CSCI 572 Homework 2 Report

Tasks:

1. Google Hangout Session with Dr. Burgess

We learned a quite a few relevant things about the assignment from our Google Hangout session with Dr. Burgess. She told us some facts about the AMD dataset, such as the fact that there is always a spatial coordinate listed for data, and that it contains more than just snow and ice data (for example, human dimensions and biological data). In addition, she discussed her interests as they pertain to the field of climate science. For example, she is studying spatio-temporal changes.

With regards to the actual queries, Dr. Burgess gave us a lot of helpful tips. The one we followed most closely is that, since we are looking at metadata, it is easier to answer questions such as "Are we paying more attention to sea level rise?" than it is to answer questions like "Is sea level rising?".

This piece of advice, in addition to Dr. Burgess' stated interest in spatio-temporal changes, helped guide us in creating our science questions.

2. Develop an indexing system using Apache Solr and its ExtractingRequestHandler ("SolrCell")

We began by building Tika trunk (1.8-SNAPSHOT). We made changes to the tika-trunk/tika-parsers/pom.xml file so that it could be integrated with Solr. Then, since we had already built Tika with GDAL and OCR support during HW1, we upgraded Tika to support FFMPEG files. Then we built SOLR to contain the latest version of Tika by making changes to the Solr/lucene/ivy-settings.xml & Solr/lucene/ivy-versions.properties files.

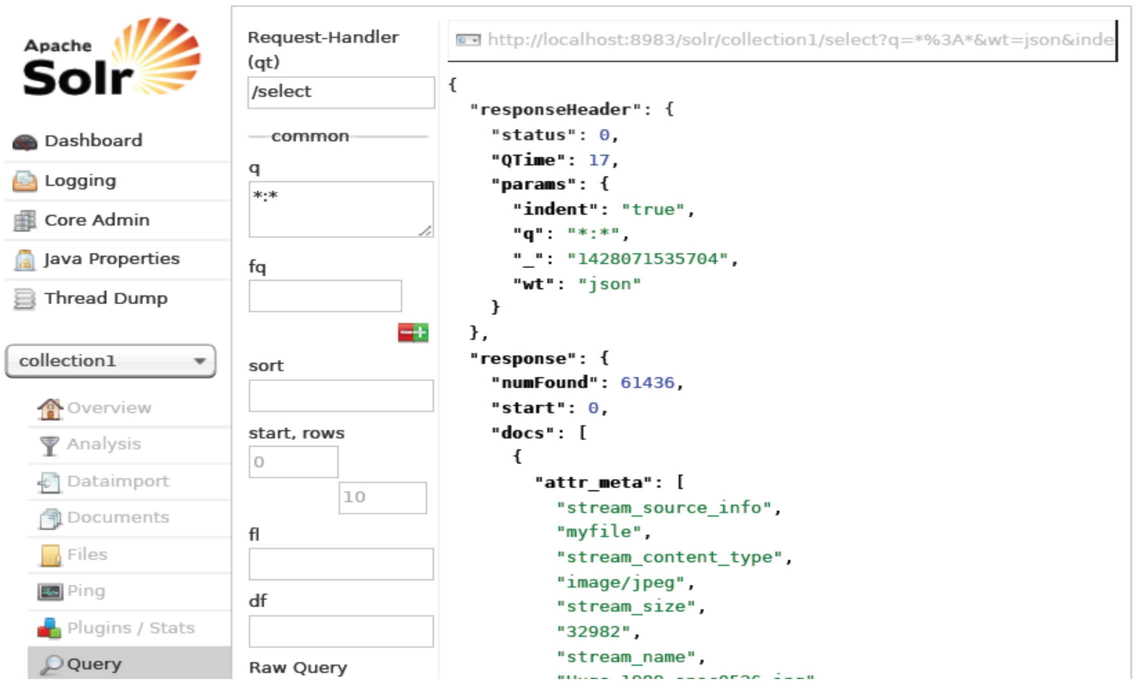
After successfully building Solr with support for GDAL, OCR and FFMPEG we were able to test it by indexing a small set of different file types. We used curl to index the file in Solr. After testing, we ran the script to index 61,500 documents in Solr.

