Pull Data from OMDB API

June 20, 2022

1 Final Project - Movie Reviews Analysis

- **Team 13: Jimmy Nguyen, Dallin Munger, Tyler Wolff**

2 Packages

[3]:
```
import pandas as pd
import numpy as np
import requests
from requests import TooManyRedirects
import re
import omdb
import time
from collections import Counter, defaultdict
```

3 Pulling Data from API

[4]:
```
# set timeout of 5 seconds for this request
# Pull 500 pages of movies (5000 movies) with the word 'one' in the title
imdb_ids = []
for i in range(1,501):
    year_df = pd.DataFrame(omdb.search_movie('one', page=i, timeout=5))
    # Store the ids in a list
    imdb_ids.append(year_df['imdb_id'].tolist())
imdb_ids = sum(imdb_ids, [])
```

[5]:
```
# Use ids to obtain movie information -
movies_info = pd.DataFrame([omdb.imdbid(i) for i in imdb_ids])

# View the dataframe
movies_info.sample(15)
```

[5]:
```
                                  title  year  rated  \
4893             One Nation Under Mike  2015    N/A
2163               Before One More Day  2019    N/A
3108      Secret Access: Air Force One  2008    N/A
```

```
4884         ONE Fighting Championship 3: War of Lions  2012    N/A
4931                                       One Liners  2008    N/A
4803                               One Last Love Song  2009    N/A
2263  Reclamation - chapter one - a Star Trek fan pr…  2019    N/A
4157                                   Batman: Year One  2019    N/A
3262                          A Thousand and One Nights  1941    N/A
3769                          Five times smile plus one  2010    N/A
2040                          One Minute for Conductors  2013    N/A
400                                      One Last Shot  1998  TV-MA
4777                                     Table for One  2020    N/A
190   One Piece: Episode of Alabasta - The Desert Pr…  2007  PG-13
2638   Yum Ciao News: The Revenge of the Number One Fan  2011    N/A

         released runtime                          genre  \
4893          N/A  11 min                  Short, Comedy
2163  06 Aug 2019  16 min                   Short, Drama
3108          N/A     N/A                    Documentary
4884  31 Mar 2012     N/A                          Sport
4931  09 Nov 2008   2 min                          Short
4803          N/A     N/A                   Short, Drama
2263  22 Dec 2019     N/A                  Short, Sci-Fi
4157  01 Nov 2019     N/A        Action, Horror, Mystery
3262          N/A     N/A       Adventure, Comedy, Drama
3769  03 May 2010     N/A                          Short
2040  23 Nov 2013  87 min             Documentary, Music
400   30 Oct 2017  30 min           Short, Comedy, Drama
4777  20 Aug 2020     N/A             Documentary, Short
190   03 Mar 2007  90 min  Animation, Action, Adventure
2638  01 Feb 2011   7 min                  Short, Comedy

                                        director  \
4893  Vera Abrams, August Hartwell, Jessica Miner
2163                                Renata Abreu
3108                               Peter Schnall
4884                                         N/A
4931                                    Tim Best
4803                    Keith Mackin, John Reck
2263                                  Greg Ogles
4157                     Jose Luis Garcia Baylon
3262                                Togo Mizrahi
3769                               Loris Arduino
2040               Angel Esteban, Elena Goatelli
400                             Mike Clattenburg
4777                                   Jenny Gao
190                              Takahiro Imamura
2638                           Harrison Berenger
```

	writer \
4893	Jillian Sanders
2163	Renata Abreu
3108	Don Campbell
4884	N/A
4931	N/A
4803	Keith Mackin, John Reck
2263	Greg Ogles
4157	Darren Aronofsky (based on the screenplay by),…
3262	N/A
3769	Loris Arduino
2040	Angel Esteban (original idea), Elena Goatelli …
400	Mike Clattenburg, John Paul Tremblay, Robb Wells
4777	N/A
190	Eiichiro Oda, Hirohiko Uesaka
2638	Harrison Berenger

