**University of Southern California**

Viterbi School of Engineering

CSCI 599: Content Detection and Analysis for Big Data

Instructor:Dr. Chris Mattmann

Assignment 1: MIME Diversity in the Text Retrieval Conference (TREC) Polar

Dynamic Domain Dataset

**GitHub repository**: *http://www.github.com/harshfatepuria/data-analysis-test*

**Github.io website**: *http://harshfatepuria.github.io/*

Date: 03/02/2016

Report submitted by:

Harsh Fatepuria, fatepuri@usc.edu

Warut Roadrungwasinkul, roadrung@usc.edu

Rahul Agrawal, rahulagr@usc.edu

(Graduate Students, Department of Computer Science)

Specifics:

Write a short 4 page report describing your observations, i.e. what you noticed about the dataset as you completed the tasks. Were you able to discern any new MIME types within the 200, 000 application/octet-stream (“unknown”) types? How well did BFA work, compared to FHT? Did the D3 interactive visualizations help you understand the byte frequencies, and to identify patterns? Describe your results from BFA, BFC, and FHT. What MIME types did you pick, and why? Thinking more broadly, do you have enough information to answer the following: 1. Why Tika’s detector was unable to discern the MIME types? 2. Was it lack of byte patterns and specificity in the fingerprint? 3. An error in MIME priority precedence? 4. Lack of sensitivity in the ability to specify competing MIME magic priorities and bytes/offsets? Also include your thoughts about Apache Tika – what was easy about using it? What wasn’t?

1. **Installations, Downloads and File preparation**
2. Downloaded and installed Tika Python
3. Downloaded and Installed D3.js
4. Downloaded the Amazon S3 based TREC-DD-Polar data (full dump ~ 60 GB)
5. Created D3 Pie Chart using existing JSON breakdown from Github (Shown below).

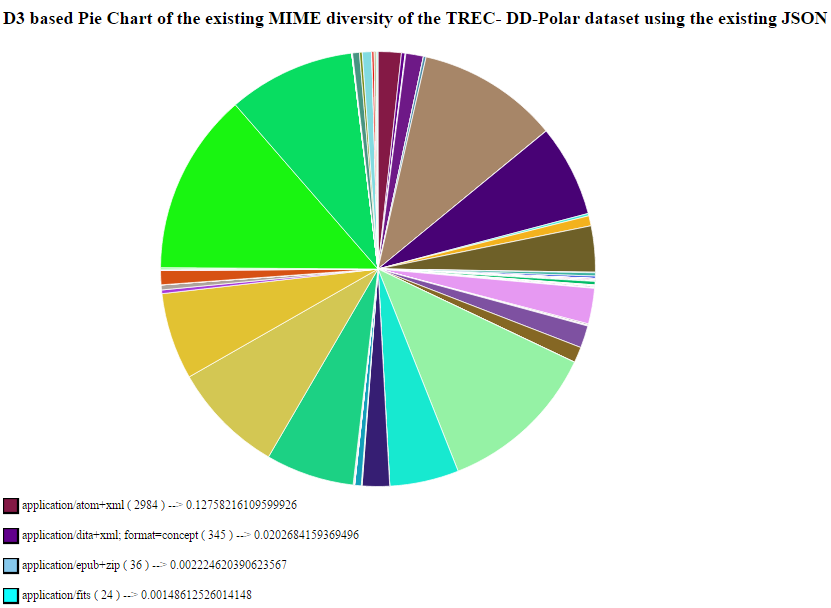


Fig 1: Pie Chart of MIME diversity of TREC-DD-Polar dataset (JSON from Github)

1. Since the dataset is very large and is not associated with file type, we did some preparation to work with it more easily.
2. First, we used Apache Tika to detect file type and then indexed file path by its file 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.
3. To do the analysis, we separated files from each type to be training samples and test samples by the ratio of 75% to 25% except for the type *application/octet-stream* which uses 50,000 files each as training and test set. This was done by running another Java class **typedetect.runner.SeparateTestTrainDataRunner**.
4. The above steps create one JSON file for each file type **that indicate paths to training samples and test samples**.
5. **Byte Frequency Analysis on files of the selected 15 MIME types**
6. We performed Byte Frequency Analysis on the chosen 15 MIME types (14 types + Octet Stream). 75% of the files in the full dump dataset were used for this purpose except octet stream where 50,000 files were used. An automated script was written for this analysis.
7. The script generates 15 JSON files, one for each MIME type. The JSON files were used to generate D3 visualizations of the signatures. An example visualization is shown below.