3. Leverage the Nutch indexing system to build up an Apache Solr index

a. Compare the metadata extracted from using Tika in Nutch during crawling upstream compared to your SolrCell based Tika run generated in Task #2.

i. What did Tika extract in Nutch compared to what SolrCell extracts?

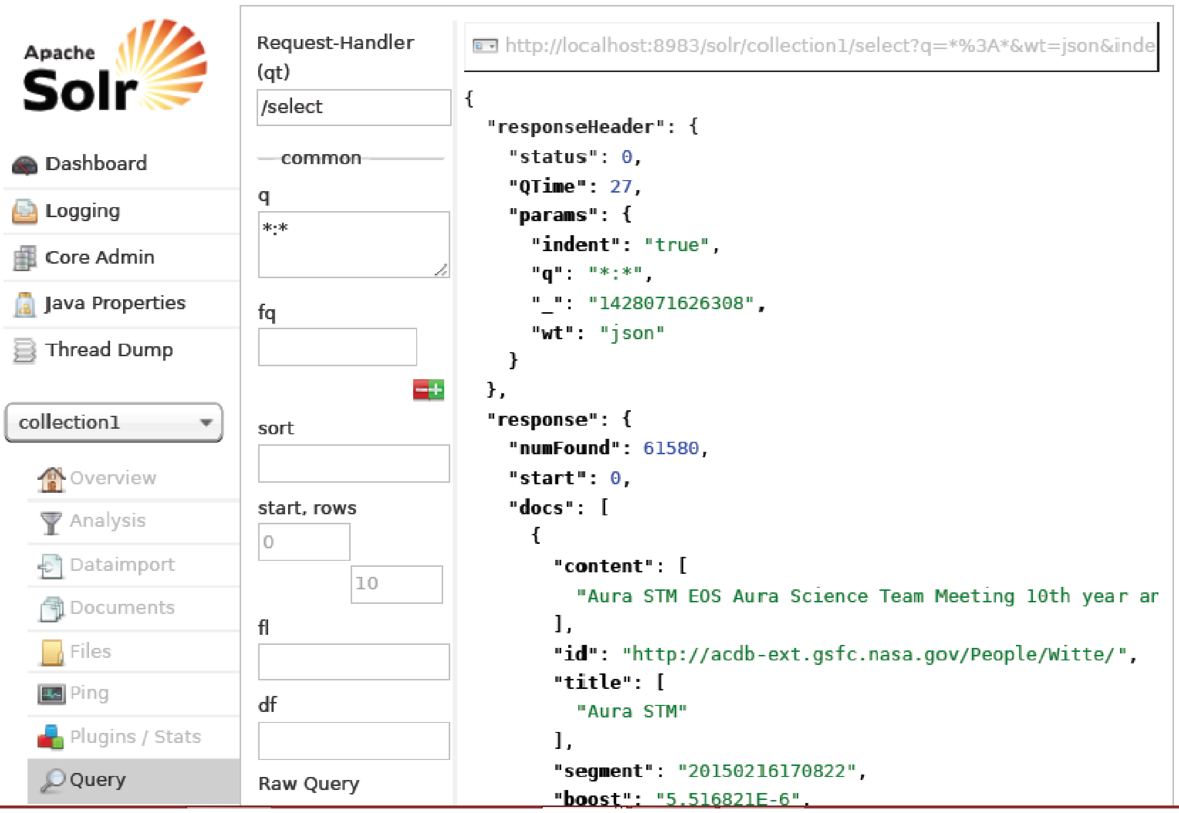
We found out that the metadata generated by the SolrCell had many more attributes as compared to the Nutch solrindex. We have attached pdf file to show the difference between the two methods. Here are the screenshots of the pdf (**Nutch.pdf, Solr.pdf**).



