# WA_Analysis_

August 29, 2018

```
In [1]: # install these libraries before proceeding
        """
        !pip install numpy
        !pip install pandas
        !pip install wordcloud
        !pip install nltk
        !pip install matplotlib
        !pip install nltk
        """

Out[1]: '\n!pip install numpy\n!pip install pandas\n!pip install wordcloud\n!pip install nltk\

In [2]: import pandas as pd
        import numpy as np
        import nltk
        from nltk.corpus import stopwords
        from wordcloud import WordCloud
        import matplotlib.pyplot as plt
        from IPython.core.display import display, HTML
        from collections import Counter
        from collections import OrderedDict
        import re
        import os

In [3]: # uncomment the below line to download english stopwords
        # nltk.download('stopwords')

In [4]: file_name = 'WhatsApp Chat with AppliedAI.txt' # replace the file name by your file na
        fh = open(file_name,"r")
        data = fh.read()
```

#### 0.0.1 Util

```
In [5]: def custom_display(txt):
            display(HTML("<h2>" + txt + "</h2>"))
            print("_____"*23)
```

## 0.1 Getting datetime & user name from the text file

```
In [6]: time_and_names = re.findall(r'(\d{2}/\d{2}/\d{2},\s\d{1,2}:\d{1,2}\s(PM|AM)\s-\s.*?:)'
        data_and_time = []
        users = {}
        for time_and_name,a in time_and_names:
            data_and_time.append(time_and_name.split(' - ')[0])
            name = time_and_name.split(' - ')[1].split(':')[0].strip('\u202c').strip('\u202a')
            if name in users.keys():
                users[name]+=1
            else:
                users[name] = 1
```

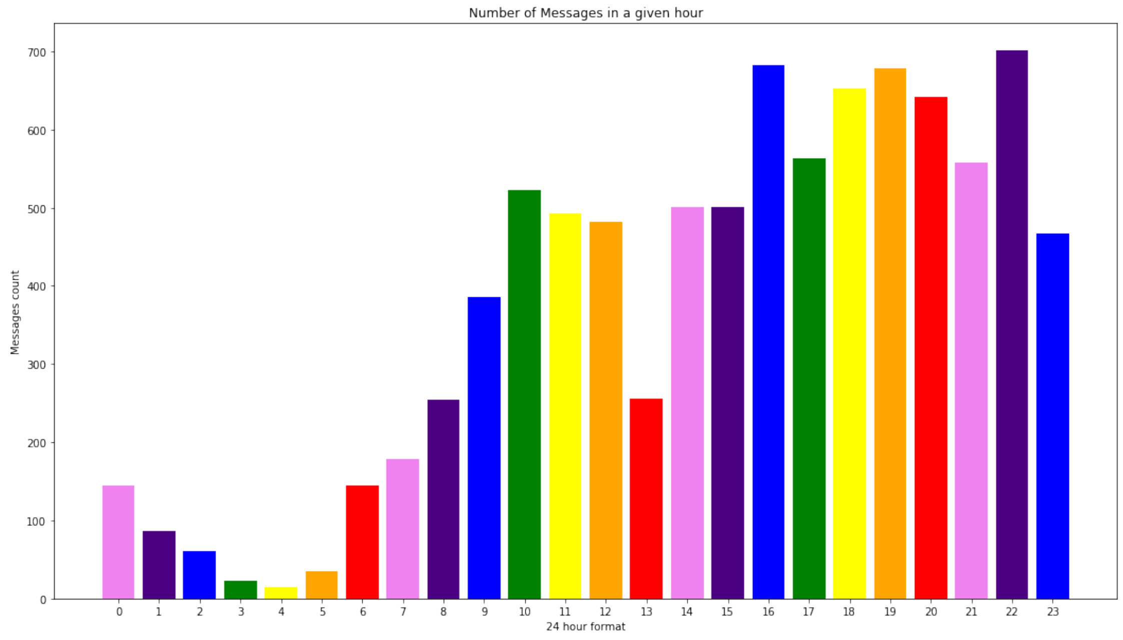
```
In [7]: data_and_time[:2]
```

```
Out[7]: ['19/02/18, 9:44 PM', '19/02/18, 10:27 PM']
```

```
In [8]: message_timings = {}
        for message_time in data_and_time:
            message_time = message_time.split(',')[1].strip().split()
            meridian = message_time[1]
            message_time = message_time[0].split(':')
            key = int(message_time[0])
            if meridian == 'PM':
                key += 12
            key -= 1
            if key not in message_timings.keys():
                message_timings[key] = 0
            message_timings[key] += 1
```

