CSCI 572: Assignment II Report
Building an Apache-Solr / ElasticSearch based Search Engine, Ranking Algorithms and NER for Weapons Datasets

Team number: 33
Team members: Gaurav Shenoy, Mahesh Kumar Lunawat, Pramod Nagarajarao, Presha Thakkar, Karthik Manipur Kini

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

The following steps explains the installation of Solr and the necessary plugins for tika-parser to build the Inverted Index:

- Apache Solr was built from the 4_10 branch of Lucene-Solr to take advantage of integration of Tika with SolarCell and OCR. Web application server - Jetty that comes with Apache Solr is being used in this assignment.
- Tika-1.11-SNAPSHOT version is downloaded and “maven install” command is run.
- GeoTopicParser was successfully installed. The GeoTopicParser combines a Gazetteer and a Named Entity Recognition (NER) modeling technique that identifies names and places in text to provide a way to geotag documents and text.
- Installed OCR which is used to capture the text present in an image.
- Installed cTakes successfully. Tika now has the ability to leverage cTakes for use in parsing biomedical information and dates from text.
- At this step Tika 1.11 installed has the GeoTopicParser, OCR and cTakes capabilities. A python program second_index.py is used to index the metadata given out by Tika parser including geotags/locations, text present in images and dates present in the metadata. Refer

The crawled data is present in the “crawl” directory of nutch runtime containing the crawldb, linkdb and segments directory. Following steps are carried out to build the index from nutch:

- To index all the incoming anchor text with pages we run “invertlinks” command as given below:
  ```
  bin/nutch invertlinks crawl/linkdb -dir crawl/segments
  ```
- Execute the following command to index:
  ```
  bin/nutch solrindex http://localhost:8983/solr crawl/crawldb/ -linkdb crawl/linkdb/
  crawl/segments/ -filter -normalize
  ```

The dump of the segments directory present in nutch runtime is taken and an Index will be built as explained in the following steps:

- bin/nutch dump -outputDir images -segment crawl/segments/ command is run to take dump of segments directory containing images and text content.
- A python program second_index.py has been written to post metadata of dumped files and extra metadata obtained from GeoTopic Parser, cTakes Parser, tika OCR and tessaract to Solr.

Task 2: Leverage the Nutch indexing system to build up an Apache Solr index or and ElasticSearch index

Upgrading Tika instance with the content parser support such as GeoTopicParser, OCR and cTakes has been carried out as explained in Task 1.

Comparison of metadata:

1. Index obtained by running bin/nutch solrindex command on the crawl directory is the first set of Index obtained. The metadata fields obtained are description, author, keywords, title, resource name, content type, subject, URL and links.

2. Running the dump command on the segments directory, we get images and HTML content. A python program "second_index.py" has been written which executes curl command by running respective server for Geotopic Parser, cTakes Parser and Tesseract OCR. It includes Geographic_name, Geographic_Latitude, Geographic_Longitude, text extracted from images using tesseract OCR parser and published date, last modified date and time from dumped content as part of its metadata. The metadata obtained is indexed using Solr. In addition to the metadata fields obtained in the earlier index, we obtained following relevant information such as Name, Address, Phone Number, item model type for guns, Publish Date for seller/Buyer/Manufacturer.

Overall we found that the second index is better than the first as we get more number of metadata fields and the the ability to control certain fields.

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

a. Describe content based algorithm

The content based algorithm that we implemented uses a combination of 2 metrics: TFIDF (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:

$$idf(word) = Total\ number\ of\ documents / Number\ of\ documents\ containing\ the\ given\ word\ given\ word$$

We use the TFIDF metric to represent each document in our index and the query itself as vectors. We use the cosine similarity metric to determine similarity between two vectors. In this case, the two 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.

Algorithm:

1. Given a query q. We represent it as a vector with each dimension of the vector corresponding to the product of the normalized term frequency of each word in the query and its inverse document frequency (IDF) over the entire set of indexed documents. Note: The IDF of a word is a constant for a set of documents that we have indexed.
2. For each indexed document d, we create a vector as we did for the query.

3. Let the query vector be q and $document i$'s vector be di.We then compute the cosine similarity between q and di as:
   Cosine Similarity (q, di) = $q.di$ / ||q|| ||di||
4. We calculate the cosine similarity between a given query and all documents that we have indexed. We then return the documents in decreasing order of cosine similarity.