	actors \
4893	Phil Abrams, John Glouchevitch, Jessica Miner,…
2163	Thais Castro, Flávio Dias, Thalita Franco
3108	N/A
4884	Kevin Belingon, Jan Kai Chee, Nicole Chua
4931	Tim Best
4803	Allan Richardson, Matt Watkins, Rholda Wilson
2263	Haley Barnes, Crimson Kromer, Hassel Kromer, R…
4157	Alan Acosta, Alejandro Fierro
3262	Ali Al-Kassar, Aqila Ratib
3769	Luigi Caso, Andrea Ciardulli, Nicola Cuomo
2040	N/A
400	Robb Wells, John Paul Tremblay, John Dunsworth
4777	N/A
190	Charles Baker, Troy Baker, Anthony Bowling
2638	Harrison Berenger, Samuel Berenger, Tom Crane

	plot	…	metascore \
4893	N/A	…	N/A
2163	One family suffers from the death of their mot…	…	N/A
3108	N/A	…	N/A
4884	N/A	…	N/A
4931	Movies have permeated the consciousness of Ame…	…	N/A
4803	N/A	…	N/A
2263	A diabolical foe resurfaces during the Federat…	…	N/A
4157	The story recounts the beginnings of James Gor…	…	N/A
3262	N/A	…	N/A
3769	N/A	…	N/A
2040	More than 130 young conductors participate in …	…	N/A
400	Two best friends, Rob and GW, spend a night ou…	…	N/A

```
4777  Against all odds, an aging table tennis icon r…  …       N/A
190   A re-telling of the Alabaster Arc from One Pie…  …       N/A
2638                                               N/A  …       N/A

     imdb_rating imdb_votes     imdb_id   type          dvd box_office  \
4893         N/A        N/A   tt5541410  movie          N/A        N/A
2163         N/A          8  tt11597840  movie          N/A        N/A
3108         N/A        N/A   tt1567417  movie          N/A        N/A
4884         N/A        N/A   tt6217432  movie          N/A        N/A
4931         N/A        N/A   tt1337526  movie          N/A        N/A
4803         N/A        N/A   tt1387245  movie          N/A        N/A
2263         7.7          7  tt11534274  movie          N/A        N/A
4157         N/A        N/A   tt7829938  movie          N/A        N/A
3262         N/A        N/A   tt0280411  movie          N/A        N/A
3769         N/A        N/A   tt9878362  movie          N/A        N/A
2040         8.5         10   tt3314488  movie          N/A        N/A
400          6.5        523   tt0380601  movie          N/A        N/A
4777         N/A        N/A  tt12788300  movie          N/A        N/A
190          6.9      1,631   tt1037116  movie  03 Dec 2020     $6,587
2638         N/A        N/A  tt11651744  movie          N/A        N/A

     production website response
4893        N/A     N/A     True
2163        N/A     N/A     True
3108        N/A     N/A     True
4884        N/A     N/A     True
4931        N/A     N/A     True
4803        N/A     N/A     True
2263        N/A     N/A     True
4157        N/A     N/A     True
3262        N/A     N/A     True
3769        N/A     N/A     True
2040        N/A     N/A     True
400         N/A     N/A     True
4777        N/A     N/A     True
190         N/A     N/A     True
2638        N/A     N/A     True

[15 rows x 25 columns]
```

[6]:
```python
# Dimensions of the data frame
movies_info.shape
```

[6]: (5000, 25)

[7]:
```python
# Columns of the data frame
movies_info.columns
```

[7]:
```
Index(['title', 'year', 'rated', 'released', 'runtime', 'genre', 'director',
       'writer', 'actors', 'plot', 'language', 'country', 'awards', 'poster',
       'ratings', 'metascore', 'imdb_rating', 'imdb_votes', 'imdb_id', 'type',
       'dvd', 'box_office', 'production', 'website', 'response'],
      dtype='object')
```

[8]:
```
# Write to a csv
#movies_info.to_csv('Raw Movie Data.csv', index= False)
```

Created in Deepnote

Descriptive Statistics on API Data

June 20, 2022

1 Final Project - Movie Reviews Analysis

- **Team 13: Jimmy Nguyen, Dallin Munger, Tyler Wolff**
- Date: 06/20/2022

2 Packages

[1]:
```
import pandas as pd
import numpy as np
import requests
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from requests import TooManyRedirects
import re
import omdb
import time
from collections import Counter, defaultdict
import nltk
#nltk.download('stopwords')
from nltk.corpus import stopwords
from string import punctuation
sw = stopwords.words("english")
```

