

# 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>

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

Date: 04/05/2016

Report submitted by:

Harsh Fatepuria, [fatepuri@usc.edu](mailto:fatepuri@usc.edu)

Warut Roadrungrasinkul, [roadrungr@usc.edu](mailto:roadrungr@usc.edu)

Rahul Agrawal, [rahulagr@usc.edu](mailto:rahulagr@usc.edu)

(Graduate Students, Department of Computer Science)

## I. Common Codes for all the following Parsers

1. **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.
2. **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

## II. Tag Ratio Tika Parser and Measurement Extractor

1. 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).
2. 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.
3. 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:

- a. Extract text using Text-to-Tag ratio analysis
- b. Tokenize text using Stanford CoreNLP DocumentPreprocessor
- c. Extract number using Stanford CoreNLP NumberNormalizer
- d. 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.
- e. Extract numbers and their 2 following tokens
- f. 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
- g. Output measurement (unit) and its number (magnitude).

Algorithm 1: Extract Measurements from documents

4. 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]**
5. 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

### III. URL Shortner

1. 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.
2. Python Program to be executed: aURL\_shortner.py
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

4. 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.

### IV. Content Extraction and NER using Grobid Journal Parser

1. Grobid Journal Parser in Tika was used to extract TEI annotations from the PDF files in the dataset.
2. Out of roughly thirty-seven 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.
3. For each of the PDF files in the dataset, their TEI annotations are stored as JSON data. (using GetTEI.java)
4. Next, the modified version of scholar.py program (getRelatedPublications.py) 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.
5. Relevant information such as Authors, Publication Year, Affiliations, Citations, Excerpt, and Title were extracted and identified for each of the new papers/ journals. These data are embedded into the metadata of that file.
6. NOTE: Fields used for extraction of papers- To extract the most relevant publications, we used Title field and keywords from Abstract field. We used stop words list to ignore the irrelevant words.

### V. Geo-Topic Parsing using Tika GeoTopicParser

1. 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.
2. 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]
```

3. 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

4. Later, these output files were dumped in solr to generate an inverted index.

## VI. Intersecting the above results with SWEET

1. 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.
2. 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.
3. Wrapped in `sweet.SweetParser`, SWEET concepts extraction can be done by the following steps.
  - a. Perform Named Entity Recognition to extract entities using Stanford CoreNLPNERRecogniser
  - b. For each extracted entities, match these entities with SWEET concepts. Select the best matched concept that the distance is within tolerance
4. 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]
```

5. Implication: (Step V)  $\cap$  (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

6. 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
Grobid	37810 (PDF files)	1711

Table 1: Number of files used for our analysis

## VII. Metadata Quality Score

1. We used the standard Tika Parser to extract the metadata information present in the files, and checked whether it contains Title, License, Description, Version, and Alias fields.
2. We calculated the total number of metadata fields available for each MIME type in the data set.
3. We divided the number of metadata fields present in each file by the total number of metadata fields available for that MIME type to get the metadata score for every file. This data is stored in the `metadataScore.json` file.
4. Java program for Metadata Quality Score Analysis: `MetadataScore.java`

## VIII. Inverted Index Generation using Apache Solr

1. 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.
2. A program/ script to iterate over all the extracted or parsed content was developed (`createSolrIndex.py`). The program ingests the output JSON files into the index. However, the rate of ingestion is very slow.
3. The proposed schema for Solr to represent our data is present in the file `Schema.xml` present in the submission folder.
4. Important fields used in visualizations and searching relevant information: `metadata.Geographic_NAME` , `metadata.measurement_unit` , `metadata.RelatedPublications.0.author` , `metadata.filePath` , `metadata.TEI.teiHeader.fileDesc.titleStmt.title.content` , `metadata.TEI.teiHeader.fileDesc.sourceDesc.biblStruct.analytic.author.persName.forename.content` , `metadata.sweet_concept` , `metadata.shortURL`

## IX. MEMEX GeoParser

1. 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. More screen shots are available in the MEMEX GEOPARSER directory provided in the submission folder.

