# CSCI 572 Fall 2015 - Team 36 - Assignment 2 Report

**Thammegowda, Nii, Rakshith and Rahul**

## 1. Nutch index step

**a.** We used Apache Solr(V 4.10) throughout this assignment. We also enabed **indexer-solr, index-(basic|more|anchor|geoip|static|metadata)** plugins.

**d.** Index via SolrCell and the issues we faced:

We dumped the crawl content via '`bin/nutch dump`' and wrote a python script (`dump-poster/dump_post.py`) to POST the files to solrCell handler. However, the following issues were encountered.

- Most of the metadata fields were ignored by SolrCell.
- The 'url' of document was lost after duping nutch content to file system.
- We couldn't control field mappings, based on value types.

So, we wrote a program to read nutch content, reparse using Tika at client side and then post to Solr. We enhanced Solr schema by defining a dynamic fields for metadata. All the metadata fields end with '*_md*' suffix. In addition, metadata were mapped to field based on value data types, For example, integers, text_general and strings were mapped to following fields:

| Value type | Field suffix |
|---|---|
| Single Integer value field | _i_md |
| Multivalued integer field | _is_md |
| Single text value | _t_md |
| Multivalued text field | _ts_md |

**NOTE:** Please check schema.xml for full list of fields.

## 2. Index comparison

**a.** The latest version of Tika was built from trunk and GeoTopic parser, OCR and cTAKES were configured.

**b.** Metadata comparison

i. Fields obtained from Nutch index

| | | |
|---|---|---|
| content | boost | contentLength |
| Title | Digest | Date |
| Host | Tstamp | lastModified |
| Segment | Id | |

ii. Fields obtained by reparsing using Tika

| | | | |
|---|---|---|---|
| Id | Host | contentType | dataprecision |
| Content | x-parsed-by | tiff:bitspersample | component2 |
| tiff:imagewidth | tiff:imagelength | numberofcomponents | resolutionunits |
| imagewidth | imageheight | filesize | jpegcomment |
| filename | filemodifieddate | compressiontype | component3 |
| xresolution | yresolution | w:comments | comment |
| software | dates | locations | persons |
| organizations | weaponnames | weapontypes | Geographic_LATITUDE |
| Geographic_NAME | Geographic_LONGITUDE | Optional_LATITUDE1 | Optional_LONGITUDE1 |

This parsing was performed in two iterations.

**i.** Default parser with Tesseract OCR and cTAKES enabled.

**ii.** Parse text content with GeoTopicParser, NamedEntityParser(people, location, dates, organizations), RegexNER(Weapons and Weapon Types).

## 3. Ranking Algorithms

**a.** Content Based Algorithm

Content detection and analysis:

- We configured tokenizer to remove stopwords. We followed instructions given in solr wiki.
- We carefully mapped metadata fields to appropriate schema field types to ensure that proper lucene analyzers are applied to the content. We wrote a java program to automatically map value to proper types (The program is located in edu.usc.cs.ir.cwork.solr.schema.FieldMapper class in the project).
- We used Stanford CoreNLP's CRF classifier with 7 class model to detect named entities from text content. Using this we are able to detect: Location, Person, Organization, Money, Percent, Date, and Time. All these values are also copied to "text" field.
- We used Natty Date Parser to parse dates and then indexed as date field in solr.
- We scraped wikipedia pages for weapon names and created a regex for extracting weapon names in text. (The regex can be found in src/main/resources/org/apache/tika/parser/ner/regex/ner-regex.txt in the project).
- We found synonyms and related utterances of weapons, then created another regex for detecting them in text. This is also included in previously mentioned regex file.
- The content relevancy score of documents to the input query can be retrieved from Solr by requesting `fl*,score` query parameter.

Explanation of content based relevancy:

In this assignment we came to know that the Lucene library internally works on top of Vector Space Model with TFIDF weight scheme. This has been explained in lucene docs here. The stopwords were removed at the index time since they are ubiquitous and do not contribute any meaning to determine relevance. By applying proper analyzers to content, we ensured correct tokenization and vectorization. We recognized various entities like people, locations, weapon names, weapon types in the input and indexed them in separate fields. This allowed us to do content match on those fields. The queries are also vectored, similar to the documents during index time, and then cosine similarity between documents and the query is computed to determine the relevance.