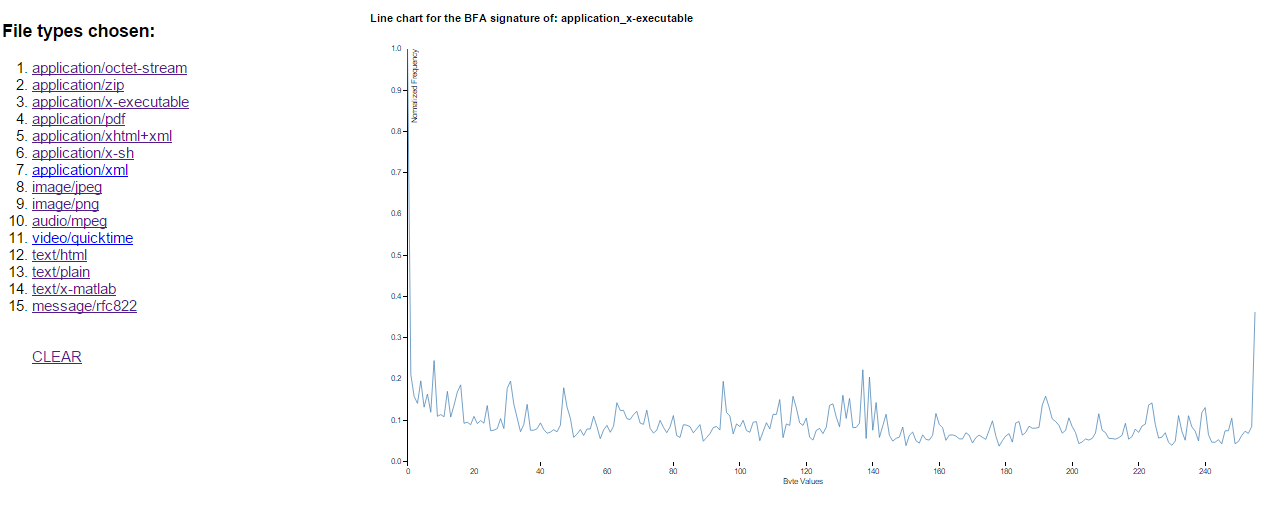


Fig 2: Companded BFC Signature of application/x-executable type

1. **Byte Frequency Correlation and Byte Frequency Cross Correlation**
2. We developed a program which computes the Byte Frequency correlation between an input file and its MIME type’s BFA Signature. The input to this program is a file type (as described in Tika MIME repository) and a file path (of the file to be analysed).
3. [Extra Work] For all the files present in the test data, we calculated a correlation coefficient (Pearson Correlation, algorithm for which has been given in appendix section). We stored the results for all the file types in separate text files. For each of the files, Pearson Coefficient gives a value in range [-1, 1], 1 meaning that the files are highly correlated and -1 meaning that they are not correlated at all.
4. We then developed two D3 visualizations. A multi-line chart depicting BFA signature for the MIME type, and Byte Frequency Distribution of a file. Another visualization depicts the difference between the two (signature and BFD of a file) and shows areas of high and low correlation.

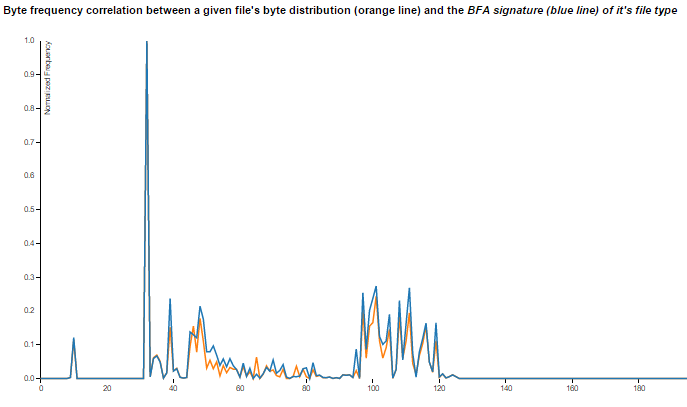


Fig 3: Multi line chart b/w BFA Signature (Blue Line) and BFD of File (Orange Line)