Request-Handler (qt)
/select
common
q
`*:*`
fq
sort
start, rows
0
10
fl
df
Raw Query

http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&inde

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 17,
    "params": {
      "indent": "true",
      "q": "*:*",
      "_": "1428071535704",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 61436,
    "start": 0,
    "docs": [
      {
        "attr_meta": [
          "stream_source_info",
          "myfile",
          "stream_content_type",
          "image/jpeg",
          "stream_size",
          "32982",
          "stream_name",
```



Request-Handler (qt)
/select
common
q
`*:*`
fq
sort
start, rows
0
10
fl
df
Raw Query

http://localhost:8983/solr/collection1/select?q=*%3A*&wt=json&inde

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 27,
    "params": {
      "indent": "true",
      "q": "*:*",
      "_": "1428071626308",
      "wt": "json"
    }
  },
  "response": {
    "numFound": 61580,
    "start": 0,
    "docs": [
      {
        "content": [
          "Aura STM EOS Aura Science Team Meeting 10th year ar
        ],
        "id": "http://acdb-ext.gsfc.nasa.gov/People/Witte/",
        "title": [
          "Aura STM"
        ],
        "segment": "20150216170822",
        "boost": "5.516821E-6",
```

As you can see in the above screenshots, we indexed more fields in the first one which was using Solr.

4. Design and implement two ranking algorithms for your Polar data documents

a. Content-based Algorithm

Since Solr does not index other(it only indexes xml,html,pdfs) multimedia mime types such as jpeg, docx, pptx automatically, we wrote a script to parse such files into Tika app jar and get meta-data files for these type of application files. These meta-data files were indexed into solr.

The benefit of such a method was that tika worked like a charm and we got lot of new fields which were indexed under the attr_* fields.

These new fields are of the field-type 'text_general'. The analyzer for this field-type is 'solr.StandardTokenizerFactory'. We did not modify this, however we have added more filters which are an appropriate fit for our data. We added 'KeyWordRepeatFilterFactory' and 'RemoveDuplicatesTokenFilterFactory'. This definitely increases the number of tokens we index but it helps us improve the results,since the if we have an exact match,it is assigned a higher score. We also added common stop words found in the english language in the **stopwords.txt** file which was previously empty.This is needed by the StopWordFilter.

The crux of the content based algorithm is about boosting the correct fields at the query time. We boost a token if it present at the right location. We have boosted the tokens which are found in the title and in the keywords fields. We feel that title and keywords correctly summarize the document and its contents and thus we give this bigger boost than query tokens matching in the content but nevertheless they have to be present in the content too, for the score to be boosted.

Also we have used the some of the parameters of the dismax query syntax. We chose to include the dismax options as they help us take care of lousy user entered query syntax. Thus we can include multi word queries.

By boosting only these fields we quickly get the score of each document and rank them on their content

b. Link-based Algorithm

For our linked based algorithm, we implemented a modified version of PageRank. The original PageRank algorithm is based on inlinks and outlinks. Our algorithm differs in that is is based on geographical properties rather than inlinks and outlinks.

The input for the original PageRank algorithm is a graph of the pages in which a page **u** has an edge to a page **v** if page **u** contains a link to page **v**.

To generate the input for our algorithm, we used the CLAVIN Java library in order to get a list of the latitudes and longitudes of all resolved locations in all documents. Then we round all of the latitudes and longitudes to the nearest 1 and remove the duplicates from each list. (This benefits us in two ways: first, it helps us to link documents which reference nearby locations which are don't have the exact same latitude and longitude, and second, it can reduce the overall number of latitude-longitude pairs, so we save some space and future processing.) (The code which computes these latitudes and longitudes is found in our file **GetLocations.java** as well as **LatLon.java**.) Finally, we link all those files which share a latitude-longitude pair. By this, we mean that if the latitude-longitude pair (**x**,**y**) is contained in document **u**'s latitude-longitude list as well as document **v**'s latitude-longitude list, we make an edge from **u** to **v** as well as an edge from **v** to **u**.

