

# Team Name: Team 6

## Assignment 1

### Task 1: Nutch Configuration

Property	Original	Modified	Reasons
http.agent.name		Team6	Bot name
http.agent.rotate	False	True	Politeness
http.agent.rotate.file		Agents.txt	Rotating IDs
scoring.similarity.model.path		Goldstandard.txt	Relevant words
db.max.outlinks.per.page	100	75	Faster crawl
scoring.similarity.stopword.file		Stopwords.txt	Cosine similarity enabled in nutch
file.content.limit	65536	6553600	Manage size issue
selenium.driver		firefox	Focused crawl
plugin.includes	protocol-http urlfilter-regex parse-(html tika) scoring-similarity urlnormalizer-(pass regex basic)	protocol-http protocol-interactive selenium urlfilter-regex parse-(html tika) scoring-similarity urlnormalizer-(pass regex basic)	Handlers for efficient crawling
fetcher.threads.fetch	10	15	Parallel crawl

Attached here is the list of all changes in nutch-site.xml, regex-urlfilter, naivebayesword-list.txt-filters which we used while crawling.

#### **List of Commands used:**

Crawl Statistics:-

```
Shivens-MacBook-Pro:local shivensaiwal$ bin/nutch readddb ../../shiven/crawl_13/crawldb/ -stats
```

Mime Statistics-

```
Shivens-MacBook-Pro:local shivensaiwal$ bin/nutch dump -outputDir ../../shiven/mimstats -segment ../../shiven/crawl_13/segments/ -mimeStats
```

## Task 2: NutchPy

a). and b). We downloaded and installed nutchpy along with anaconda for parsing the crawled data. To see the crawl metadata for which we used sequence reader to extract each url and its corresponding metadata along with a mimetype.txt which gave us a list of mimetypes it could encounter.

c). Mime types we encountered

image/png, image/jpeg, image/gif, image/svg+xml, image/vnd.microsoft.icon, image/jpg, image/x-ms-bmp, image/ico, image/x-jg, image/vnd.dwg, image/fif

d). List of 100 urls that are creating a problem:



q2-urliststats.docx

e). Crawl Statistics:

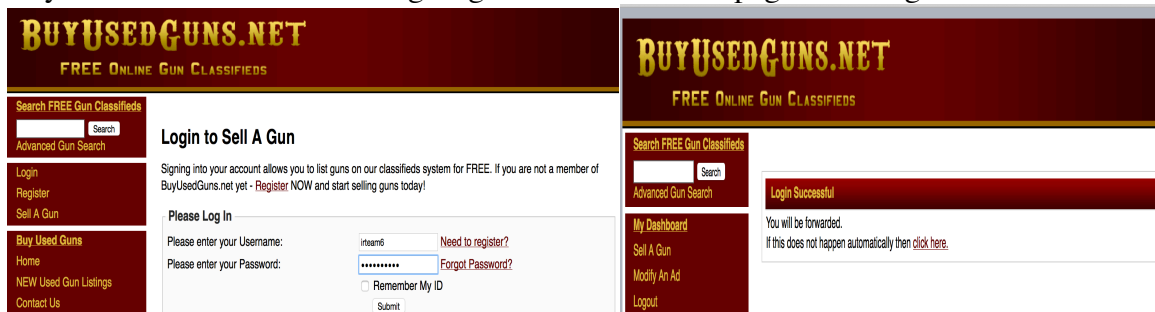
```
CrawlDb statistics start: /Users/pranali/haveri/Desktop/nutch/pranali/crawldata,
crawl_finaldump1retry/crawlddb
Statistics for CrawlDb: /Users/pranali/haveri/Desktop/nutch/pranali/crawldata/c
awl_finaldump1retry/crawlddb
TOTAL urls:      372819
retry 0:         328911
retry 1:         43908
min score:       0.0
avg score:       4.2511245E-5
max score:       1.118
status 1 (db_unfetched): 270483
status 2 (db_fetched):  94416
status 3 (db_gone):     3029
status 4 (db_redir_temp): 903
status 5 (db_redir_perm): 1082
status 7 (db_duplicate): 2906
CrawlDb statistics: done
```

## Task 3: Selenium Plugin

a) We have implemented the interactive protocol selenium plugin by creating a custom handler that utilizes the css properties. We could then successfully then login into the website and using the search box to input value like “gun” we got good amount of relevant images. Please find Team6Handler.java attached.

b) Example:

BuyUsedGuns.net wasn't letting us go ahead to the next page but using selenium we could do so.



