**University of Southern California**

Viterbi School of Engineering

CSCI 599: Content Detection and Analysis for Big Data

**Instructor**:Dr. Chris Mattmann

Assignment 2: Scientific Content Enrichment in the Text Retrieval Conference (TREC) Polar

Dynamic Domain Dataset

**TEAM 22**

**GitHub repository**: *http://www.github.com/harshfatepuria/ Scientific-Content-Enrichment-in-the-Text-Retrieval-Conference-TREC-Polar-Dynamic-Domain-Dataset*

**Github.io website**: [*http://harshfatepuria.github.io*](http://harshfatepuria.github.io)

*(All the visualizations with interactive capabilities available on this website)*

Date: 04/03/2016

Report submitted by:

Harsh Fatepuria, *fatepuri@usc.edu*

Warut Roadrungwasinkul, *roadrung@usc.edu*

Rahul Agrawal, *rahulagr@usc.edu*

(Graduate Students, Department of Computer Science)

1. **Common Codes for all the following Parsers**
2. **TikaExtractedTextBasedParser**: An abstract class extended from Tika AbstractParser. Parsers that perform on text extracted from documents using Tika should extend this class. This class also provide a method to get extracted text from a document. Most of the parsers created in this assignment extend this class.
3. **AbstractParserRunner**: A utility class to run a specific parser using all the documents in a specified folder. It can be configured through different parameters. Start parsing by invoking runParser() method.

|  |  |
| --- | --- |
| baseFolder | base folder of documents to be parsed |
| resultFolder | base folder to keep result files |
| markerFile | marker, for skipping the files that are already been parsed |
| overwriteResult | set to overwrite existing results (default : false) |
| documentsInCborFormat | setting that tell whether documents to be parsed are in CBOR Format, for parsing data in common-crawl (default : false). CBOR support can be used separately using cbor.CborReader |

1. **Tag Ratio Tika Parser and Measurement Extractor**
2. Apache Tika was to convert files in the dataset into an intermediate XHTML file. Then, the Text-To-Tag ratio for the file was calculated using the formula: **No of text characters in the line/ No of tags in a line**. The overall TTR is the average of the TTR for individual lines in a file. The irrelevant text where the TTR is lesser than the average TTR ratio is discarded. Only the remaining text is chosen for next stage (measurement analysis).
3. The code to perform this is in the class TTR.TTRAnalysis. It is also wrapped as a Tika parser, measurement.TagRatioParser, and can be used separately.
4. Next, a wrapper class was created (wrapped in measurement.MeasurementParser), which utilized this parser’s text extraction capabilities to extract measurement based data from the files. We developed an algorithm for extracting the measurements, which is outlined below:
5. Extract text using Text-to-Tag ratio analysis
6. Tokenize text using Stanford CoreNLP DocumentPreprocessor
7. Extract number using Stanford CoreNLP NumberNormalizer
8. Concatenate consecutive text tokens that can be normalized to number, these tokens normally are part of the same number so they should be treat as a whole, reevaluate the text.
9. Extract numbers and their 2 following tokens
10. For each extracted 3-gram, match tokens with Unit concepts defined in SWEET Ontology (reprSciUnits.owl) and some predefined symbols. The two tokens will be match separately and combined using both exact and fuzzy string matching. Concepts that are within tolerance will be considered a measurement
11. Output measurement (unit) and its number (magnitude).

Algorithm 1: Extract Measurements from documents

1. To run this across the dataset, initialized measurement.MeasurementParserRunner and provide its parameters then invoke runParser() method or invoke the main class with following parameters.

