**MIME Diversity in the Text Retrieval (TREC) Conference-**

**Polar Dynamic Domain Dataset**

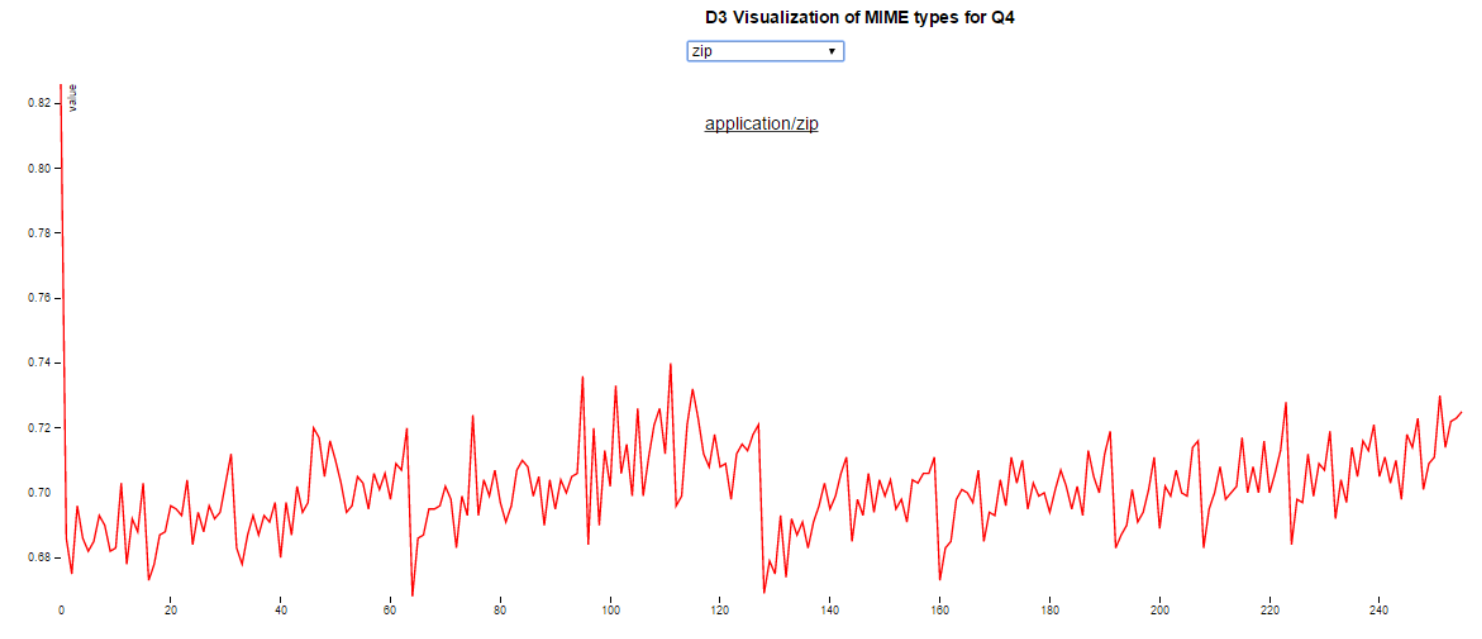
**Introduction**

This experiment aims to study and implement the Automatic File Type Detection [1] mechanism on the TREC-DD-Polar dataset, collected over years with large diversity, to contribute towards Apache Tika. The project adapted the concept of frequency analysis for detecting the file types using the byte frequency for the dataset. We first training the models by parsing and detecting file type using Apache Tika to generate fingerprints and then comparing the results on the test data result set.

