The Whatsapp Investigator

The forensic tool that investigators deserve AND also need right now!

- Ajay Shaan Shanmugam

I wanna give it a spin! How do I begin?

Your interest delights me! Please make sure you have the following first.

1. Anaconda (for accessing jupyter notebook)
2. Following Python libraries:
   a. pandas
   b. numpy
   c. datetime
   d. re
   e. difflib
   f. bokeh
   g. itertools
   h. os
   i. matplotlib

Whew! Now that everything's set up, how do I explore this?

Great! Now the fun begins! Pay attention, my friend.

All functions of the Whatsapp Investigator work with pandas dataframes. Pandas dataframes are very wieldy, super cool tables (fun fact: they have numpy arrays under the hood) that hold data.

The Whatsapp Investigator can do the following:

Data Extraction:

1. Extract all texts between 2 timestamps from a dataframe
2. Extract texts containing any of given list of keywords from a dataframe
3. Chronologically order texts in a dataframe
4. Extract texts containing phone numbers from a dataframe

5. Extract texts containing email ids from a dataframe
6. Extract texts containing web hyperlinks from a dataframe

Data Visualization:

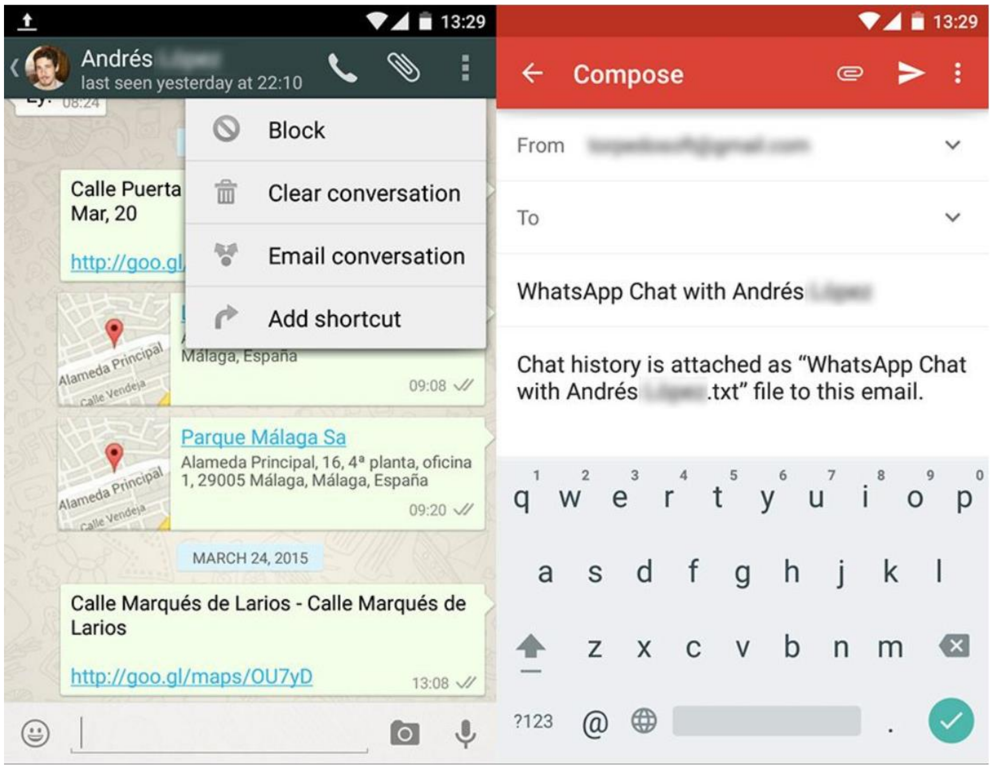
7. Visualize number of texts sent to and received from each contact
8. Visualize number of texts sent to and received from each contact per date
9. Visualize number of search hits on texts sent to and received from each contact
10. Visualize number of search hits on texts sent to and received from each contact per date
11. Visualize wordclouds of frequent words in conversation with each contact.

Whoa! Care to detail?

With pleasure! Please allow me to walk you through each of these little utilities.

Before we explore these, we have two more itty bitty things to do.

First, please email desired whatsapp chat conversations without media to your email id.



