Evaluation of Content in the Polar Dynamic Domain Dataset

Introduction

This experiment aims to study and implement the evaluation of the content of the TREC-DD-Polar dataset, collected over years with large diversity, to scientifically enrich the Polar dataset. As a part of the assignment we will working on answering questions like are the mime detection techniques good, did the parsers that we used collected the right data, are we getting the right call to the parser for each mime type from tika, is the metadata collected good, what is the distribution of the the file size diversity, is the language detection of tika working fine or is it getting confused among multiple languages, the units of content collected e.t.c. Each of the above question will be using its own D3 visualization to better analyze the data. The important part of this study is the evaluation and assessment of collected results.

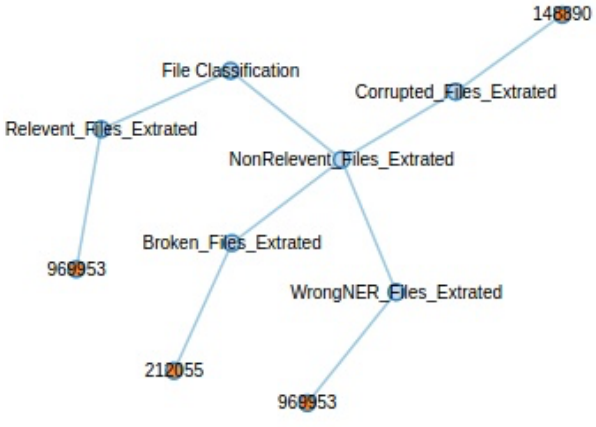
1. Path From Request to Content

Implemented a python script whose task was to read the response part of each file from the common crawl dataset and identify the following:-

a) Categories of pages that were part of the request before reaching the landing pages.

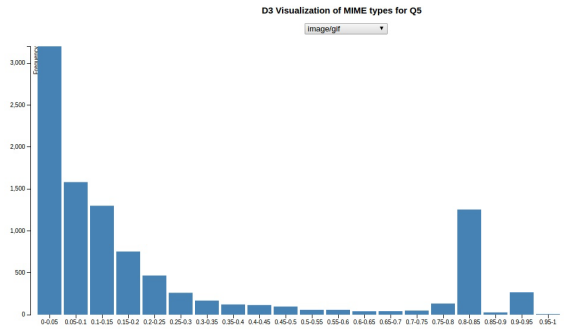
b) The NER entities that were present at the arriving page.

The motive behind the above rationale was to find if the requests made to collect the common crawl dataset were collecting proper pages and the the NER entities were also matched to further find the accuracy of data collected by the crawler used to collect the common crawl data set. Most of the pages collected as part of the crawler were found to be relevant but majority of the data had many broken or corrupted files. This shows that there was a timeout or connection issue or size of data collected issue which resulted in the broken and corrupted files.



2. File Size Diversity of CCA Dataset

Implemented a python script to find the size of each file collected and was grouped on the basis of the mime type. Further each of the file was compared with the respective data collected as a part of indexing by Solr. A ratio matrix was created for each file in the common crawl dataset i.e size of file/size of respective data collected by Solr. A D3 visualization was used to study the file size diversity for further clarity of the distribution. The bar chart was found to be an effective tool for this purpose. Since this was file size the sizes were normalized and frequency of each file range was taken into consideration. Following is the D3 visualization for one of the type:-



As a part of the visualization we realized the following:-

“we noticed that most of the msword files were in the mid ranges of around 50-60kb, whereas most of the gif files were very short in the range of 10-40kb.”

