

CSCI 599 - Content Detection & Analysis of Big Data

Brian Cohn

Hang Guo

Dhruv Bhatia



Code and instructions for viewing results are available here:

<https://github.com/Kaspect/polar/blob/master/README.md>

Overview

Here we document how we approached a challenging document classification problem across an exceptionally large dataset. We performed a MIME diversity analysis of the TREC-DDPolar dataset.

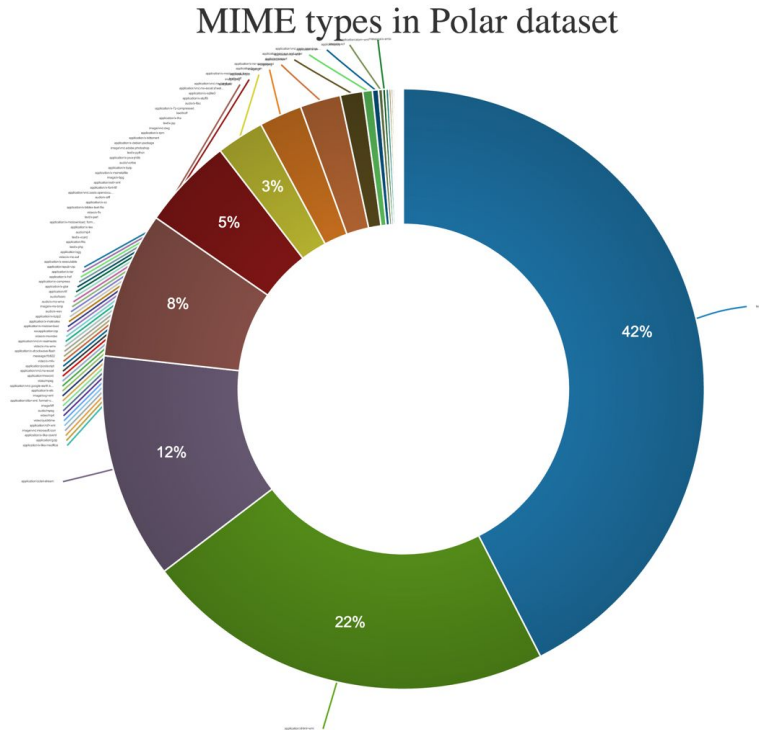


Figure X. Mimeypes and their percentages in the polar dataset (as defined on the chrismattmann/trek-dd-polar repository). With so many mimetypes, it was difficult to see them all in the piechart. We leveraged the d3pie.org styling so we would have a cleaner chart view. That said, we still wanted to understand how the file counts were distributed across file types; as a supplement, we generated some summary statistics to tell us more about the data within:

| Min | 1st Quarter | Median | Mean | 3rd Quarter | Max |
|-----|-------------|--------|-------|-------------|--------|
| 1 | 5 | 39 | 18140 | 328 | 739600 |

Table X. Statistical distribution of the mimetype counts across all filetypes observed (including application/octet-stream). We generated this summary with a simple command in R on our permuted JSON file:

```
summary(read.csv('polar_mime_counts_converted.csv', header=TRUE))
```

3.4a

Perform Byte Frequency Analysis on at 15 (14+ application/octet-stream) MIME types present in TREC-DD-Polar dataset

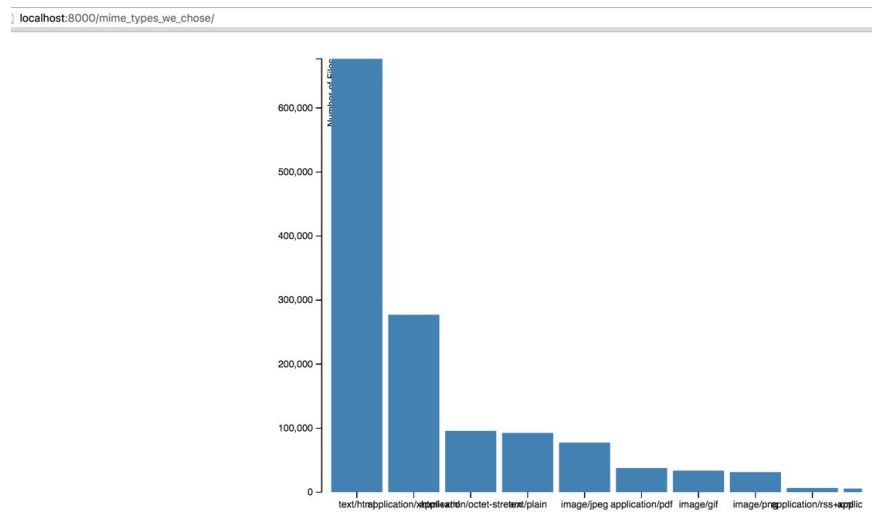


Figure X. The number of files for each of the 15 filetypes we chose to investigate. Application files we used were atom+xml, dif+xml, gzip, octet-stream, pdf, rdf+xml, rss+xml, xhtml+xml, xml. Image files were gif, jpeg, png, vnd.microsoft.icon. Text files were html, and plain.