Calle Puerta
Mar, 20
http://goo.gl

Málaga, España
09:08

Parque Málaga Sa
Alameda Principal, 16, 4ª planta, oficina 1, 29005 Málaga, Málaga, España
09:20

MARCH 24, 2015

Calle Marqués de Larios - Calle Marqués de Larios
http://goo.gl/maps/OU7yD
13:08

From [redacted]

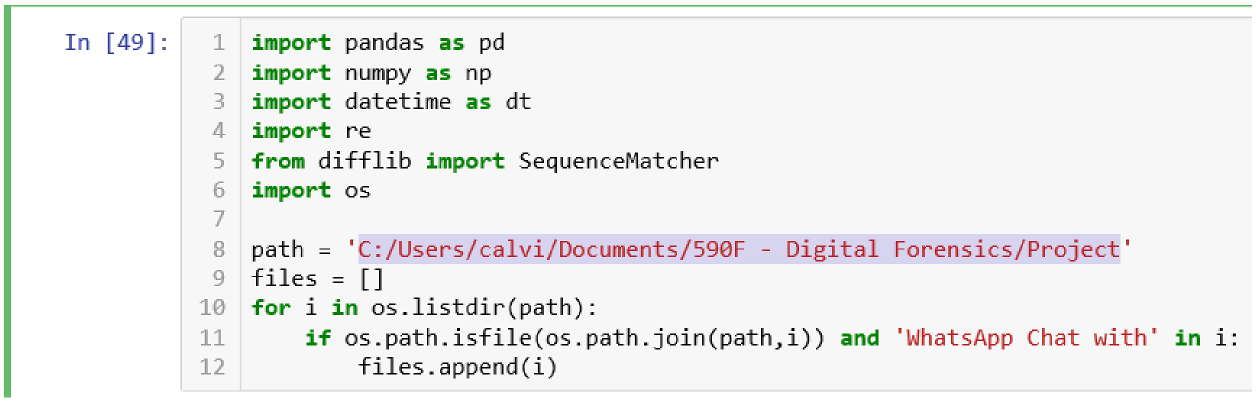
To

WhatsApp Chat with Andrés [redacted]

Chat history is attached as "WhatsApp Chat with Andrés [redacted].txt" file to this email.

Download all these txt files into the desired folder.

Second, specify this folder name as path in the highlighted part of the Whatsapp Investigator Ipython notebook.


```
In [49]:
import pandas as pd
import numpy as np
import datetime as dt
import re
from difflib import SequenceMatcher
import os

path = 'C:/Users/calvi/Documents/590F - Digital Forensics/Project'
files = []
for i in os.listdir(path):
    if os.path.isfile(os.path.join(path,i)) and 'WhatsApp Chat with' in i:
        files.append(i)
```

The second cell of the ipython notebook cleans data in these txt files, converts them into dataframes and creates a list of dataframes. The third cell creates the master dataframe aggregating ALL texts from all contacts.

Now let’s see how each function aids the investigation.

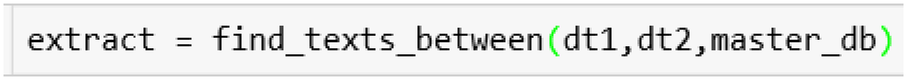
Note: Remember that all functions in data extraction gives one the flexibility to use any dataframe extract to be passed as an argument.

Data Extraction:

1. Extract all texts between 2 timestamps from a dataframe

Wanna extract all texts between 2 timestamps from the master dataframe?

Call find_texts_between with arguments: start timestamp, end timestamp, dataframe


```
extract = find_texts_between(dt1,dt2,master_db)
```

Wanna extract all texts between 2 timestamps from a dataframe of which contents you spent hours extracting?

Simply pass the timestamps and the required dataframe as arguments! This function returns dataframe containing all texts between given timestamps.

Here dt1 and dt2 should be strings of the form – MM/DD/YY, HH:MM in 24hr time format.
You may either specify these timestamps one above the other in a txt file called ‘textsbetween.txt’ in the same folder, or if you’re feeling adventurous, comment out cell 4 and define dt1 and dt2 with the start and end timestamps.

Why is this function cool?

This helps you extract texts between timestamps from any dataframe of your choice! Needless to say, this is an essential forensic tool for timeline analysis and zeroing in on artifacts.

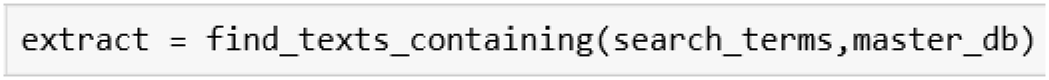
2. **Extract texts containing any of given list of keywords from a dataframe:**

I repeat, any dataframe extract can be passed. Alright, I won’t say it again for the other functions, I promise!

Specify search terms/keywords one below the other in a txt file ‘searchwords.txt’ in the same folder, or if you’re feeling adventurous, specify keywords as a list of strings and define it as search_terms.

Wanna extract texts containing any of the given list of keywords from a dataframe?