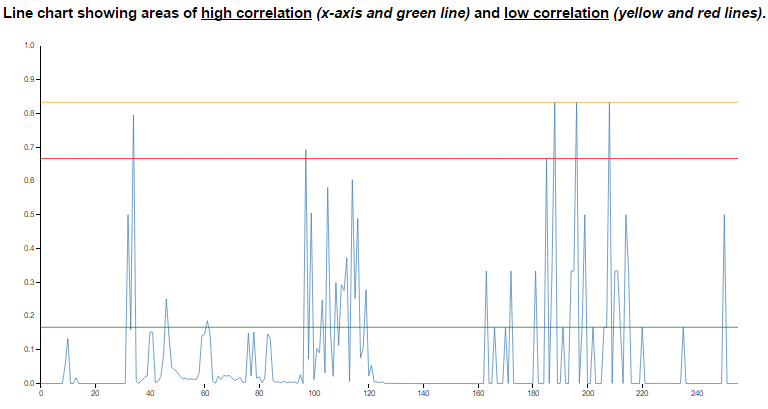


Fig 4: Graph showing BFC (difference between BFA Signature and BFD of file). Also shows areas of

high and Low correlation (Low Correlation= high difference, high Correlation= low difference)

1. We then performed BFC cross correlation on the test data to generate cross correlation matrix for the 15 MIME types. Also, we generated a D3 heat-map visualizing the same.

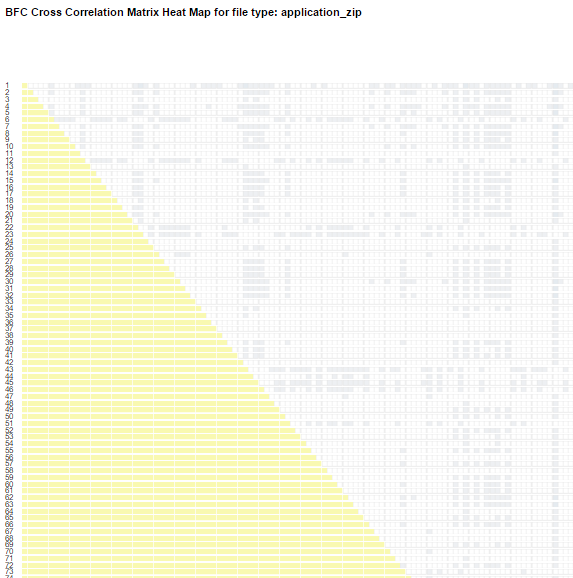


Fig 5: Cross correlation heat map for application/zip type

1. **File Header trailer Analysis (First 4,8 and 16 bytes)**
2. We developed a program which computes FHT analysis on the first 4,8 and 16 bytes of all the files in the training set for all the 15 chosen MIME types.
3. We then generated a FHT heat map of size 256 x 16 which depict the overall data distribution of first 16 bytes of all the 15 chosen MIME types.

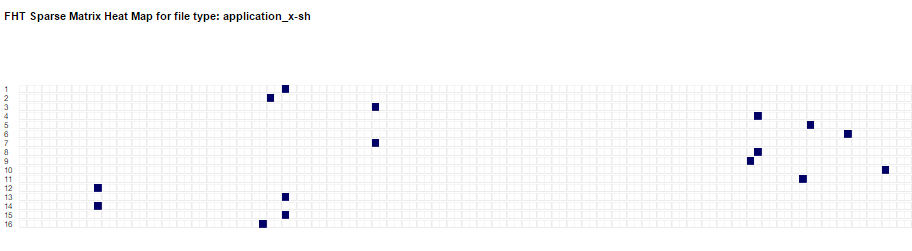


Fig 6: FHT heat map of application/x-sh type

1. [Extra Work] We calculated the Correlation Coefficient for the first 4, 8 and 16 header bytes for each file in the test data compared to FHT Signature procured in the previous step. The Analysis shows some interesting observations. For ex. the first 16 bytes in all the files of type application/x-sh match the FHT Signature 100%.