Once we have generated all of these geographical links among our documents, our algorithm proceeds in the same way as the original PageRank. We first give each document the initial PageRank score of $\frac{1}{N}$, where d is the damping factor, and N is the total number of documents. We used the widely recommended damping factor of 0.85.

Next, we iterate through our list of documents, updating the PageRank score for each document according to the following formula.

$$PageRank_{new}(doc_i) = \frac{1-d}{N} + d \sum_{x \varepsilon X} \frac{PageRank_{old}(x)}{\#LinksFrom(x)}$$

where X is the set of documents which have a link to doc_i, and $\#LinksFrom(x)$ is the number of documents which are geographically linked to document x.

This code is found in our file **pagerank.py**.

So, once each document has a page rank, we index that into Solr as an additional attribute via a curl request, in which we use literal.pagerank = <the pagerank> . For specifics on the curl syntax, see **post_pagerank.sh**, our bash script which performs these requests.

Finally, to test out our link-based algorithm on a query we ordered the query results based on pagerank associated with each page. We defined a new field name **pagerank** in the schema file so that we can index it and so it for linked based queries.

5. <u>Queries to demonstrate answers to the scientific questions</u>

Below are the nine queries to demonstrate answers.

- ❏ Are there any changes in migratory patterns of animals or birds due to rise in sea level?
- ❏ What natural resources are present in the Arctic Region?
- ❏ What military operations are currently being conducted in Antarctica?
- ❏ How do scientists monitor the Arctic wildlife?
- ❏ What oil exploration is happening in polar regions?
- ❏ Have sea ice changes led to new international shipping routes?
- ❏ What minerals can be found in Alaska?
- ❏ What programs are present for endangered species?
- ❏ How do polar species adapt to changing environmental conditions?

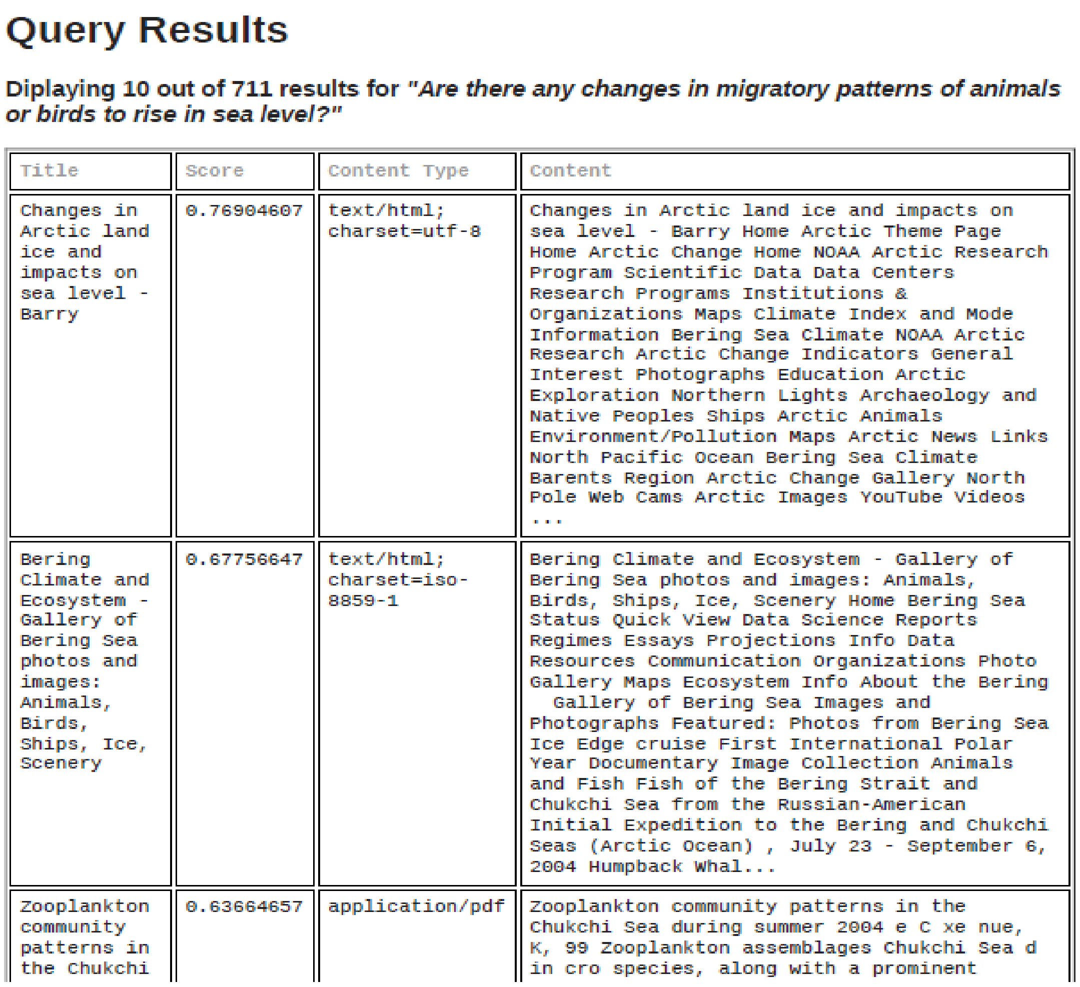
6. Develop a program in Java, Bash, and/or Python to run the queries against Solr index and outputs the results