## 0.2 Peak hour of chat

```
In [9]: plt.figure(figsize=(18,10))
        hours = sorted(message_timings.keys())
        message_counts = [message_timings[hour] for hour in hours]

        plt.bar(np.arange(len(hours)), message_counts,color=['violet','indigo','blue','green',
        plt.xticks(np.arange(len(hours)),hours)

        plt.title('Number of Messages in a given hour')
        plt.xlabel('24 hour format')
        plt.ylabel('Messages count')
        plt.show()
        custom_display("Chats will be more during " + str(hours[np.argmax(message_counts)]) +
```



```
<IPython.core.display.HTML object>
```

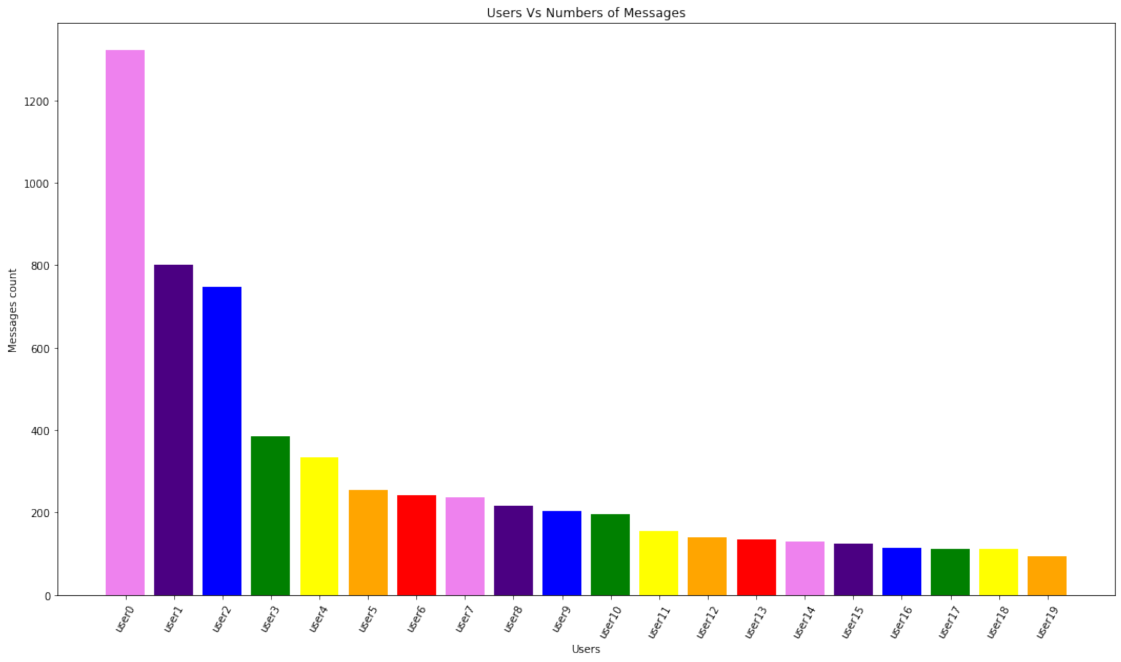
```
------------------------------------------------------------------------------------------------
```

```
In [10]: users_df = pd.DataFrame({'id': np.arange(len(users.keys())), 'user':list(users.keys()
         users_df=users_df.sort_values(['messages'], ascending=False)
         total_active_users = users_df.shape[0]
```

