Scientific Content Enrichment in the Text Retrieval Conference (TREC) Polar Dynamic Domain Dataset

Team 20

Shashank Vernekar – 1698224250 Rashmi Nalwad – 2021179341
Aditya Ramachandra Desai – 5246496115
MS Computer Science Spring 2016
Viterbi School of Engineering
University of Southern California
Los Angeles, California USA

3. Tag Ratio:

A method to extract content text from diverse webpages by using the HTML document's tag ratios. We describe how to compute tag ratios on a line-by-line basis and then cluster the resulting histogram into content and non-content areas. Tag ratios can be computed by looking at the ratio of the number of HTML tags in a line of code to the number of non-HTML-tag characters

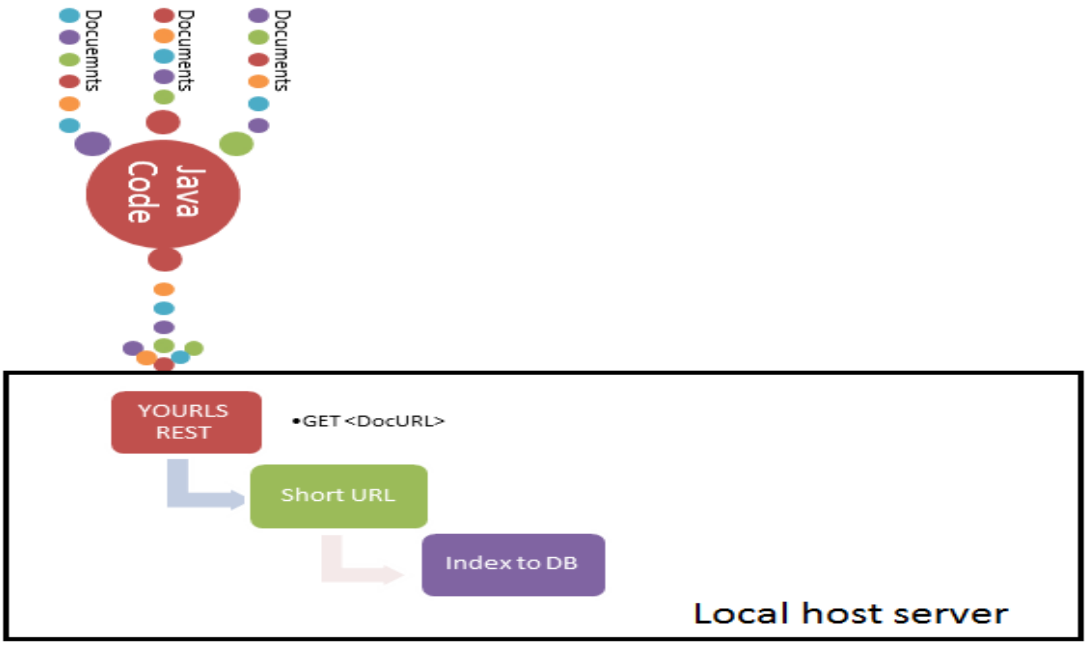
Algorithm implemented to compute the Text-Tag Ratio:

1. Read each data file from the TREC Polar Dataset.
2. Using TIKA's Auto detect Parser to convert the file types identified into XHTML.
3. Use the generated XHTML from the TIKA's Auto detect parser and parse the file line by line.
4. For each line do the following
    - Look the matching '<' tag and which indicates start of the HTML tag and increment the tag count by 1. Then look for the matching '>' tag which indicated the tag opened had ended.
    - Anything identified apart from the above step will be text so text count will be incremented by 1
    - For each line text to tag ratio will be calculated and the result will be stored into the Map<line_number, text-tag-ratio>
5. The Tag ratio thus obtained for each line will now be smoothened using Gaussian smoothing so as to have better accuracy and clear visualization for the content section within the file.
6. The Smoothened Tag ratio values for each file are now given as input to TIKA NER Regex Parser to identify the measurements (Temperature in Degree and Kelvin, Pressure, Miles, Kilogram, Kilometer, Feet, Nanometers, Money, Meters, Inch, Ph No etc) by writing the regular expressions into the ner-regex.txt file which is given in LISTING 3b and running the RegexNERecogniser via the TIKA app.