- **Input to the algorithm** is the Query String and a file containing the contents of all the indexed documents separated by new lines.
- **Output of the algorithm** is the pages in decreasing order of relevancy score computed.
- When the algorithm is run, the score calculated by the algorithm is calculated. The files included are in the folder "ContentBasedAlgorithm". This includes ContentBasedAnalysis.java which is the main class incorporating the algorithm.

Later in the assignment we figured out that the DefaultSimilarity scoring class of Lucene has TFIDF algorithm implemented and uses it internally while indexing.
To view the TFIDF score while querying Solr, follow the below steps:
- Add the below code in "schema.xml" in installed Solr directory.
  <similarity class="org.apache.lucene.search.similarities.DefaultSimilarity"/>
- Add the following two parameters in the query.
  fl=score
  debugQuery=true

b. Describe link based algorithm.

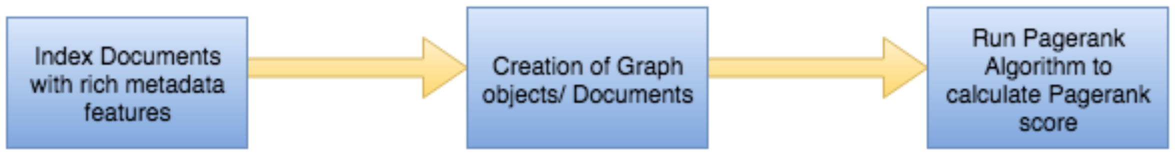
Keeping the given queries in mind, we started analysing the metadata associated with the initial dataset. The best features among these were chosen in creating a link-based graph. The features chosen are
- Location categorized as state Ex: California, Oregon, Washington etc.
- Gun Type Ex: shotgun, rifles, pistols etc
- Make/model Ex: AK47 Rifle, Hot Purple Nano etc
- Manufacturer Ex: Bushmaster, Daniel Defense, Stag arms etc.
- Keywords Ex: 8mm Remington magnum, 300 Blackout (7.62x35mm), Double Action etc.

A Hash-Map is created with features as keys and documents as values. Each Key in the Hash-Map has list of documents associated with it as values. We created graphs for each of the features separately. The nodes of this graph are the documents. The bidirectional graph created has an edge from one node to another if there are common features shared between them. Pagerank algorithm is ran on top of the graph created. The output of the algorithm would be a JSON file containing various objects where key would be "document" and value would be "pagerank score". Each of the obtained pagerank scores are appended to the Solr Index. We boost the particular fields based on the obtained pagerank scores for a given query.

Pagerank calculation:
Pagerank calculation:
If pages $T1$,$T2$,$T3$ .... Tn are linked to page A, pagerank of A PR(A) is given by:

$PR(A) = PR(T1)/C\ (T1) + PR(T2)/C\ (T2) + PR(T3)/C\ (T3) + \cdots + PR(Tn)/C\ (Tn)$
Where $C(Ti)$ is the number of links emanating from Ti.

As we can see from the equation, calculating the pagerank of a page involves finding the page ranks of all the pages linking to it.


Index Documents with rich metadata features
Creation of Graph objects/ Documents
Run Pagerank Algorithm to calculate Pagerank score
Fig1: Flow diagram of the linked based algorithm

The process of calculating page ranks is hence an iterative process. We assume an initial set of page ranks for all the pages. We then calculate the pagerank of each page as per the above formula, continuously using the page ranks just derived. We stop when two consecutive iterations throw up the same set of page ranks for all the concerned pages.

Add the following code to the “schema.xml” in the core of Solr:

```
<field name="pgrk_guntype" type="float" indexed="true" stored="true" multiValued="true"/>
<field name="pgrk_keywords" type="float" indexed="true" stored="true" multiValued="true"/>
<field name="pgrk_locations" type="float" indexed="true" stored="true" multiValued="true"/>
 <field name="pgrk_manufacturer" type="float" indexed="true" stored="true" multiValued="true"/>
```

The folder “LinkBasedAlgorithm/Graph generation and Page Rank” contains all the code related the implementation. Folder “scripts” contains the code “guntype.py”, “keywords.py”, “locations.py” and “manufacturer.py” which implements PageRank algorithm. Folder “resultJSONs” contains the results. The Python script “indexDataTrial2.py” present in “LinkBasedAlgorithm” folder helps us in integrating pagerank Score with Solr Index.

Task 4: Develop a suite of queries that demonstrate answers to the relevant weapons related questions

Content-Based Queries:

A. What time-based trends exist in Gun ads? Can you correlate temporal and spatial properties with buyers? For example, can you identify based on ad time-window and/or based on geospatial area places where people try and purchase guns on behalf of someone unauthorized to purchase them?

