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

TEAM – 20

Team Members
Shashank Vernekar – 1698224250 Rashmi Nalwad – 2021179341
Aditya Ramachandra Desai – 5246496115
MS Computer Science Spring 2016
Viterbi School of Engineering
University of Southern California
Los Angeles, California USA

1. Apache Tika Installation

Apache Tika, version 1.12 ships with core bindings in Java. This was used in the project to detect MIME types of files in TREC Polar data set.

2. D3.js Installation

Data-Driven Documents (D3) [1] is the JavaScript library for producing the dynamic and interactive data visualization in web browser. We used D3.v3.min to visualize the output of BFA (Byte Frequency Analysis), BFC (Byte Frequency Distribution and Cross Correlation) and FHT (File Header Trailer) Algorithm results.

3. The TREC –DD-Polar data.

a) The Polar data set was downloaded using Cyberduck application. Approximately around 60GB data was downloaded from the Amazon S3 bucket for performing Content Detection and Analysis of MIME Types.

b) Pie chart for existing Polar data set was created using the JSON file provided. “Chart” SVG of D3 was used to plot Pie chart layout. PieChart.html is a file to generate Pie chart visualisation.

(For details refer “Team20/D3 Visualisations”)

4. Byte Frequency Analysis[2,3]

a) Algorithm Explanation

i. We have used Tika to identify the MIME type of all the documents present in the TREC-DD-Polar data set. Tika could not identify the MIME types of certain documents and classified them as application/octet-stream. These were separated out for further analysis.

ii. For the files identified by Tika in step 1, fingerprint was generated by reading the bytes from the input file and computing corresponding byte frequencies. Hash Map was used with Key as MIME Type of file and value as the fingerprint so obtained.

iii. The results obtained in step ii were normalised and then combined using the formula given below:

$$\textbf{New Score} = \frac{\textbf{(Old FP Score * previous number of files) + New file score.}}{\textbf{Previous number of files + 1}}$$

This results in a simple average, where the previous fingerprints score is weighted by the number of files already loaded in the fingerprint.

iv. A correlation factor is calculated by comparing each file to the frequency scores in the fingerprint. The correlation factor of each byte value for an input file is calculated by taking the difference between that byte value’s frequency score from the input file and the frequency score from the fingerprint. We have used the following function to calculate the correlation

$$F(x) = e^{(-x^2/2\sigma^2)}$$

Where *F(x)* is the correlation factor between the new byte difference and the average byte difference in the fingerprint, and x is the difference between the new byte value frequency and the average byte value frequency in the fingerprint. We have made use of the latest version of Apache Commons Math library to calculate the Standard Deviations in most accurate manner.

v. Application octet stream files are read and matched with the fingerprint of each file obtained in step iii. The fingerprint with closest match (within range of +/- 0.1) of byte frequency will be considered as a potential MIME type for the unknown file.

vi. We considered 125000 (25% of 500000) files to perform BFA analysis. Fingerprint for 36 MIME Types are computed.

b) Above step outputs data.tsv files for each type which is used to plot visualization using D3 for 36 MIME file types as a line chart. The visualization is available at Team20/BFAOutput.

5. Byte Frequency Cross Co relation

a) Algorithm Explanation

While BFA compares overall byte frequency distributions, other characteristics of the frequency distributions are not addressed. BFC compares the cross-correlation, between byte value frequencies.

i. Tika was used for detecting the MIME types of the input file.

ii. For the files identified by Tika in step 1, fingerprint was generated by reading the bytes from the input file and computing corresponding byte frequencies. Hash Map was used with Key as MIME Type of file and value as the fingerprint (256*256 Matrix).

iii. The results obtained in step 2 were normalised and then combined using the formula given below

$$\textit{New Score} = \frac{\textit{(Old FP Score * previous number of files) + New file score}}{\textit{Previous number of files + 1}}$$

iv. If byte value i is being compared to byte value j, then array entry (i, j) contains the frequency difference between byte values i and j while array entry (j, i) contains the difference between byte values j and I which contains the correlation strength for the byte pair.

v. A correlation factor can be calculated for each byte value pair, by comparing the frequency differences in the input file to the frequency differences in the fingerprint. The Correlation Strength Average Frequency Difference correlation factors can then be combined with the scores already in the fingerprint to form an updated correlation strength score for each byte value pair. If no files have previously been added into a fingerprint, then the correlation factor for each byte value pair is set to 1.

$$\textbf{NewCorrStrength} = \frac{\textbf{(OldCorrStrength * previous number of files) + NewCorrFactor}}{\textbf{Previous number of files + 1}}$$

vi. File types that have characteristic cross-correlation patterns should have high assurance levels, while those that do not have characteristic cross-correlation patterns should have low assurance levels and the assurance level is computed as a simple average of the correlation strengths of each byte value pair. The higher the assurance level, the more weight can be placed on the score for that fingerprint.

6. Frequency Header Trailer Algorithm

a) Algorithm Explanation

i. Tika was used for detecting the MIME types of the input file. Files with MIME type application/octet stream were separated as unknown files for which MIME types were supposed to be identified.

ii. For the files identified by Tika in step 1, fingerprint was generated by reading the bytes from the input file and computing corresponding byte frequencies. Hash Map was used with Key as MIME Type of file and value as the fingerprint so obtained.
Fingerprint is obtained by reading first 4,8,12 bytes of header and 4,8,12 bytes of trailer in a matrix of size Header Length*256 and Trailer Length * 256 for Header and Trailer finger print generation respectively.

