**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: 03/03/2016

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. **File preparation**

We did some file preparation to work with the dataset files easily. First, we used Apache Tika to detect file type, and then indexed file path by its MIME type in JSON format, by running a Java class **typedetect.runner.TypeDetectRunner**. This would help us when we would work on a specific file type.

1. **Tag Ratio Tika Parser and Measurement Extractor**
2. The Tag Ratio algorithm allows us to extract relevant text from XHTML files. We used Tika to convert any file in the dataset into an intermediate XHTML file. Then, we calculated the Text-To –Tag ration for the file 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 ration is discarded. Only the remaining text is chosen for parsing for next stage. An example file and its extracted contents are shown below:
3. Fig Example file and content extracted
4. We applied this parser to all the files within the TREC Polar Domain Dataset
5. Next, a wrapper class was created, 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:
6. Fig: Extracted Measurements Algorithm
7. Measurements extracted by the parser using the above algorithm are:
8. Fig: Extracted measurements from the files
9. **URL Shortner**
10. 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.
11. A sample JSON containing the short URL and relative file path is shown below:
12. Fig: Sample JSON URL Shortner
13. 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.
14. **Content Extraction and NER using Grobid Journal Parser**
15. Grobid Journal Parser in Tika was used to extract TEI annotations from the PDF files in the dataset.
16. 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.
17. For each of the PDF files in the dataset, their TEI annotations are stored as JSON data.
18. 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, a total of --------- files were pulled.
19. Relevant information such as Authors, Publication Year, Affiliations were extracted and identified for each of the new papers/ journals.
20. Fig: Extracted papers for one file, Draw dendogram if possible.
21. 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, and used the remaining keywords as input to the scholar.py program.
22. Show example field input.
23. **Geo-Topic Parsing using Tika GeoTopicParser**
24. 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.
25. The results obtained were stored as JSON objects. A sample result file is shown below:
26. Fig: Sample result from GeoTopic Parsing
27. Later, these output files were dumped in solr to generate a search inverted index.
28. **Intersecting the above results with SWEET**
29. 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.
30. Implication: (Step V) ∩ (Step VI) gives us the extracted environment related terms form all the files in the dataset. Sample results are shown below:
31. Fig: Result from Sweet Ontology
32. If Possible, make a D3 for some concept here.
33. **Metadata Quality Store**
34. **Inverted Index Generation using Apache Solr**
35. 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. A sample snippet of the same is shown below:
3. Fig: snippet of solr schema.
4. Explanation of what fields are important to visualize, and what are important to search and to find the data
5. **MEMEX GeoParser**
6. The MEMEX GeoParser is run over the index generated by us in the last step (VIII). The location map hence generated is shown below:
7. Fig: Show Location Map
8. Interesting Observations: write some observations from the map/ some inferences.
9. **Tika Similarity**
10. Deployed Tika Similarity library to connect to Solr Index with extracted NER features and Text.

a. Cluster the data according to Measurement extractions.

b. Cluster the data according to related publications and authors.

c. Cluster the data according to extracted locations.

d. Cluster the data according to SWEET features.

e. Develop a program that connects to your Solr and/or ElasticSearch and that performs either the k-means and/or hierarchical clustering technique using Jaccard similarity, edit distance, and/or cosine similarity and that produces output D3 visualizations. Please choose carefully and describe your thought process behind your distance metric and clustering decision.

A workflow of clustering and visualization using Tika Similarity is as follows:

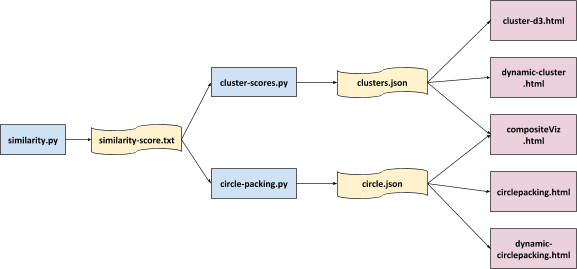


Fig 5: Workflow of clustering and visualization using Tika Similarity

1. By default, Tika Similarity uses Jaccard similarity. However, there is no such concept using Cosine similarity and Edit distance. There is an implementation of K-means clustering already in the project which can be modified to suit our needs.