Simply pass the search_terms (list of keywords) and the desired dataframe. This function returns dataframe containing hits.



```
extract = find_texts_containing(search_terms,master_db)
```

Why is this function cool?
Unlike searching for a keyword in whatsapp on the phone, one can:

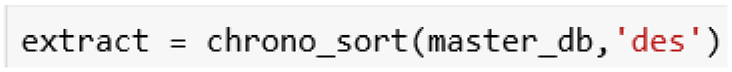
- Search for a list of keywords across texts
- Perform search across texts from any contact (or any desired dataframe extract, remember?)
- Search for texts containing similar words! Eg: Searching for ‘meme’ finds you texts containing perceivable misspellings of ‘meme’. Imagine searching texts based on keywords in a dyslexic suspect’s chat conversations!

3. **Chronologically order texts in a dataframe:**

   This is especially useful for chronological sorting of the master dataframe.

   How to use this function?

   Simply pass the desired dataframe and the sort order (ascending or descending). Returns chrono sorted dataframe.



```
extract = chrono_sort(master_db,'des')
```

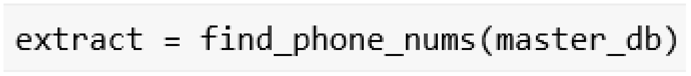
   Why is this function cool?

   Chrono sorting the master dataframe will give the investigator a good idea of what the suspect has been recently upto or who the suspect has been communicating with during the timeline of interest. Also, depending on the order, the investigator can analyze the timeline from both ends.

4. **Extract texts containing phone numbers from a dataframe:**

   How to use this function?

   Simply pass the desired dataframe as argument. Returns dataframe containing hits.



```
extract = find_phone_nums(master_db)
```

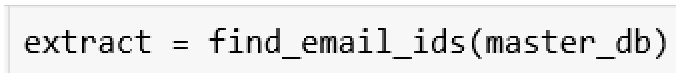
   Why is this function cool?

   Proves to be most useful for moments when our suspect has a lot of lady friends. Jk! It can find texts containing phone numbers regardless of the format. Eg: 413-695-xxxx or 413695xxxx or 1413695xxxx

5. **Extract texts containing email ids from a dataframe:**

   How to use this function?

   Simply pass the desired dataframe as argument. Returns dataframe containing hits.



```
extract = find_email_ids(master_db)
```

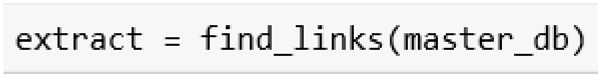
   Why is this function cool?

   Even though emailing someone in informal occasions is out of fashion, emails still are a prominent way of communication in business, and surprisingly among cyber criminals (especially the dabblers in technology). Therefore, extracting email contacts may give the investigator as many leads as phone numbers can.

6. **Extract texts containing web hyperlinks from a dataframe:**

   How to use this function?

   Simply pass the desired dataframe as argument. Returns dataframe containing hits.



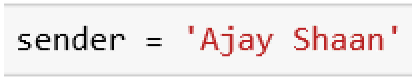
```
extract = find_links(master_db)
```

   Why is this function cool?

   One's true identity reveals itself shielded by the façade of his online persona. Nuff said. This function helps the investigator know the web links the suspect shared or was shared.

Data Visualization:

Note: Before running the visualizations, please set sender to suspect's name.