To generate the BFA histograms, we ran it for each filetype individually.

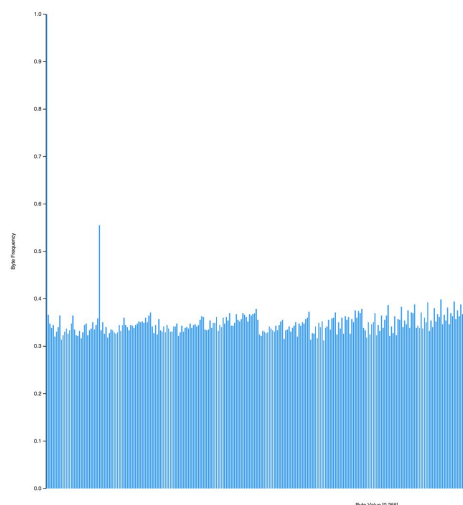


Figure X. BFA Fingerprint for image/png files in the Polar dataset. Available at http://localhost:8000/BFA_Dhruv/d3_histograms/ after running python -m http.server in the main repo folder.

















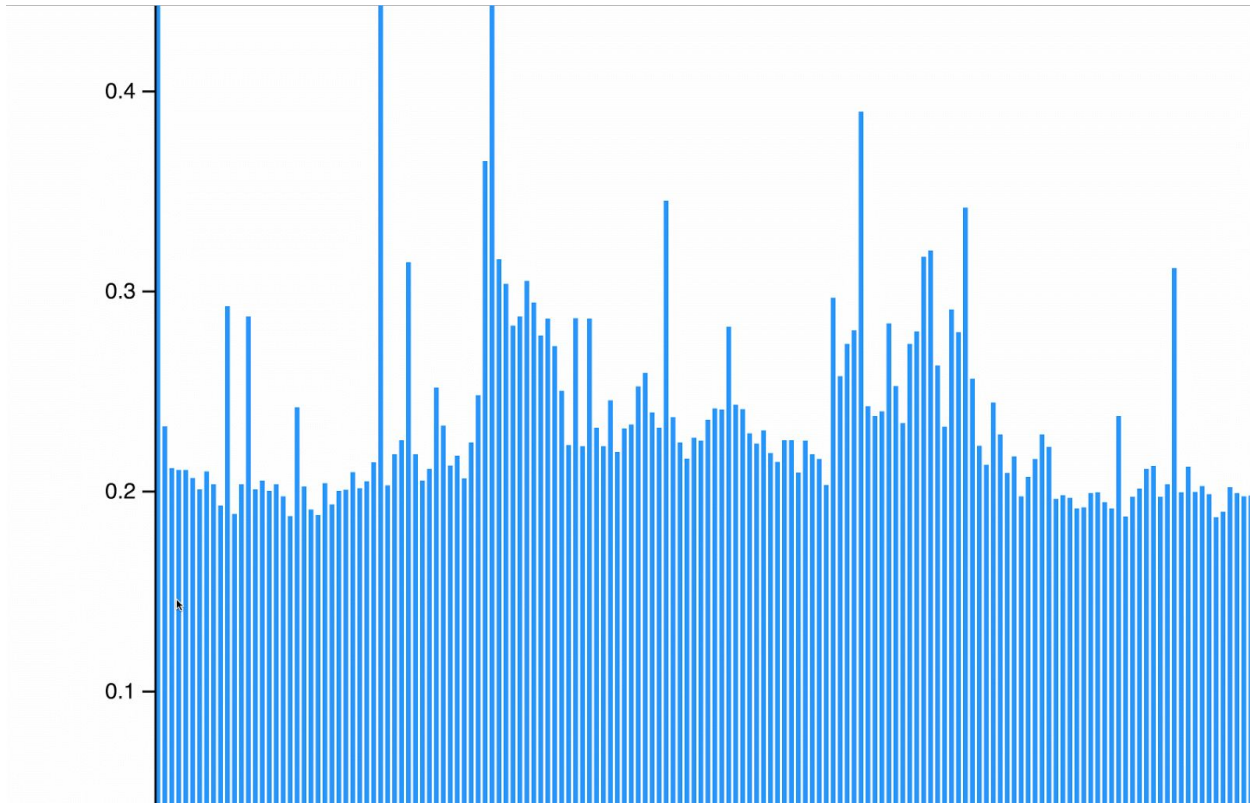
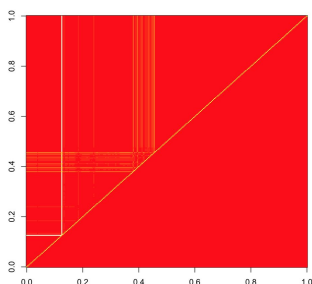
| |
|--|
|  application_atom+xml_signature.json |
|  application_dif+xml_signature.json |
|  application_gzip_signature.json |
|  application_octet-stream_signature.json |
|  application_pdf_signature.json |
|  application_rdf+xml_signature.json |
|  application_rss+xml_signature.json |
|  application_xhtml+xml_signature.json |
|  application_xml_signature.json |
|  conglomerate_15_types_bfa_distributions.json |
|  image_gif_signature.json |
|  image_jpeg_signature.json |
|  image_png_signature.json |
|  image_vnd.microsoft.icon_signature.json |
|  text_html_signature.json |
|  text_plain_signature.json |