**b.** Link Based Algorithm

For computing the page ranks of documents, we built different graphs based on location, dates and organizations. Each graph is computed by using Solr join query to get all the all the connected documents which have same values in the field of interest.

This is spread across three steps:

**i.** Generate Graph
**ii.** Compute Page Rank
**iii.** Update Ranks

These three steps are performed for graphs of locations and dates.

**i. Generate Graph:**

The implementation is in edu.usc.cs.ir.cwork.relevance.GraphGenerator. Here is the pseudo code:

```
for each document D in the index:
   S = documents that have same value as of D in field F
   //S = ALLDOCS(?q={!join from=F to=F}+id:D.id)
   for each document D2 in S:
         output edge between D.id and D2.id
```

This step produced a text file, say "EdgesFile" containing all edges in graph.

**ii. Compute Page Ranks**

The implementation is in the edu.usc.cs.ir.cwork.relevance.SparkPageRanker class. Here is the pseudo code:

```
d = 0.85            // damping factor
nIterations = 5     // num iterations
G =   READ:EdgesFile.Stream(LINE)
-> MAP : (V1, V2)
-> GROUP : groupByKey(V1)
-> cache()
ranks = G.mapValues(1.0)              // initial ranks for all
vertices
for 1 to nIterations:
   JOIN : (G + ranks)         // join edges and pagerank scores
   -> REDUCE : count the edges of V
   -> REDUCE : CONTRIB = sum up the inlinks pagerank contributions
   -> MAP : ranks = (1-d) + d * CONTRIB      //new pageranks
// in the end
ranks -> MAP : ("V \t SCORE")
-> WRITE : ranks.asText())
```

This step produced "PageRanks" file

**iii. Update Ranks**

This step performed solr atom updates on documents with the scores obtained in previous steps. The implementation is in edu.usc.cs.ir.cwork.solr.SolrPageRankUpdater class.

## 4. Challenge Questions and Queries

**a.** Time-based trends in Gun ads.

i. Facet on 'dates' field showed that

- The first spike is on Sept. 11, 2001. We correlate this to September 11 attack.


```
"2001-09-06T00:00:00Z": 1,
"2001-09-09T00:00:00Z": 1,
"2001-09-10T00:00:00Z": 1,
"2001-09-11T00:00:00Z": 16,
"2001-09-12T00:00:00Z": 1,
```


- Another interesting spike is on 07/1/2015. We correlate this response of people to Boko Haram extremists' attack that is reportedly happened on the same day.


```
"2015-06-30T00:00:00Z": 729,
"2015-07-01T00:00:00Z": 27604,
"2015-07-02T00:00:00Z": 1091,
```


ii. Location based trend.
We were able to perform facet search on locations and we found that the top three states with highest number of gun ads are North Carolina (17728), Texas (13327) and Virginia (10331).

iii. By applying location filter to Texas and then performing faced search, we found a spike on 6/20/2009.


```
"2009-06-16T00:00:00Z": 2,
"2009-06-17T00:00:00Z": 5,
"2009-06-19T00:00:00Z": 2,
"2009-06-20T00:00:00Z": 53,
"2009-06-23T00:00:00Z": 4,
"2009-06-24T00:00:00Z": 2,
```


iv. We found that guns were illegal to purchase in the state if Illinois prior to Dec, 2012. We found ads related Illinois posted prior 2012. Hence, we may conclude that there might have been unauthorized purchase of firearms.

**b.** Similarity in firearms image types.
We are able to determine weapon types and weapon model names. We used regex NER to find this information from page text and page URL. For example, we did a query to know top traded weapons in Anacortes on 07/01/2015 and found the following results:


```
  "C9",
3270,
  "Bor",
1533,
  "Beretta 92"
1087,
  "AKM",
1082,
```


It does indicate some event happened in Anacortes. We think it might be correlated to Independence Day sale from Anacortes Gun Shop. We found that "O.D. Green" as a top user name with 853 associated ads. We also used 'More Like This (MLT)' handler to discover results similar to the given results. We used MLT handler parameters to look inside location, weapontypes, weaponnames fields for performing the match.