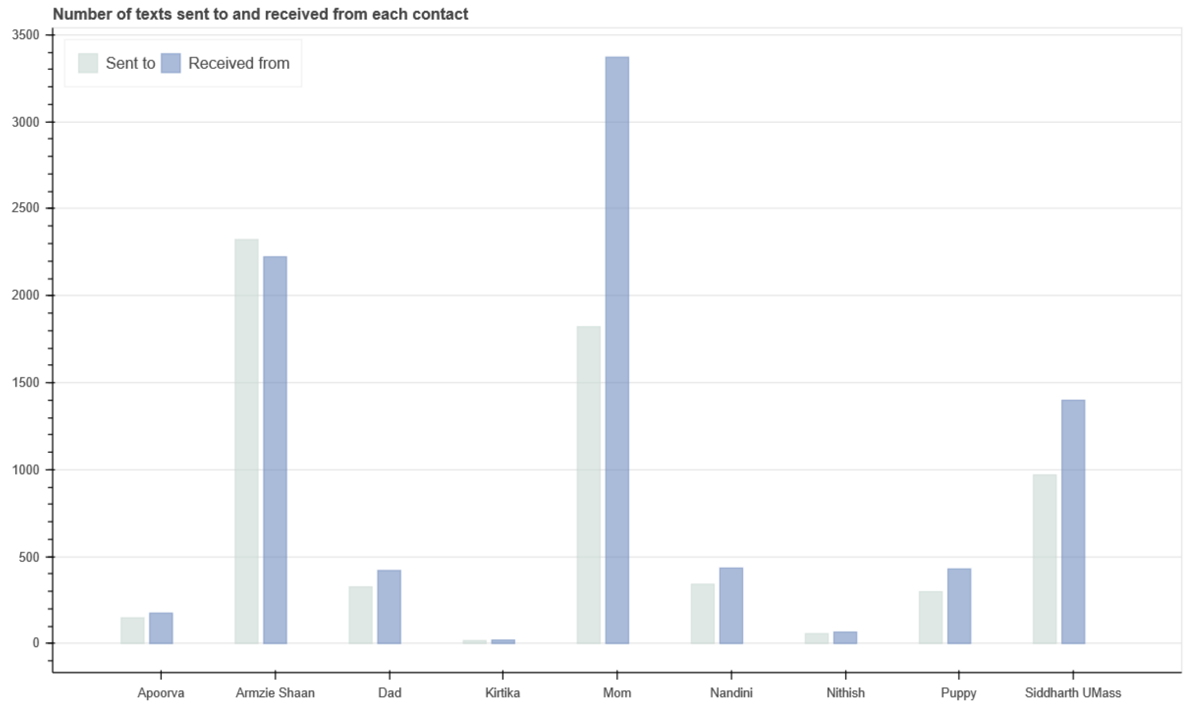


```
sender = 'Ajay Shaan'
```

7. **Visualize number of texts sent to and received from each contact:**

   This visualization tells you the text traffic between suspect and each of his contacts.

   How to use this?
   There's no need to invoke them. Just set the sender before running the visualizations!

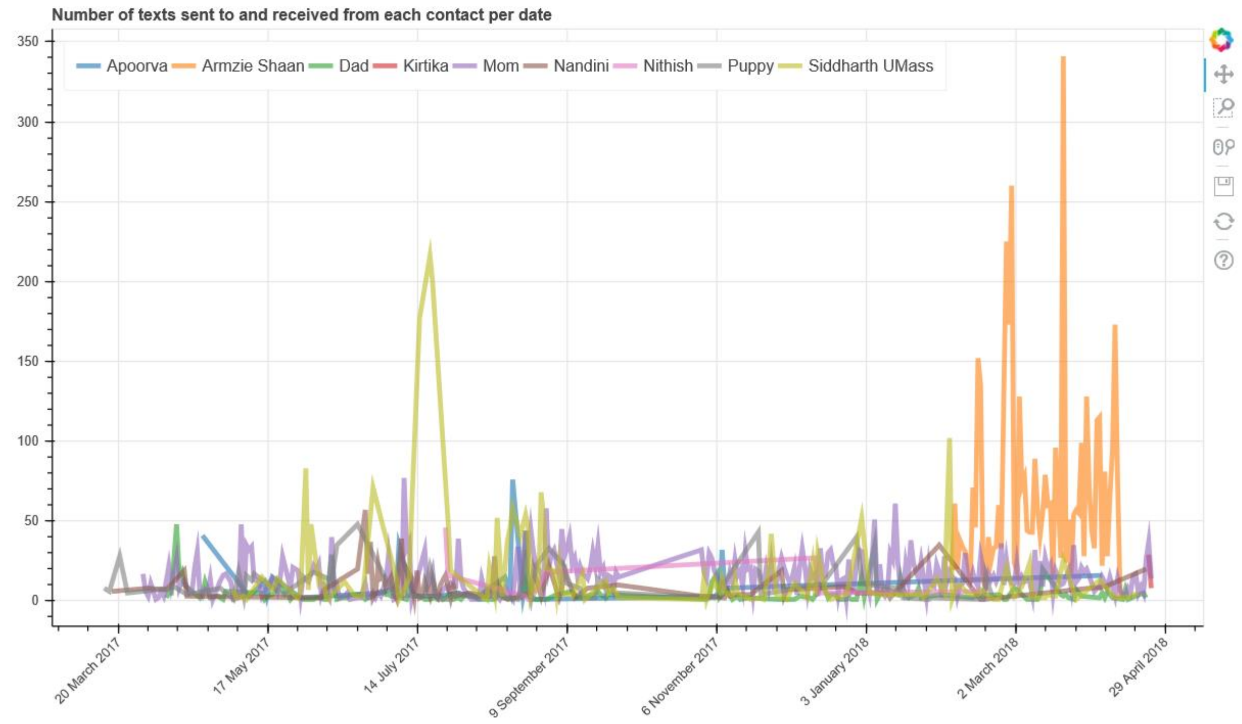


Why is this visualization cool?

This gives the investigator a quick overview of the two-way traffic of communication between the suspect and all his contacts. This helps him understand who the suspect interacts most with or otherwise.

8. Visualize number of texts sent to and received from each contact per date:

This visualization is a more granular view of the last one. Instead of just depicting two-way communication traffic with each contact, it visualizes this per date!