## 0.3 Frequency of messages by Users

```
In [11]: total_msgs = users_df.messages.sum(axis=0)
         total_medias = len(re.findall(r'<Media omitted>', data))
         hyper_links = re.findall(r'(http[s]?://[^\s\n]+)', data)
```

```
In [32]: number_of_users_to_be_shown = 20
         plt.figure(figsize=(18,10))

         message_count_by_users = users_df.messages[:number_of_users_to_be_shown].tolist()
         user_names = users_df.user[:number_of_users_to_be_shown].tolist()
         # user_names = []
         # for i in range(number_of_users_to_be_shown):
         #     user_names.append('user'+str(i))
         plt.bar(np.arange(number_of_users_to_be_shown), message_count_by_users,color=['violet

         plt.xticks(np.arange(number_of_users_to_be_shown),user_names,rotation=60)
         plt.title('Users Vs Numbers of Messages')
         plt.xlabel('Users')
         plt.ylabel('Messages count')
         plt.show()
         # custom_display("User X" + " messaged highest number of times with "+ str(users_df['
         custom_display(users_df['user'].iloc[0] + " messaged highest number of times with "+ s
```



```
<IPython.core.display.HTML object>

------------------------------------------------------------------------------------------------
```

```
In [13]: custom_display("Total active users: "+ str(total_active_users))
         custom_display("Total messages: "+ str(total_msgs))
         custom_display("Total medias shared: "+ str(total_medias))
         custom_display("Total  hyper links: "+ str(len(hyper_links)))
         # uncomment the following lines to get the hyper links
         # for link in hyper_links:
         #     print(link)
```

```
<IPython.core.display.HTML object>

------------------------------------------------------------------------------------------------
```

```
<IPython.core.display.HTML object>
```

```
------------------------------------------------------------------------------------------------
```

```
<IPython.core.display.HTML object>
```

```
------------------------------------------------------------------------------------------------
```

```
<IPython.core.display.HTML object>
```

```
------------------------------------------------------------------------------------------------
```

## 1 Top 3 highest messagers

```
In [ ]: users_df.head(3)
```

```
In [15]: user_contents = []
         for user_name in users_df['user']:
             user_id = int(users_df[users_df['user'] == user_name]['id'])
             if user_name[0] == '+':
                 user_name = "\u202a\\" + user_name + '\u202c'
             contents = re.findall(r''+user_name+ ':(.*)', data)
             for content in contents:
                 user_contents.append([user_id, content.strip()])
```

```
In [16]: content_df = pd.DataFrame(user_contents,  columns=['user_id', 'message'])
         content_df['message_length'] = content_df['message'].apply(lambda x: len(x))
         content_df.head()
```

```
Out[16]:    user_id                      message  message_length
         0        1                          Yes               3
         1        1                 I have done.              12
         2        1  https://youtu.be/n92U_y5PwhU              28
         3        1                         fact               5
         4        1    https://awwcart.wooplr.com              26
```

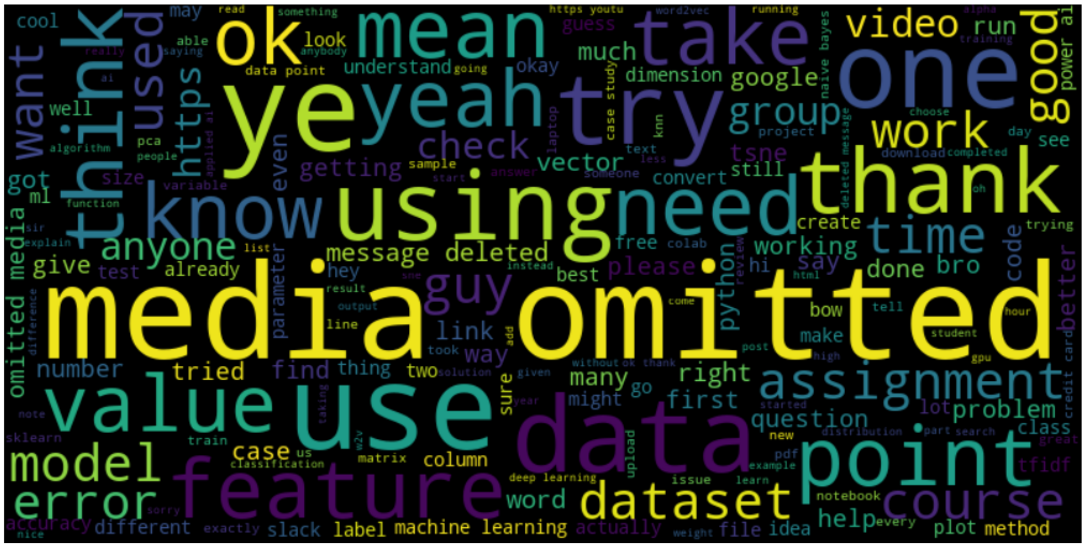
```
In [31]: long_content = content_df[content_df.message_length == max(content_df.message_length)]
         longest_message = long_content['message'].tolist()[0]
         user_name = users_df[users_df['id'] == int(long_content['user_id'])]['user'].tolist()
         # custom_display("User X " + " wrote the longest message with " + str(len(longest_mes
         custom_display(user_name + " wrote the longest message with " + str(len(longest_messag
         print(longest_message)
         custom_display('====='*20)
```

```
<IPython.core.display.HTML object>



------------------------------------------------------------------------------------------
Suppose you have a dataset with 25 features, but you decide you dont want to analyze all those


<IPython.core.display.HTML object>



------------------------------------------------------------------------------------------
```