**c.** Correlation between stolen bulk shipment and spike in sales.
We analyzed ads trend for weapon named "Beretta 92". We found the following influxes.

i. On June 16, 2010


```
"2010-06-05T00:00:00Z": 1,
"2010-06-11T00:00:00Z": 1,
"2010-06-15T00:00:00Z": 88,
"2010-06-16T00:00:00Z": 1004,
"2010-06-17T00:00:00Z": 1,
```


ii. On July 26, 2011, there's another influx.


```
"2011-07-12T00:00:00Z": 4,
"2011-07-15T00:00:00Z": 1,
"2011-07-26T00:00:00Z": 457,
"2011-07-30T00:00:00Z": 1,
"2011-08-01T00:00:00Z": 7,
```


iii. The top locations for the above influx are as follows


```
  "California",
1958,
  "Anacortes",
1101,
  "Dayton OH",
667,
```


May be a shipment of "Beretta 92" weapons were stolen. We could find some stories of UPS driver stealing gun shipment on top news websites.

**d.** Ads by underage persons.
We are able to detect names of people who posted the ads using Stanford CoreNLp NER. Using a regular expression we wrote, we are able to extract and index phone numbers. We tried to determine people age via gun registry. We also tried to reverse lookup on phone numbers to determine age of person who owns it. Both of these attempts failed because we could not find a publicly available dataset for lookup. The tesseract OCR failed to recognize Serial Numbers of guns in images.

**e.** Ads related to weapons of mass destruction.
We extracted weapon types and performed a faceted search on it. We found that the weapon type "bomb" existed in our faceted search with 1741 ads. We also found many other weapon types which can be referred to in the ***Queries.pdf*** file. The top 4 weapon types (gun, ammo, firearm and rifle) have been included for scale

To see the result on the browser, open **Queries.pdf** and click the corresponding queries.
**Note:** Make sure Solr is running before executing the queries.

## 5. Query Program

The query program is a Python script that takes a text file as an input with queries separated line by line. For example,

*query?q=*:*&facet=true&facet.field=weapontypes&rows=0*

The program allows you to pass the solr host url, and collection name as command line arguments. For each query, it outputs the JSON results of the query to stdout.

Example usage: python query-runner.py -u http://localhost:8983/solr -c collection2 -f queries.txt

## 6. Lucene-latent Dirichlet allocation

We used the https://github.com/chrismattmann/lucene-lda/ repository

Following files were used to configure the lucene lda

- files.dat
- vocab.dat
- words.dat
- theta.dat

We first used the following commands

```
bin/indexDerectory t/t001/code index1 ldahelper.obj --ldaConfig
64,t/t001/lda/64
```

This command generates us the index directory

```
bin/queryWithLDA index1/ t/t001/lda/ldaHelper.obj t/t001/bugs
latestresults/ --scoringCode 1
```

This command generated query results

Re-run your queries and examine the results from Task #4. What differences do you see?

- We saw that rankings assigned to the documents changed.
- Documents now were labeled and we were now able to generate rankings based on these labels.

Can you explain them?

- This technique could be used to generate a custom vocabulary which could be used to generate topics related to labels of interest e.g if we want to give higher index to documents that mention more content related to a particular type of gun like Shotgun.
- Advanced text modelling can be used to generate a sample vocab.dat which understands the words commonly used by teenagers. Such a technique can be used to find which documents were written by teenager who are not allowed to trade guns by law.

## 7. Named Entity Parser

We developed Named Entity Parser and supplied the following three different implementation of Named Entity Recognizers.

i. Open NLP NER.
ii. Stanford Core NLP NER.
iii. RegEx NER.

Please visit the URL https://github.com/apache/tika/pull/61 for the pull request.

## 8. D3 Visualization

We could able to generate D3 visualization of our link-based relevancy based on page rank and the URL. We wrote a python script to which given a page rank file of the form `<URL> <Page rank>` it converts it to a set of url/id page rank pairs to a JSON file consisting of this structure

{ "d3" : [ {"url" : <URL>, "pr" : <PAGE_RANK>] }

Since we have lots of URLs D3.js failed to render for the entire set. Hence, we reduced the set and generated the visualization. You can find the python script and visualization under *project/d3 folder*.

## 9. Conclusion

Link based relevancy algorithm are better for finding the documents which have higher static value like popularity score. While content based algorithms are helpful for finding best match results for the query. The queries related to filtering of documents with respect to geographical area and time-window showed better results with the content based algorithm. However, queries related to finding stats like most popular documents showed better results with linked based algorithm. Nutch/Tika + SolrIndexing was easier compared to SolrCell. Finally, we would like to contribute our indexed dataset to Memex project.