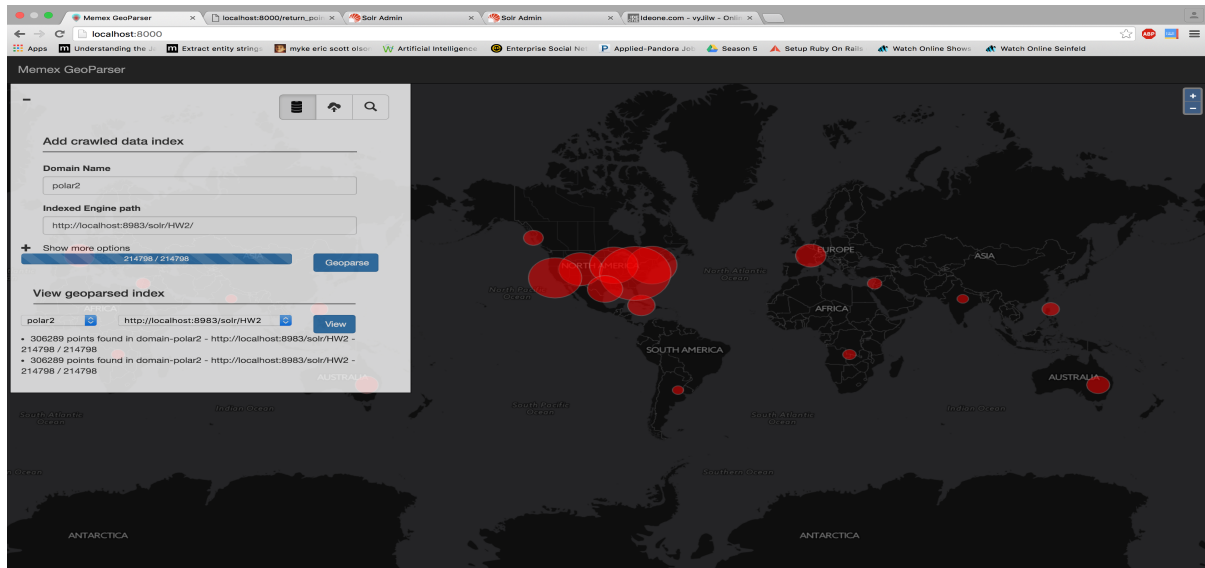


Fig 1: MEMEX GeoParser result

2. 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

## X. Tika Similarity

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.

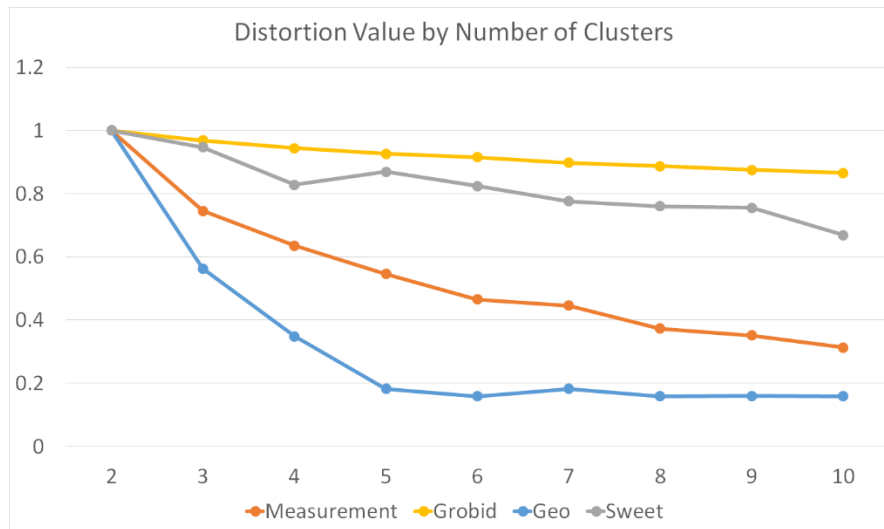


Fig 2: Tika Similarity Results- Distortion Values v/s Number of Clusters for various parsers

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`

## XI. D3 Visualizations

1. Website for viewing the visualizations: <http://harshfatepuria.github.io>
2. The following D3 visualizations represent our extracted features and some analysis on them using the parsers developed throughout the past month.

