Data Challenge 2024

Experimentation, Analysis, and Conclusions

James Kane

06 May 2024

1. Purpose

This project is part of an open-ended data challenge event hosted by EOP, with the purpose of developing tools and methods to add to their provided toolbox.

2. Dataset

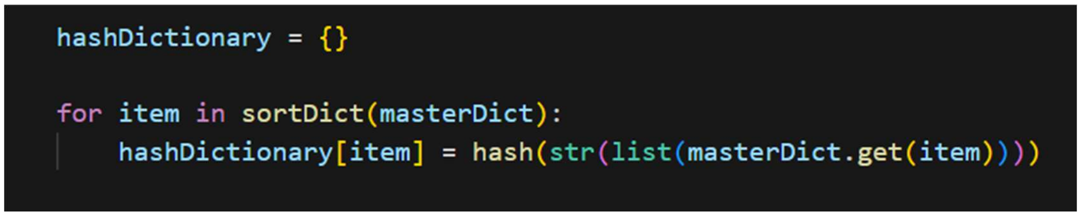
To better understand some of the objectives explored in this data challenge, it is best to explain the dataset, its contents, and how the dataset was perceived in this experiment.

The dataset contains three major components: raw data, parsed data, and media files (e.g. audio, video, images, documents, etc.). The raw data were in excel (.xlsx) format with each file containing several different spread sheets. These spreadsheets had titles like “Device Information”, “Audio”, “Contacts”, “Video”, “Call Log”, etc. The parsed data was just the raw data in JavaScript Object Notation (JSON) format. The media files contained various file extensions but were well organized into their respective extension classes (i.e. .mp4 files were in a folder titled ‘Video’ and .mp3 files were in a folder titled ‘Audio’).

3. Preliminary Analysis

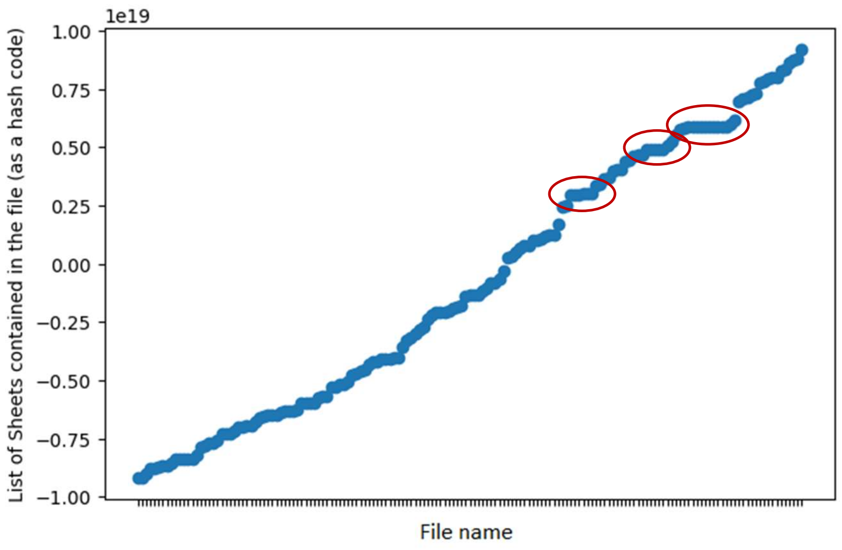
Since the excel sheets contained information like International Mobile Equipment Identity (IMEI) numbers, call logs, and various phone records, it is highly probably that each .xlsx or JSON file pertains to a specific device, and that the data was skimmed from these devices and output into an excel file. This is what is assumed for the remainder of the experiment.

At first, it appeared that there may have been a similar structure to the excel files, enough so that they could be grouped together and processed together. The list of sheet names in each excel file were converted to a hash value and plotted on a graph (below). If there were definitive clusters of hash values that were the same, then the excel files could be organized into categories. This might imply that devices were skimmed in the same manner and came from the same source.