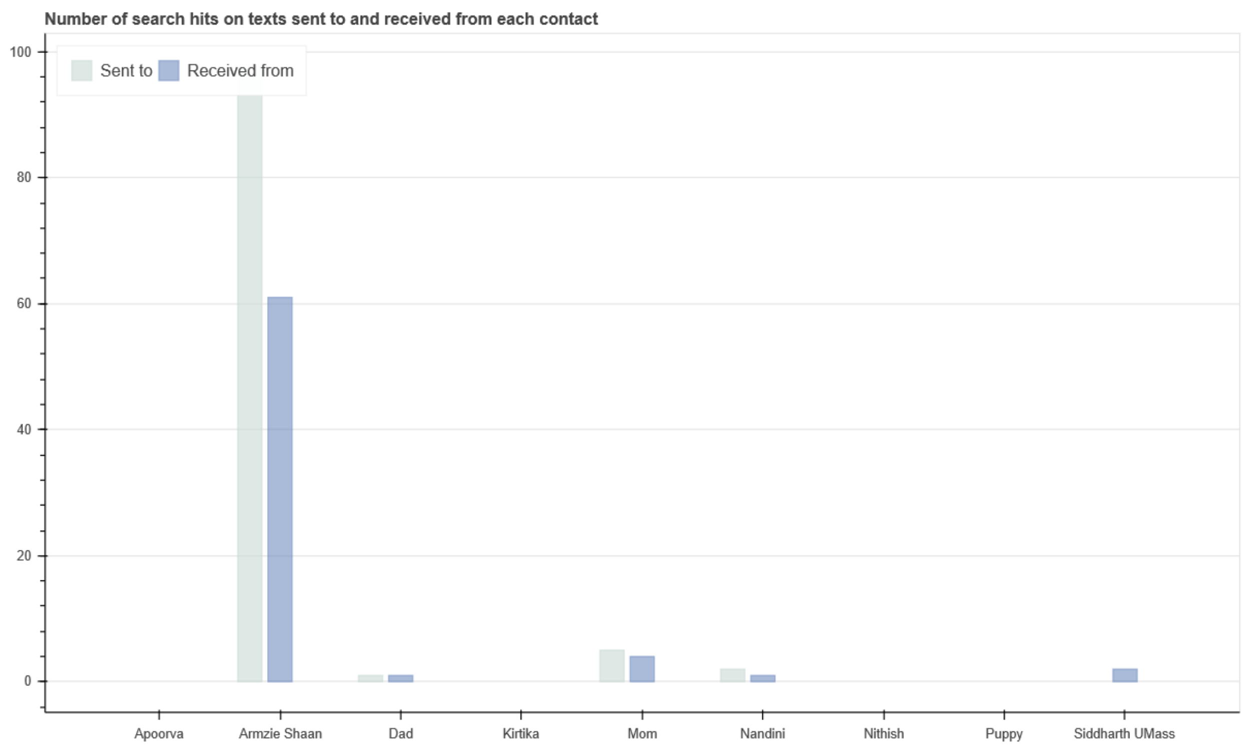


Why is this visualization cool?

First of all, this visualization gives the investigator a very good understanding of the traffic of texts per date, for each contact. Spikes in the figure for a certain date would then set red flags and he can make notes to scrutinize texts from these dates. In fact, the visualizations help the investigator form theories that he can explore using the data extraction tools that we talked about earlier. Now, you may be wondering why the dates in the figure are so far apart. Here's the second cool thing about this visualization. It's interactive! Use the box zoom or scroll zoom tools on the right to zoom in and out on specific dates. Use the pan tool to click-drag omni-directionally. Use the reset tool to reset figure to original scale and position.

9. **Visualize number of search hits on texts sent to and received from each contact:**

This visualization portrays the number of texts sent to and received from each contact that contain the search-terms/keywords (the list of strings that you specified, remember?).