---

3 Loading the Raw Data from API

[2]:
```
# Read in csv data as pandas data frame
movies_info = pd.read_csv("Raw Movie Data.csv")
# see a random subset of 15 samples
movies_info.shape
```

[2]: (5000, 25)

[3]:
```
movies_info.sample(10)
```

[3]:

```
                                               title  year  rated  \
351                                One Under the Sun  2017  TV-14
2313                                      Square One  2013    NaN
2518  You Are the One: The Claudine-Raymart Love Story  2006    NaN
4525                                 One Day Over L.A.  2014    NaN
38                                 The Son of No One  2011      R
3123                              One Sight, One Sound  2009    NaN
4544                         One Day in Perfect Health  1950    NaN
2639                                  One Black Coffee  2019    NaN
507    One Damned Day at Dawn… Django Meets Sartana!  1970    NaN
1224           One Direction: What Makes You Beautiful  2011    NaN

         released  runtime                          genre  \
351   14 Mar 2017  101 min         Drama, Mystery, Sci-Fi
2313  15 Aug 2013   15 min                  Short, Comedy
2518  26 Mar 2006      NaN             Documentary, Music
4525          NaN    3 min  Documentary, Short, Adventure
38    09 Jul 2011   90 min            Action, Crime, Drama
3123          NaN   75 min             Documentary, Music
4544          NaN   18 min             Documentary, Short
2639  30 Mar 2019      NaN                          Short
507   25 Jun 1970   90 min                        Western
1224  19 Aug 2011    3 min                   Short, Music

                              director                                  writer  \
351   Riyaana Hartley, Vincent Tran      Katherine Tomlinson, Vincent Tran
2313                   Emanuel Parvu                           Emanuel Parvu
2518                             NaN                                     NaN
4525                     Cole Kawana                     Cole Kawana (story)
38                      Dito Montiel                            Dito Montiel
3123                   Josh Pomponio                                     NaN
4544                      John Krish                                     NaN
2639                    Manoj Mathew                            Manoj Mathew
507                  Demofilo Fidani  Demofilo Fidani, Mila Vitelli Valenza
1224                     John Urbano                                     NaN

                                                actors  \
351             Pooja Batra, Gene Farber, Michael Keeley
2313  Dorian Boguta, Dorina Lazar, Emanuel Parvu, Co…
2518  Claudine Barretto, Raymart Santiago, Dennis Pa…
4525                                               NaN
38          Channing Tatum, Al Pacino, Juliette Binoche
3123                                               NaN
4544                                               NaN
2639  Manoj Mathew, Babli Das, Kasturi Banerjee, Tar…
507                Jack Betts, Fabio Testi, Dino Strano
1224  One Direction, Harry Styles, Louis Tomlinson, …
```

```
                                                   plot  … metascore  \
351   Astronaut Kathryn Voss, sole survivor of a dis…  …       NaN
2313                                              NaN  …       NaN
2518                                              NaN  …       NaN
4525  Harvard-Westlake sophomore Cole Kawana flies h…  …       NaN
38    A young cop is assigned to a precinct in the w…  …      36.0
3123  September-November 2008; three months with the…  …       NaN
4544                                              NaN  …       NaN
2639                                              NaN  …       NaN
507   Framed for a bank robbery, bounty killer Djang…  …       NaN
1224  Official music video for "What Makes You Beaut…  …       NaN

     imdb_rating imdb_votes     imdb_id   type          dvd  box_office  \
351          3.5        612   tt5110386  movie  14 Mar 2018         NaN
2313         6.6          7   tt6479182  movie          NaN         NaN
2518         5.3          5   tt0787250  movie          NaN         NaN
4525         NaN        NaN   tt4540802  movie          NaN         NaN
38           5.1     17,137   tt1535612  movie  21 Feb 2012     $30,680
3123         NaN        NaN   tt1567653  movie          NaN         NaN
4544         NaN        NaN   tt2064890  movie          NaN         NaN
2639         NaN        NaN  tt11062440  movie          NaN         NaN
507          4.9        329   tt0067643  movie          NaN         NaN
1224         7.2         45   tt7318548  movie          NaN         NaN

     production website response
351         NaN     NaN     True
2313        NaN     NaN     True
2518        NaN     NaN     True
4525        NaN     NaN     True
38          NaN     NaN     True
3123        NaN     NaN     True
4544        NaN     NaN     True
2639        NaN     NaN     True
507         NaN     NaN     True
1224        NaN     NaN     True

[10 rows x 25 columns]
```