## Task 4: Tika and Tesseract

- a) We installed the latest version of Tika and Tesseract to get the parse data unavailable before in the metadata. Using the OCR properties we got the text from the images that qualified for the same. Here is a sample of the image and the corresponding text value in the parse-text repository as best recognized by TikaOCR



Extracted text:

<http://www.armsandammo.com/images/logo.png>  
MGA'IO THE SHOOTER'S DISCOUNT SUPPLIER

## Task 5: Statistics of Enhanced Tika and Nutch Selenium

- a) Link to 100 Urls:



q5-urliststats.docx

- b) After using Tika and Selenium we figured that we could access few of the urls which had login or search functionality and could extract better and relevant images as we had specific search criterias put in.

For eg. Arguntrader.com has a login and search functionality which we handled using our selenium plugin later which gave us better and specific guns images.

- c) From what we have noticed, Tika though difficult to work around with gave us quite good results. We had couple of sites that were not being parsed due to some specific mime types. After using Tika we were successfully able to parse a few of them. Although there were cases which still threw errors, we could get some relevant data from the sites that were parsed by Tika. Eg: sites like [https://tacticaloffense.com/img/FlashArea\\_night\\_vision.png](https://tacticaloffense.com/img/FlashArea_night_vision.png) had problems parsing data with png extension which was successful with Tika parser but it still could not parse types like pdf and css and we had to explicitly enable these parsers in Tika.

- d) **Updated Crawl Statistics:**

```
Shivens-MacBook-Pro:local shivensaiwal$ bin/nutch readddb ../../shiven/crawl_12/crawldb/ -stats
Crawldb statistics start: ../../shiven/crawl_12/crawldb/
Statistics for Crawldb: ../../shiven/crawl_12/crawldb/
TOTAL urls: 190625
retry 0: 184774
retry 1: 5851
min score: 0.0
avg score: 5.047397E-4
max score: 1.333
status 1 (db_unfetched): 168965
status 2 (db_fetched): 18328
status 3 (db_gone): 1649
status 4 (db_redir_temp): 381
status 5 (db_redir_perm): 1132
status 7 (db_duplicate): 170
Crawldb statistics: done
Shivens-MacBook-Pro:local shivensaiwal$
```

## Task 6: Deduplication

### a. Algorithm to calculate exact duplicates:

We have used the Simhash algorithm on the extracted text using Tika-Python. Simhash algorithm maps high-dimensional vectors to small-sized fingerprints. The extracted metadata from images was used to generate a hash value and based on the distance the algorithm works as follows:

- First, we apply the near duplicate algorithm and check for those images that satisfy the threshold condition.
- Next, we extract the metadata (image height, image width, mimetype) and the extracted text (X:Content) was generated using Tika OCR with Tesseract using Tika Python.
  - `parsed_1=parser.from_file('image_1.png')`
  - `str = parsed_1["metadata"]`
  - `contentdata = parsed_1["content"]`
- After extracting the content data and parsed text and we now apply the Simhashing algorithm on both.
  - `Simhash(str(parsed_1))`
- Similarly, we calculate the same for all the other images.
- The hamming distance between images was calculated next and .....
  - **`Simhash(imageparsed_1).distance(Simhash(imageparsed_2))`**
  - if (both distance between the metadata and parse text is same )  
then considered as exact duplicates.

Note: For Simhash, type has to be string i.e. imageparsed\_1 and imageparsed\_2 are string buffers.