Each query given in the question has a suite of queries associated with it. We present two python programs - “WriteFile.py” and “WriteFileByDate.py” which both helps in achieving the task. The python scripts generate the queries and execute it. The corresponding output files and queries generated are placed inside the “Queries: 1” folder. The following are the set of inferences we found by running the query.

- On year-month 2015-05-24, in the state of Connecticut, Delaware, Hawaii, Idaho and Utah there were same number of ad sales for Rifles and shotguns. The number of undefined weapons ad sales were significantly large compared to others.
- There were no rifles and shotguns sold in the state of Maine on 2015-05-10, 2015-05-12, 2015-05-13, 2015-05-14, 2015-05-17, 2015-05-19, 2015-05-22, 2015-05-24, 2015-05-25 and 2015-05-27.

B. Can you identify similar firearms image types (e.g., shotguns) that are sold in the same region and time? Does this indicate influx related to stolen goods?

We present two python programs - “WriteFileByDate.py” to do the task. The python scripts generate the queries and execute it. The corresponding output files and queries generated are placed inside the “Queries : 1/ Queries : 2” folder. The following are the set of inferences we found by running the query.

- On 2015-05-30 same number of shotguns were sold in the state of Kansas and Kentucky.
- The Seller’s name and description are obtained in the execution of the query. The “Description” field is the manufacturer’s name. Though we have the data, we can’t entirely be sure that if an ad query result which doesn’t have the manufacturer’s name can be classified as stolen goods.

C. When a shipment of bulk firearms is stolen, the rate of ads and images may indicate an increase in sales of that particular make/model – can you identify these?

We present a python program - “queryGenerator2.py” to do the task. The python scripts generate the queries and execute it. The corresponding output files and queries generated are placed inside the “Queries: 3” folder. The following are the set of inferences we found by running the query.

- The seller by Name “Darin”, who works for “The Armory Company” bearing the phone number “+1-9122851288” sells many different type guns through ads having almost the same “Maintained Conditions”.
- Some of the make/model of the guns sold by “Darin” are “'M&P15-22'”, “G30gen4” etc.
- Some Geo-locations like “Texas” has a surge in the number of the arms being sold/bought during certain period of time (In span of days-months).
- The seller by Name “Joanna”, who works for manufacturer “Plano Pawn Shop” being the phone number “+1-9124246911” sells different models with guns conditions.

D. Can you identify ads and/or weapons images that are posted by persons, whom are underage or in which the weapons are de-identified (by type and/or serial number, etc.)?

We present a python program - “queryGenerator3.py” to do the task. The python scripts generate the queries and execute it. The corresponding output files and queries generated are placed inside the “Queries: 4” folder. The following are the set of inferences we found by running the query.

- The Seller with Name “Steve” works for two companies or associated with two manufacturers - “Texas Jewelry Pawn” and “B & S Guns”. In the State of Texas, there are 47 manufacturers that sell guns or subscribed to gun related ads.
- But, underage and unauthorized or weapon being de-identified might not be deducted with the information we have. The Index constructed by our team doesn’t posses the necessary fields to come up

E. Can you identify ads and/or images that relate to the unlawful transfer, sale, and possession of explosives, WMD devices, and precursors?

We present a python program - “qqueryGenerator4.py” to do the task. The python scripts generate the queries and execute it. The corresponding output files and queries generated are placed inside the “Queries: 5” folder. The following are the set of inferences we found by running the query.

- This query is one of the most interesting query we came across. When we queried “Chemicals, Nuclear, WMD, Missile etc.”, we obtained results from the index.
- We went one step ahead of re-checking the URL’s obtained. The following URL’s are some of them.

Missile Examples:

- http://www.armslist.com/posts/4067140/washington-nfa-firearms-for-sale--fs--gulf-war-souvenir-scud-missile-

- http://www.armslist.com/posts/4330798/new-hampshire-misc-for-sale--military-warhead-shipping-container
- http://www.armslist.com/posts/4354719/hampton-roads-virginia-rifles-for-sale--adcore-a-556-elite-16--piston-ar15

Nuclear Examples:

- http://www.armslist.com/posts/3840344/wilmington-north-carolina-misc-for-sale--childrens-gas-masks--5-of-them--lowered-price
- http://www.armslist.com/posts/4356769/st-louis-missouri-misc-for-sale-trade--gas-mask

WMD:

- http://www.armslist.com/posts/4295509/nashville-tennessee-gun-parts-for-sale--nickel-boron-upper-assembly--upper--rail--charging-handle--forward-assist---dust-cover--wmd-nib-x
- http://www.armslist.com/posts/4463846/east-st-louis-illinois-rifles-for-sale--wmd-beast
- Even though the query output gave output to the keywords mentioned as related to the query, upon scrutiny of the URL's most of them were not related to the WMD or deadly weapons ads.