**java** main.Main -t **measurement** -b *baseFolder* -r *resultFolder* [-m *markerFile*] [-cbor]

1. Sample measurements extracted by the parser using the above algorithm are:

|  |
| --- |
|  |
| year 296930 |
| kilometre 151741 |
| degrees 135431 |
| hour 107379 |
| kelvin 79598 |
|  |

Snippet 1: A Sample of Extracted measurements with their counts

1. **URL Shortner**
2. In order to give unique short identities to each files, YOURLS (Your Own URL Shortner), an open source tool was used. Each file in the TREC Polar Dataset was mapped to a unique 8-character long alphanumeric hash. The resulting key-value pairs were stored as JSON objects containing the file path and the short URL.
3. A sample JSON containing the short URL and relative file path is shown below:

{"metadata":

{

"filePath": “org/aoncadis/www/96DEB8E3B9… CA50B668CAB77D03392AC7F4A790670706662D030",

"shortURL": "polar.usc.edu/2acb03f4"

}}

Snippet 2: Example snippet showing mapping between filePath and shortURL

1. These JSON files are dumped along with outputs from other parsers developed as a part of this project into the solr index. This helps us in identifying and retrieving a file using its shortened URL.
2. **Content Extraction and NER using Grobid Journal Parser**
3. Grobid Journal Parser in Tika was used to extract TEI annotations from the PDF files in the dataset.
4. Out of roughly forty five thousand such PDF files, majority of them were either truncated or image files stored in PDF format. Grobid Journal Parser gives exceptions on such files, and we ignore such files for the purpose of this project. Furthermore we only consider the files containing title and abstract fields in their TEI information to retrieve a rich set of documents in the following steps.
5. For each of the PDF files in the dataset, their TEI annotations are stored as JSON data.
6. Next, the scholar.py program was used to pull 20 related publications for each of the PDF files above. Since there is a limit of requests that can be made using the API, not all publications were pulled.
7. Relevant information such as Authors, Publication Year, and Affiliations were extracted and identified for each of the new papers/ journals.
8. NOTE: Fields used for extraction of papers- To extract the most relevant publications, we used its title and keywords from abstract. We used stop words list to ignore the irrelevant words.
9. **Geo-Topic Parsing using Tika GeoTopicParser**
10. The Tika GeoTopic Parser allows us to extract location related content from the files. A program to run this parser on the entire dataset was developed. The GeoParser class is already defined. We wrap it in our GeoWrapperParser so it can accept every type of documents that are supported by Tika. This parser will extract GeoName, Latitude and Longitude as metadata, along with optional locations.
11. To run this across the dataset, after running the Geo Gazetteer server, initialized geoparser.GeoParserRunner and provide its parameters then invoke runParser() method or invoke the main class with following parameters.

**java** main.Main -t **geo** -b *baseFolder* -r *resultFolder* [-m *markerFile*] [-cbor]

1. The results obtained were stored as JSON objects. A sample result file is shown below:

{ "metadata": {

"Geographic\_LONGITUDE": [ "-86.73611" ], "Geographic\_NAME": [ "Medium United Methodist Church" ],

"Geographic\_LATITUDE": [ "35.26341" ], filePath": [ "aero/weather/982DA6E97F400……………..……A406" ]

}}

Snippet 3: Sample result from GeoTopic Parsing

1. Later, these output files were dumped in solr to generate an inverted index.
2. **Intersecting the above results with SWEET**
3. Semantic Web for Earth and Environment Terminology (SWEET) contains 600 files with Environment related terminologies. These annotations were intersected with the results obtained from GeoTopic Parsing to extract relevant environment related data.
4. SWEET Ontologies are written in OWL format. We use **Sesame** (rdf4j.org) as an engine to store and query these ontologies. We create and store this repository within sweet.SweetOntology class. The class also provide methods for querying the concept by fuzzy string matching using Levenshtein Distance provided by Apache Commons library.
5. Wrapped in sweet.SweetParser, SWEET concepts extraction can be done by the following steps.
6. Perform Named Entity Recognition to extract entities using Stanford CoreNLPNERecogniser
7. For each extracted entities, match these entities with SWEET concepts. Select the best matched concept that the distance is within tolerance
8. To run this across the dataset, initialized sweet.SweetParserRunner and provide its parameters then invoke runParser() method or invoke the main class with following parameters.