The code to perform the Tag Ratio and identify the measurements across the Polar dataset can be obtained at JAVA Code LISTING3a

4. DOI Generation

Digital Object Identifier (DOI) is a serial code used to uniquely identify objects. The DOI system is particularly used for electronic documents such as journal articles. The DOI system began in 2000[1]. These DOI will help us in maintaining the online documents without bothering about the change in metadata and/or content. In this project the DOI is generated with the help of YOURLs library. The local host server that has the YOURLs library configured and the database called SURLs created using MySQL. The following diagram depicts the how the DOI is generated in the project



DOI generation procedure:

- Initially, the YOURLS library is configured on the server to generate the short URL with the keyword polar.usc.edu. We also edited the host.txt to accommodate http://polar.usc.edu/<shortURL>
- The absolute path of a document in the given polar data set is fetched.
- The LISTING4a is the JAVA code that fetches this absolute path and initiates the REST API call to the PHP code listed in LISTING4b.
- With the help of LISTING4b the PHP code on the server will call the YOURLS library to generate the short URL.
- The generated short URL is indexed to the database for future use.
- Following is the sample for REST request and response to generate the short URL

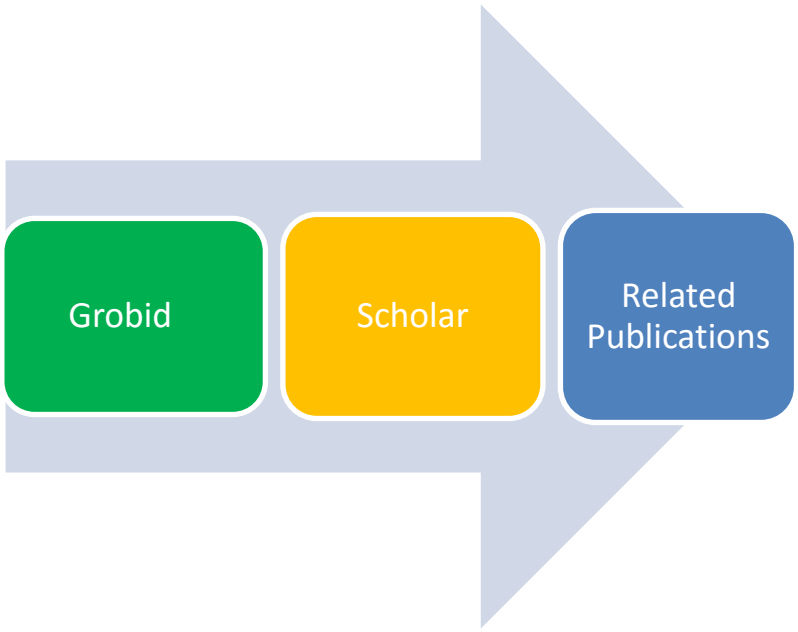

```
E:\myDrive
0000A126198BF1B241733526392CDB9BDA146DD2218521A4AE828D0DBC79C8410000
A126198BF1B241733526392CDB9BDA146DD2218521A4AE828D0DBC79C841
{"url":
{"keyword":"polar.usc.edu\/2Yj",
"url":"http:\/\/E:%5CSampleData%5CdataCLEAN4kNASA%5C0000A126198BF1B2417335
26392CDB9BDA146DD2218521A4AE828D0DBC79C841",
"title":"0000A126198BF1B241733526392CDB9BDA146DD2218521A4AE828D0DBC79C8
41",
"date":"2016-04-03 18:12:02",
"ip":"127.0.0.1"},
"status":"success",
"message":"http:\/\/E:%5CSampleData%5CdataCLEAN4kNASA%5C0000A126198B[...]
added to database",
"title":"0000A126198BF1B241733526392CDB9BDA146DD2218521A4AE828D0DBC79C8
41",
"shorturl":"http:\/\/localhost\/YOURLS\/polar.usc.edu\/2Yj",
"statusCode":200}
```