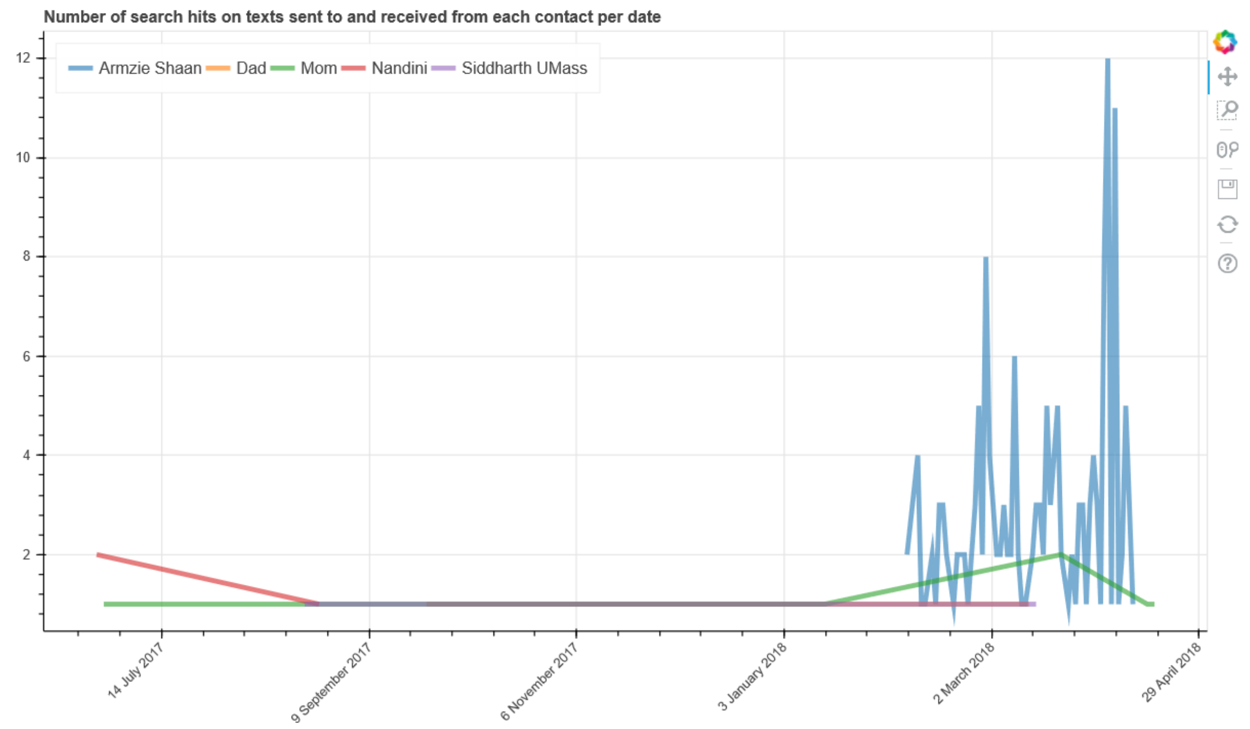


Why is this visualization cool?

This visualization helps the investigator find who the suspect has been communicating with about subjects of interest to the case. How cooler can things get? Wait till you see the next visualizations!

10. Visualize number of search hits on texts sent to and received from each contact per date:

This builds on top of the visualization above. It portrays the number of texts sent to and received from each contact that contain the search-terms/keywords per date!

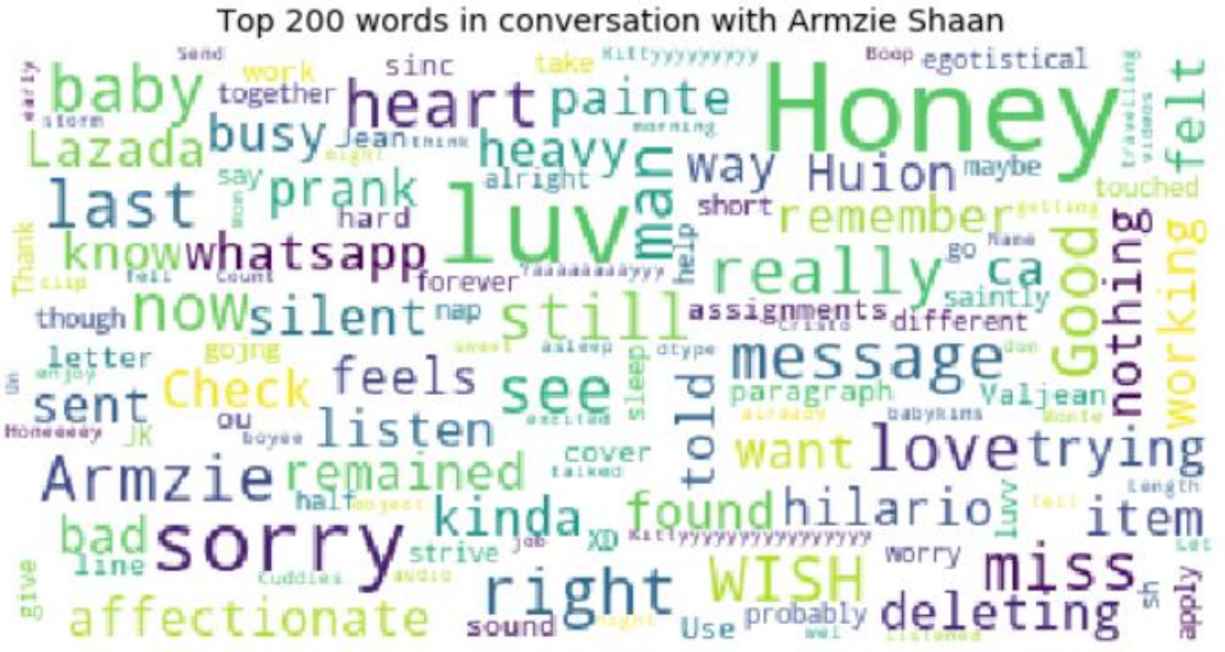


Why is this visualization cool?