**java** main.Main -t **sweet** -b *baseFolder* -r *resultFolder* [-m *markerFile*] [-cbor]

1. Implication: (Step V) ∩ (Step VI) gives us the extracted environment related terms form all the files in the dataset. Sample results are shown below:

http://sweet.jpl.nasa.gov/2.3/realm.owl#Earth 133751

http://sweet.jpl.nasa.gov/2.3/matrSediment.owl#Boulder 50560

http://sweet.jpl.nasa.gov/2.3/matrCompound.owl#CO 18179

http://sweet.jpl.nasa.gov/2.3/realmOceanFeature.owl#SouthernOcean 14449

Snippet 4: SWEET ontology intersected with Extracted Dataset and extracted counts

1. A brief summary of extracted metadata types on the dataset with the number of files are shown below:

|  |  |  |
| --- | --- | --- |
| **Metadata Type** | **No. of files** | **No. of files with metadata** |
| Measurement | 1360304 | 370284 |
| Geo | 641653 | 268763 |
| SWEET | 1360313 | 348741 |

1. **Metadata Quality Store**
2. **Inverted Index Generation using Apache Solr**
3. Technology Used: Apache Solr.

Reasons for choosing Solr over Elastic Search: Solr is slower, yet more powerful as compared to Elastic Search. Also,

MEMEX GeoParser is easier to connect to Solr index.

1. A program/ script to iterate over all the extracted or parsed content was developed. The program ingests the output JSON files into the index. However, the rate of ingestion is very slow.
2. The proposed schema for Solr to represent our data is present in the file Schema.xml present in the submission folder.
3. Explanation of what fields are important to visualize, and what are important to search and to find the data
4. **MEMEX GeoParser**
5. The MEMEX GeoParser is run over the index generated by us in the last step (VIII) from the GeoTopicParser. The location map hence generated is shown below:

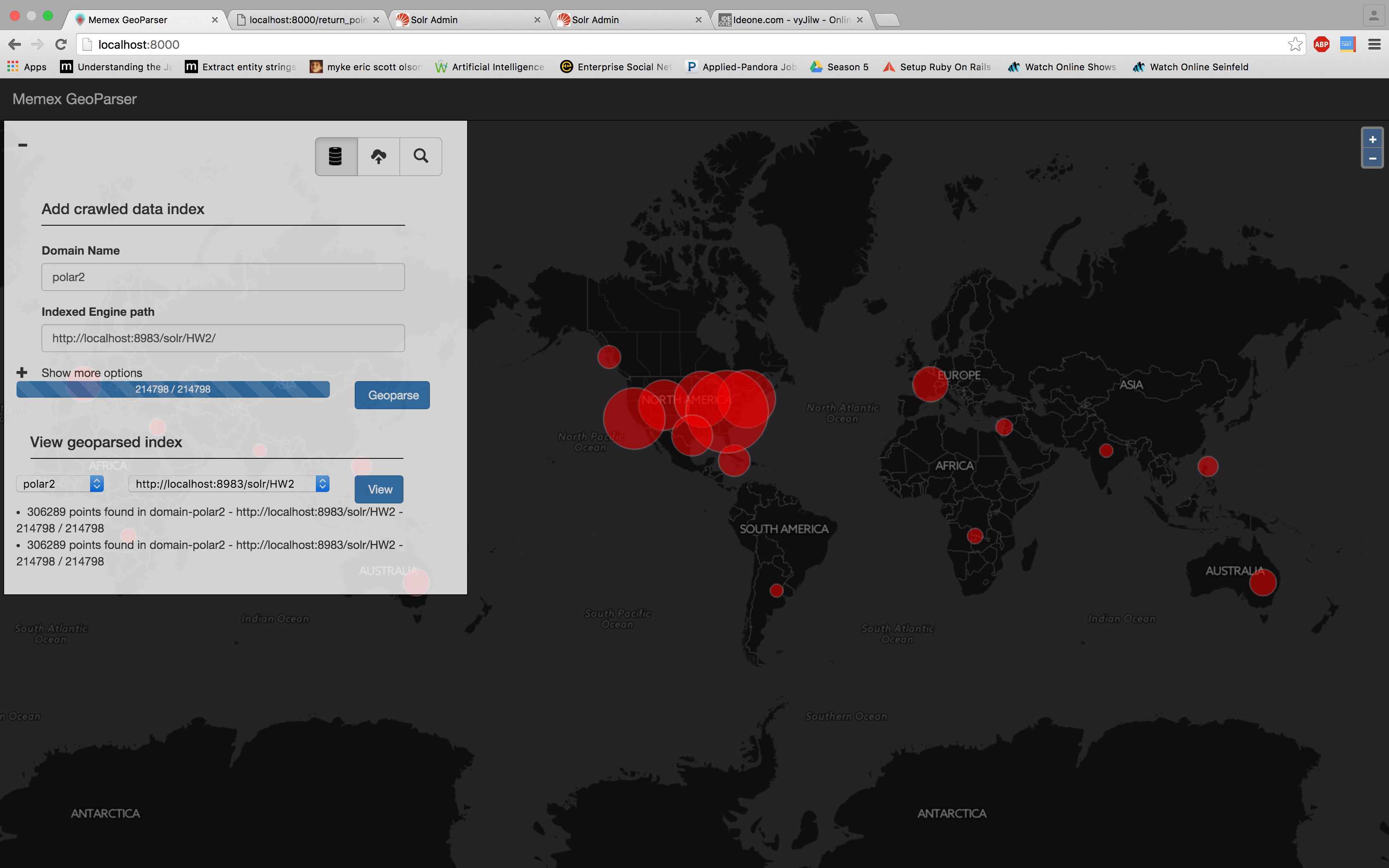


Fig 1: MEMEX GeoParser result

1. A snippet of the Solr Index containing the above data is shown below:

{

"points":["[{'loc\_name': 'RepublicofIndia', 'position': {'y': '79.0', 'x': '22.0'}}]"],

"id":"7e889d67-d25b-4e22-90cd-d47962999e7a",

"\_version\_":1530476767183110144

}

Snippet 5: Solr Index. MEMEX GeoParser used to generate the world map shown above

1. **Tika Similarity**
2. From assignment 1, we had already modified codes in tika-similarity to support clustering using cosine distance and edit distance. In this assignment, we will modify the codes to be able to represent Solr index of each extracted metadata type by extending existing Vector class and clustering them using specific distance measure.
3. For measurement extraction, we create MeasurementVector class. It will store extracted measurement units and its average magnitude. We will use Cosine Distance to cluster this.
4. For related publication and author, we create GrobidVector class. It will store only authors and related publications information. Since this data are mostly string based, we will use Edit Distance to cluster this.
5. For extracted locations, we create GeoVector class. It will store extracted latitude and longitude. Since these values are points in 2-D space, we will use Euclidean Distance to cluster this.
6. For SWEET features, we create SweetVector class. It will store set of extracted SWEET features. Since these values are parts of a finite set (all SWEET concepts), we will use Jaccard Similarity to cluster this.

We will use K-mean as a clustering algorithm. We will cluster a sample set of documents that are ingested to Solr index. The set contains documents in /gov/nasa/climate in polar-fulldump dataset which are about 1200 files but when extracted there will be around 300 files for each metadata type. For related publication and author we will use another set of documents extracted from various folder which are around 45 files.

To select number of clusters (k), we will run the clustering with number of clusters vary from 2 to 10. We then plot normalized distortion value and select number of cluster by Elbow method. Selected number of cluster for measurement, related publication and author, locations and SWEET concepts are 6, 4, 5 and 4 respectively.