Visualization	Explanation
a. SWEET Concept	Gives a count analysis on the extracted SWEET terms from the text. Restricted to top 100 results. File Name: sweetOntologyCountBarChart.html
b. Grobid + Google Scholar	A dendogram of publications with authors and authors of related publications. Helped in finding overlap between authors in related publications. File Name: relatedPublicationsDendogram.html
c. Metadata Score Analysis	Gives a visual representation of Metadata score based on number of fields present using the Zoomable Circle Packing D3. Files are grouped by MIME type. The larger the circle, the higher the score. File Name: metadataScoreCirclePacking.html
d. Measurement Extraction	Gives a count analysis of various types of measurement terminologies extracted from the data. File Name: measurementCountBarChart.html
e. Geo Location Analysis	Gives an analysis of various locations extracted from the dataset. File Name: geoLocationCountPieChart.html
f. Geo Location Clustering	Clustering based on locations on a world map. File: geoLocationCluster.html

Table 2: List of D3 visualizations

3. These visualizations are dynamic and connect to our solr index. We have contributed our web pages as pull request to **polar.usc.edu** as well.

## XII. [Extra Credit] Connecting GeoParser Application to actual data in Solr Index

1. We connected the GeoParser application to the actual data in the solr index. Furthermore, we developed a Geo Location map which shows extracted location metadata upon hovering.

Geo Location Clustering (documents in /gov/nasa/climate/)

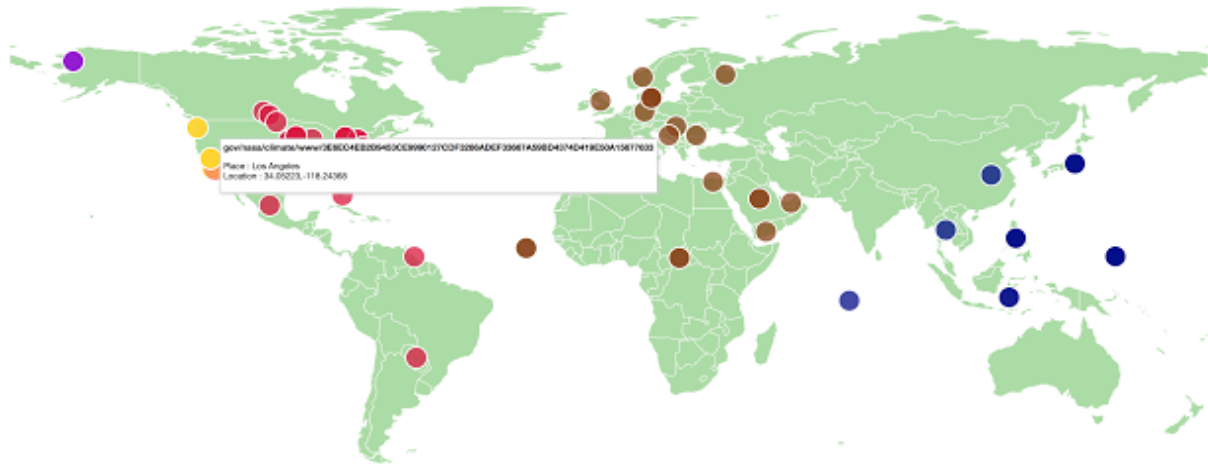


Fig 3: Metadata visualization over world map on hovering

## XIII. [Extra Credit] Feature/ Content Extraction using Tesseract OCR

1. There are many image files in the polar dataset. These files may contain valuable information in the form of embedded text. Using Tesseract OCR, Tika will be able to extract text embedded in various types of image. By installing Tesseract OCR library, Tika will use it automatically when parsing files with supported types in combination with other parsers.