---

4 Exploratory Data Analysis

1. Examine a five-number summary of the numerical and categorical columns
2. Checking for Missing Data
3. Plotting Value Distribtuions
4. Comparing Value Distributions Across Categories

4.1 1. Calculating Summary Statistics for Columns

[4]:
```python
# Create a new column to look at the length of each plot
movies_info['plot_length'] = movies_info['plot'].str.len()

# 5 number summary of the numerical columns
movies_info.describe().T
```

[4]:
```
              count         mean        std     min     25%     50%     75%  \
year         5000.0  2002.897000  23.082681  1887.0  2002.0  2011.0  2016.0
metascore     115.0    60.026087  18.559747    16.0    47.0    62.0    74.0
imdb_rating  2442.0     6.591155   1.386458     1.0     5.8     6.7     7.5
plot_length  3560.0   159.808989  70.963086    16.0   109.0   170.0   208.0

                max
year         2023.0
metascore      93.0
imdb_rating    10.0
plot_length  1324.0
```

Interpretation 1. **Year:** The range of the movies pulled from the API is from the year 1887 to 2023. This may seem plausible, but requires more drilling down in the data to figure out if the first movie ever was actually made in 1887. For movies in the year 2023, this may be upcoming movies that will be released then.

2. **Metascore**: The metascore is a weighted average of many reviews coming from reputed critics. The Metacritic team reads the reviews and assigns each a 0–100 score, which is then given a weight, mainly based on the review's quality and source. That means the higher the metascore, the more positive reviews a movie has. In our summary, we can see that the range for our movies in this sample is from 16 as the lowest and 93 as the highest. The average metascore is 60, where as the median is 62. This can be interesting later as we dive into the average metascore over time.

3. **imdb_rating**: IMDB rating allow users to rate films on a scale of 1-10. As expected, the range for this variable is 1 as the lowest and 10 as the highest. However, the average IMDB rating is 6.6 and the median is 6.7.

4. **plot_length**: This column displays the length of each movies' plot. Movies plots length range from 16 words as the lowest to 208 as the highest. On average, a movie plot has the length of 160 words whereas the median is 170. This could also indicate that a longer plot description will provide more information to understanding the movies' genres.

[5]:
```python
movies_info[['awards','runtime', 'language', 'country']].describe(include =␣
↪'O').T
```

[5]:
```
         count unique          top  freq
awards     840    177  1 nomination   139
runtime   3826    173        4 min   163
language  4586    233      English  3414
```

```
country    4797    321            USA   1444
```

Interpretation

1.**awards** This variable shows that 139 movies out of 5000 were able to receive 1 nomination for an award. However, due to the number of unique values, we may need to consider that awards recorded down for each movie is not consistent since this has a high cardinality. Therefore, this may not be a reliable insight for the awards variable

2. **runtime** This variable also sees a high cardinality, but at a quick glance we can see that there are 163 movies that has a runtime of only 4 minutes.
3. **language** There number of unique languages here is 233, while that may seem plausible it is also expected to see that movies in English was most prevalent.
4. **country** Understandably, the country with the most movies are from the United States of America (USA). Exactly 1444 movies out of 5000 in this API sample are American.

