Data Project Pt. 1: Webscraping the script of Neon Genesis Evangelion

In this notebook, I'll be pulling scripts from each episode of Neon Genesis Evangelion using the BeautifulSoup and Requests packages.

In [1]:
```
import pandas as pd
import bs4 as bs
import requests
import re
import time
```

To get the scripts I'll be using for this analysis, I'll be pulling from scripts available online at https://www.animanga.com/scripts/anime_scripts_english.html. The urls follow a pattern in naming conventions, so I can pull each episode script sequentially with a loop.

The resulting dataset is created by parsing through each page for separate lines of dialogue using regular expressions. I created two separate dictionaries to track the series on an episodic level and as a whole.

The first dictionary contains every characters' lines, line count, and word count for the entire series, while the second is a dictionary of dictionaries containing the same data, but with keys for each episode.

In [2]:
```
character_lines = {}
lines_by_episode = {}

for i in range(1,27):
    current_ep = {}

    webpage = requests.get(f'https://www.animanga.com/scripts/textesgb/eva{i}.html')
    soup = bs.BeautifulSoup(webpage.text, 'lxml')
    page_text = soup.get_text()

    #-------------------------------------------------------------------------

    regex = re.compile(r"(?m).+:[^*#:]+\n (?:(?![#*]).)*") # regular expression to match
    result = re.findall(regex,page_text)

    #-------------------------------------------------------------------------

    # iterating through each line of text, creating a key and word count
    # if character not already in character_lines, otherwise adding to the existing entr
    for text in result:
        character_name = re.findall('[^:#]*',text)[0] # regex to pull the name of the ch
        character_name = character_name.strip('!"#$%(*+, ''-./:;<=>?@[\]^_`{|}~') # stri
        word_count_line = len(text.split())-1
        text=text.split('------------')[0].split('----------------------')[0].split('---
        if character_name in character_lines:
            character_lines[character_name]['Lines'].append(text)
            character_lines[character_name]['Line Count'] += 1
            character_lines[character_name]['Word Count'] += word_count_line
        else:
            character_lines[character_name] = {}
            character_lines[character_name]['Lines'] = []
            character_lines[character_name]['Lines'].append(text)
            character_lines[character_name]['Line Count'] = 1
            character_lines[character_name]['Word Count'] = word_count_line
```

```
    for text in result:
        character_name = re.findall('[^:#]*',text)[0] # regex to pull the name of the ch
        character_name = character_name.strip('!"#$%(*+, ''-./:;<=>?@[\]^_`{|}~') # stri
        word_count_line = len(text.split())-1
        text=text.split('------------')[0].split('----------------------')[0].split('---
        if character_name in current_ep:
            current_ep[character_name]['Lines'].append(text)
            current_ep[character_name]['Line Count'] += 1
            current_ep[character_name]['Word Count'] += word_count_line
        else:
            current_ep[character_name] = {}
            current_ep[character_name]['Lines'] = []
            current_ep[character_name]['Lines'].append(text)
            current_ep[character_name]['Line Count'] = 1
            current_ep[character_name]['Word Count'] = word_count_line

        # ------------------------------------------------------------------------

    lines_by_episode[i] = current_ep

        # there shouldn't be a risk of overloading the server, but added a short wait in
    time.sleep(1)
```

Looking through the compiled list of characters and lines, there are clearly a lot of typos, ranging from misspelled names, to translator notes being included, to whitespace or punctuation causing errors in dialogue attribution.

Next steps in order:

1. Done in the previous step: leading punctuation and whitespace was stripped so that names with errors (ex: 'Shinji' and ' Shinji ') are counted for the same key in each dictionary.
2. Sorting dictionaries and filtering out "names" which only appear due to being in the translator notes on each page
3. Converting into dataframes
4. Merging the misspelled names of characters into the correctly spelled dataframe column and dropping the typos.

