1. Download and configure Nutch to crawl Weapons images as identified in the seed list that will be sent to you by the graders

a. We set the agent ID with usc-572-group17.

b. We try and change the configurations below in nutch-site.xml
http.content.limit, db.update.max.inlinks, db.ignore.external.links, db.max.inlinks, fetcher.server.delay, plugin.includes ...etc

2. Perform crawls of Weapons images sites

a. We write a python program nutchpy_content.py to get the content-type from data in crawl/segments/.../crawl_fetch directory. It based on nutchpy.sequence_reader and reg-ex to get the mime-type begin with "image". Here's what we get:

image/jpg image/jpeg image/vnd image/bmp image/svg
image/png image/gif image/x-ms-bmp
image/tiff image/x-icon image/vnd.microsoft.icon

b.Among 100 failed fetching urls that we attached ("Q2b.txt")

c. In this part of crawling, we separate the seed into several partitions, ex: seed1-1, seed1-2, seed2… All of the seed files, configurations and python files are in the "source files" directory.

d. crawl statistics are in "crawl statistics" folder. The readdb_out/ and segment_out/ folders contain all the urls we fetch and parse, also with their http status and metadata response. The all_pic_urls.txt contains all the image we get. The format is the same as question 5d and 7d, which also ask for crawl statistics

e. In the crawling pictures we found that the types of weapons are correlated with certain websites. For example, some websites focus on used guns, some focus on gun accessories ...etc. Further, subfolders in websites sometimes have relationships with types of gun, like rifles/ or pistols/

3. Installing selenium:

Problems we met:

a. For OSX users, it's impossible to download X11, or xvfb, by following the guideline in the wiki page (https://github.com/…/tree/trunk/src/plugin/protocol-selenium) since the command "sudo apt-get install" is not allowed. We finally install xQuartz, the Apple Inc.'s version of X11, to solve the problem.

b. Version capatibility for Nutch selenium and Firefox:
Initially, we use the selenium 2.47.1 and the Firefox 41 to crawl urls, but it causes many errors such as "Failed to connect to Firefox binary, ""Unable to bind to locking port 7054 within 45000 ms, " and "Firefox profile cannot be loaded". We finally downgrade Firefox from Firefox41 to Firefox 33, and the frequency of these errors is reduced.

c. Some webpage need to login or pass CAPTCHA verification before entering the main page. For these special cases, we have to extend the selenium plugin and let the selenium knows how to handle these problems.

4. Installing Tika:

The Nutch 1.11 truck that is required in this assignment has already contained Tika. The only thing that we have to do is to upgrade Tika.

5. Re-run your Weapons crawls with enhanced Tika and Nutch Selenium

a. Among 100 failed fetching urls that we attached ("Q5a.txt"), the most of reasons that Tika cannot fetch are:
   (1) Pop up dialog (mark as the blue urls): users have close the pop up dialog before crawling main pages
   (2) Form issue (mark as the blue urls): such as selected options, accept the agreement, and log in or registration
   (3)Http 404 (mark as the black urls): the web page is not found

b. All 100 failed fetching urls that we selected are not contained in that of the original Nutch.

c. In order to compare the original Nutch truck with the Nutch selenium plus Tika, we select 14 urls that the original Nutch truck has some difficulties to crawl and use the Nutch selenium with Tika to crawl these 14 difficult cases. The urls that we select are:

d. After fetching these 14 urls four times, the number of failed fetching cases for Tika is 62, while the original Nutch truck results in more than 800 failed fetching cases. And all Tika's failed fetching cases are not included in that of the original Nutch truck.

6. Develop two deduplication algorithms to use on the extracted text and metadata in the Parsed Content from Nutch

a. In this program, we implement the algorithm of exact duplicates. From each image's metadata, we extract six features such as image length, compression type name, transparent color index, color space type, image width, image height and combine those strings into a string. Then, we use md5 cryptographic hashing methods to turn the string into a hashing code. Comparing each image's hashing code and only keep the unique one. This code is under folder source files/deduplication/exactMatch.py. In this algorithm, once one of the parameters is different and this images will be considered as differed one.

b. In this program, we implement the algorithm of near duplicates. From each image's metadata, we extract six features such as image length, compression type name, transparent color index, color space type, image width, image height and combine those strings into a string. Then, we made each image's fingerprint with shingling in 3-grams and compare every images' fingerprint with Jaccard similarity method. This code is under folder source files/deduplication/nearMatch.py. In this algorithm, even if the image's length and width are different but content are the same, it still has the chance to be considered as similar one via adjusting the threshold.

7. Enable the Nutch Similarity Scoring Filter Focused Crawling Plugin

c. We consider that our deduplication algorithms especially near duplication is far more effective than Nutch similarity scoring filter focused crawling plugin. Since we have discussed with Sujen—the Nutch maintainer, he said that "The plugin works only against what comes from the parsing step in nutch. That is the parseText object only (which is mostly the text content of a url that Nutch parses). It currently does not support any forms of metadata or images of yet." Besides, this plugin does not support any media format yet. So, even if we write any text in gold standard text file or stop word text file, these only leads our crawler to have more chance to crawl the content about gun's webpage instead of the images.

However, our near exact algorithm takes out the features of images from their metadata which is more relevance to images. It's really arduous to do fingerprint with every images merely with their metadata, some images might have the same content but with different parameters like width and height. If I use image's dhash function to do fingerprint, it easy to consider as the same image, however, metadata gives merely limited information to use.
In addition to features, my near duplicate can customize the size of shingling and threshold which can best suit for making filter restrictions that I want. Therefore, our near duplicates algorithm is way more effective.

8.configure Nutch-Python to control crawling

a. The program "nutch-python" performs crawling using REST server. I believe this is the standard way to call and use REST for nutch server. However, it seems to have some "method" issue that we are not able to use "post" to create. We keep receiving 415 and 500, and we think this might be solved after some changings in nutch-python source code. This code is under folder source files/Q8/ nutch-python.py

9. Dump the crawled data out of your Nutch content and then run https://github.com/chrismattmann/tika-similarity/ over it.

a. I find out that the clusters formation is according to the metadata of images. For example, it may according to the color space type or the color shows in the images. If those images with most red color or yellow color, then those images have the possibility to become one cluster. So, it will count the percentage of each color occupied in the image as one important consideration. Besides, I found that different clusters still have the possibility to contain the same images. Therefor, I consider that the tika-similarity will check images' metadata information and cluster images with the number of features from metadata.

10. Memex Explorer

a. I am able to run Memex Explorer for my crawling. However, the result is much less than using nutch. I found Memex Explorer loss some of the web pages even I set nutch to default configuration and compare these two.
b. The most convenient part using Memex Explorer is that is doesn't provide immediate viewing logs. If I run a large amount of seed for several runs, I can only get the logs when I want to see it, and the log content stop at the time I dump it out. And after the crawling, the visualizing part is not working well. I tried the patterns for a lot of time for the results which are easy to dump in nutch.
c. Memex Explorer doesn't need to deal with the complex configuration as nutch does. This is the best part and the worst part at the same time. What attracts people the use Memex Explorer is that we don't have to do so much setting and can easy do the crawling through web. However, after getting familiar with nutch, it's more powerful to get involved in and control the crawling.