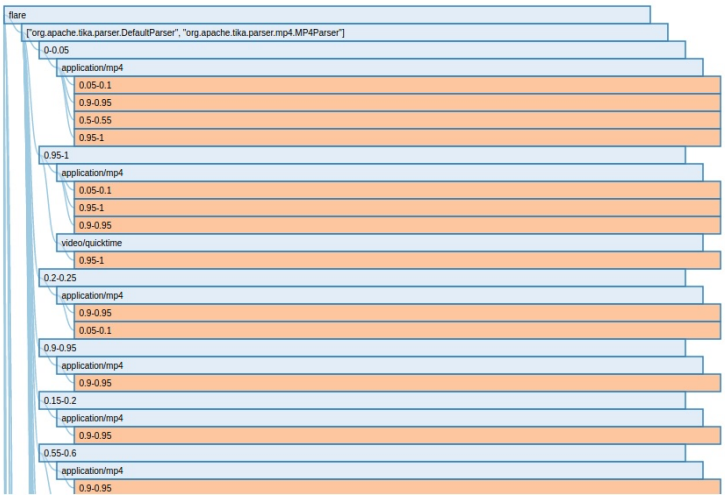
Sample Output:-

letter	frequency
0-0.05	49
0.05-0.1	20
0.1-0.15	71
0.15-0.2	2
0.2-0.25	3

0.25-0.3	0
0.3-0.35	7
0.35-0.4	0
0.4-0.45	10
0.45-0.5	220
0.5-0.55	31
0.55-0.6	2
0.6-0.65	0
0.65-0.7	18

3. Parser Chain Call

One of the important step in tika while detecting the content type is the call to the proper parser. Tika usually calls to the default parser and this default parser handles the call to other parsers. It might call the correct parser directly or call the composite parser which will call multiple parsers in turn to extract the content or it may call external parser to get the content. To analyze this we wrote a script to find for each mime type which parser was collected and how much content and metadata it collected. This was done to find which type of parser collects what amount of data and also find for a particular mime type what is the parser chain call that collect the proper data. Similar to above cases a D3 visualization was created for content and metadata separately to analyze the result collected. A indented tree visualization was used for this case. The indentation used for this was the parser chain call, then comes the mime-type including its subtype and then in the indentation comes the amount of data collected. The following is the example foe a particular mime-type:-



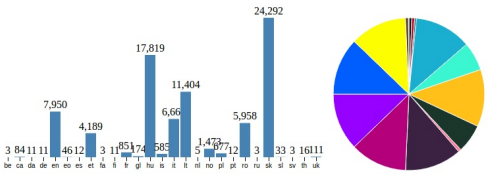
Looking at the visualization we realized that

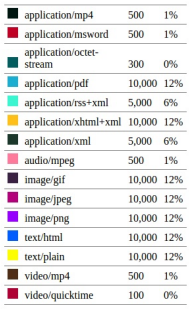
Sample Output:-

{“array":
{"parser_chain":"org.apache.tika.parser.DefaultParser>org.apache.tika.parser.pdf.PDFParser","Values":{"100-500":
[13212,143.015625,19.0234375],">500":
[234,177.015625,37.0234375],"0-10":
[23399,2334.015625,221.0234375],"10-50":
[345,12.015625,12.0234375],"50-100":
[234,11.015625,7.0234375]}}}

4. Language Identification & Diversity

Another tika component that helps tika a lot during content extraction is its language identification library. If we pass the content of the file then it identifies the language of the file. As a part of the assignment we tried to find the diversity of file languages in the Polar dataset. A dash board D3 visualization was chosen to study the data collected. We tried to categorize on the basis of mime-type and for each mime the language that were contained as a part of mime type were represented as a part of bar charts. The bar charts represented the count of files for each language type. The following is the D3 visualization for a particular mime type:





Most of the files were correctly identified but the language detector failed for some of the files which had multiple languages in them. It classified them to english by default.

Sample Output:-

{
"freq": {
"application/mp4": 0,

```
  "application/msword": 333,
  "application/octet-stream": 18,
  "application/pdf": 4,
  "application/rss+xml": 2378,
  "application/xhtml+xml": 1209,
  "application/xml": 502,
  "audio/mpeg": 1,
  "image/gif": 18,
  "image/jpeg": 1,
  "image/png": 1,
  "text/html": 1025,
  "text/plain": 2460,
  "video/mp4": 0,
  "video/quicktime": 0
 },
 "State": "en"
}
```


5. Word Cloud

The word cloud visualization was used to find some of the most common and repeating words in the entire polar dataset. This visualization showed the important words in the documents in the form of a Word Cloud. Before forming the Word Cloud, I removed the common stop words so that the D3 is not cluttered and we can gather useful information out of it. We noticed that certain keywords like Ozone, Heat, National, NASA, Climate and Minerals are very common in our dataset. This is not surprising as our data is from the TREC Polar dataset. The word cloud D3 visualization was found most suitable for this purpose because it highlighted the most important words in the polar data set. Following is the D3 visualization generated for the polar data set.