4.2 2. Checking for Missing Data

[6]:
```
movies_info.isna().sum()
```

[6]:
```
title            0
year             0
rated         4245
released      1349
runtime       1174
genre          159
director       415
writer        1435
actors         754
plot          1440
language       414
country        203
awards        4160
poster        2180
ratings          0
metascore     4885
imdb_rating   2558
imdb_votes    2446
imdb_id          0
type             0
dvd           4497
box_office    4884
production    4965
website       4996
response         0
plot_length   1440
dtype: int64
```

[7]:
```
# Only working with rows where plot AND genre is not null
movies_info = movies_info[(movies_info['plot'].notnull()) &␣
↪(movies_info['genre'].notnull())]
movies_info.shape
```

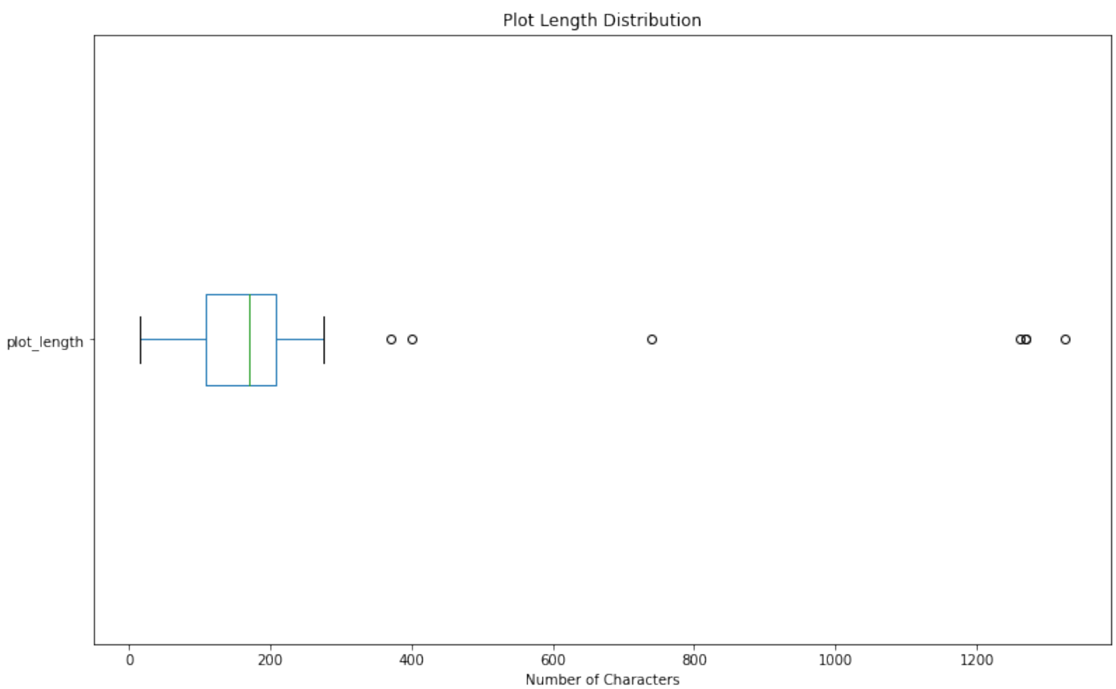
[7]: (3510, 26)

Interpretation

Since our project is based on classifying the first genre of every movie based on its plot, then we only need to take into consideration the `plot` and `genre` columns to prepare for modeling