We wrote a java program that given the queries, generates URLS that have to be called to query the solr server. The java program is called **QueriesURLGenerator.java**. It will output Query1.txt, Query2.txt etc. These files contain the URL which have to called on the Solr server. It will produce twice the number of output files as the number of queries where the initial files are Content based and next files are for the page rank.

For example if there 2 user queries defined we will have 4 output files Query1.txt,Query2.txt..Query4.txt. Then Query1,2.txt will have content based algo URL and Query3,4.txt will have Pagerank algorithm URL.

These URLS are encoded in bash script files named **queriesCB.sh & queriesPR.sh**. This bash scripts takes a command line argument of the query number and calls the Solr server with the query URL. It downloads the generated json file and stores it. Then we open the file **resultCB.html or resultPR.html** in the browser and displays the json data. It gives the top 10 result based on the content score or the pagerank. We can further see all the fields of the output by cliking on the respective row of the output table. We have tried to make the output as easy to understand and visually pleasing as possible. We included the fields which we think are important and which depict and explain why the results are ordered this way in the primary output table. Thus for the content based algorithm the table contains the 'score' column and for the link based algorithm we have included the 'pagerank' column.

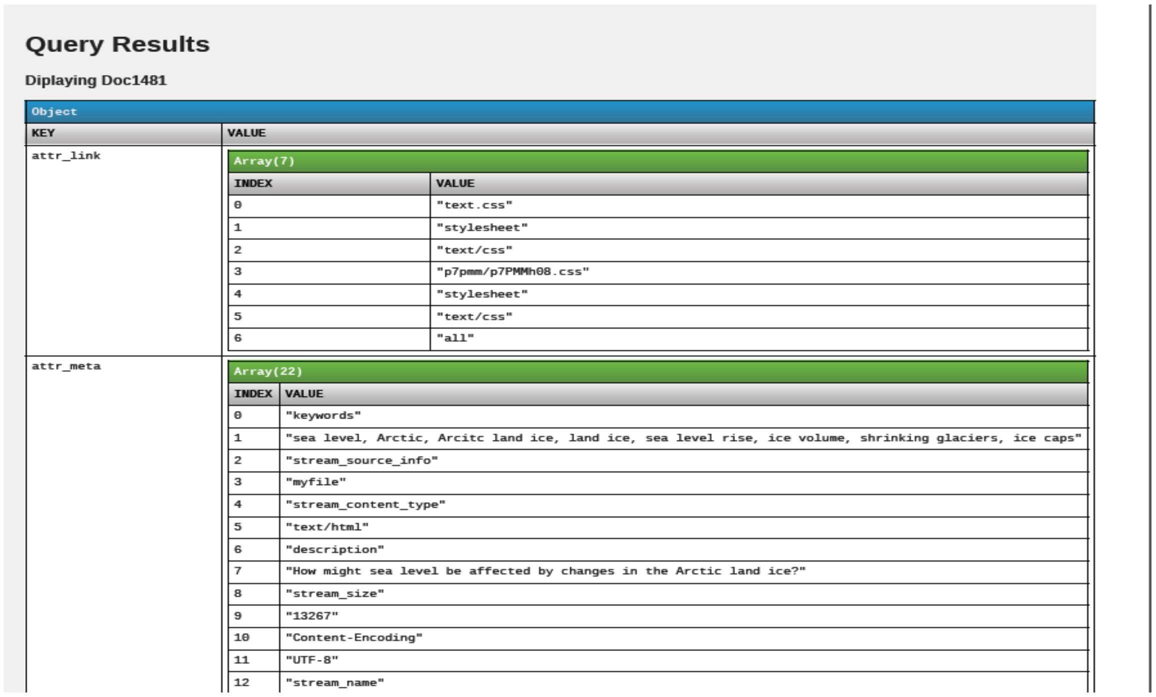
We have included a screenshot to illustrate what we want to show.



Query Results

Diplaying 10 out of 711 results for *"Are there any changes in migratory patterns of animals or birds to rise in sea level?"*

Title	Score	Content Type	Content