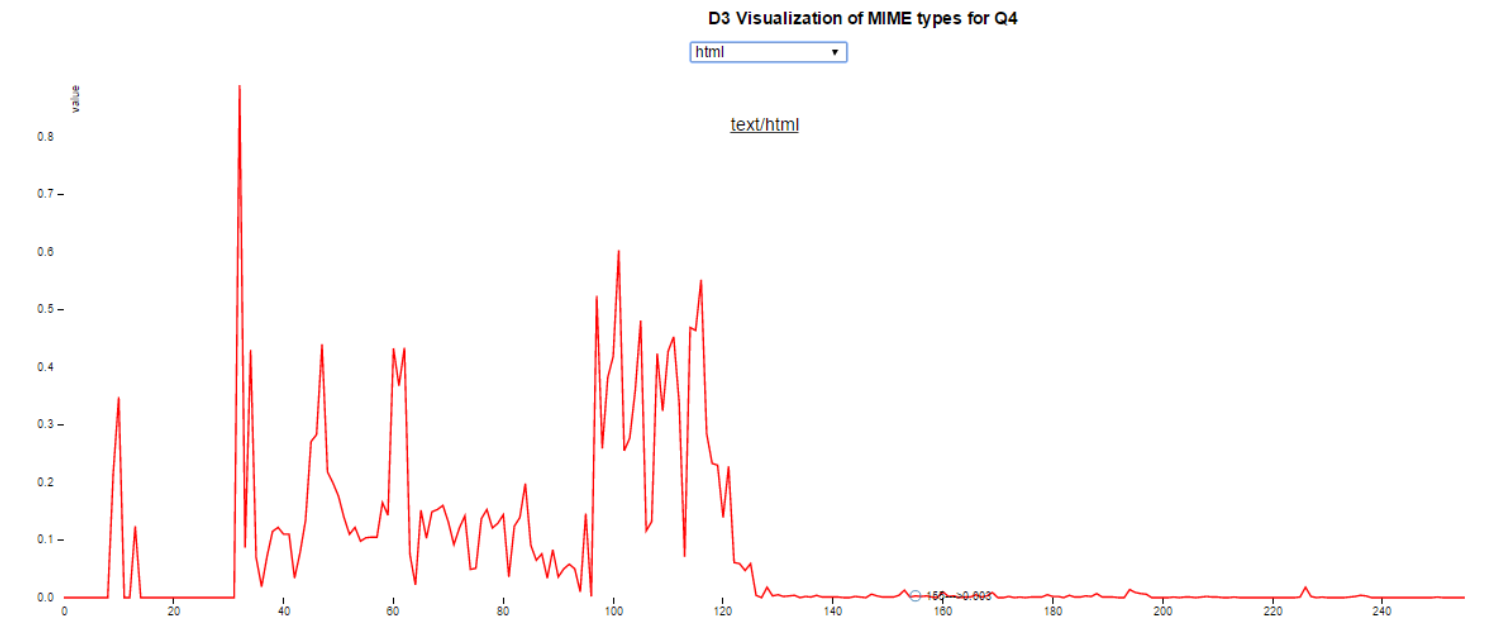
1. **Byte Frequency Analysis - BFA**

The byte frequency distribution for the 75% of the dataset which was clearly classified and 25% of the application/octet, was calculated and normalized to generate the fingerprints. We used over 1.2 million files as our training dataset. The main challenge was to read all these files byte by byte and generating the byte frequency distribution. It was observed in some of the cases some typical bytes value are very high, resulting in skewing of data for high frequency bytes, and low frequency byte pattern was. This problem can be overcome with companding (compression and expanding) and amplifying the low frequency bytes value. In telecommunication industry the A-law and µ-law algorithm are used for companding, though these process require high computation complexity. The result of applying the lower degree companding like square root and cube root resulted in better representing the features of these low frequency bytes. We used the square root for damping this effect, as shown below in Figure 1.1 below.