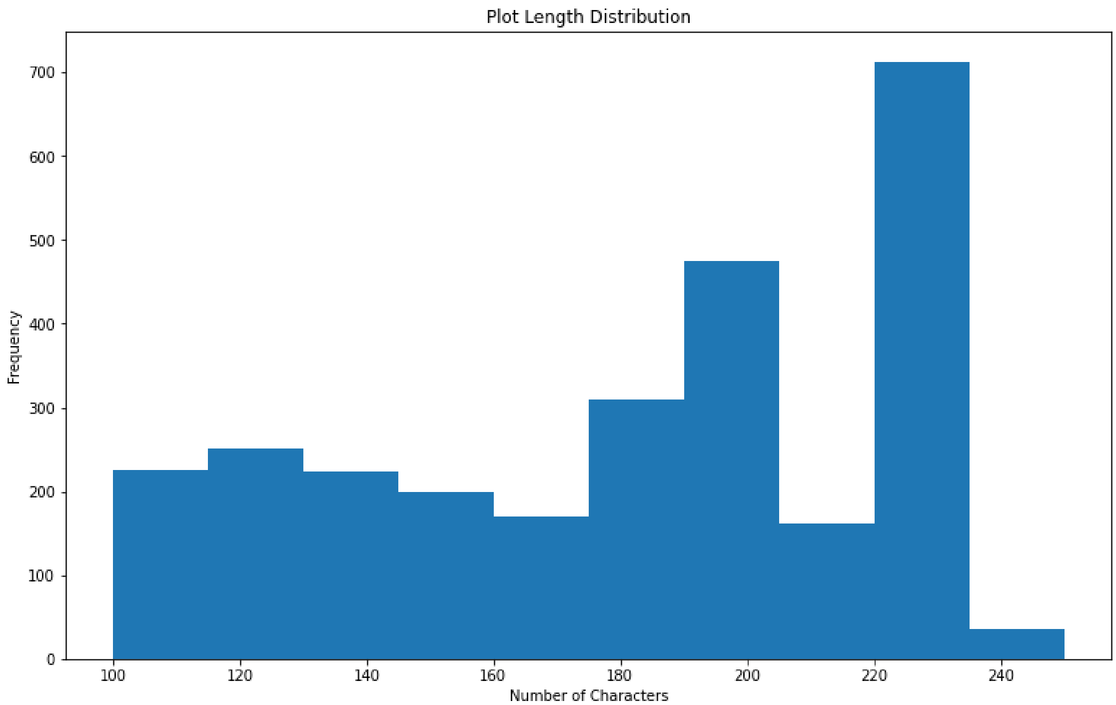
4.3 3. Plotting Value Distribtuions

[8]:
```
## Plotting Value Distribtuions
plt.figure(figsize=(13,8))
movies_info['plot_length'].plot(kind='box', vert=False)
plt.xlabel('Number of Characters')
plt.title('Plot Length Distribution')
plt.show()
```



[9]:
```
# Histogram distribution of movie plot lengths
plt.figure(figsize=(13,8))
movies_info['plot_length'].plot(kind = 'hist', range = (100,250))
plt.xlabel('Number of Characters')
```

```
plt.title('Plot Length Distribution')
plt.show()
```



Interpretation

After removing the missing values, 50% percent of the plot descriptions have a length between roughly 150 and 230 characters, with the median at about 180 with many outliers to the right. The distribution is obviously left-skewed.

The histogram is showing the bins for the number of characters between the ranges of 100 to 250.

4.4 4. Comparing Value Distributions Across Categories

[10]:
```
# top 10 countries with most movie plot descriptions
movies_info['country'].value_counts().nlargest(10)
```

[10]:
```
United States     885
USA               883
UK                174
United Kingdom    155
Canada            139
Japan              81
Australia          78
France             62
India              57
```

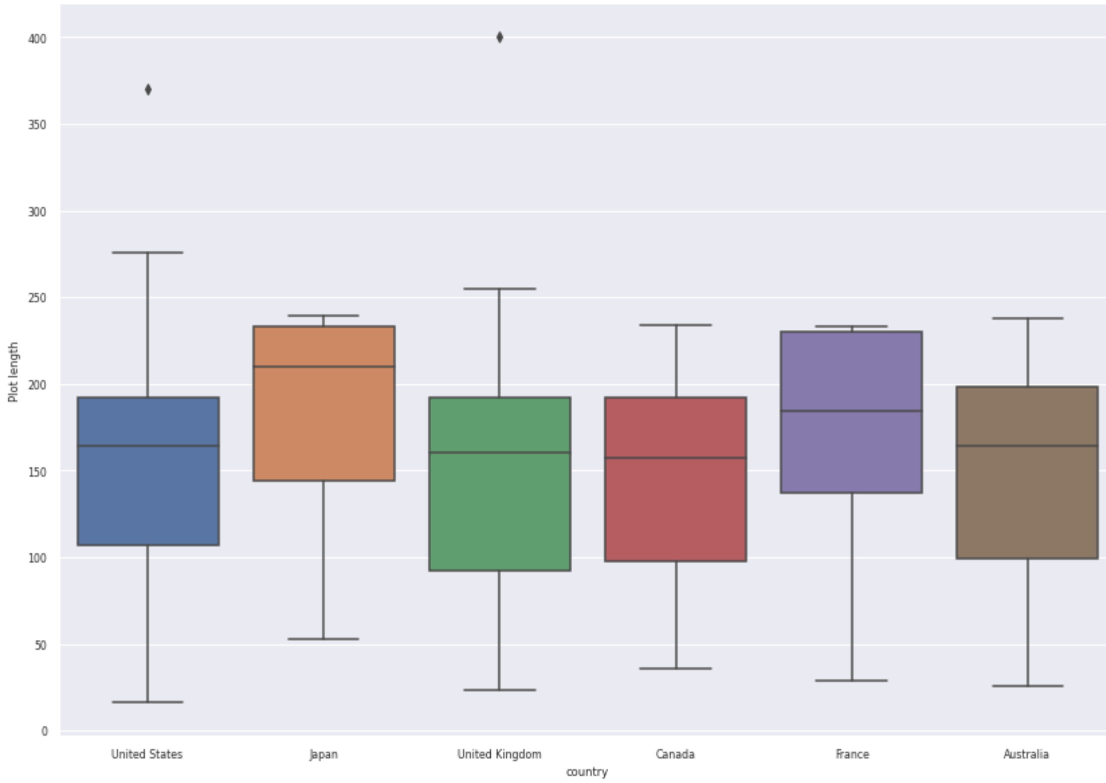
```
Italy                43
Name: country, dtype: int64
```