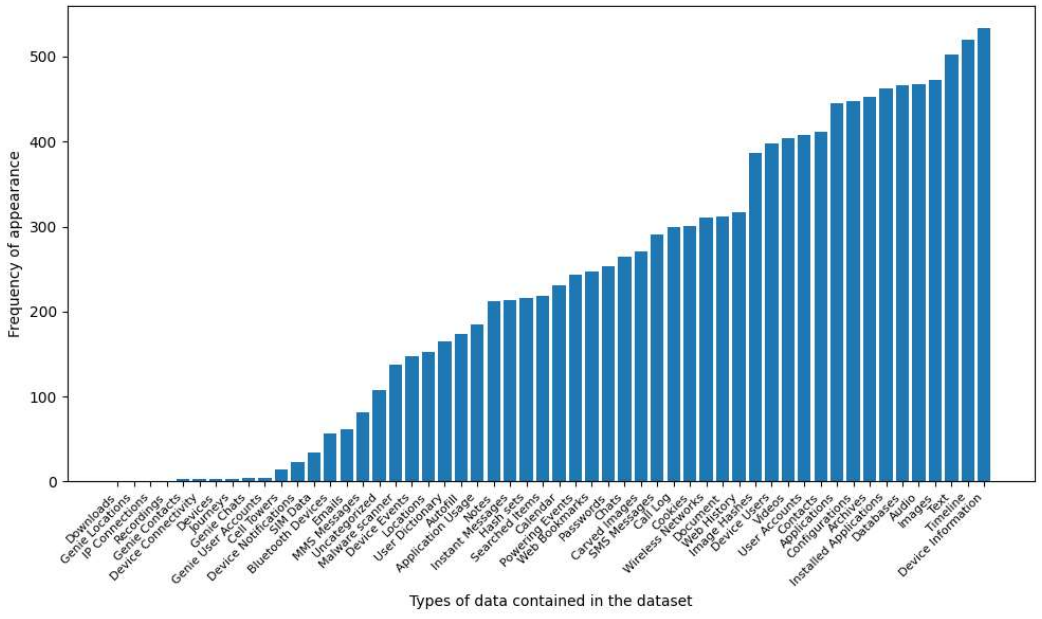
```python
hashDictionary = {}

for item in sortDict(masterDict):
    hashDictionary[item] = hash(str(list(masterDict.get(item))))
```

(Figure 3.1) Above is the python code used to calculate the hash value of the list of sheet names in each excel file, read as a string. The hash value is assigned to its respective file (by name) in a dictionary for later processing (@jkane, @parens, @jhoskisson, 2024).



(Figure 3.2) Above is the graph plotting the files by the hash value output of the contained sheet names. There are few large clusters where the hash values appear to be equal circled in red (@jkane, @parens, @jhoskisson, 2024).

Looking at figure 3.2, there are few significant features from the plotted data where files have the exact same associated hash value, which would imply that these files are mostly unique in their contents. It is good, however, that this data covers a wide spectrum of device contents as it could more accurately reflect a larger population. Each device has a user, who is inherently unique. Instead of evaluating the spreadsheets in categories, the data must be evaluated based on which of their aspects are most significant.



(Figure 3.3) Above is a bar chart of sheet names present in the excel files, plotted by their frequency that they appear in the dataset (@jkane, @parens, @jhoskisson, 2024).

The frequency at which the sheets appeared in the dataset were plotted from least to greatest. To confirm the methodology, the ‘Device Information’ sheet is something that is present in every excel file, and so it is true that it should be the most frequent. Among the most frequent sheets are contents that pertain to media files: audio, images, and video. This became the basis for the proceeding experiment.

Within the dataset, there remained media files which were not immediately connected to any excel file (i.e. any device). Within the sheets pertaining to media files, there were columns which listed some "MD5" hash values.

> The MD5 hash algorithm is a commonly used function for validating data integrity. The name is derived from Message-Digest algorithm 5. The algorithm is applied against the source data (typically a file and its content) in order to generate a unique, 128-bit hash value… (FADGI).