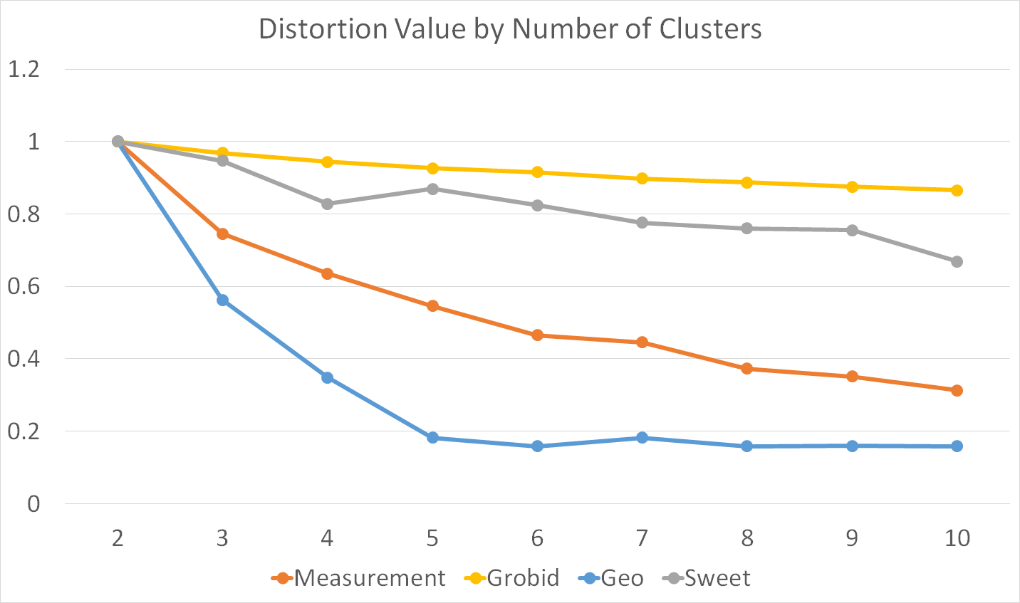


Fig 2: Tika Similarity Results- Distortion Values v/s Number of Clusters for various parsers

Clusters visualizations are located in tika-similarity/result folder. The code can be executed using the following command: python **solr\_metadata\_cluster.py** *type*(measurement|grobid|geo|sweet) *solrUrl noOfCluster*

Or, execute this command for sample results: python **solr\_metadata\_cluster.py** sample

1. **D3 visualizations**
2. Website for viewing the visualizations: [*http://harshfatepuria.github.io*](http://harshfatepuria.github.io)
3. The following D3 visualizations represent our extracted features using the parsers developed throughout the past month.
4. Fig 1, Explanation
5. Fig 2, Explanation
6. Fig 3, Explanation
7. Fig 4, Explanation
8. Fig 5, Explanation
9. Fig 6, Explanation
10. **[Extra Credit] Connecting GeoParser Application to actual data in Solr Index**
11. Created pop up in GeoParser that displays the metadata record
12. Search Box- allows both the locations and documents to be returned in a search list
13. **[Extra Credit] Feature/ Content Extractions listed on Tika wiki page**

a. Why did you chose the Content Extractions?

b. What additional knowledge did you gain from the features?

**Key Observations**:

1. What features did you find most useful in exploring the Polar data?
2. Were you able to take advantage of Tag Ratios to isolate the measurement data?
3. Did NER and SWEET terminology mapping work well – was the NER unable to identify SWEET categories and concepts?
4. Did the D3 interactive visualizations help you understand the data?
5. Were particular features that you extracted such as the geo-locations more effective in producing clusters?
6. Were particular cluster techniques e.g., k-means, more meaningful than hierarchical clustering?
7. What about distance metrics – which ones were more effective (Jaccard, Edit Distance, etc.) Why?
8. Was your metadata quality score something that you could leverage to find richly curated records and ultimately is it something that could be leveraged to point users to the more meaningful polar data?
9. Were you able to find related scientific publications, and did the authors you found both inside the dataset and using Google Scholar have a high degree of overlap with the existing Polar dataset?