- Following is the configuration file snippet that highlights the usage of http://polar.usc.edu in the short URL. The complete YOURLs library with the configuration is provided at LISTING4b.

```
//Providing the custom prefix for the short URLs
$keyword = 'polar.usc.edu/' . $keyword;
yourls_do_action( 'add_new_link_custom_keyword', $url, $keyword, $title );
$keyword = yourls_escape( yourls_sanitize_string( $keyword ) );
$keyword = yourls_apply_filter( 'custom_keyword', $keyword, $url, $title );
        if ( !yourls_keyword_is_free( $keyword ) ) {
        // This shorturl either reserved or taken already
        $return['status']  = 'fail';
        $return['code']    = 'error:keyword';
        $return['message'] = yourls_s( 'Short URL %s already exists in database or is reserved', $keyword );
        }
```

- Once we have the short URL generated, we have retrieved the short URLs and embedded in the enriched metadata JSON.

5. GROBID



- The Grobid Journal Parser uses the GROBID (or Grobid) GeneRation Of BIbliographic Data machine learning framework to parse PDF files and to extract information such as title, abstract, authors, affiliations, keywords, etc. from journal publications. The process flow we have followed to extract the required author information is as shown in the above process diagram.

- We made use of TIKA Parser to detect and separate all application/pdf files from Polar dataset using the JAVA code LISTING5. In order to extract the TEI annotations from the scientific publications from the polar data set we have git clone the Grobid repository as mentioned in the README_GROBID.txt

- Tika Parser uses Google Scholar API to extract Related Publications based on author name and title name. Run "python D:\\CSCI599\\scholar.py-master\\scholar.py-master\\scholar.py –c 30 -author author" command to extract related publications of author.

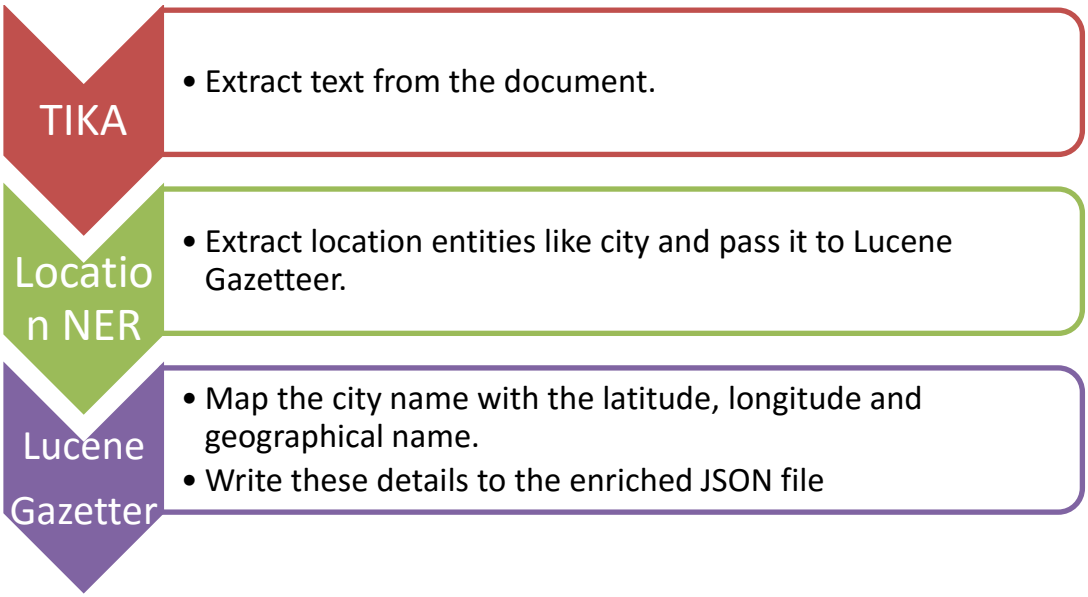
Further details are mentioned in README_SCHOLAR.txt, LISTING53c.