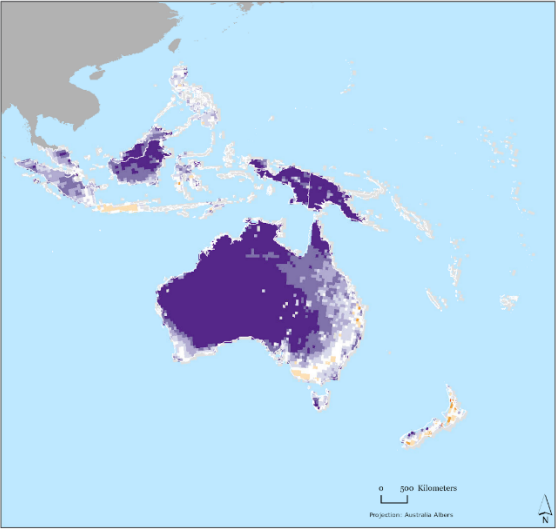
File:/edu/columbia/ciesin/sedac/47049B413D345D87D20F3C6153AD055E258177FCF01BA96812E1C1F924C889DE	Extracted Text
<p><b>Oceania Nitrogen in Manure Production</b> Global Fertilizer and Manure, Version 1</p>  <p>Amount of nitrogen in manure produced within the 0.5 degree grid cell. Grid cell values are expressed in kilograms per hectare (kg/ha) ranging from 0 to 370. The data values were derived based on the nutrient content of the manure produced by the total number of livestock located within each grid cell.</p> <p>Kg/ha of Nitrogen Manure produced per grid cell: 0 0.1-2 2.1-5 5.1-10 10.1-20 20.1-40 40.1-75 75.1-200 200.1-370</p> <p>Center for International Earth Science Information Network Copyright 2011. The Trustees of Columbia University in the City of New York Source: Potter, P., and N. Ramankutty, et al. (2010). Global Fertilizer Application and Manure Production. Data distributed by the NASA Socioeconomic Data and Applications Center (SEDAC): <a href="http://sedac.ciesin.columbia.edu/data/collection/fertilizer-andmanure">http://sedac.ciesin.columbia.edu/data/collection/fertilizer-andmanure</a> This document is licensed under a Creative Commons 3.0 Attribution License EV <a href="http://creativecommons.org/licenses/by/3.0/">http://creativecommons.org/licenses/by/3.0/</a> Science Information Network</p>	<p>Oceania Nitrogen in Manure Production Global Fertilizer and Manure, Version 1 :4 0 500 Kilometers</p> <p>Projection: Australia Albers A</p> <p>Kg/ha of Nitrogen Manure produced per grid cell: ~</p> <p>Amount of nitrogen in manure produced within the 0.5 degree Q :19 59 r Q39 ,9Ŧ grid cell. Grid cell values are expressed in kilograms per N3KN,ŦN)GĐ5K5K5K635K(195 hectare (kg/ha) ranging from 0 to 370. The data values were Q Q~ (1, 6y *9 '19 {&gt;9 ŦŦŦ derived based on the nutrient content of the manure produced by the total number of livestock located within each grid cell.  :_:</p> <p>Center for Tntematkma] Earth Copyright 2011. The Trustees of Columbia Universityinthe City of New York. Pubncatwn Date: 1/24/2011 Source: Potter, P., and N. Ramankutty, et al. (2010). Global FertilizerApplication and Manure Production. Data distributed by the NASA Socioeconomic Data and Applications Center (SEDAC): <a href="http://sedac.ciesin.columbia.edu/data/collection/fertilizer-andmanure">http://sedac.ciesin.columbia.edu/data/collection/fertilizer-andmanure</a> This document is licensed under a @ Creative Commons 3.0 Attribution License EV <a href="http://creativecommons.org/licenses/by/3.0/">http://creativecommons.org/licenses/by/3.0/</a> Science Information Network</p>

Fig 4: Extracted text from Images (Using Tesseract Extractor)

2. We developed `tesseract.TesseractOCRParserRunner` which is a utility class to run string parsing across image files with supported types. To run it, initialized `TesseractOCRParserRunner` and provide its parameters then invoke `runParser()` method or invoke the main class with following parameters.  

```
java main.Main -t ocr -b baseFolder -r resultFolder [-m markerFile]
```
3. By examining the results, some of them are maps with information or description embedded. Parsing by Tesseract OCR will extract this information into plain text. The extracted text will give us more information about the image. Since it can be used in combination with other parsers, we might be able to extract more metadata and might be able to geotag these map files.