Changes in Arctic land ice and impacts on sea level - Barry	0.76904607	text/html; charset=utf-8	Changes in Arctic land ice and impacts on sea level - Barry Home Arctic Theme Page Home Arctic Change Home NOAA Arctic Research Program Scientific Data Data Centers Research Programs Institutions & Organizations Maps Climate Index and Mode Information Bering Sea Climate NOAA Arctic Research Arctic Change Indicators General Interest Photographs Education Arctic Exploration Northern Lights Archaeology and Native Peoples Ships Arctic Animals Environment/Pollution Maps Arctic News Links North Pacific Ocean Bering Sea Climate Barents Region Arctic Change Gallery North Pole Web Cams Arctic Images YouTube Videos ...
Bering Climate and Ecosystem - Gallery of Bering Sea photos and images: Animals, Birds, Ships, Ice, Scenery	0.67756647	text/html; charset=iso-8859-1	Bering Climate and Ecosystem - Gallery of Bering Sea photos and images: Animals, Birds, Ships, Ice, Scenery Home Bering Sea Status Quick View Data Science Reports Regimes Essays Projections Info Data Resources Communication Organizations Photo Gallery Maps Ecosystem Info About the Bering Gallery of Bering Sea Images and Photographs Featured: Photos from Bering Sea Ice Edge cruise First International Polar Year Documentary Image Collection Animals and Fish Fish of the Bering Strait and Chukchi Sea from the Russian-American Initial Expedition to the Bering and Chukchi Seas (Arctic Ocean) , July 23 - September 6, 2004 Humpback Whal...
Zooplankton community patterns in the Chukchi	0.63664657	application/pdf	Zooplankton community patterns in the Chukchi Sea during summer 2004 e C xe nue, K, 99 Zooplankton assemblages Chukchi Sea d in cro species, along with a prominent



Query Results

Diplaying Doc1481

Object

KEY	VALUE
attr_link	Array(7)