Sample Output:-


```
{"Word": "Originator/Creator:","Count": 89298},{"Word": "tEducation/Outreach","Count": 6877}, {"Word": "Larsen","Count": 17823}, {"Word": "AEROSOL","Count": 11300}
```


6. NER and Maximal Joint

The NER tools that were used as a part of this question were NLTK, Core NLP, Open NLP and Grobid Quantities. Each has its own set of named entities that it was able to extract from the polar data files. The aim of this question was to find a maximal joint agreement. The maximal joint agreement is the intersection of all the data collected from the above 4 NER tools. This is done to find the common data that has been extracted by all the 4 NER tools. This gained information was then used to enrich the Solr index. For each file the Solr index was compared with the maximal agreement. If we were able to discern any new properties that are not present as a part of Solr index then this data was added to enrich the Solr index. Most of the files had some entities missing in the Solr index because last time we only used Open NLP. The addition of these content improved the searching capability of solr and gave better results than earlier.Folder wise description to facilitate usage of the NER modules separately as follows:

a. NLTK Rest Server- User can use the python files in 2 modes. Use NLTKRESTServer.py to start a localhost server at 8881 serverUse the NLTK library to point to files in the local file system. Use nltk.download('all') if your program is not able to find the correct models to load. This will create a nltk_data folder in temp directory for its use.

Description - The nltk uses the POS tokenizer to read the content and mark as pos token ,for NER we are going to filter on 'NE' for named entity tags, for more chunking you can also consider 'NN' for Nouns. Dependency on nltk python library.Output as entity in set "NAMES".

Now we have NLTK Rest server running we can integrate a progarm in python or java to talk to the using a http client to talk to the nltk server. Refer the NLTKRest.java file for explicit use of NLTKNERRecognizer.java . It returns the list of NER entities.

b. CoreNLP.java integrates the tika with the CoreNLPRecognizer. We have given a facility for you to give the path to your classifier as required in the constructor to take 3,5 or all 7 (Refer folder classifier) entities - [LOCATION, ORGANIZATION, DATE, MONEY, PERSON, PERCENT, TIME]

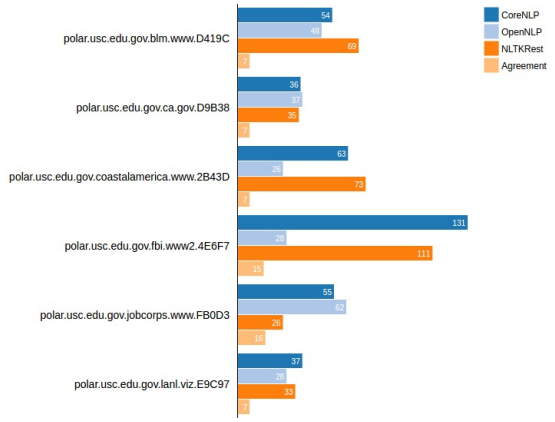
c. OpenNLP.java integrates the tika with the OpenNLPRecognizer. We have given a facility for you to give the path of your MODEL_DIR as required in the constructor. There are mainly 7 (Refer folder models) [LOCATION, ORGANIZATION, DATE, MONEY, PERSON, PERCENT, TIME] models, also available in language like es, fr etc . It uses the opennlp-tools.jar which are bundled inthe tika.

d. Grobid quantity extrcats the measurements used in the content as a special measurement NER, which are very useful in extracting the range,max and min spectrums.

e. CompositeNERParser lets you combine and analyze the NER extrcation from files, and also gives a join maximal agreement between the three techniques mentioned above. These techniques differ in their manner of tokenization and chunking and has some overall between the extracted named entities. It generates the intersection and union between all NER techniques.

f. The program in e generates the JointAgreement.json for non empty sets and can be used to compare how each NER performed. The d3 gives a powerful mean to compare these side by type with detailed int=formation.

g. We found that the joint agreement to be very useful but not necessary to be sufficient in most of the cases.Hence, shoving all the 3 NER or superset of them is very useful in enriching the



metadata. You can use the SOLRAddCompositeNER.py to post the requests to the running SOLR over http connection. You can exploit this module to mould your decision to post your metadata for documents seemlessle identified with DOI as id.

