CSCI 599 Assignment 1 Report
Team 17

Assignment:
MIME Diversity in the Text Retrieval Conference (TREC)
Polar Dynamic Domain Dataset

Overview:
In this assignment, we explored various mime types using the following algorithms: BFA, BFCC, FHT.
We built a system that allows visual interaction and introspection of the MIME diversity
Using D3 on the dataset extracted from Amazon s3. (polar full dump)
We also studied the application/octet-stream and identified 2 new mime-types that were previously unidentified by Tika.
We updated Tika's mime repository with the knowledge gained form the analysis.
Finally, we applied Tika-similarity and content-based mime detector on TREC-DD Polar Data.
All the findings have been listed below in the report.

Organizing the data dump:
The data was downloaded from Amazon s3 using AWS CLI. The initial bucket size was around 60GB. We then ran a Java utility program to sort the data according to their mime-types. This was achieved using *tika.detect()*. A pie chart was shown using d3 for the existing mime diversity of TREC-DD- Polar Dataset.
The criteria for selecting 14 of the mime types were:

1. Diversity in the super classes of the mime-type hierarchy
2. Size of the files of a particular type
3. Variation in the byte frequency distribution

Based on the above, we selected the following 14 types, in addition to the mandatory application\octet-stream:

Text	-	html, plain
Application	-	pdf, xhtml+xml, x-tar, x-executable, zip, octet-stream
Image	-	jpeg, png, gif
Message	-	rfc822
Audio	-	wav, basic, mpeg
Video	-	quicktime, mp4

Observation:
Following three file types were predominant:
1. text/html
2. text/plain
3. application/xhtml+xml

Nearly 90% files of the application/octet-stream type were empty, making them unfit for analysis. Some of the files were truncated.

Algorithms:

(Note: Java bytes are signed, byte index needs to be manipulated for fingerprint generation)

We did each of the algorithms development in Java.

1. Byte Frequency Analysis:
   We generated the 256 byte-vector fingerprint.
   75% of the data was used for generating the fingerprint and on the remaining 25%, we calculated the correlation strength.
   For octet-stream, only 25% of the data was used for fingerprint generation.
   Beta value used for companding function - 1.5
   Sigma value used for correlation strength - 0.0375

   Visualization:
   We represented the fingerprint and correlation distribution strength using Bar chart of D3.
   Wav mime type had a very distinct fingerprint pattern while Exe mime type had a lot of zero values.

2. Byte Frequency Cross Correlation:
   The 256*256 fingerprint consists of 2 triangular sections.
   The lower section represents the pair wise byte differences while the upper triangle represents the correlation scores.
   In the visualization, the dark (red and blue) points signified areas of high correlation while the fainter ones (tending towards white) signified low correlation.
   Though interesting patterns were observed, the patches in the visualization did not provide much insight into the mime type.

3. File Header Trailer:
   The algorithm was implemented for header and trailer sizes of 4, 8 and 16 bytes.
   Most of the valuable information was obtained from the headers.

Updating tika-mimetypes.xml:

Most of the byte values/string patterns that we obtained from the visualization of the FHT outputs, were already present in tika-mimetypes.xml for offset 0.
We added 2 mime magic byte fingerprints: jpeg for offset 2 and postscript for offset 0 with our own findings. We then recompiled Tika with this new xml and ran it on the original unsorted data set.
All the files were classified exactly as with the original Tika run.

Application/octet-stream observation:

A varied number of unidentified file types were observed. Most of them were not readable in ascii.
Some files had content similar to that of email content.
Two other files types that were observed were JavaScript(.js) and Standard Formatted Data Unit (SFDU)

Tika similarity:

We installed and ran Tika similarity on our data set and different observed cluster patterns.
After running it on the data set, we found that cosine-distance and edit-distance produced similar clusters whereas Jaccard distance produced a different pattern.
This is because Jaccard Distance compares file types with the golden set which is a union of all the features derived from the metadata of all files. Cosine and edit distance compare each file with the other.