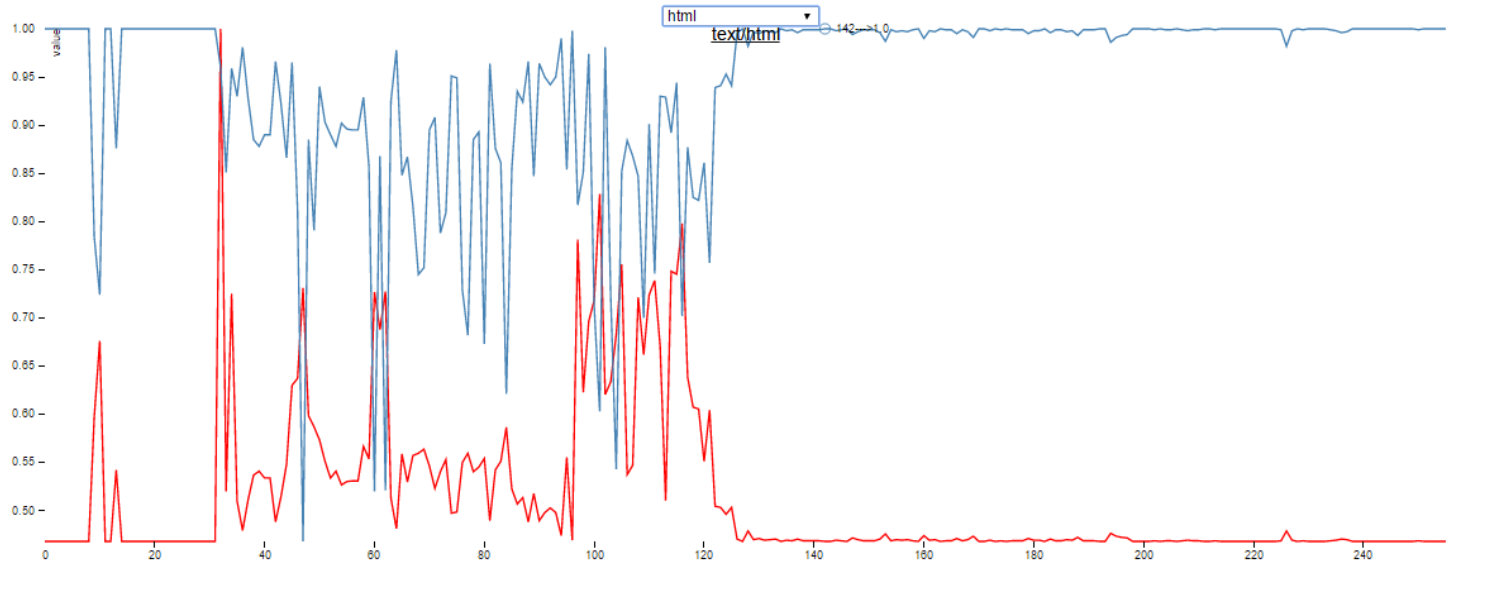
Now, we are ready to compare the test files to detect the file type based on its byte frequency distribution by computing the correlation strength for the test dataset. We compared the performance of BFA fingerprint based on correlation and Apache Tika for the test dataset.



