Exploratory Data Analysis DoQA Movie Output

```
pip install textatistic
```

```
Requirement already satisfied: textatistic in /usr/local/lib/python3.10/dist-packages (0.0.1)
Requirement already satisfied: pyhyphen>=2.0.5 in /usr/local/lib/python3.10/dist-packages (from textatistic) (4.
Requirement already satisfied: wheel>=0.36.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->t
Requirement already satisfied: setuptools>=52.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5
Requirement already satisfied: appdirs>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->
Requirement already satisfied: requests>=2.25 in /usr/local/lib/python3.10/dist-packages (from pyhyphen>=2.0.5->
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from request
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.25->pyh
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.2
```

```
import json
import pandas as pd
from textatistic import Textatistic
from pathlib import Path
import re
```

```
with open("/content/doqa-movies-test-v2.1.json", 'r') as file:
    # Load the JSON data
    json_data = json.load(file)
```

```
# Access the 'data' key
data = json_data['data']
```

```
# Create lists to store the extracted data
titles = []
backgrounds = []
all_paragraphs = []

for entry in data:
   title = entry["title"]
   background = entry["background"]
   paragraphs = [c["context"] for c in entry["paragraphs"]]
   # Store each variable in separate lists
   titles.append(title)
   backgrounds.append(background)
   all_paragraphs.append(paragraphs)
#three separate lists: titles, backgrounds, and all_paragraphs
title_df = pd.DataFrame({'Title': titles})
background_df = pd.DataFrame({'Background': backgrounds})
paragraphs_df = pd.DataFrame({'Paragraphs': all_paragraphs})
```

```
title_df = title_df['Title'].tolist()
background_df = background_df['Background'].tolist()
paragraphs_df = paragraphs_df['Paragraphs'].tolist()
```

```
def clean_string(text): # Cleaning
    """re.sub(pattern, repl, string).
    Returns the string obtained by replacing the leftmost
    non-overlapping occurrences of pattern in string by the
    replacement thus removing any urls
    """
    return " ".join(re.sub("([^0-9A-Za-z \t])|(\w+:\/\/\S+)", "", str(text)).split())
```

```
TextOnlyTitle = [clean_string(Title) for Title in title_df]
TextOnlyBackground = [clean_string(Background) for Background in background_df]
TextOnlyParagraphs = [clean_string(Paragraphs) for Paragraphs in paragraphs_df]
```

```
TextOnlyTitle[:2]
```

```
['Whats the meaning behind the phrase The blood stays on the blade',
 'Is Franz Oberhauser related to the original Blofeld']
```

```
TextOnlyBackground[:2]
```

```
['At the beginning of the movie Gangs of New York Priest Vallon tells his son not to wipe the blood of the
blade after hes cut himself shaving saying The blood stays on the blade Ive watched the movie multiple times
but can never find what the meaning behind that phrase is Anyone able to shed some light on this',
 'Note spoilers for Spectre ahead In Spectre we meet Franz Oberhauser Christoph Waltz He later reveals he now
goes by the name Ernst Blofeld and has a scar just like the original Blofeld But he also states he took the
name from his mothers side My question is is Oberhauser a descendent of the original Blofeld from On Her
Majestys Secret Service among others or is he supposed to be the same Blofeld character I know the Daniel Craig
series of Bond films are a rebootprequel of sorts Is Spectre showing the genesis of the character']
```

```
TextOnlyParagraphs[:2]
```

```
['I just did a bit of digging on your behalf regarding this line and there seems to be no concrete definition
of it from any official source That said I found several interpretations along the way including 1 You cannot
wash away your sins 2 The blood is worn as a badge of merit on the blade 3 The blood remains on the blade and
not on your hands meaning you are relatively guiltfree Personally considering the overtly religious themes
within the film I would side with option 1 especially since Priest Vallon is saying the line CANNOTANSWER',
 'Yes Hes supposed to be the same character The timeline of the story arc is all over the place now but Spectre
could be regarded as a prequel to the series before Daniel Craig Some say a reboot but apparently its all
supposed to be the same story This falls apart because of Judi Denchs M Bonds previous encounters with Spectre
and Blofeld etc though CANNOTANSWER']
```

Title

```
a_list = TextOnlyTitle
```

```
new_list_a = [x for x in a_list if len(x) < 1000]
```

```
textfile_a = open("fileA_file.txt", "w")
```

```
for element in new_list_a:
    textfile_a.write(element + "\n")
```

```
textfile_a.close()
```

```
textA = Path('fileA_file.txt').read_text()
```

```
a_string = textA
```

```
Str_EndFixA = a_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixA)
```

```
%precision 3
```

```
     '%.3f'
```

```
readability.dict()
```

```
     {'char_count': 17831,
      'word_count': 3541,
      'sent_count': 400,
      'sybl_count': 4763,
      'notdalechall_count': 1261,
      'polysyblword_count': 246,
      'flesch_score': 84.054,
      'fleschkincaid_score': 3.735,
      'gunningfog_score': 6.320,
      'smog_score': 7.609,
      'dalechall_score': 9.699}
```

For Background

```
b_list = TextOnlyBackground
```

```
new_list_b = [x for x in b_list if len(x) < 1000]
```

```
textfile_b = open("fileB_file.txt", "w")
```

```
for element in new_list_b:
    textfile_b.write(element + "\n")
```

```
textfile_b.close()
```

```
textB = Path('fileB_file.txt').read_text()
```

```
b_string = textB
```

```
Str_EndFixB = b_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixB)
```

```
%precision 3
```

```
     '%.3f'
```

```
readability.dict()
```

```
     {'char_count': 129707,
      'word_count': 29307,
      'sent_count': 378,
      'sybl_count': 37670,
      'notdalechall_count': 6312,
      'polysyblword_count': 1701,
      'flesch_score': 19.399,
      'fleschkincaid_score': 29.815,
      'gunningfog_score': 33.334,
      'smog_score': 15.248,
      'dalechall_score': 10.883}
```

For Paragraph

```
p_list = TextOnlyParagraphs
```

```
new_list_p = [x for x in p_list if len(x) < 1000]
```

```
textfile_p = open("fileP_file.txt", "w")
```

```
for element in new_list_p:
    textfile_p.write(element + "\n")
```

```
textfile_p.close()
```

```
textP = Path('fileP_file.txt').read_text()
```

```
p_string = textP
```

```
Str_EndFixP = p_string.replace("\n", ".")
```

```
readability = Textatistic(Str_EndFixP)
```

```
%precision 3
```

```
'%.3f'
```

```
readability.dict()
```

```
{'char_count': 154673,
 'word_count': 33753,
 'sent_count': 329,
 'sybl_count': 44683,
 'notdalechall_count': 7607,
 'polysyblword_count': 2387,
 'flesch_score': -9.292,
 'fleschkincaid_score': 40.042,
 'gunningfog_score': 43.866,
 'smog_score': 18.517,
 'dalechall_score': 12.284}
```