#### XIV. Important Observations:

1. We used the parsers developed as a part of this project to extract measurement data, location data, research publications and named entities from the text (Intersected with SWEET). These features were used to enrich the content of the dataset. A conglomerate of these features when put in Solr index helped us create a fast search engine to query the data effectively. All the above features provide good insights into the data. For Example, Location based data shows us that most of the locations mentioned in the dataset were concentrated in North America.
2. The file sin the dataset had a lot of irrelevant information. Tag Ratio Analysis helped us isolate the relevant data in the files. Measurements were then extracted from these relevant portions of the file. Advantages: Performance boost up as lesser text had to be parsed.
3. NER in its native form was not able to identify SWEET categories. Stanford Core NER was used to extract entities present in the data (Eg. Names of Organizations, Locations etc.). We used minimum Edit Distance to find the entities that were similar to concepts present in SWEET categories and concepts. These entities were stored in respective JSON as metadata information for all the files.
4. D3 visualizations were of great help to understand the data. Zoomable circle packing D3 of our metadata quality score analysis helped us to visually understand the difference in metadata quality for some files in all MIME types. Location map helped us to visualize various locations referred to in the dataset. Bar charts and Pie Charts were used to analyze the counts of various locations and concepts in the dataset.
5. Geo Location Distortion feature was helpful in producing clusters. With lesser distortion, the number of clusters increased linearly (approximate). See Tika Similarity (Part X) for more information. Also, Geo Location features like Longitude and Latitute are easy to visualize on a world map (See Part XII)
6. Based on our observation, we did not notice a hierarchical structure in the extracted data. Hence we used K-means clustering for our analysis. Also, since we can specify the number of clusters the data and study the distortion values, using K-means is more meaningful here.
7. Jaccard similarity resemblance value of each file is calculated from metadata key, so a file type that has the same metadata key should produce similar resemblance, thus will be in the same cluster. Clustering using Euclidean and Cosine distance should be quite the same because both of them use the length of metadata values as features. Edit distance use actual metadata values so the cluster might be different. Overall, Edit distance proved to be slightly better than other distance metrics.

Smaller Dataset ("/com/ytimg")					Larger Dataset ("/info")				
Jaccard					Jaccard				
Cluster	Type	Majority	Size	Accuracy	Cluster	Type	Majority	Size	Accuracy
cluster0	text/plain	13	15	86.66667	cluster0	application/xhtml+xml	1	1	100
cluster1	image/gif	2	2	100	cluster1	image/gif	9	10	90
cluster2	image/jpeg	31	31	100	cluster2	application/xhtml+xml	288	521	55.27831
Overall		46	48	95.83333	Overall		298	532	56.01504



Euclidean					Euclidean				
Cluster	Type	Majority	Size	Accuracy	Cluster	Type	Majority	Size	Accuracy
cluster0	text/plain	13	15	86.66667	cluster0	application/xhtml+xml	7	8	87.5
cluster1	image/gif	2	2	100	cluster1	application/xhtml+xml	218	253	86.16601
cluster2	image/jpeg	31	31	100	cluster2	text/html	29	44	65.90909
Overall		46	48	95.83333	cluster3	text/html	97	227	42.73128
					Overall		351	532	65.97744
Cosine					Cosine				
Cluster	Type	Majority	Size	Accuracy	Cluster	Type	Majority	Size	Accuracy
cluster0	image/jpeg	31	31	100	cluster0	text/html	45	85	52.94118
cluster1	text/plain	13	15	86.66667	cluster1	application/xhtml+xml	226	262	86.25954
cluster2	image/gif	2	2	100	cluster2	text/html	81	185	43.78378
Overall		46	48	95.83333	Overall		352	532	66.16541
Edit Distance					Edit Distance				
Cluster	Type	Majority	Size	Accuracy	Cluster	Type	Majority	Size	Accuracy
cluster0	image/jpeg	31	31	100	cluster0	text/html	62	135	45.92593
cluster1	text/plain	13	13	100	cluster1	text/html	99	192	51.5625
cluster2	image/vnd.microsoft.icon	2	4	50	cluster2	application/xhtml+xml	199	205	97.07317
Overall		46	48	95.83333	Overall		360	532	67.66917

Table 3: Accuracy in Tika Similarity using various distance measures (Edit Distance gives highest accuracy in the larger dataset)

8. The metadata quality score helps us to segregate the files having sparse metadata fields. Metadata fields are important since they provide valuable insights about the data files. The more the number of metadata fields present in the files, the more is the metadata quality score, and more inferences can be made about the data.
9. We were able to find twenty related scientific publication for each research publication identified in the dataset. We found that there was overlap in authors between the publication in the dataset and the publications pulled from the Google Scholar API. However, due to the restricted nature of API usage, we used other parameters (title, keywords from abstract) to fetch the related publications. This gave us a good variety of the publications which were highly relevant to the publications present in the dataset.