**Fig 1.1 BFA ZIP fingerprint- companded**



**Fig 1.2 - BFA HTML fingerprint**



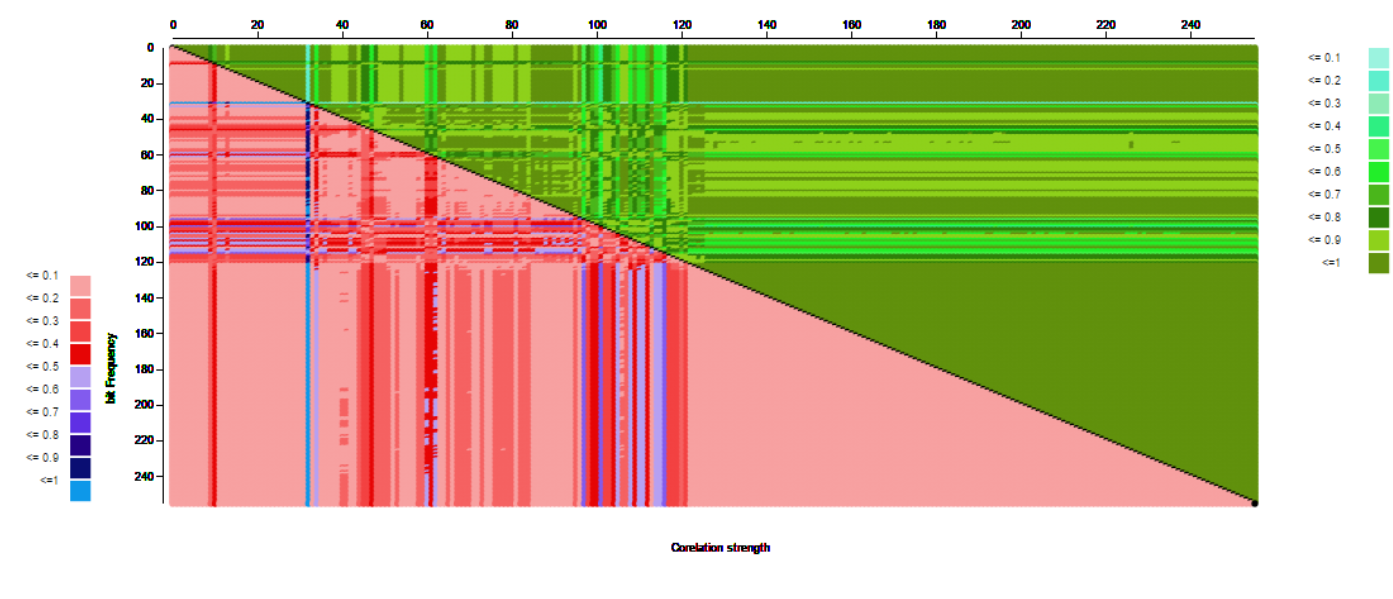
**Fig – Byte Value vs Correlation strength**

*Comments:*

This method is slow and time consuming, however the same can be used with parallel computing paradigm or Hadoop to reduce the computation time for averaged byte frequency distribution.

1. **Byte Frequency Cross-Correlation**

It can be observed based on the BFA chart that some of the bytes have relative similar frequency and can be an importance feature in detection in mime type e.g. – HTML. This feature extraction can be better studied by the byte cross correlation algorithm and generating a visualization for each mime type. We computed the cross correlation matrix for the test data based on the average byte frequency as well as with the correlation strength.



**Fig – BFC HTML -Cross Correlation**

*Comments:*

The way of identifying the features of the MIME types was possible with help of the BFC matrix and correlation. This approach was faster than BFA in identifying the correlation strength, though not very efficient in detecting the MIME types.

1. **File header/ Trailer Analysis**

File header and trailer byte analysis might indicate the content of the files and we conducted it on the entire dataset. Determining how many bytes to read is an important question. We ran our algorithm on 4, 8, and 16 bytes and found that choosing 16 byte is most appropriate considering that some bytes are bit farther. We were able to determine the magic bytes more accurately, and updated to mimetype.xml. The trailer analysis was not found to be that effective given some of the files were observed to be truncated files giving rise to misleading results.



**Fig – FHT Sparse Matrix**

*Comments:*

The file header/trailer analysis is found to most efficient so far. The accuracy is way higher for FHT than in the previous mechanisms and appears to most promising. On the contrast the ,there is an extra overhead of reading entire file content for the trailer version and wait associated in reading in real time may discourage its usage.

1. **Conclusion**

As we have can conclude that most of the files can be easily identified by the header and FHT and given to be fastest, over other two techniques. The list below gives the description of the changes made to mimetype.xml to tune Apache Tika in support of the observations.