- Tika is used to parse the URL which is extracted from the related publications, download the document, extract and display the author and other affiliation information from the related Publications. LISTING5d.

6. GeoTopic Parser

The GeoTopicParser combines a Gazetteer (a lookup dictionary of names/places to latitudes, longitudes) and a Named Entity Recognition (NER) modelling technique that identifies names and places in text to provide a way to geo tag documents and text i.e., to identify places in the text, and then to look up the latitude/longitude pairs for those places. Steps followed for the GeoTopic Parser is provided in the README_GEOPARSER.txt

Following is the pipeline mechanism we have followed to get the latitude and longitude in the document text. The program depicting the flow is provided using JAVA code listed in LISTING6



7. Sweet Ontologies

NASA's Semantic Web for Earth and Environmental Terminology (SWEET) is developed as a comprehensive upper level ontology of Earth system science concepts that enables scalable classification of Earth science and associated data concepts. SWEET is a collection of ontologies covering a wide range of concepts and relations among them in the domain of Earth and environment SWEET is widely used by project managers and knowledge engineers who need to make use of ontologies in their applications. Moreover, SWEET is commonly reused by ontology developers and has been extended to support low level specific domain ontologies.

In this project we concentrated on the Realm Super Class of the SWEET ontology. To find out the overlapping of SWEET ontologies on our polar dataset the following algorithm was used

Algorithm

1.The OWL files were parsed using Tika to identify the ontologies to be mapped with the TREC polar dataset
2.All the identified ontologies were stored in a HashMap<Ontology,Count>
3. Initially all the unique ontologies added were having count as 1.
4. The count values incremented are incremented as and when we find a match with the the data in the dataset. This count is being used to map the percentage overlap of the SWEET ontology on the polar dataset.
5.Each of the files and read from the polar dataset and matched with each of the SWEET ontology identified in the previous step and count

Finally, D3 visualization is plotted to represent the overlapping percentage of the owl ontologies on our Polar dataset. The code to perform the parsing is provided in LISTING 7a.

8. Metadata Quality

We have enriched the metadata of the documents in the Polar Data Set. We have extracted the various measurements, phone numbers, currencies, geographical coordinates, authors, related publications, citations etc. present in the Polar Data Set. These documents metadata is carefully studied and an attempt is made to quantify the metadata with the help of the constant scores.

There are 3 metadata type viz. structural metadata, administrative metadata and descriptive metadata. We have made an attempt to consider these metadata types and assign the scores. The base metadata model we have considered is Dublin Core.

Before proceeding further, we have evaluated various tools that can extract metadata and help us in quantifying the metadata. We have studied the Metadata Extraction tool like Metadata Extraction tool of the National Library of NZ.

Metadata Extraction tool of the National Library of the New Zealand [4]. This tool is licensed under Apache and is capable of handling both individual files and also batch of files. When we tested on various types of files we were able to get less number of metadata for the same file using this tool when compared with Apache TIKA. TIKA extracts all possible valid metadata from the give file. The screenshot comparing the metadata for the same JPEG file using TIKA and Metadata Extraction tool is provided at SCREEN 8.

Using Apache TIKA we have got the extracted metadata and also the enriched JSON data for the individual files. We have the following algorithm that will help us in having the constant score that can be used as the measure to know how good the metadata of the document is. A brief description is provided here followed by the complete algorithm

- The algorithm assigns the score for each metadata value, a score out of 100 points for any given document. The algorithm assigns high value scores for descriptive metadata and mix of high-mid scores for structural and administrative metadata. The other metadata like *digital zoom ratio, camera serial number* which are very specific for JPEG have lower scores.
- For instance the metadata '*rights-90*' assigns score of 90 for rights metadata in Dublin core. *keywords-97* assigns the score 97 for PDF documents, which follows the content specific model for the metadata.
- Similarly *revision-number, preservation-format, title, content-type, page-count* (for OfficeOpen metadata model) have high scores. These scores are on higher end as they are important in document preservation for longer time and comes under the administrative metadata.