Sample Output:-

"DOI":"polar.usc.edu.eu.axes-project.www.4ADBF3E888B77791C8508DA6997695038994AAB8043EF0114C9BE953410BF93C"},
{"Agreement":[Results Partners Contact, Berlin, Marc Berenguer, Yuri Borisov]}

7. Grobid Quantities Measurements

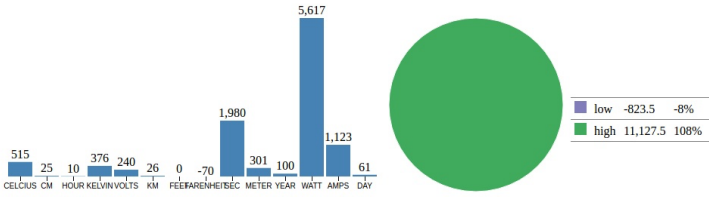
The Grobid quantities finds the measurements and the units present in each file. It normally processes the xml, html and pdf file. But as we know that tika converts each file contents into XHTML format so we can use it for every type of files. The measurements are categorized and for each type a min and max value is found. So basically it finds the range of values that can be found for a type of measurement for the polar data set. A dashboard D3 visualization was used to analyze the data collected from grobid quantities. Each type of measurement collected from grobid is represented by a bar graph which shows the sum of the min and max value and once you click on it it shows the minimum and maximum value on a separate table. The algorithm was run for the following categories:

a. Compute for all measurements of a particular type

b. Compute for all pages from a particular domain

c. Compute for all files of a particular MIME type

From the various types of measurements we were able to discern the range of values present in the polar data set and the units of measurements present.



Sample Output:-

[{"measures": [2.0, 2.0], "title": "WEEK"}, {"measures": [-40.0, 8.0], "title": "CELSIUS"}, {"measures": [18.0, 26.0], "title": "CM"}, {"measures": [1.0, 720.0], "title": "HOUR"}, {"measures": [30.0, 45.0], "title": "MIN"}, {"measures": [12.0, 12.0], "title": "MM"}, {"measures": [1.0, 500.0], "title": "KELVIN"}, {"measures": [1.0, 5794.0], "title": "METER"}, {"measures": [2.0, 15.0], "title": "MONTH"}, {"measures": [0.0, 500.0], "title": "FEET"}, {"measures": [-15.0, 75.0], "title": "FAHRENHEIT"}, {"measures": [13.0, 1000.0], "title": "MILE"}, {"measures": [5.0, 1990.0], "title": "SEC"}, {"measures": [1.0, 5000.0], "title": "KM"}, {"measures": [0.0, 200.0], "title": "YEAR"}, {"measures": [27.0, 222.0], "title": "WATT"}, {"measures": [-1.0, 2010.0], "title": "AMPS"}, {"measures": [0.0, 1014.0], "title": "VOLTS"}, {"measures": [6.0, 10.0], "title": "DAY"}, {"measures": [20.0, 2004.0], "title": "GRAM"}]

8. Conclusion

After an extensive study of the various evaluation techniques for the content extracted we found that most of the data collected from polar data set had variety of mime types and some of which were more prevalent than others. We also discovered the effectiveness of tika parsers and the language identification library. We also enriched the Solr index by including new named entities collected as a part of this assignment. The new tool to use as a part of this assignment was grobid quantities and it had some flaws.

9. Q&A

b) Is your MIME detection good? Define "good".