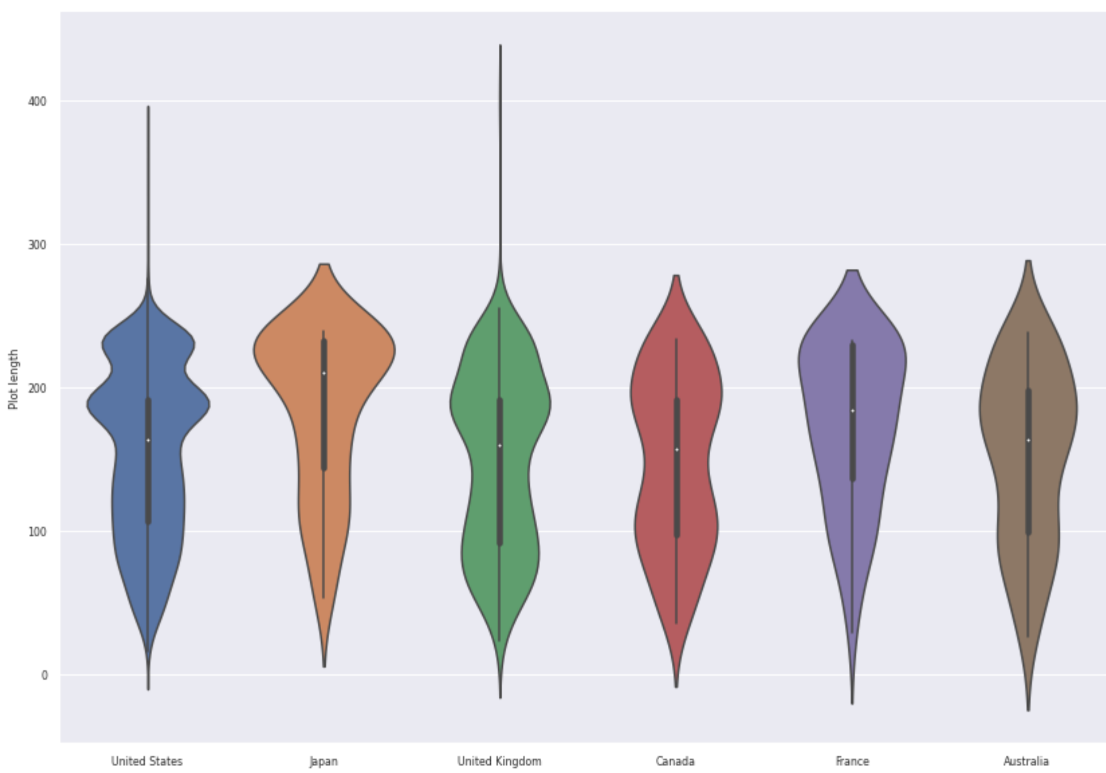
[11]:
```
# Replace United States Values
movies_info['country'] = movies_info['country'].str.replace(r'USA','United␣
↪States', regex=True)

# Replace UK Values
movies_info['country'] = movies_info['country'].str.replace(r'UK', 'United␣
↪Kingdom', regex=True)

# top 10 countries with most movie plot descriptions
movies_info['country'].value_counts().nlargest(10)
```