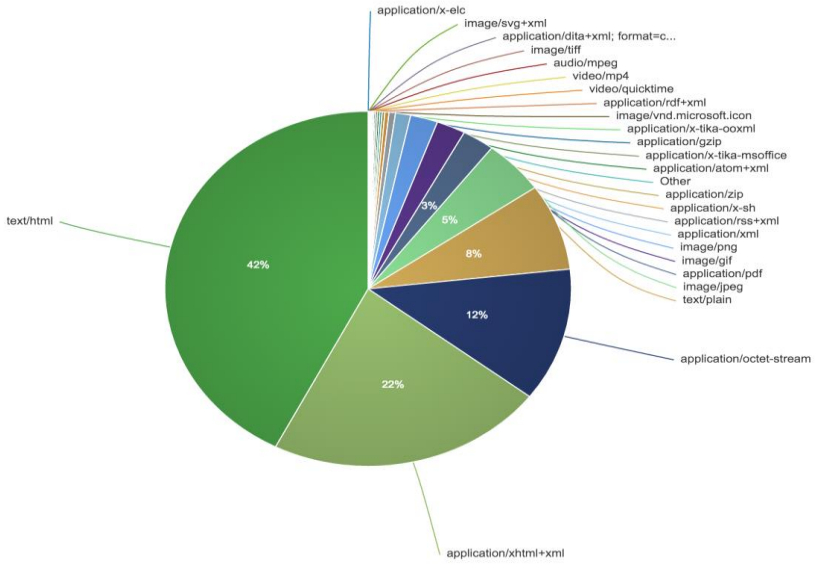
Content Based MIME Detector:

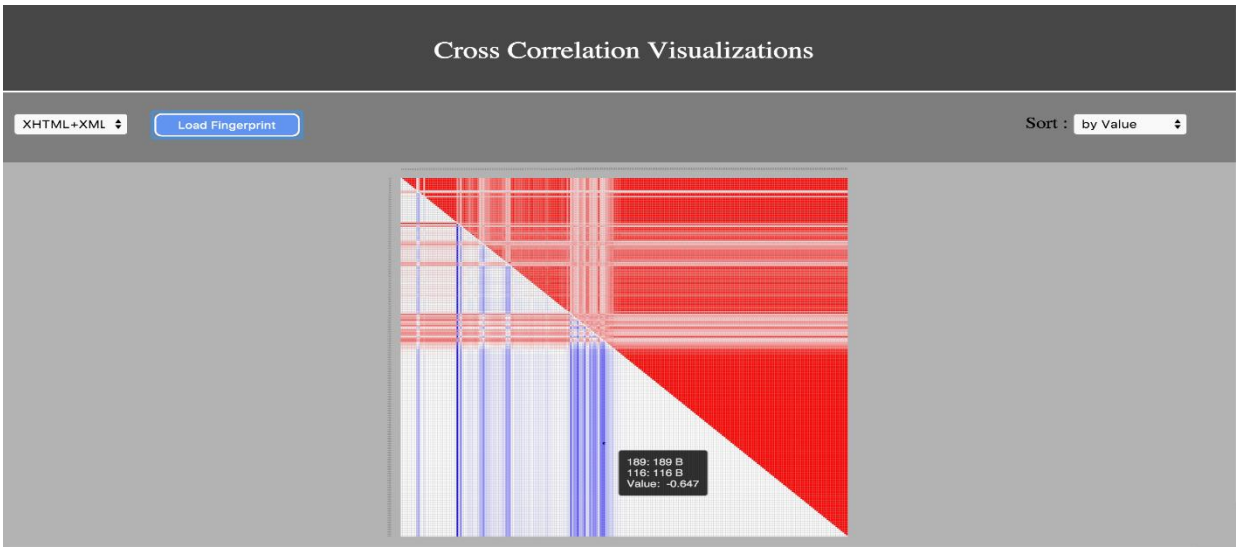
Application/pdf was chosen as the mime type for testing. We wrote a java utility to generate train, validation and test inputs. The three sets comprised of positive (pdf files) and negative (plain, html files) examples.