- Since the metadata score for a file is for 100 points, each point being allocated for specific metadata, there is enough scope to add new scores for any metadata that comes up in coming days. This is very active area of research and we have tried to accommodate this in our algorithm

Algorithmic approach to evaluate the metadata for the documents

```
QUALITY ANALYSIS (DataSet DS)
{
 /*
Provide the pre-defined metadata quality score considering the importance of descriptive metadata, structural metadata and administrative metadata higher values and custom specified metadata values lower scores.
*/
   for(each document di in the dataset DS)
   {
       MIME_TYPE= Use Apache TIKA to identify the MIME type for di;
       META_DATA= Metadata extracted using TIKA;
       function calculateMetaDataScore(META_DATA for di)
  }
function calculateMetaDataScore(META_DATA for di)
{
   Calculate the metadata scores using content specific metadata model
   //comparePDF(), compareGIF(), compareJPEG() etc.
 If no specific metadata model can be specified, fall back on Dublin Core metadata model and calculate the metadata score;
 Write output to the file <FileName, Score>;
 function calculateStatistics();
}
function calculateStatistics()
{Calculate the Standard Deviation, Average, High and Low scores from the output file to plot on D3 visualization.}
}//END QUALITY ANALYSIS
```

Calculating the metadata scores is one of the active areas of the research and challenging. Here are some of the metadata quality observations we have made in the polar data set.

- There are some metadata tags in HTML like <rel=" license"> which will help in identifying if the document has the license information or not.
- Text file metadata is always complete, no fields are left empty. This brings us to assumption that this is most widely used format in preferable format to store.
- Interoperability index for JPEG helps to know how well the file can be used in different environment. The score we have provided for this index is 90. Other metadata qualities in this same interoperability are *xmp:creatortool* and *producer, creator.*
- We have created the DOI using YOURLS and for some of the exceptional files that cannot be opened or parsed or mal-formatted or YOURLS could not generate DOI for them. In our Polar Data Set we have around nearly 0.5-1% for which DOI is not generated.
- Long term management terms like *preservation-format, aliases, date and last-modified* etc. have been identified and given a suitable constant score in our JAVA code LISTING8.
- Metadata like *resourceName, author, content-type and title* are seen in majority of the documents.

9. Apache Solr

- Apache Solr 4.10.4 is chosen as the indexing technology for indexing the metadata extracted from the parsers.

- SolrDataFeeder.java indexes the extracted data from the parsers into Apache Solr
- Schema.xml includes 14 fields used for indexing extracted data from parsers into Apache Solr

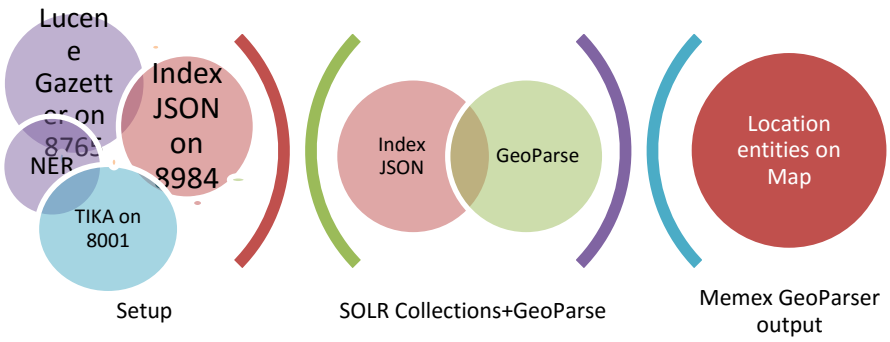
```
<field name="id" type="string" indexed="true" stored="true" required="true" multiValued="false" />

<field name="Title" type="text_general" indexed="true" stored="true" multiValued="true"/>

<field name="Author" type="string" indexed="true" stored="true"/>

<field name="Related_Publications" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="Year" type="int" indexed="true" stored="true"/>

<field name="DOI" type="string" indexed="true" stored="true"/>

<field name="Geographic_LATITUDE" type="float" indexed="true" stored="true"/>

<field name="Geographic_LONGITUDE" type="float" indexed="true" stored="true"/>

<field name="Geographic_NAME" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="Geographic_ALTITUDE" type="float" indexed="true" stored="true"/>

<field name="NER_PHONE_NUMBER" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="NER_MEASUREMENT_MONEY" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="NER_MEASUREMENT_TEMP" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="NER_MEASUREMENT_PRESSURE" type="string" indexed="true" stored="true" multiValued="true"/>

<field name="SWEET" type="string" indexed="true" stored="true" multiValued="true"/>
```