Link Based Queries:

In the Link-based algorithm, we first generate the pagerank score for the selected metadata fields by generating graph based on similarity measures (GeoTopicParser, OCR and cTakes) which is explored in earlier section. Later we integrate these pagerank scores in our Solr Index. Depending on the queries we boost certain combination of fields to give us an optimal result. Following is the query results and field boosted are mentioned along with the solution.

A. What time-based trends exist in Gun ads? Can you correlate temporal and spatial properties with buyers? For example, can you identify based on ad time-window and/or based on geospatial area places where people try and purchase guns on behalf of someone unauthorized to purchase them?

We boost the "Location/Geo location PageRank field for this query. Minnesota, Missouri and Michigan has more handgun ads than in Montana. The Python code and the necessary outputs are provided in "Queries 1 and 2 Link Based" folder.

B. Can you identify similar firearms image types (e.g., shotguns) that are sold in the same region and time? Does this indicate influx related to stolen goods?

We yet again boost the geo location pagerank field for obtaining good refined results for the above query. The Python code and the necessary outputs are provided in "Queries 1 and 2 Link Based" folder.

C. When a shipment of bulk firearms is stolen, the rate of ads and images may indicate an increase in sales of that particular make/model – can you identify these?

The boosted fields for this query are the "Manufacturer" and "Gun type". More particular results related to the query are obtained. The changes in the python code helps us in retrieving good relevant results.

- defType = edismax
- qf = pgrk_locations ^ 200 AND pgrk_keywords ^ 100
- stopwords = true
- It is observed that the sellers in the state of Alaska is absolutely zero.
- Texas, Florida and Georgia has the highest number of gun ads being sold.

- More refined results are obtained when queried due to the boosting.

The Python code and the necessary outputs are provided in “Queries 3 and 4 Link Based” folder.

D. Can you identify ads and/or weapons images that are posted by persons, whom are underage or in which the weapons are de-identified (by type and/or serial number, etc.)?

The Seller named “steve” works at more than two gun manufacturing companies. Steve’s personal details, phone number could be retrieved due to the added boosting. The Python code and the necessary outputs are provided in “Queries 1 and 2 Link Based” folder.

E. Can you identify ads and/or images that relate to the unlawful transfer, sale, and possession of explosives, WMD devices, and precursors?

The last query we boost only “keywords” metadata field to 200 times as it is very much relevant.
The boosting of “keywords” field helped us in getting appropriate results. Though the results for “nuclear, WMD etc” were very sporadic. Genuine WMD ads were not detected.

Comparison of Link Based with Content based algorithm results for the given set of queries:

Queries	Content-Based or Link-Based Relevancy
a. What time-based trends exist in Gun ads? Can you correlate temporal and spatial properties with buyers? For example can you identify based on ad time-window and/or based on geospatial area places where people try and purchase guns on behalf of someone unauthorized to purchase them?	Link-based algorithm performs better because of boosting (query level). It helps us in fetching top results where especially geo-location and Dates are present.
b. Can you identify similar firearms image types (e.g., shotguns) that are sold in the same region and time? Does this indicate influx related to stolen goods?	Link-based algorithm performs better because of the query level boosting techniques incorporated. The Manufacturer, gun types, geolocation and dates fields present.
c. When a shipment of bulk firearms is stolen, the rate of ads and images may indicate an increase in sales of that particular make/model – can you identify these?	Link Based algorithm performs better

d. Can you identify ads and/or weapons images that are posted by persons, whom are underage or in which the weapons are de-identified (by type and/or serial number, etc.)	Link-based algorithm refines the results because of boosting factor. This query is much of a corner case and hence with the index we built, content-based algorithm gives more number of results.
e. Can you identify ads and/or images that relate to the unlawful transfer, sale, and possession of explosives, WMD devices, and precursors?	With the data set we obtained and index built, we were not able to comment arguably about one algorithm performing better than other.

Task 5: Develop a program that runs your queries against your Solr index and outputs the results in list of results demonstrating your relevancy algorithms.

For each of the above mentioned set of queries, for both content-based and link based algorithm, python scripts has been written to execute all the queries against the Solr Index and output the results.

The python script,queries and corresponding results is available in the “ContentBasedAlgorithms/Queries” and “LinkBasedAlgorithms/Queries”.