The mime detection used by tika is quite impressive and appears to correctly identify most of the files. The main issue with the polar data set was that most of the files were either corrupted or were broken. This is why tika identifies them as octet stream which is the default type if the subtype for a file is not identified. It is also good because once we analyzed the octet stream files in the first assignment and enriched the mimetype.xml file the accuracy of tika further improved which shows that Tika does a fine job in identifying various mime type in very less time. The tika batch module helps us to process multiple files one after the other which makes big data analysis an easier process.

c) Are your parsers extracting the right text? Define "right".

For most of the files that are not corrupted or broken tika uses the correct parser to collect the content of the files. The parser chain calls extracted the right amount of data and metadata. We can say that it extracted the right amount of data because for each mime-type we took a ratio of data collected to the total amount of data for that mime-type and it was nearer to 1. This means that all the data available to that parser was extracted.

d) Are we selecting the right parser? Define "right".

The parser selection is correct in most of the cases. We have used most in the above cases because for the broken files since the parser chain failed there was no content collected. But for the rest of cases where there was no issue with the input files tika's default parser did a

great job in calling the right parser chain. It called a variety of parsers like direct call to the correct parser or call to a composite parser which in turn called other parser or called an external parser.

e) Is your Metadata appropriate? What's missing? You can use your Metadata score generated from assignment #2 here, and also your results from this assignment.

The metadata collected as a part of the final assignment was quite appropriate and it increased the efficiency of Solr search. In the second assignment we collected some metadata using Open NLP as NER tool. But as a part of this assignment we used many other NER tools and found a maximal joint which helped us in increasing the quality of metadata collected for each file as well as enrich the Solr index. After the metadata file was enriched we ran the same algorithm used in assignment 2 for metadata quality score calculation and found an increase in value of the quality for most of the file. So basically in the 2nd assignment some of the entities from the maximal joint were missing.

f) How well is my language detection performing? Comment based on the diversity of the languages derived in this assignment. Are there mixed languages? Did it affect your accuracy?

The language detection of Tika is working just fine. It fails when the file is broken or corrupted and when there are multiple language it identifies only one of the language. Ya there were some files with multiple languages. I wouldn't say that it affected the accuracy because the files with multiple languages were less and the files with single language were perfectly identified. The language detection of tika did an impressive job on the files of polar data set. It fails for corrupt files because the Tika parser is not able to extract any content for the corrupt files and the tika language detector wants content of files to detect the language for the file. So basically with proper content in file it was able to identify the correct language.

g) Do your Named Entities make sense?

The named entities collected made perfect sense. We have prior knowledge that the data we are working on is the polar data set and were expecting the NER tools to collect data related to climate change, various locations, dates, names of organization and other such things. All the 4 NER tools used as a part of this assignment were able to categorize data to one of the above categories. For example:

"Daniel Sempere-Torres Radar QPE, Yuri Pavlyukov, Central Aerological Observatory,, November 22nd, Daniel Michelson, SMHI OPERA, BALTRAD+, Berlin, Australian Weather and Climate Research Operational Russian Weather Radars"

The above data clearly shows that the data collected was from polar data set and it made perfect sense when we built the word cloud from these data. We noticed that certain keywords like Ozone, Heat, National, NASA, Climate and Minerals are very common in our dataset. This is not surprising as our data is from the TREC Polar dataset.

Google Drive Link For Code and D3 Visualizations ->

https://drive.google.com/open?id=0ByOclo6bVRsEbVNMREFkWVlGVVE

GitHub Link for the repository->

https://github.com/anirbanmishra/Content_Evaluation

D3 Visualization GitHub Pull Request for Question 11->

https://github.com/USCDataScience/polar.usc.edu/pull/136

Grobid Quantities Pull Request GitHub for Extra Credit->

As a part of this extra credit we know that Grobid only took content as input to the curl command. We used the Tika parser to first convert the file into XHTML and fed it to the curl command.

https://github.com/apache/tika/pull/113

Grobid Quantities Documentation Pull Request for Extra Credit->

For the time being this can be a document correction because Tika python works only with Java 1.7 and Grobid quantities works only with Java 1.8 as a result both cannot be combined to achieve good result.

https://github.com/chrismattmann/tika-python/issues/101

Youtube Video Link->

https://youtu.be/wy_KOyjugTM