The results obtained in step 2 were combined using the formula

$$\textbf{New Score} = \frac{\textbf{(Old FP Score * previous number of files) + New file score.}}{\textbf{Previous number of files + 1}}$$

This results in a simple average, where the previous fingerprints score is weighted by the number of files already loaded in the fingerprint.

iii. Application octet stream files are read and matched with the fingerprint of each file obtained in step 3. Score for file header or trailer was computed using the formula

$$S = \frac{(C1*G1) + (C2*G2) + \ldots\ldots.. + (Cn*Gn)}{G1 + G2 + \ldots\ldots\ldots\ldots\ldots\ldots + G3}$$

Where C is the correlation strength for the byte value extracted from the input file for each byte position, and G is the correlation strength of the byte value in the fingerprint array with the highest correlation strength for the corresponding byte position.

iv. We considered 125000 (25% of 500000) files to perform FHT analysis. Fingerprint for 34 MIME Types are computed.

b) Above step outputs ‘fileType.tsv’ for each type which is used to plot visualization using D3 for 34 MIME types as a scatter plot. The visualization are available at Team20/FHTOutput

7. **a**) Based on our research on the resources available in the public domain of the internet, we could identify MIME type of the empty file as **application/x-zerosize** [4]. We have implemented the new parser and detector in the source code for Tika 1.12 (tikacore/src/main/java/org/apache/tika/parser/zerosize) and reported in TIKA-1883.

We identify the following file types, added them to Tika 1.12 mimetypes.xml. The issue of the same has been reported in TIKA-1884

i. **.SFDU** - Standard Formatted Data Unit [5].

The SFDU file format is added in the mimetypes.xml. These files can be identified by and the XML code snippet is here. The magic bytes are NJPL2I00 with offset as 4

```
<mime-type type="application/x-sfdu">
        <_comment>SFDU Documents File Detection, found during TREC Polar Data
Project USC March 1 2016</_comment>
        <magic priority="50">
                <match value="NJPL2I00" type="string" offset="4"/>
        </magic>
        <sub-class-of type="text/plain"/>
        <glob pattern="*.sfdu"/>
</mime-type>
```

ii. **.CDF** – Common Data Format.

The CDF file format is added in the mimetypes.xml. These files can be identified by and the XML code snippet is here. The magic bytes are CDF with offset as 0

```
<mime-type type="application/x-netcdf">
   <glob pattern="*.nc"/>
   <glob pattern="*.cdf"/>
    <magic priority="50">
     <match value="CDF" type="string" offset="0"/>
</magic>
```

The CDF [6, 7] and NetCDF [6, 7] are two different versions, with CDF being with parent of NetCDF, considering this hierarchy; we have added the above XML snippet.

b) Tika was recompiled with the new types mentioned in 7.a).Please find updated jar at Team20/Tika1.2UpdatedJar.

c) We wrote a Java program along with the new updated Tika 1.12 jar. This program identified the new added MIME types along with the existing file types and also the Tika GUI app displays appropriate Mime Type. (Please refer Team20/7c for Code.)

d) D3 visualization of the results obtained by running tika on Polar dataset after changes can be found at Team20/D3 Visualisation.

8. Tika Similarity

a) Used Tika Similarity to understand the similarity between the documents in the given data set.

b) The results of the Tika Similarity for Jaccard, Edit Cosine Distance and Edit Similarity are available here.

c) The Jaccard is calculated on the size of the intersection divided by the size of the union of sample documents, so we can see a large very large circle. In cosine distance similarity we can notice that the clusters being grouped by the metadata. We can see a small cluster at the left in the circle packaging. This small cluster has all the metadata that is mostly related to multimedia, for instance Exposure mode, Self-timer, Camera Type etc. But the similar behaviour is not observed in edit distance circle packaging.

9. Application of Content Based MIME Detector into Tika

Added a detector in the path tikacore/src/main/java/org/apache/tika/parser/zerosize to detect empty files and classify them as MIME type application/x-zerosize. The Tika source code is updated with these changes and the updated Tika JAR is available at (Team20/9).

10. Observations

We are listing some of the observations that we have made on this dataset:

a) The results of BFA were very poor when compared to FHT. We could identify the file type using FHT than BFA.

b) Tika is more capable of identifying the MIME type for any given data set. There are some MIME types which Tika is not able to identify. There are possible answers on why this is so. Tika will not consider empty files for Mime Type identification. This particular Polar Data Set has high percentage of empty files. We have identified the MIME type of empty files as application/x-zerosize. Also we cannot rule out the possibility of how the data crawling might have actually performed, the configurations of the web crawler is also an important point that needs to be considered here. Apart from that there might be some proprietary MIME types for which Tika cannot detect them.

c) Considering the large amount of the data we deal in Data Science, Tika considers the magic bytes and the header bytes to identify the MIME type along with glob pattern. We have added the new MIME types based on byte patterns and this will improve the Tika's performance.

d) We noticed that there was no MIME priority precedence error.

e) We have also noticed some of the documents having the Chinese characters (GB 2312) embedded with English and Tika is identifying them as application/octet-stream. We are exploring the options of using Tika's language detection capabilities to identify such documents and classify them separately.

11. References

[1] https://github.com/mbostock/d3/wiki/Gallery
[2] https://www.computer.org/csdl/proceedings/hicss/2003/1874/09/187490332a.pdf
[3] Mason McDaniel, Automatic File Type Detection Algorithm, Master's Thesis, James Madison University, 2001.
[4] https://marc.info/?t=101528310400007&r=1&w=4
[5] http://www.srl.caltech.edu/galileo/ipf/SFDUformat.html
[6] http://www.unidata.ucar.edu/software/netcdf/docs/
[7] http://cdf.gsfc.nasa.gov/