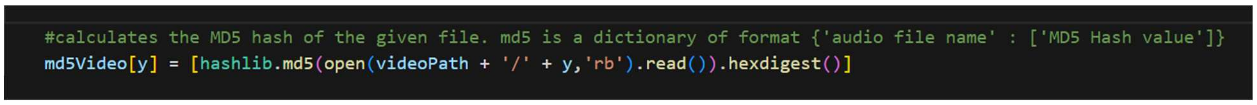
It should be noted that this algorithm is not one-to-one. One-to-one means that for every element in the domain, there is another unique element in the codomain (range). Instead, the MD5 algorithm is surjective or "onto", which means that for every element in the domain, there is at least one element in the codomain (this implies the output is not necessarily unique). It is entirely possible for two different media files to have the exact same hash value output from the MD5 algorithm, however the dataset is small enough that the probability of this happening in negligible.

4. Methodology

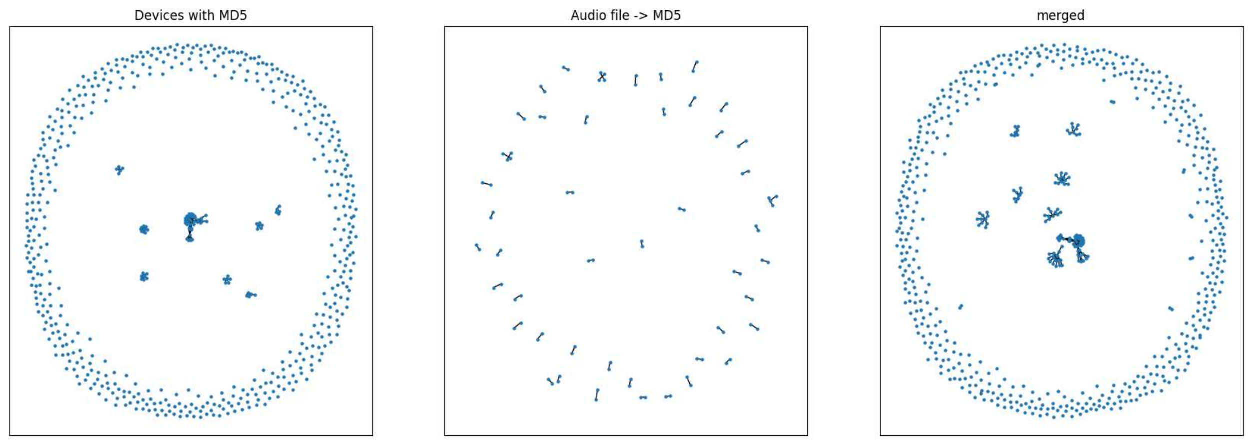
If the hash values listed in the spreadsheets pertained to the media files in the original dataset and were indeed calculated using the MD5 algorithm, then each media file could be connected to the device(s) it came from by evaluating its contents to a hash value via the MD5 algorithm as well. The procedure for the code, pertaining to the audio files only, is as follows:

1. Create a dictionary containing the file name and their respective hash values. Write it to a text file for later use.
2. Calculate MD5 Hash Values: Calculate the MD5 hash values for each audio file in the specified media folder and store them in a dictionary. Write it to a text file for later use.
3. Find Common MD5 Hash Values: Compare calculated MD5 hash values with those found in the dataset to identify common hash values.
4. Write Matched MD5 Values to File: Write the matched MD5 hash values to a text file.
5. Prepare Final Dataset: Initialize a new dictionary for drawing graphs and import data from text files into this dictionary. This dictionary only includes devices with matched hash values.
6. Generate Graphs: hash values should be connected to at least one audio file and one device.
7. Output: Display a plot showing the relationships between audio files based on MD5 hash values along with the largest components of the network.

This process was repeated for the video files as well, and the final graph contains both datasets from the video and audio methods.


```
#calculates the MD5 hash of the given file. md5 is a dictionary of format {'audio file name' : ['MD5 Hash value']}
md5Video[y] = [hashlib.md5(open(videoPath + '/' + y,'rb').read()).hexdigest()]
```

(Figure 4.1) Above is the method and code used to calculate the hash value of the contents of each audio file via the MD5 algorithm (@jkane, @parens, @jhoskisson, 2024).

