along with the metadata that have been discussed above. This included stream source info, stream content type, stream size, content encoding, stream name and content type. Also additional HTML tags, Images attributes were also generated.

Overall, the metadata generated by SOLRCELL was more detailed and descriptive with regard to using this for the page ranking algorithms. Also there was more flexibility for user defined indexes which was not present using NUTCH Solrindex.

## Question 3: Design and implement two ranking algorithms for your Weapons data documents

1. The *content based* algorithm that we implemented was one similar to the algorithm employed by google. The algorithm uses a combination of 2 metrics: “tf.idf” (term frequency and inverse document frequency) and cosine similarity.

Term frequency of a word given by tf(word) in a document is simply the number of times the word appears in the document.

Inverse document frequency of a word given by IDF (word) is given by :

IDF (word) = Total Number of Documents / Number of documents containing the given word

We use the tf.idf metric to represent each document in our index and the query itself as vectors. We use the cosine similarity metric to determine how similar 2 vectors are to each other.

In this case, the 2 vectors that we compare are the query vector and a document vector. The document whose corresponding document vector resulted in the highest cosine similarity when computed with respect to the query vector is considered most relevant.

1. The *Link based* algorithm

## Question 4: Develop a suite of queries that demonstrate answers to the relevant weapons related questions below.

1. Test
2. Test
3. Test
4. Test
5. Test

Question 5: Develop a program in Python that runs your queries against your Solr index and outputs the results in an easy to read list of results demonstrating your relevancy algorithms and answers to your challenge questions from Task #4.

We implemented a python program for the above.

EXTRA CREDITS

Question 6: Develop a Lucene-latent Dirichlet allocation (LDA) technique for topic modeling on your index and use it to rank and return documents.

Question 7: Figure out how to integrate your relevancy algorithms into Nutch

We went through the documentation of Similarity Scoring Filter and understood the

Question 8: Create a D3-based visualization of your link-based relevancy. Provide a capability to generate D3 relevancy visualizations as a Nutch REST service using Apache CXF. Integrate the service into nutch-python.

1. We went over the documentation and wrote a python program to crawl with Nutch using Nutch rest server and Nutch python.
2. The python program *Nutch\_python\_crawl.py* is attached.
3. We went through the documentation and implemented the best practices for crawling using Nutch rest server.
4. NutchPy is a python library for working with Apache Nutch. It has functionality to work with Nutch data structures – to read Sequence files, LinkDb, etc. We feel these functionalities can be added inside the *Nutch REST server* as an API. It can accept a link to the the segments generated and further processing of can be performed by the *Nutch REST server.*

CONCLUSION

The goal of the assignment was to collect as many unique images of guns and weapons on the provided seed lists. We could get around 107.000 images.

We faced multiple challenges throughout this project. Here are the major challenges:

* Developing an interactive selenium to interact with dynamic Ajax based content which enabled us to get into deep web and collect more images of weapons. We had to downgrade the Firefox version to 33.0 to successfully enable the protocol-interactive-selenium plugin.
* Identifying the duplicate images was one another major challenge. The project led us to a thought process as to why the images were duplicated. Most of the times it happens that there may exist the same content in different regions (example, we may have same page in .com domain, .[co.uk](http://co.uk/), .[co.in](http://co.in/), etc.) which may result in duplication of web content. We developed algorithms to identify exact and near duplicates. The challenge was in extracting the metadata for the images based on which they could be categorized into exact or near duplicate images. In addition, the duplicated URLS were filtered in the crawling phase itself.
* Initially we faced one of the challenge in increasing the nutch heap memory size where in the crawling aborted after running for 2 days due to insufficient heap memory. We figured out the configuration parameter JAVA\_HEAP\_MAX in bin/nutch script to increase the size of the heap memory.
* Configuring the nutch crawler to deal with politeness and completeness. This enabled the crawler to increase its sensitivity to the way it appeared in web logs for servers providing weapons oriented content on the Internet.
* Getting to work with the various open source technologies. Most of the times we couldn't get much documentation in installing or integrating the technologies with apache nutch. This led us to actively engage in the discussions in apache nutch forums, post our queries in the forums and to even contribute back to the forum by answering other's queries.