Sorting, filtering

In [3]:
```
# entire series: dictionary - {character name:[line count, word count]}
sorted_lines = dict(sorted(character_lines.items()))

# individual episodes: dictionary of dictionaries - {episode number: {character name:[li
sorted_eps = {}
for i in range(1,27):
    sorted_eps[i] = dict(sorted(lines_by_episode[i].items()))

    #------------------------------------------------------------------------

# # filtering out common translator notes not in the script
filter = ['Neon','EVA','Email','E-mail','http','title','episode','Episode','EPISODE','Na

for i in range(1,27):
    sorted_eps[i] = {k:v for k, v in sorted_eps[i].items() if not any(x in k for x in fi

sorted_lines = {k:v for k, v in sorted_lines.items() if not any(x in k for x in filter)}
```

Converting to dataframe, merging misspelled names

To my knowledge, there isn't really a simple way to automate this part, so I had to look through the dataset manually to find typos and also looked through the script to determine where some ambiguous names should be attributed (ex: 'Ikari' is the last name of three different characters, but the lines were spoken by Ikari Gendo)

In [4]:
```
# converting to dataframes
df = pd.DataFrame(data=sorted_lines).T
df.columns = ['Lines','Linecount','Wordcount']

df1 = pd.DataFrame(data=sorted_eps)
df1 = df1
```

With pandas, we can very simply combine dataframes. for this analysis, I'm just going to compile typos and aliases for the top 10 characters and a couple other important ones from a historical character popularity poll

In [5]:
```
# merges all names in the list into the first name of the list in the df dataframe
def character_merge(list1):
    result = df.loc[list1[0]].copy()
    for name in list1[1:]:
        result += df.loc[name].copy()
    df.loc[list1[0]] = result

shinji_list = ['Shinji','Shiji',"Shinji'","Shinji '",'Shinji&Asuka']
asuka_list = ['Asuka','Little Asuka','Shinji&Asuka']
misato_list = ['Misato','Misato (thinking)','Mistato','Phone(Misato)']
ritsuko_list = ['Ritsuko','Ritusko','Ritsukko','Rituko']
ryoji_list = ['Ryoji','Ryouji','Ryouji (voice from the telephone)']
gendo_list = ['Gendo','Gendou','Gendow','Ikari']
fuyutsuki_list = ['Fuyutsuki','Fuyutsuki (voice)','Fuyutsuki(mono)','Fuyuzuki','Kouzou',

character_merge(shinji_list)
character_merge(asuka_list)
character_merge(misato_list)
character_merge(ritsuko_list)
character_merge(ryoji_list)
character_merge(gendo_list)
character_merge(fuyutsuki_list)

# dropping the columns that were merged into the correct column
df.drop(['Shiji',"Shinji'","Shinji '",'Shinji&Asuka','Little Asuka','Shinji&Asuka','Misa
         'Phone(Misato)','Ritusko','Ritsukko','Rituko','Ryouji','Ryouji (voice from the
         'Gendow','Fuyutsuki (voice)','Fuyutsuki(mono)','Fuyuzuki','Kouzou','Kozo','Kozo

df1.drop(['Shiji',"Shinji'","Shinji '",'Shinji&Asuka','Little Asuka','Shinji&Asuka','Mis
         'Phone(Misato)','Ritusko','Ritsukko','Rituko','Ryouji','Ryouji (voice from the
         'Gendow','Fuyutsuki (voice)','Fuyutsuki(mono)','Fuyuzuki','Kouzou','Kozo','Kozo
```

In [6]:
```
# converting to dataframe and exporting to excel sheet and JSON data
df.to_excel('src/NGE_entire_series_lines.xlsx')
df1.to_excel('src/NGE_lines_by_episode.xlsx')

df.to_json('src/NGE_entire_series.json',orient='columns')
df1.to_json('src/NGE_by_episode.json',orient='columns')
```

The dataframes are now exported to Excel files and JSON files for further analysis.