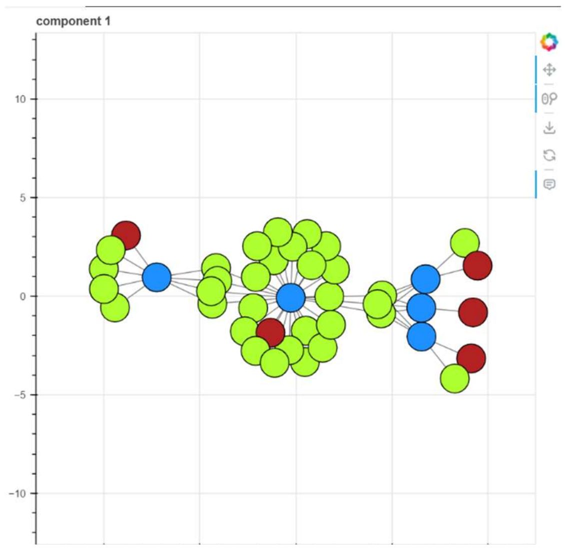


(Figure 4.2) Above are the 3 preliminary graphs created using NetworkX (Python library). ‘Devices with MD5’ (left) plots devices with common/shared hash values. The devices on the outside appear to not have common hash values with any other device. ‘Audio -> MD5’ (center) is just a one-to-one evaluation of audio files to their respective MD5 hash value. The ‘merged’ graph combines the previous two graphs into one, connecting devices to audio files via a common hash value (@jkane, @parens, @jhoskisson, 2024).

5. Discoveries

In the first experiment (pertaining to just the audio files), the graphs revealed that some of the audio files seemed to be clustered together based on naming convention or file extension. For example, the third largest component revealed that ‘file_37.json’ and ‘file_38.json’ shared 6 advanced audio coding (abbr. “.aac”) files. These files also shared a similar naming convention.

In the largest component of the graph, 26 devices shared an audio file with the name “____ ___ _____.mp3”. Seemingly, the title of the audio files was either corrupted or removed intentionally. The shared file ended up being a song, so perhaps this was a popular song from the culture that these devices originated from.



(Figure 4.3) Above is a plotted graph of the largest component of the network. The blue nodes are hash values, the red nodes are audio files, and the green nodes are devices. This plot was created using Bokeh (Python library) (@jkane, @parens, @jhoskisson, 2024).

The 8 largest components of the network clustered audio files together with similar naming convention, so it could be the case that some Natural Language Processing (NLP) on just the file names could lead to similar results for grouping the files together. However, titles of files are much more likely to be changed based on the user, and changing the file name will not change its MD5 hash value.

The procedure for the video files resulted in similar results; similarly named video files and file extensions were again clustered together, and 'file_37.json' and 'file_38.json' were once more linked via common video files, this time sharing 3 different video files. As it turns out, these two devices share the same International Mobile Equipment Identity number (IMEI) and many other traits within the general device information, so it is most likely that they are the same device.

Sharing media files does not mean that the connecting media file was sent directly from one device to another. Take the song shared amongst the 26 devices mentioned earlier. It's more probable that all 26 devices downloaded the song from the same source (e.g. iTunes, Spotify, etc.) than the devices directly sending that mp3 file across 26 different devices. Although connecting devices can be critical for identifying major components in social network analysis, its is equally important to determine the strength or meaning of said connection.

In general, the interconnectedness of any set devices might indicate a similarity between the users. The dataset that this experiment was performed on appears to be a subset of a much larger dataset, and so the components of these networks are far too small to draw any conclusions with a high degree of probability. Still, it remains entirely possible to link any number devices if they share media files using the MD5 algorithm.

6. Conclusion

Using the MD5 algorithm, media files were able to be linked to devices (.json files) in the dataset, thus creating a network of devices that shared media files. In the scope of identifying interconnectedness between devices, the methodology performed in this experiment proved to be useful. It should be noted that devices can further be connected by parsing through their call logs, contacts, and other media files which could lead to larger networks and stronger links between devices.

References

Federal Agencies Digital Guidelines Initiative (FADGI). (n.d.). *Term: MD5 (checksum)*. MD5 (checksum) - Glossary - Federal Agencies Digitization Guidelines Initiative. https://www.digitizationguidelines.gov/term.php?term=md5checksum

@jkane, @jhoskisson, @parens (2024) team007-data-challenge-2024 (Version 1.0) [Source Code]. https://gitlab.wildfireworkspace.com/jhoskisson/team007-data-challenge-2024/