|  |
| --- |
|  |
| Mime Type | Training Data | Test Data | Fingerprint 1 | Fingerprint 2 |
| atom+xml | 1992 | 664 | 3C 3F 78 6D 6C 20 76 65  72 73 69 6F 6E 3D |  |
| html | 505346 | 168449 | 3C 21 64 6F 63 74 79 70 | \<title |
| office | 1176 | 392 | CF 11 E0 A1 B1 1A E1 00 | D0 CF 11 E0 A1 B1 1A E1 |
| gif | 25256 | 8419 | 47 49 46 38 37 61 or  47 49 46 38 39 61 | 7A  or  8A |
| png | 25167 | 8389 | 89 50 4E 47 0D 0A 1A 0A | \x146\x031\x0\x0 |
| jpeg | 59536 | 19846 | FF D8 FF E0 xx xx 4A 46  49 46 00 | \xFF\x4F\xFF\x51\x00 |
| pdf | 28330 | 9444 | 25 50 44 46 | %PDF-3. |
| gzip | 1197 | 399 | 1F 8B 08 | \037\036 |
| zip | 1104 | 368 | 37 7A BC AF 27 1C | 50 4B 03 04/  50 4B 05 06/  50 4B 07 08 |
| sh | 799 | 267 |  |  |
| rss | 5639 | 1880 | 0x107 |  |
| xml | 4222 | 1408 | 3C 3F 78 6D 6C 20 76 65  72 73 69 6F 6E 3D | 3C 3F 78 6D 6C 20 76 65  72 73 69 6F 6E 3D 22 31  2E 30 22 3F 3E |
| xhtml | 207004 | 69002 | \<title |  |
| plain text | 67372 | 22458 | 7B 5C 72 74 66 31 | 46 4F 52 4D  46 54 58 54 |
| octet | 23946 | 21332 | NA | NA |

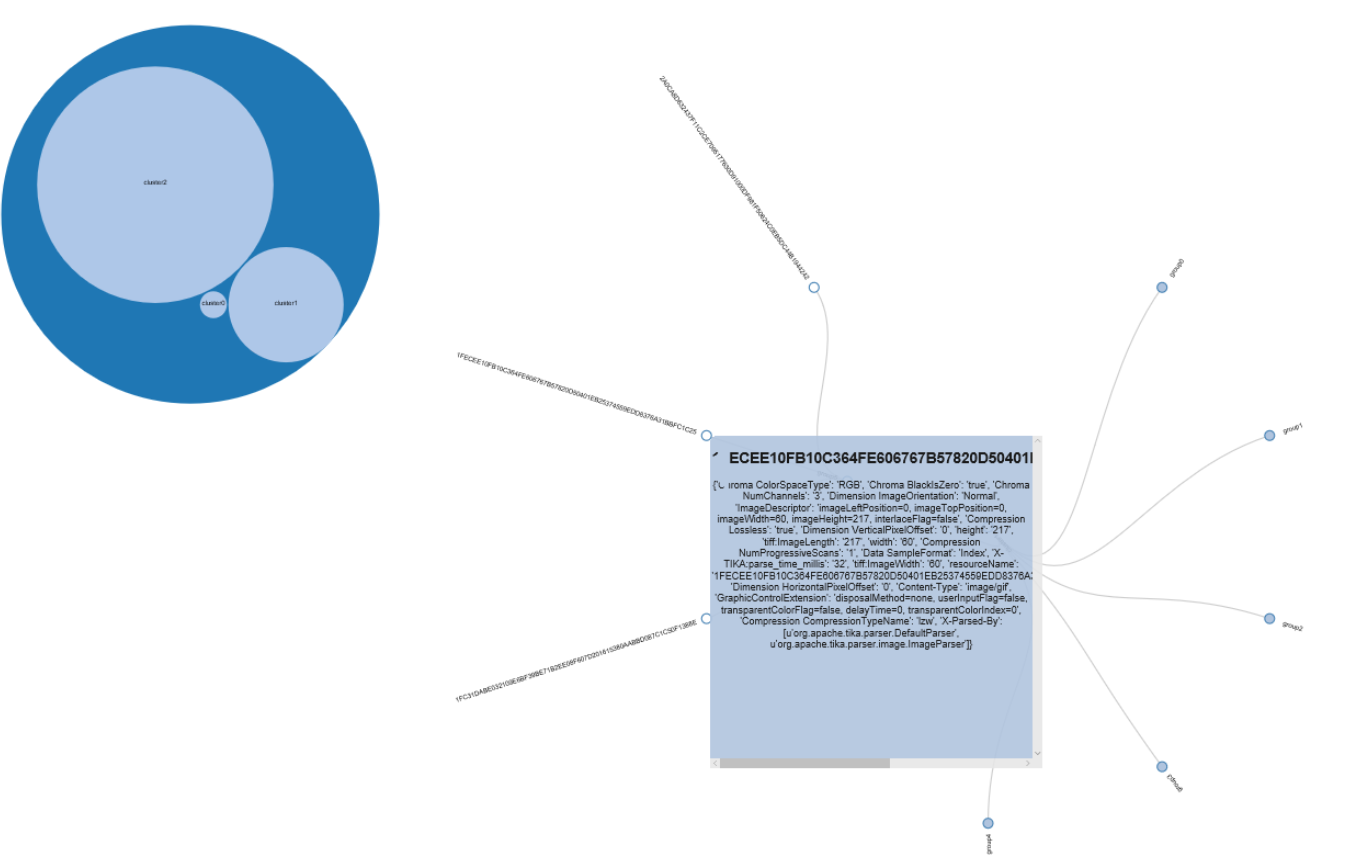
­­­We further go ahead and run the configured Apache Tika(v 1.11) based on our observation to show improvement over version. We were able to identify 29 files out of 322 application/octet-stream files.