### b. Algorithm to calculate near duplicates:

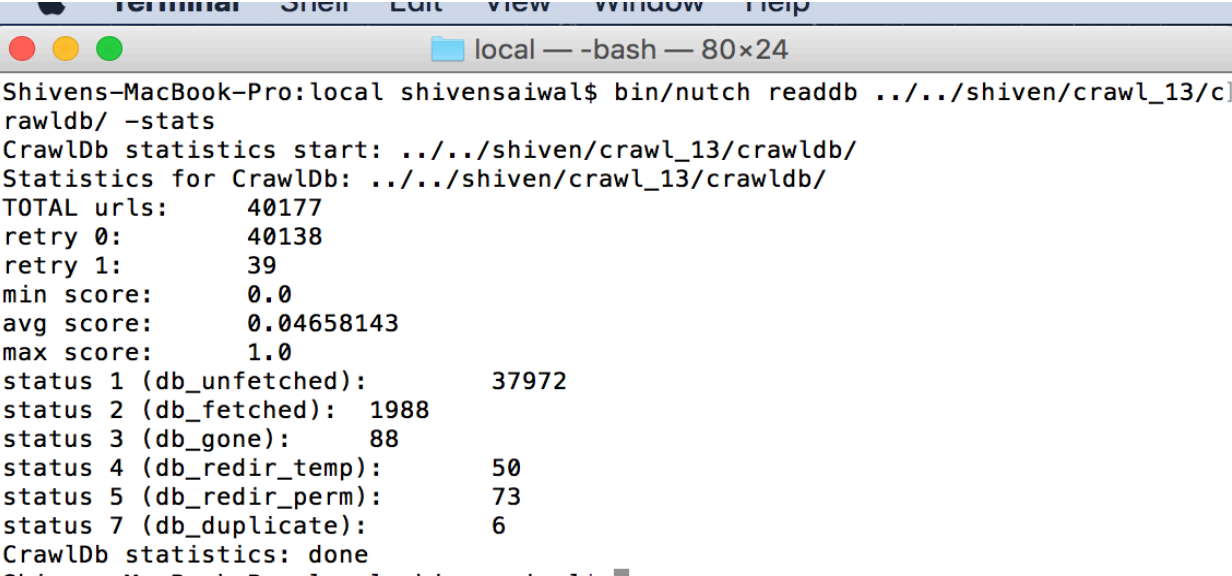
We are using the vector distance algorithm to fetch the near duplicate values.

- All the images are scaled to the same size using a java program, here 180\*120 pixels
  - `BufferedImage resizedImage = new BufferedImage(width, height, img.getType());`
  - `Graphics2D g = resizedImage.createGraphics();`
  - `g.drawImage(img, 0, 0, width, height, null);`
- All images are divided in a matrix format of 50\*50 and average intensity of 500 such random boxes is calculated using the RGB component.
  - `int sRgbColor = resizedImage.getRGB(50, 50);`
  - `Color c = new Color(sRgbColor);`
  - `IntensityVector v = new Vector(3);-> v(R,G,B)`
- The **Euclidean** distance between intensities of these images on respective boxes is now generated.
- Formula for distance:  **$d = \sqrt{(r_2 - r_1)^2 + (g_2 - g_1)^2 + (b_2 - b_1)^2}$**
- Now, we calculate the distance percentage for better accuracy
  - **$P = d / \sqrt{255^2 + 255^2 + 255^2}$**
- Then, find the average percentage ( **$p/500$** ) and compare it with the threshold value (Assuming 3%)
  - If (the threshold condition satisfies <3%)
    - Output: Near duplicates
  - Else
    - Not a Near duplicate.