Id field acts as primary key and holds unique keyword generated in step 4 (Short URL generation for each document). Author, title, Geographic name fields are used for searching in the indexed data. Geographic latitude, Geographic longitude, Related_Publication, SWEET and other fields are added for D3 visualisation in step 12. LISTING9.

10. MEMEX Geo Parser

MEMEX GeoParser, extracts the geographic coordinates from any text and visualize the location on the global map. This parser scans the text or any given document, forms the string and passes to the GeoParser which is embedded in the MEMEX parser as well. The location coordinates provided by GeoParser are indexed in Apache SOLR 5.3 which comes with MEMEX Geo Parser. The screen shot showing the global map with all location is available at SCREEN10

For this particular step, we are considering the already indexed SOLR documents. The URL of already indexed SOLR collection is provided as the input to MEMEX GeoParser. The detailed steps are illustrated in the following process diagram as follows. README_MEMEX_GeoParese.txt has the setup details



11. Tika Similarity

- Measurements such as NER_MEASUREMENT_TEMP are extracted from files and clustered such that the data with same units falls in to one cluster. For ex. Degree and Kelvin are two units of temperature data extracted from Polar Data set. Measurement_similarity.py clusters all files with degree measurements and kelvin measurements into 2 different clusters. K means clustering technique based on edit distance is used to cluster Polar data as algorithm classifies documents into K clusters based on the units of temperature measurement. Edit_cosine_cluster.py is used for producing clusters.json for output D3 visualisation. LISTING11a
- Authors and related publication information extracted from files are clustered such that data with same author names fall in to one cluster. K means clustering technique based on edit distance is used to cluster Polar data as algorithm classifies documents into K clusters based on the name of the Author of the publication. Edit_cosine_cluster.py is used for producing clusters.json for output D3 visualisation. LISTING11b.
- Geographic information such Geographic names are extracted from files and clustered such that the data with same location name falls in to one cluster. K means clustering technique based on edit distance is used to cluster Polar data as algorithm classifies documents into K clusters based on the name of the geographical location. Edit_cosine_cluster.py is used for producing clusters.json for output D3 visualisation. LISTING11c.
- SWEET features with same ontology information are grouped into one cluster. K means clustering technique based on edit distance is used to cluster Polar data as algorithm classifies documents into K clusters based on the sweet features. Edit_cosine_cluster.py is used for producing clusters.json for output D3 visualisation. LISTING11d.
- PySolr 3.1.0 is a light weight wrapper for Apache Solr. It provides an interface that queries the server and returns results based on the query. SolrPython.py uses PySolr library to connect to Apache Solr and query all the results which will be then fed to clustering algorithm edit value similarity.py for producing similarity scores. K means clustering technique with edit distance metric is chosen over hierarchical clustering technique with

Jaccard distance metric, because K means algorithm is used for clustering data into K separate clusters i.e. similar item falls in same clusters. This algorithm was more appropriate as we are trying to group data into different clusters based on the extracted metadata. Hierarchical clustering is not very appropriate in this use case, as it produces a sequence of clustering's in which each clustering is nested into the next clustering in the sequence. And also since hierarchical clustering is a greedy search algorithm based on a local search, the merging decision made early in the agglomerative process are not necessarily nice ones. LISTING11e.

- Existing edit_cosine_cluster.py had an issue in the clustering the files as it was trying to cluster based on file names rather than on similarity scores. Issue #66 was reported and fixed; this updated code performs clustering based on similarity scores.