## 1.1 Word Cloud

```
In [18]: english_words = set(stopwords.words('english'))
         content_df['message_modified'] = content_df['message'].apply(lambda txt: ' '.join(t fo
         word_cloud_text = ' '.join(content_df['message_modified'].apply(lambda x: ''.join(x))
```

```
In [19]: wordcloud = WordCloud(width=800, height=400).generate(word_cloud_text)
         plt.figure(figsize=(18,10))
         plt.imshow(wordcloud, interpolation='bilinear')
         plt.axis("off")
         plt.show()
```



## 1.2 Top 3 most commonly used words

```
In [20]: whole_text_list = word_cloud_text.replace("<media omitted>", "forwarded_image").split
         whole_text = Counter(whole_text_list)
         custom_display(str(whole_text.most_common(3)))
```

```
<IPython.core.display.HTML object>
```

```
--------------------------------------------------------------------------------
```

### 1.3 Frquency of Medias (Image or Video) sent by users

```
In [21]: def get_media_count(user_id):
             return content_df[(content_df['message_modified'] == '<media omitted>') & (conten
         users_df['total_media_sent'] = [get_media_count(user_id) for user_id in users_df.id.to
```

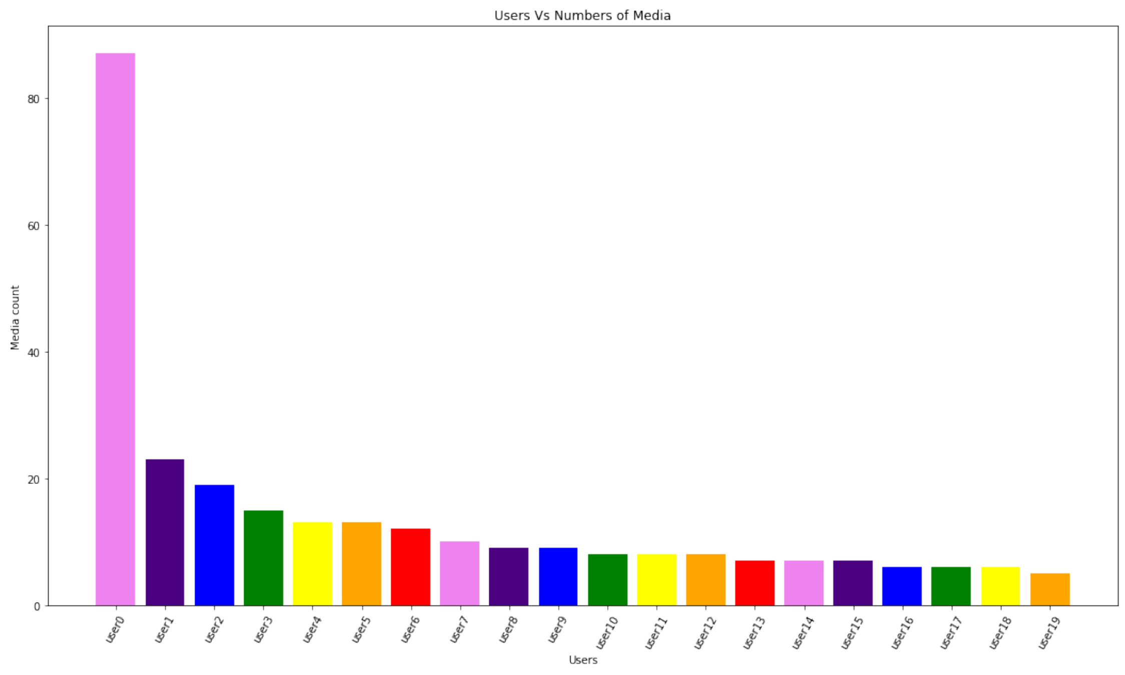
```
In [29]: temp_users_df = users_df.sort_values(['total_media_sent'], ascending=False)
         total_medias_sent = users_df.total_media_sent.sum().tolist()
         custom_display("Total medias shared: "+ str(total_medias_sent))
         #custom_display("User X" + " has sent highest number of medias with a count of "+ str
         custom_display(temp_users_df['user'].iloc[0] + " has sent highest number of medias wi
         number_of_users_to_be_shown = 20
         plt.figure(figsize=(18,10))

         message_count_by_users = temp_users_df.total_media_sent[:number_of_users_to_be_shown]
         user_names = temp_users_df.user[:number_of_users_to_be_shown].tolist()
         # user_names = []
         # for i in range(number_of_users_to_be_shown):
         #     user_names.append('user'+str(i))
         plt.bar(np.arange(number_of_users_to_be_shown), message_count_by_users,color=['violet
         plt.xticks(np.arange(number_of_users_to_be_shown),user_names,rotation=60)
         plt.title('Users Vs Numbers of Media')
         plt.xlabel('Users')
         plt.ylabel('Media count')
         plt.show()
```

```
<IPython.core.display.HTML object>
```

```
--------------------------------------------------------------------------------
```

```
<IPython.core.display.HTML object>
```

```
--------------------------------------------------------------------------------
```



```
In [23]: # to get messages by a specific user
         # re.findall(r'(\d{2}/\d{2}/\d{2},\s\d{1,2}:\d{1,2}\s(PM|AM)\s-\s'+user_name+ ':+)',
```