## **Task 7: Similarity Scoring Filter**

a). and b). We have successfully created a similarity scoring filter where we created a goldstandard.txt which is attached herewith. We also enabled the scoring similarity dependency in nutch-site.xml

c). After putting the similarity filter we noticed that it is definitely giving us better results. We used the deduplication algorithm to remove the duplicate images from our current dump which was a huge task but gave us good results eventually. Near duplicates is efficient because if we give exact duplicates there are chances that all conditions might not match. Scoring similarity filter on the other hand was quite helpful that it gave us less but quite relevant results. The total number of urls fetched drastically reduced since the last crawl but we got relevant images compared to what we had



```
Shivens-MacBook-Pro:local shivensaiwal$ bin/nutch readddb ../../shiven/crawl_13/crawl_13/crawldb/ -stats
CrawlDb statistics start: ../../shiven/crawl_13/crawldb/
Statistics for CrawlDb: ../../shiven/crawl_13/crawldb/
TOTAL urls:      40177
retry 0:         40138
retry 1:         39
min score:       0.0
avg score:       0.04658143
max score:       1.0
status 1 (db_unfetched):      37972
status 2 (db_fetched):      1988
status 3 (db_gone):         88
status 4 (db_redir_temp):    50
status 5 (db_redir_perm):    73
status 7 (db_duplicate):     6
CrawlDb statistics: done
```

### **Overall results:**

Overall while we ran our headless crawls, we found that a lot of times we got a large amount of irrelevant data due to outlinks such as craigslist, classifieds and the like. We found a lot of duplicate images as well as the same seller has hosted his weapons to multiple websites. Also, some of the websites are on sale so the same vendor moves to posting to another site that creates redundancy.

As we were aiming to reach a large number of images in total it was difficult to find exact images manually. Hence we used the SimHash and Euclidean distance algorithms to remove the duplication as far as possible. We were able to complete all the tasks however, crawling took a lot of time but we felt that we could have dived more in analysis and further usage of the software if we had more time. Urls that didn't have outlinks to classifieds were easier to search and gave relevant images. E.g. [www.gandermooountain.com/guns](http://www.gandermooountain.com/guns), [www.wikiarms.com/guns](http://www.wikiarms.com/guns), [www.gunamerica.com](http://www.gunamerica.com) gave us the most relevant data.

### **Was there a particular correlation between gun type and website?**

This is not true for all the websites but there are also some exceptions. For example, In [www accurateshooter.com](http://www accurateshooter.com), we found more of long-range rifles as they primarily specialize in selling guns based on the shooting ranges. Whereas, a website such [www.nexttechclassifieds.com](http://www.nexttechclassifieds.com) has varied data ranging from knives, guns, rifles, bullets to ammunition. So, we can say that there is a correlation between the gun type and the website to an extent but it cannot be generalized.

**Were duplicates indicative of sloppy selling, or simply dealers promulgating the product to multiple sites?**

There are a few websites for eg wikiarms.com/guns in which for each gun it shows its availability in other websites also which shows that the dealers posts to multiple websites which would increase their chances of selling.

**Are there any weapons that aren't necessarily firearms, but more dangerous devices (e.g., explosives)?**

Yes, we found some explosives which were more dangerous:

We found the tannerite explosive while fetching from the following urls:

[http://cdn2.armslist.com/sites/armslist/uploads/posts/2015/09/01/4654802\\_01\\_tannerite\\_explosive\\_targets\\_640.jpg](http://cdn2.armslist.com/sites/armslist/uploads/posts/2015/09/01/4654802_01_tannerite_explosive_targets_640.jpg)

<http://pics.gunbroker.com/GB/515039000/515039438/pix251753713.jpg>

**What relationships do the clusters resultant from your deduplication algorithms tell you? Are they simply related to the ways that the pictures are edited, or indicative of anything more?**

After implementing our deduplication algorithms we found out that the best way to find near/exact duplicates is fingerprinting on smaller parts of images. We analyzed the clusters and found out that even though the images are similar they can be differentiated using their metadata as it is not just the way it was edited but also the type of camera, when it was taken apart from the compression type, height, width. We found out that pixel based comparison gave us the best results.

**Thoughts about Apache Nutch and Apache Tika:**

We realized that apache nutch is an excellent tool to start off crawling. The documentation is straightforward and the properties in nutch\_default.xml are descriptive enough to explore various options and filters for achieving a directed crawl.

Tika is a lifesaver in terms of mime types not getting parsed using nutch's default parser. The key observation from mime data was the number and variety of mime type, which we extracted. Tika assistance was crucial in the parsing step. Overall it was a good learning experience leaving us craving for more.