INDEX	VALUE
0	"text.css"
1	"stylesheet"
2	"text/css"
3	"p7pmm/p7PMMh08.css"
4	"stylesheet"
5	"text/css"
6	"all"

KEY	VALUE
attr_meta	Array(22)

INDEX	VALUE
0	"keywords"
1	"sea level, Arctic, Arcitc land ice, land ice, sea level rise, ice volume, shrinking glaciers, ice caps"
2	"stream_source_info"
3	"myfile"
4	"stream_content_type"
5	"text/html"
6	"description"
7	"How might sea level be affected by changes in the Arctic land ice?"
8	"stream_size"
9	"13267"
10	"Content-Encoding"
11	"UTF-8"
12	"stream_name"

Questions:

1. Explain how you came up with your scientific questions to ask of the data.

After reading the sample queries, we were inclined towards biodiversity data. We were interested in knowing how snow and environmental conditions affected the wildlife present and how they adapted to it. We have added some queries such as military operations to determine how it affects commerce activity and some general questions that explore the polar region and looks for resources.

Summarization of query results:

Note: Our data is mainly indexed from the ADE dataset.

Here we will discuss about the queries where content based algorithm worked wonderfully and linked based did not perform that well.

The output directory (**Query output**) contains pdfs of search results. Please look at the result of the queries. Each file also contains the queries at the top so it is easy to go through them.

The files with CB means result of content based algo and PR means result of Page Rank algo.

As we see from the file ((Query-CB1.pdf) the top ten results for the content based algo are all very relevant to the question asked. The question is about the changes in migratory patterns due to rising sea levels. We find the results are good as the terms in the title have been boosted to provide more score.

Our most happy result happens in query 6. We ask the question "Have sea ice changes led to new international shipping routes?" and the very first document returned has the title "International Group Approves New Routes for Large Ships within Monterey Bay National Marine Sanctuary", so it answers the query directly!

Another nice result comes in query 8. We search for programs for endangered species and find such results as "Recovery Plans for Endangered and Threatened Species" and many other results pertaining to fisheries and endangered species' programs.

2. How effective was the link-based algorithm compared to the content-based ranking algorithm in light of these scientific questions?

Overall, we found that our content-based algorithm produced better results than our link-based algorithm. One reason for this is the way that we have established links between documents. Because we link documents which share a location, those documents which reference popular locations will get boosted page rank scores. So, potentially more relevant documents might get a worse ranking in our results for certain queries (e.g. a query for a unique or obscure location).

3. What questions were more appropriate for the link based algorithm compared to the content one?

In general what we found was that our link-based algorithm worked better in situations where there was a lot of documentation. A specific example would be our query regarding military operations. There are several journals and official documents pertaining to these documents, and they link together geographically, boosting each other's page rank score.

4. Describe the indexing process – what was easier – Nutch/Tika + SolrIndexing; or SolrCell?

The indexing process was as follows:

For the SolrCell method we created a Bash script which first renamed the files to replace special characters with _ . Then we indexed the file using the command

```
curl
http://localhost:8983/solr/update/extract?literal.id=Doc$ID&uprefix=attr_&fmap.content=attr_c
ontent -F myfile=@$FILE
```

And finally committed the changes using the command

```
curl http://localhost:8983/solr/update?commit=true
```

For the Nutch method, we had to update the schema.xml in the collection1/conf with the schema-solr4.xml in the nutch/conf folder.

To index the files we had to run the command

```
bin/nutch solrindex http://localhost:8983/solr crawl/crawldb -linkdb crawl/linkdb -dir
crawl/segments
```

We found the SolrCell method of indexing much easier than the Nutch-based method because the documentation of ExtractingRequestHandler was easy to understand but after we were successfully able to update the schema.xml we found that it was faster to index the data directly using Nutch and it just need a single command compared to script to index the files using the ExtractingRequestHandler.