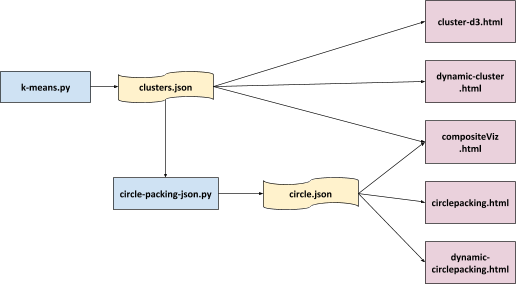


Fig 6: Workflow of clustering and visualization after adding Cosine and Edit Distance

1. K-means clustering uses Euclidean distance (existing) and outputs the clusters.json file. It transforms each file to a feature vector and calculates distance between each two vectors during the clustering process. We do the following to complete the modification.
2. Add a function to calculate Edit distance between two vectors. Also modify the feature structure to contain enough data to calculate the distance.
3. Modify the code that does K-mean clustering so we can specify which distance measure to use. Also modify centroid selection when using distance measure other than Euclidean.
4. Add a script to create circle.json by taking clusters.json (output from 2.b) as an input.

Analysis on clustering was done using 2 datasets, a smaller one containing about 50 files and the larger containing about 500 files. Screenshots of clusters and circle packing of smaller dataset are as follows:

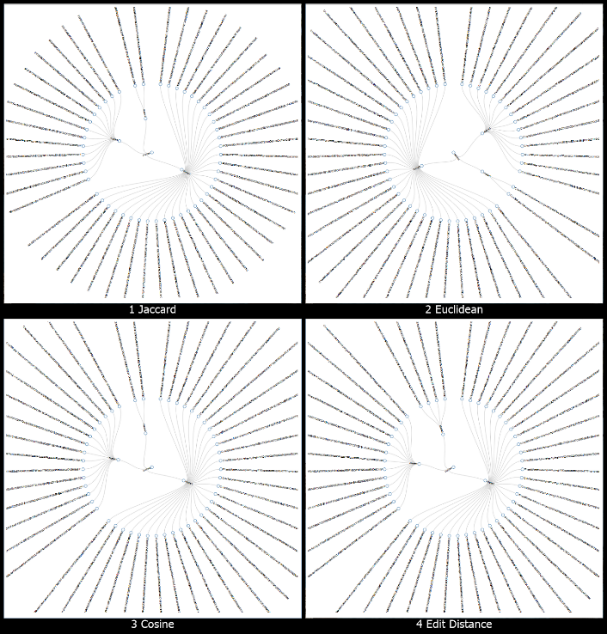
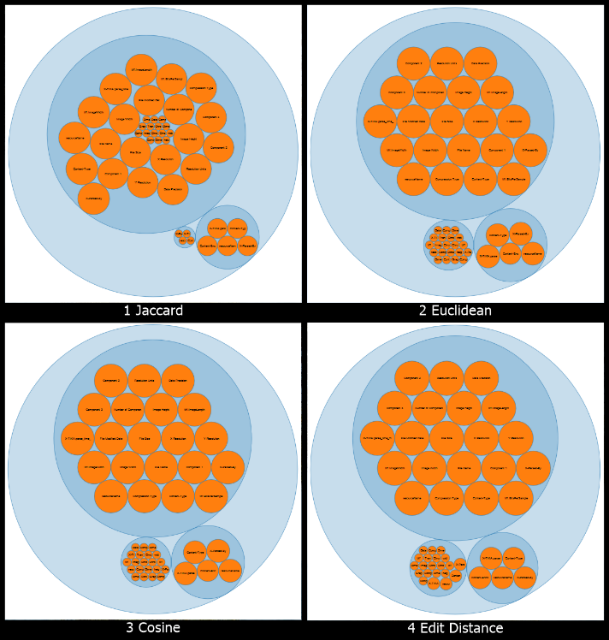
 

Fig 7: Clustering and Circle Packing of smaller dataset

**Some Observations:**

1. 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.
2. We consider the type detected from Tika as each file’s actual type and try to classify each file type in each cluster to be the same type as its cluster majority. For smaller dataset each cluster indicates its type neatly. For larger dataset, although it is not as neat as in smaller dataset but it still resembles the type.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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)

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?