## 1.4 Smiley Analysis

```
In [24]: smileys = ''

In [25]: smiley_data = {}
         for smiley in smileys:
             cnt = len(re.findall(r'' + smiley+ '', data))
             if cnt > 0:
                 smiley_data[smiley]=cnt

In [26]: ordered_smileys_by_usage_dict = OrderedDict(sorted(smiley_data.items(), key=lambda t:
         smileys_count_list = list(ordered_smileys_by_usage_dict.values())
         total_smileys_used = sum(smileys_count_list)
         custom_display("Total smileys sent: "+ str(total_smileys_used))

<IPython.core.display.HTML object>



--------------------------------------------------------------------------------------------


In [27]: top_n_smileys=10
         top_n_smileys_percentage = np.round((sum(smileys_count_list[:top_n_smileys])/total_smi
         custom_display("Top "+ str(top_n_smileys) + " used smileys:<br><br> "+''.join(list(or
         top_n_smileys_percentage = np.round((sum(smileys_count_list[:top_n_smileys])/total_sm
         custom_display("And these smileys constitute " + str(top_n_smileys_percentage)+ "% of
```

```
<IPython.core.display.HTML object>
```

```
------------------------------------------------------------------------------------------------------------
```

```
<IPython.core.display.HTML object>
```

```
------------------------------------------------------------------------------------------------------------
```

```
In [28]: html_table = """
         <table style="font-family: arial, sans-serif;border-collapse: collapse;width: 100%;">
             <tbody>
                 <tr>
                     <th style='border: 1px solid #dddddd;text-align: left;padding: 8px;'>Smil
                     <th style='border: 1px solid #dddddd;text-align: left;padding: 8px;'> Per
                 </tr>
         """
         for smiley in list(ordered_smileys_by_usage_dict.keys())[:top_n_smileys]:
             html_table += "<tr>"
             html_table += "<td style='border: 1px solid #dddddd;text-align: left;padding: 8px
             html_table += "<td style='border: 1px solid #dddddd;text-align: left;padding: 8px
             html_table += "</tr>"
         html_table += "</tbody></table>"
         display(HTML(html_table))
```

```
<IPython.core.display.HTML object>
```