**Fig 4.2**

**5 Other Approaches**

The byte frequency based content detection provided high reliability, however has little to none insight into the content of the files. We can used correlation based on how similar these documents are to each other and classifying based on resemblance.

**5.1 Tika-similarity**

Tika similarity uses basically three types of similarity between documents- Jaccard, Cosine and Edit Distance There are clusters where the similar information in different format resulted in false classification. The edit distance gives extensive classification based on content than Jaccard similarity 

**Fig-Level clusters**

**5.2 Content based Mime Detector**

Neural network training of models to identify byte histograms is an effective way to train system for differentiating between the content types. The test dataset from BFA for hmtl files was used to feed the byte frequency matrix/histogram which was further divided into training, validation and test datasets. The output – Tika model, for content based Mime detector based on newly trained model was incorporated in Tika.

**6 Q&A**

1)Discern new Mime type from /application/octet:-

Using the above fingerprints we were able to identify further mime types that were earlier identified as octet-stream. Out of all the octet types most of the files were of size 0 bytes and hence were ignored during the project. 21 files that tika identified as octet were identified as text/plain type 2 were found to be application/zip and 6 were text/html file. There are remaining further 294 types which are still classified as octet-stream type. We think some further investigation is required to identify

2)D3 helped in deciding understanding and identifying patterns?

D3 helped a lot in identifying the fingerprints. When there is this huge amount of data to work on it is very difficult to identify the fingerprints. D3 is dynamic and we just had to feed the json file and rest all was handled by d3.

3)What mime you picked and why?

The mime type used were:- application/atom+xml, text/plain, text/html, image/gif, image/jpeg, image/png, application/octet-stream, application/gzip, application/zip,application/rss+xml, application/xhtml+xml, application/x-sh, office, application/pdf. These files types had huge amount of data and was proper to train the algorithms on. They helped in identifying the fingerprints properly which led to the increase in accuracy of Tika.

4)Why Tika was not able to discern the MIME types?

Most of the files were of size 0 bytes which means there was no content in the files. If there is no content in the files then there is no way to detect the type of content for tika. That is why tika identified them as octet-stream. The other reason is some of the fingerprints were not properly updated or had lower priority.

5)Error in priority precedence?

We tried to change the priority sequence of some mime types and included some new priority values. After this we were able to identify some of the files of octet types. Which means there was some issue in the priority precedence.

6)What was easy to use in Tika what wasn’t? Tika was very easy because we just had to create the object and run the parser method. This resulted in the mime type directly and most of the time it was accurate. Tika is fast and if there is content in the file it detected the correct type. The only draw back is that it does not identify the broken files and files with some content apart from zero bytes files. It classifies all of them in to the default octet type. There should be an option to find the file types identified taking the lower priority in from the mimetype.xml

7)What was easy to use in Tika what wasn’t?

Tika was very easy because we just had to create the object and run the parser method. This resulted in the mime type directly and most of the time it was accurate. Tika is fast and if there is content in the file it detected the correct type. The only draw back is that it does not identify the broken files and files with some content apart from zero bytes files. It classifies all of them in to the default octet type. There should be an option to find the file types identified taking the lower priority in from the mimetype.xml file.