The three generated input csv files were then given to R program for Neural Network Model Generation. The output generated is the Learned Model for pdf / non-pdf detector.

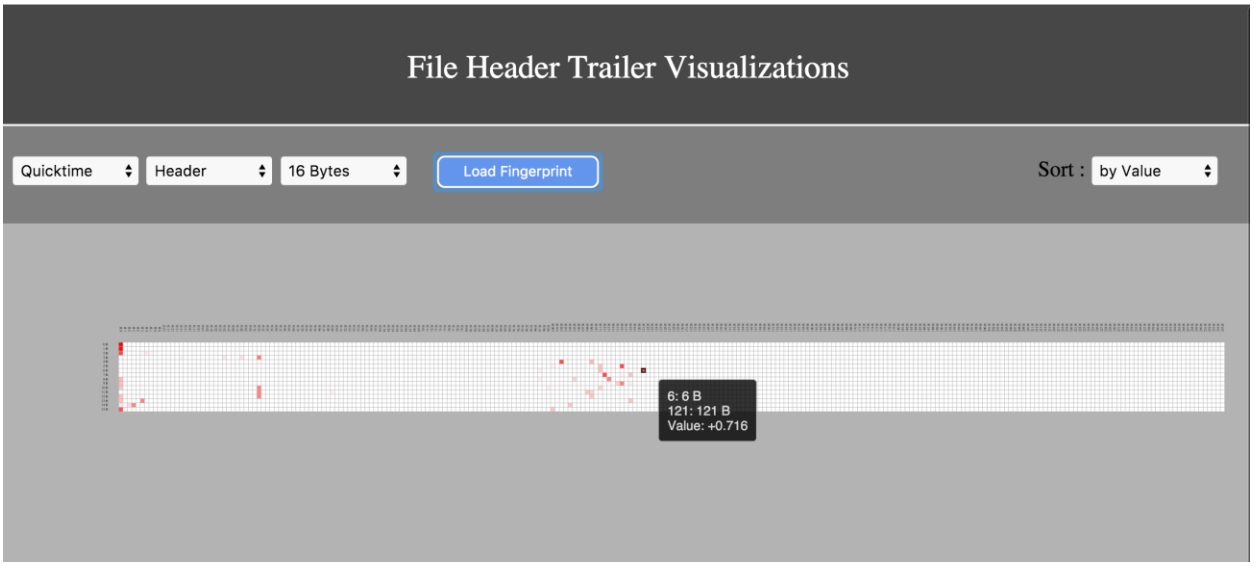
This model file was then integrated with Tika and unit test was carried out using MimeDetectionWithNNTest.java. The function was given the directory path to the data set. *assertEquals()* runs the model on given test classification files. The file is classified as application/octet-stream if it is not pdf.

Screenshots:

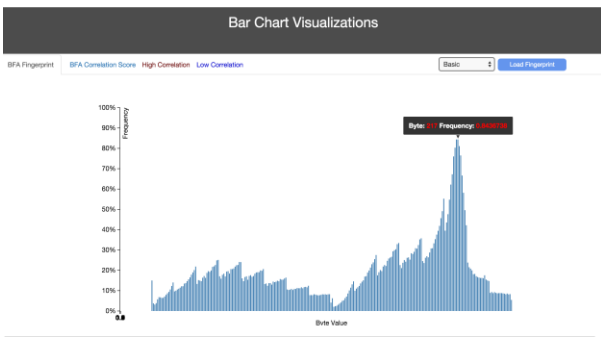




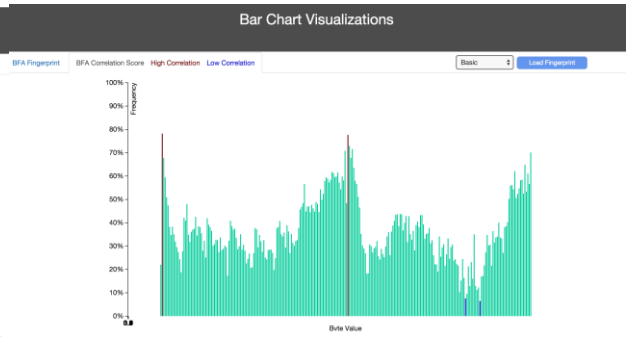
BFC fingerprint sample



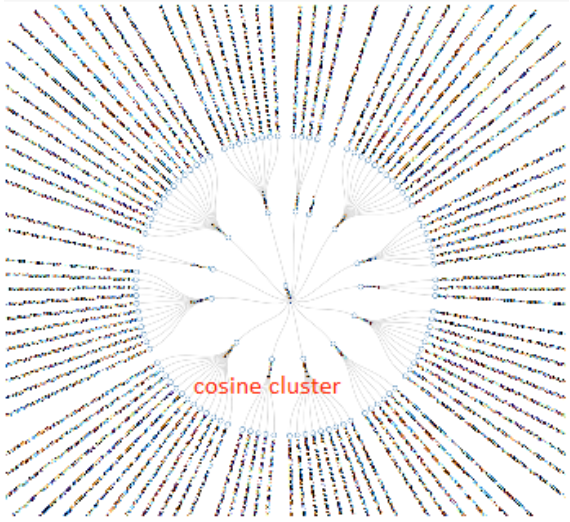
FHT fingerprint sample

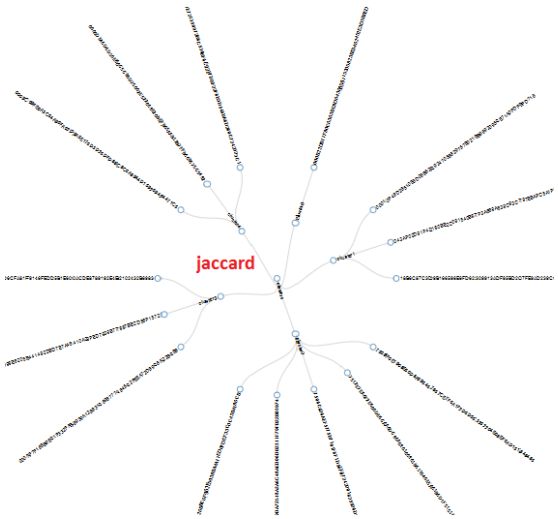


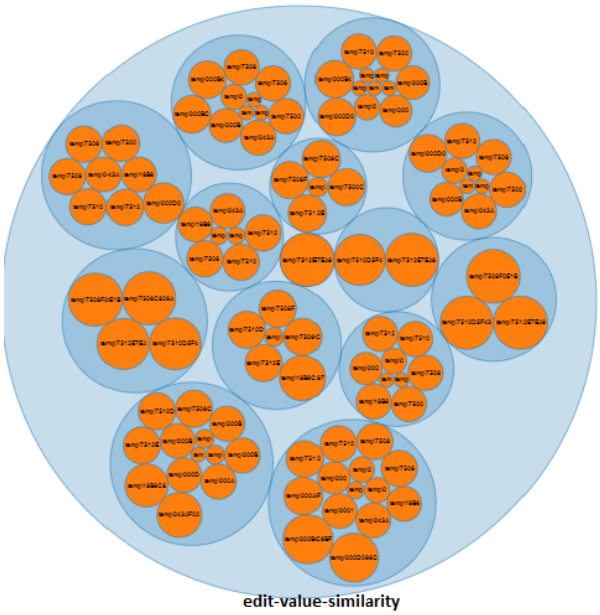
BFA fingerprint sample



Correlation score sample








```
[1] "Loading Dataset....."
[1] "Begining Training Neural Networks"
[1] "the length of weights 519"
[1] "The time taken for training: 10.310000"
[1] "The training error cost: 0.000000"
[1] "The validation error cost: 0.078841"
[1] "The testing error cost: 0.000000"
[1] "Training Accuracy: 100.000000"
[1] "Validation Accuracy: 99.196787"
[1] "Testing Accuracy: 100.000000"
```

R snapshot for pdf Model