This visualization is one of the most important indicators who the suspect has been communicating with about topics of interest AND when! As before, please use the zooms, pan and reset tools to the right to explore this insightful interactive visualization.

11. Visualize wordclouds of frequent words in conversation with each contact:

Wordclouds are attractive visualizations of texts. These wordclouds contain top 200 frequently used words with each contact.



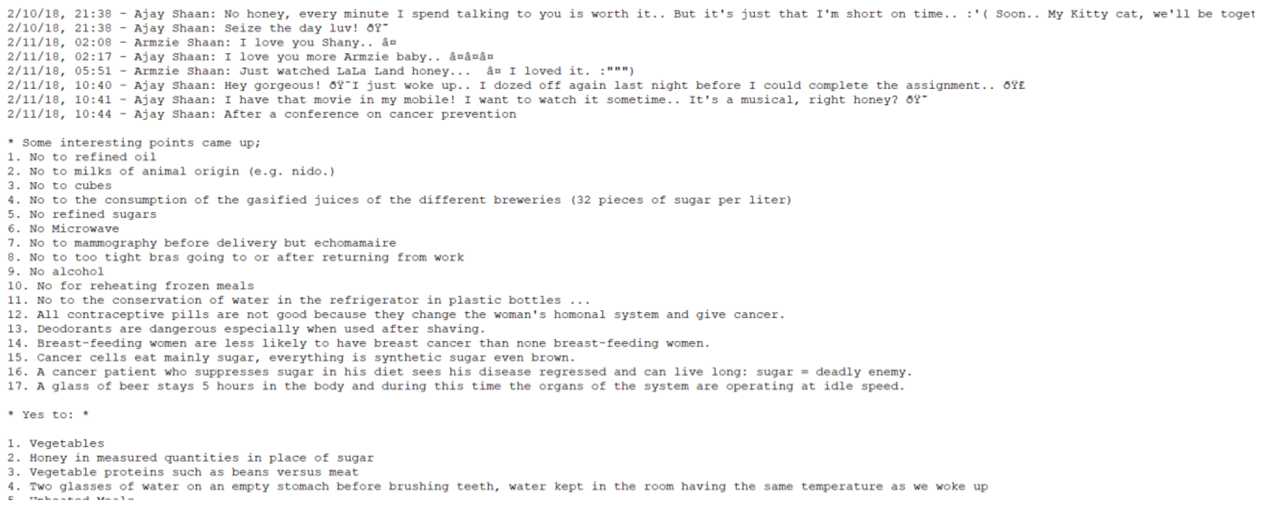
Why are these visualizations cool?

When we narrow down on a suspect/criminal's accomplices, it'd help to analyze the wordclouds to determine what kind of frequent words stick out, which are healthy fodder for keyword searches! Also, the investigator can fairly characterize a suspect and his accomplice based on this. Besides, wordclouds are inherently cool!

This is effing sweet! Let me try it already!

Dear user, please spare a few minutes to read about the challenges that were faced in building this and the attention to detail that went into this passion project.

1. Despite the chat txt files looking rather simple, cleaning them wasn't a trivial matter when care was taken to not lose a single character of information.



```
2/10/18, 21:38 - Ajay Shaan: No honey, every minute I spend talking to you is worth it.. But it's just that I'm short on time.. :'( Soon.. My Kitty cat, we'll be toget
2/10/18, 21:38 - Ajay Shaan: Seize the day luv! ðŸ˜
2/11/18, 02:08 - Armzie Shaan: I love you Shany.. â¤
2/11/18, 02:17 - Ajay Shaan: I love you more Armzie baby.. â¤â¤â¤
2/11/18, 05:51 - Armzie Shaan: Just watched LaLa Land honey...  â¤ I loved it. :""")
2/11/18, 10:40 - Ajay Shaan: Hey gorgeous! ðŸ˜I just woke up.. I dozed off again last night before I could complete the assignment.. ðŸ£
2/11/18, 10:41 - Ajay Shaan: I have that movie in my mobile! I want to watch it sometime.. It's a musical, right honey? ðŸ˜
2/11/18, 10:44 - Ajay Shaan: After a conference on cancer prevention

* Some interesting points came up;
1. No to refined oil
2. No to milks of animal origin (e.g. nido.)
3. No to cubes
4. No to the consumption of the gasified juices of the different breweries (32 pieces of sugar per liter)
5. No refined sugars
6. No Microwave
7. No to mammography before delivery but echomamaire
8. No to too tight bras going to or after returning from work
9. No alcohol
10. No for reheating frozen meals
11. No to the conservation of water in the refrigerator in plastic bottles ...
12. All contraceptive pills are not good because they change the woman's homonal system and give cancer.
13. Deodorants are dangerous especially when used after shaving.
14. Breast-feeding women are less likely to have breast cancer than none breast-feeding women.
15. Cancer cells eat mainly sugar, everything is synthetic sugar even brown.
16. A cancer patient who suppresses sugar in his diet sees his disease regressed and can live long: sugar = deadly enemy.
17. A glass of beer stays 5 hours in the body and during this time the organs of the system are operating at idle speed.

* Yes to: *

1. Vegetables
2. Honey in measured quantities in place of sugar
3. Vegetable proteins such as beans versus meat
4. Two glasses of water on an empty stomach before brushing teeth, water kept in the room having the same temperature as we woke up
```