Table X. After getting one of the BFA plots, we decided it would be best to have all 15 on the same page; we combined the fingerprint data from each filetype and combined it into one json document (highlighted)



<Brian make 15 BFC plots>



<Brian make 15 BFDmatrix heatmap plots from Json data>

<brian make 15 heatmaps for BFC plots from csv data>

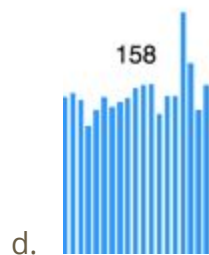
<Brian make FHT header for all 15>

<brian make FHT footer for all 15>

1. S3
 - a. We used the Amazon Web Services Command Line Tool to download the entire dataset. We successfully downloaded a total of ~1.3 million files.
2. Tika and Mime type ground truth
 - a. We checked out tika source code from git and compile it.
 - b. We used executable bash script to run tika with -m option to get MIME type ground truth for ~1.3 million file. The script also caught and recorded ~8k file that tika failed to parse.
 - c. To speed up and parallelize the parsing process, we split the task over 5 boxes mounted with NFS(Network File System).
3. 15 Mime Types we chose
 - a. We chose the top 15 frequent Mime types in polar datasets. After discussing with Dr Chris Mattmann, we count the ~8k files that tika failed to parse as application/octet-stream type
 - b. We chose: text/html, application/xhtml+xml, application/octet-stream, text/plain, image/jpeg, application/pdf, image/gif, image/png, application/rss+xml, application/xml, application/dif+xml, application/atom+xml, image/vnd.microsoft.icon, application/gzip, application/rdf+xml
4. D3
 - a. We set up multiple D3 websites using CSV and JSON input. We selected visualizations that show the data in an efficient manner. We opted out of using a Pie Chart to show the distribution of MIME types in the Polar dataset because Pie Charts are notorious for being difficult to interpret.¹
5. BFA 14 types+1
 - a. To generate byte frequency signature, we used a python script to read files for the same MIME type, count byte in them, average the byte count over files and normalize the byte count with the max byte count.

¹https://blogs.oracle.com/experience/entry/countdown_of_top_10_reasons_to_never_ever_use_a_pie_chart

- b. To speed up and parallelize signatures generating, we split the task over 5 boxes mounted with NFS.
- c. We designed a tool-tip rollover for the byte number to aid in quick visualization of the different distributions across each fingerprint. (see 4.b. below)



6. Byte Frequency Distribution Correlation

- a. To generate the Byte Frequency Distribution Correlation, we used the python script that calculated the correlation factor between the new byte difference and the average byte difference in the fingerprint which is represented in the form: $\text{math.exp}(-1 * (\text{math.pow}(x, 2)) / (2 * (\text{math.pow}(\text{sigma}, 2))))$ where x is the difference between the new byte value frequency and the average byte value frequency in the fingerprint, sigma is a constant with value 0.0375, exp is the exponent function (e) and pow is the power function in maths.
 - i. BFD Plots can be seen by running `python -m http.server` within the BFD folder, and viewing the distributions. For example, see [***bfd_json/pdf.html***](#).
- b. Once the input file's correlation factor for each byte value is obtained, these values need to be combined with the correlation strengths in the fingerprint.
- c. According to our analysis, If the difference between the two frequency scores is very small, then the correlation strength should increase toward 1. If the difference is large, then the correlation strength should decrease toward 0. Therefore, if a byte value always occurs with exactly the same frequency, the correlation strength should be 1. If the byte value occurs with widely varying frequencies in the input files, then the correlation strength should be nearly 0.

7. Byte Frequency Cross Correlation

- a. We built BFC Distribution Correlation matrix for the 15 MIME types we chose to understand the correlation between each byte pairs.

- b. We put number of files we used at the matrix's diagonal. For cell (i,j), we put the difference of occurrence between byte i and j. For cell (j,i), we put the normalized value of (i,j) .
- 8. File Header Trailer
 - a. For building the header and trailer profiles, we took 4, 8 and 16 header and trailer bytes and then built 2 two-dimensional arrays i.e. $H \times 256$ and $T \times 256$ where H and T are the respective header and trailer bytes to be analyzed.
 - b. An individual file's header array is filled by looping through each byte in the header, from byte 0 to byte H – 1. For each byte position, the array entry corresponding to the value of the byte is filled with a correlation strength of 1. All other byte value entries in that row are set to 0. After the entire array is filled, each byte position row is filled with a single 1 corresponding to the byte value at that byte position.
 - c. Similar process is followed for analyzing the trailer.
 - d. We have also analyzed one specific case where input file is shorter than the header and the trailer lengths. In this case, the fields in the missing byte position rows are filled with the value -1 to signify no data.
 - e. Our FHT script worked on sample jpeg data but failed to scale up and work on the full dataset. In the next project, we'll try to use parallel processing to scale up.
- 9. Update Tika MIME REPO
- 10.