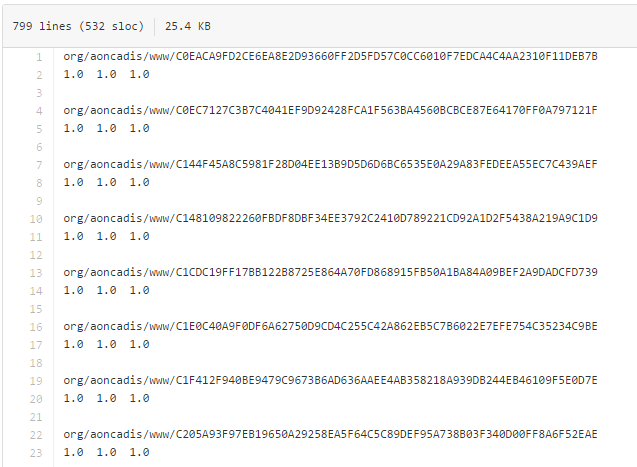


Fig 7: File showing correlation of FHT Signature and starting 4, 8 and 16 bytes of a file in application/x-sh type

1. **Updation of Tika MIME repository based on the above analyses**
2. **Tika Similarity using Cosine Distance and Edit Distance**

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

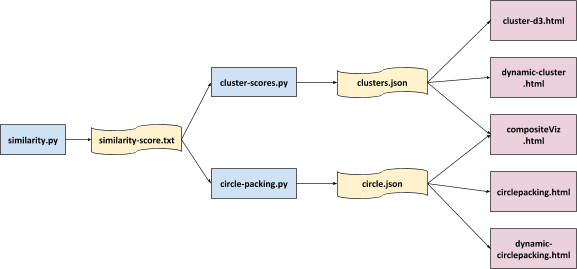


Fig 8: Workflow of clustering and visualization using Tika Similarity

1. The visualization takes data from clusters.json and circle.json as input. If we can modify the program to create these two files by using different distance measure, we can use the visualization part as is. By default, Tika Similarity use Jaccard similarity and do clustering using resemblance value of each file’s metadata. However, there is no such concept using Cosine similarity and Edit distance but each file’s pairwise distance. Fortunately, there is an implementation of K-means clustering already exist in the project so we can modify that code to suit our needs.
2. The code to do K-means clustering uses Euclidean distance and output the clusters.json file. It transforms each file to a feature vector and calculate distance between each two vectors during the clustering process. This vector class already has a function to calculate Cosine similarity. This is what we do to complete the modification.
3. Add a function to calculate Edit distance between two vectors. Also modify the feature structure to contain enough data to calculate the distance
4. Modify the code that do K-mean clustering so we can specified which distance measure to use. Also modify centroid selection when using distance measure other than Euclidean.
5. Add a script to create circle.json by taking clusters.json (output from 2) as an input.

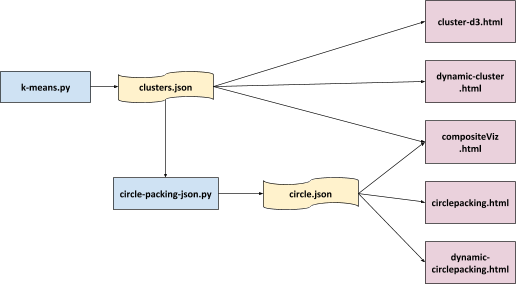


Fig 9: Workflow of clustering and visualization after adding K means

Experiment on clustering was done using 2 dataset, a smaller contains about 50 files and a larger contains about 500 files. Screenshots of clusters and circle packing of smaller dataset are as follows.