Notice the stray line breaks and lack of timestamp and sender information in cases of huge texts, especially forwarded ones. Ahem. Please don't judge. I don't usually send these to anyone. In order to push contents of these lines into the text above, the code traverses the lines in reverse while concatenating these lines of text until a timestamp is encountered.

2. As you may see, this extract from above still had to go through a few more filters before I could make a clean dataframe of timestamps, senders and messages. Please note that emojis and text in other language are supported by this tool because the txt files are read based on UTF-8 encoding, and therefore, the final cleaned texts in chat_db and master_db (concatenation of contents of chat_dbs) are a pure representation of what you see on whatsapp, but much more wieldy for our purposes of investigative analysis.

3. The find_texts_between function was initially based on parsing the start and end dates of texts. But I realized a day's worth of texts can still be a lot to go over, and therefore, I went back to square one and using the datetime module, recoded the function such that texts between even two consequtive seconds can be extracted.

4. One can search for a word in a chat conversation with a contact using the whatsapp interface. But searching for multiple words across chat conversations with all contacts is what would be most useful to an investigator without any knowledge of a suspect's contacts. Also, I wanted this search feature to be mindful of typos and not ignore texts containing keywords that are misspelled as otherwise it would result in a loss of very relevant, useful information in the search extracts. This motivated me to incorporate sequence matching. I found false positives along with true positives at first (Of course, there weren't any false negatives), and I tuned it till the FP rate got to a negligibly low level.

5. I know I said this before a couple of times already, but I'd like to explain my motivation for implementing this. Every data extraction tool in this project accepts a dataframe as a parameter. This is because I wanted the investigator to have the freedom and flexibility to dissect the chat_dbs or master_db or a dataframe that he extracted from every possible angle.

6. I realized that extracting contact information and hyperlinks from texts can give the investigator so many leads, and this prompted me to create tools to do that. I encountered a little wall when testing the find_links() function. Due to slashes in hyperlinks, the ones that occurred in close proximity to those infamous untagged long messages we talked about in point 1 were ignored by the function. I had to modify one of the filters from the data-cleaning code so as to handle this corner case, in order to extract all hyperlinks without missing anything.

7. Visualizations 1 and 3 complement visualizations 2 and 4. The per-date visualizations (2 and 4) focus on the text traffic based on the timeline but not on determining the sent-to and received-from stats. Visualizations 1 and 3 concentrate on the latter. For visualizations 1 and 3, I noticed that the chat txt files only contain sender of a text for each record and no receiver information. While this is a no-brainer when dealing with chat conversation from one contact (one chat_db), when many of these are put together (master_db), there's no way of knowing who the receiver of a text was. So I had to extract the sender and receiver counts from traversing through every chat_db in order to make these visualizations.

8. Visualizations 2 and 4 would've looked horrible in matplotlib because of the timestamp labels on the X axis. This made me make interactive visualizations using bokeh so that the timestamp labels don't make the graph an imperceptible huge mess. Fun fact: I didn't have plans of using matplotlib for visualizations 1 and 3 either even though they're static. Bokeh's visualizations (static and dynamic) are more pleasant to look at than matplotlib's crude figures.

9. When creating wordclouds for each contact, I noticed a few recurring words – 'Media', 'Omitted' and the contact's name. I realized that these are useless information despite their understandable high frequency and added them to the list of stopwords.

10. I wanted to put aside my love for jupyter notebooks and code in a proper IDE, but realizing that using a notebook would be MUCH easier for the investigator to quickly create dataframe extracts using the provided tools and do his data analysis and investigation. Besides, jupyter notebooks could display the wordclouds inline. Also, given the cell-based execution in jupyter, the investigator doesn't have to experiment with different searchterms and timestamps by updating the text files every time.

Thank you for reading! I hope you solve more cases than usual with this tool. Your feedback on this is invaluable and I look forward to hear words of gratitude as well as criticism.