12. D3 Visualisations

6 different D3 visualisations that connect to Apache Solr can be found at http://shashankjivanvernekar.github.io/ and provided the Pull Request for polar.usc.edu.

13. Search Implementation of GeoParser

- Search implementation for the MEMEX GeoParser is implemented using the AJAX call. The functionality allows us to search the locations, authors, their related publications and year of the publications. They response received from the SOLR is JSON and can be passed to any GUI object in future.
- The SCREEN13 has the screen shots describing the results and the JavaScript with search functionality is available at LISTING13.

14. TIKA OCR

We choose to run TIKAOCR on the Polar Data Set. The rationale behind this was that we came across many images in the data set. We believed that TIKA OCR will help us in extracting the required information in the form of the text.
Brief over view of TIKA OCR

- TIKA uses Tesseract OCR parser to extract the text from the images. Tesseract OCR is the open source C++ bases OCR Engine. This works on most of the file types of images and in many possible languages.
- Tesseract OCR is installed on the Linux machine by accessing the Linux package Tesseract.
- Integrating with TIKA
  We can integrate Tesseract with TIKA by the following command
  tika -t /path/to/tiff/file.tiff
- We have provided a Bash script in LISTING14 that will run the Tesseract with TIKA and provide the output in the specified file.
- Sample Output of Tesseract using TIKA on the Polar Data Set is provided in 14_TIKA_OCR and SCREEN14.

Observations made during the project

1. Geographical locations, author names, related publications, NER measurements and SWEET ontologies were the features that we found very relevant and important in the Polar Data Set.

2. The tag ratio algorithm helped us to isolate the measurement data in the polar data set. The algorithm we implemented as TIKA parser helped in parsing higher text density region in the documents. This helped in achieving faster text analysis for given documents.
3. NER was not able to identify the SWEET terminologies as expected. So we come up with the TIKA parser which will parse the OWL file using TIKA to identify the SWEET ontologies in the polar data set. The visualization with respect to REALM ontology is also provided.
4. The D3 visualizations extensively helped us in knowing the polar data set. We were able to know the percentage of SWEET ontologies in the polar data set, metadata quality of the data set, we were able to plot the geographical locations present in the dataset. Also, we were able to cluster the authors and related publications, understand the year of publications present in the polar dataset.
5. Metadata Quality evaluation is very challenging and using Apache TIKA will make the work a lot easier when compared to other metadata extraction tool. We have made an attempt to provide a constant score based considering the administrative, structural and descriptive metadata. A further scope in this field would be to have a large training set that can help in document preservation.
6. Extracting the geo coordinates using Text Extraction is made possible by Apache TIKA and Lucene Gazetteer. Without the gazetteer, it would have been the exponential time task to map the location and the respective geo coordinates.
7. Apache TIKA OCR output is really very consistence with input image and can be of great help in designing any product for visual aid. This could be the best possible future work one can proceed in the field of virtual reality.
8. Particular features that we extracted from Polar data set such geo-locations, author names did help in producing more effective clusters as we tried clustering them.
9. K means clustering technique with edit distance metric is chosen over hierarchical clustering technique with Jaccard distance metric, because K means algorithm is used for clustering data into K separate clusters i.e. similar item falls in same clusters. This algorithm was more appropriate as we are trying to group data into different clusters based on the extracted metadata. Hierarchical clustering is not very appropriate in this use case, as it produces a sequence of clustering's in which each clustering is nested into the next clustering in the sequence. And also since hierarchical clustering is a greedy search algorithm based on a local search, the merging decision made early in the agglomerative process are not necessarily nice ones.

References

[1] https://en.wikipedia.org/wiki/Digital_object_identifier
[2] http://www.joyofdata.de/blog/comparison-of-string-distance-algorithms/
[3] http://www.ijarcsse.com/docs/papers/Volume_3/7_July2013/V3I7-0565.pdf
[4] http://www.dcc.ac.uk/resources/external/nlnz-metadata-extraction-tool