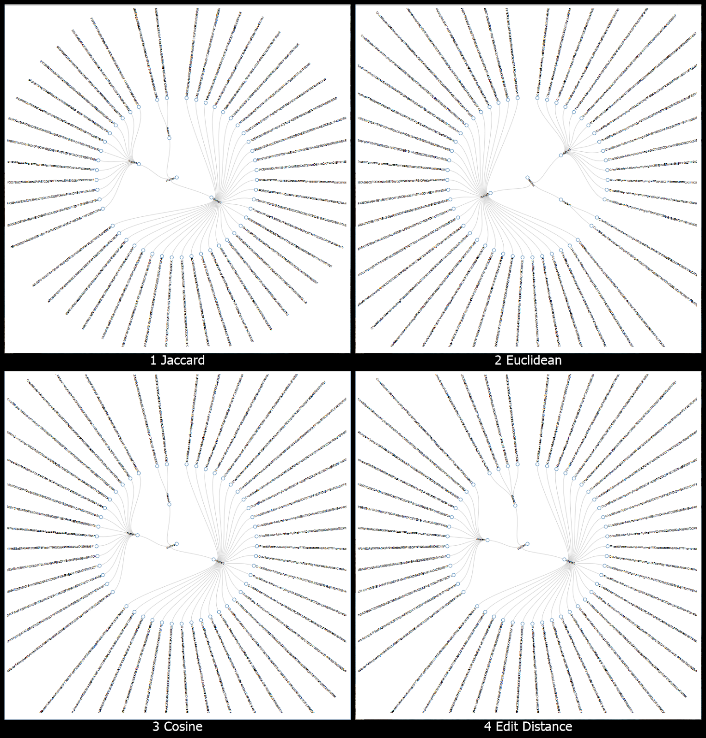
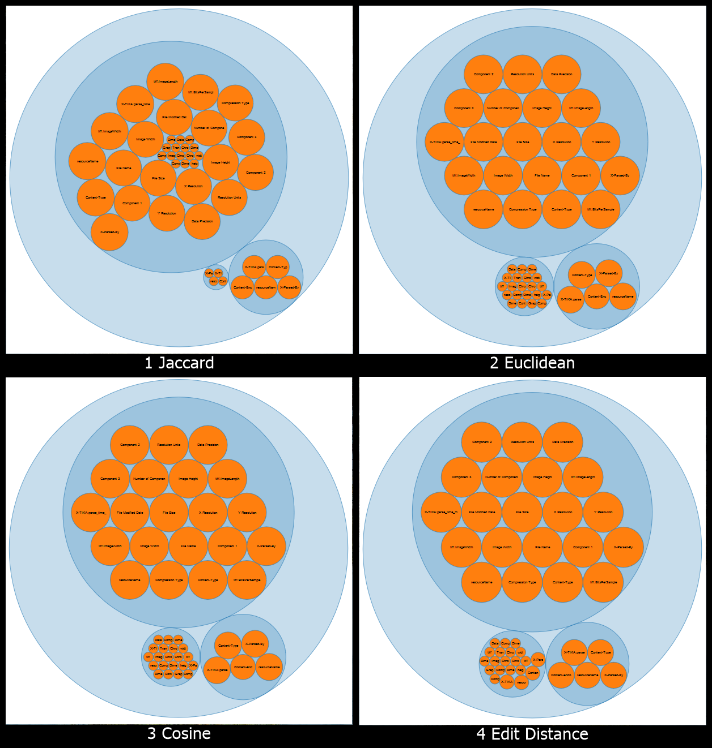
 

Fig 10: Clustering and Circle Packing of smaller dataset

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. If 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 we can see that each cluster indicates its type neatly. For larger dataset, though it’s not as neat as it be in smaller dataset but it still resemble the type. The detailed result is as followed.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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 1: Accuracy in Tika Similarity using various distance measures (Edit Distance gives highest accuracy in the larger dataset)

1. **Content Based MIME Detector**
2. In “filetypeDetection” project, the author use R scripts to train a neural network model to classify whether a file is from a specific type or not by using its byte frequency distribution. Classes to read this model and do type prediction based on the model are also implemented in Tika, so we will stick with this approach.
3. To train a model, first we prepare the training dataset. We will build the model to classify "application/xhtml+xml" type. We select 5000 files of this type to be positive examples. We also select 1000 files from each other 5 file type ("application/pdf", "image/jpeg", "application/xml", "text/html", "text/plain") to be negative examples. The validation and test dataset are built the same way.
4. Byte frequency distribution of each file is calculated as feature vector and also labeled positive or negative according to its type. We feed this data to our modified R script and a trained neural network model. We then feed the model into Tika (using *org.apache.tika.detect.NNExampleModelDetector*) and compare the result against Tika default MIME detector.
5. The accuracy of training, validation and test dataset are 98.27, 85.75 and 69.43 percent respectively which conform to the result of training script. The reason why the test accuracy is not very high might be that we only use 5000 files as training data and byte frequency distribution of files from the type "application/xhtml+xml" may vary. Detailed result is as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Train Dataset** | | **Actual** | | **Accuracy (%)** |
| **application/xhtml+xml** | **others** |
| **Predicted** | **application/xhtml+xml** | 4876 | 49 | 97.52 |
| **others** | 124 | 4951 | 2.48 |
|  | | | | 98.27 |
| **Validate Dataset** | | **Actual** | | **Accuracy (%)** |
| **application/xhtml+xml** | **others** |
| **Predicted** | **application/xhtml+xml** | 4205 | 627 | 84.1 |
| **others** | 795 | 4373 | 15.9 |
|  | | | | 85.78 |
| **Test Dataset** | | **Actual** | | **Accuracy (%)** |
| **application/xhtml+xml** | **others** |
| **Predicted** | **application/xhtml+xml** | 2427 | 484 | 48.54 |
| **others** | 2573 | 4516 | 51.46 |
|  | | | | 69.43 |

Table 2: Content based MIME Detector