[11]:
```
United States     1768
United Kingdom     329
Canada             139
Japan               81
Australia           78
France              62
India               57
Italy               43
Germany             37
China               26
Name: country, dtype: int64
```

[12]:
```
# Boxplot and violint plots for movie plot lengths by countries
sns.set(font_scale = 0.7)
where = movies_info['country'].isin(['United States', 'United Kingdom',␣
↪'Canada', 'Japan', 'Australia', 'France'])
sns.catplot(data=movies_info[where], x="country", y="plot_length", kind='box',␣
↪height=8.27, aspect=11.7/8.27)
plt.ylabel("Plot length")
sns.catplot(data=movies_info[where], x="country", y="plot_length",␣
↪kind='violin', height=8.27, aspect=11.7/8.27)
plt.ylabel("Plot length")
```

[12]:
```
Text(2.6970000000000027, 0.5, 'Plot length')
```


400
350
300
250
200
150
100
50
0
Plot length
United States
Japan
United Kingdom
Canada
France
Australia
country

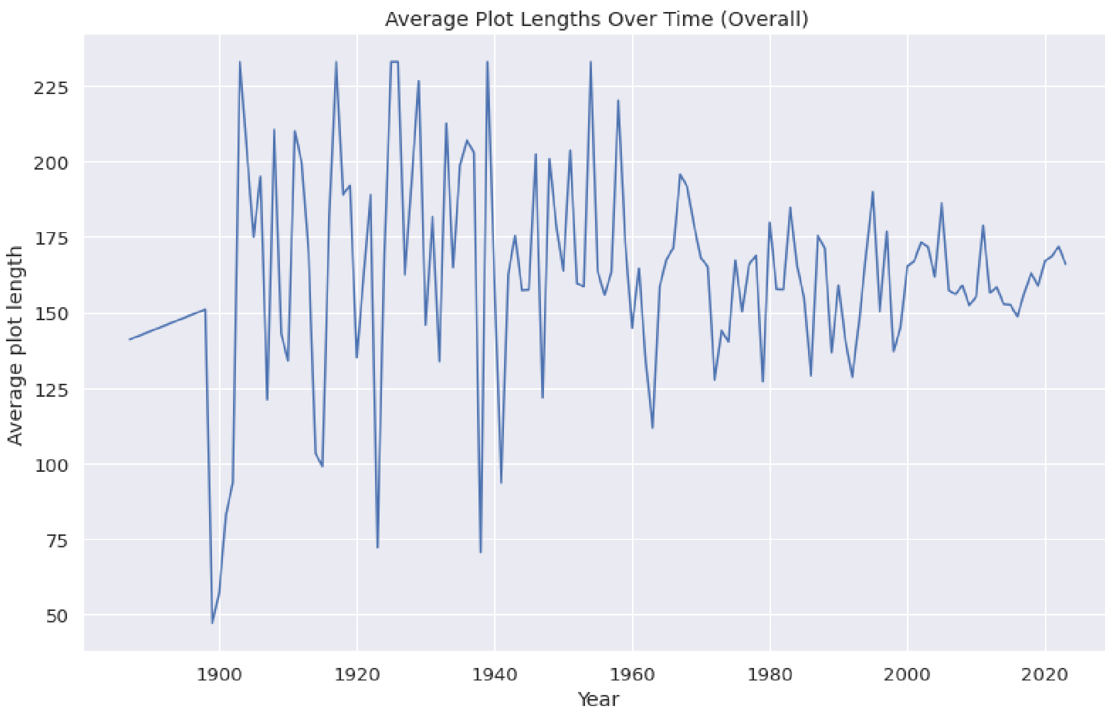

400
300
200
100
0
Plot length
United States
Japan
United Kingdom
Canada
France
Australia
country

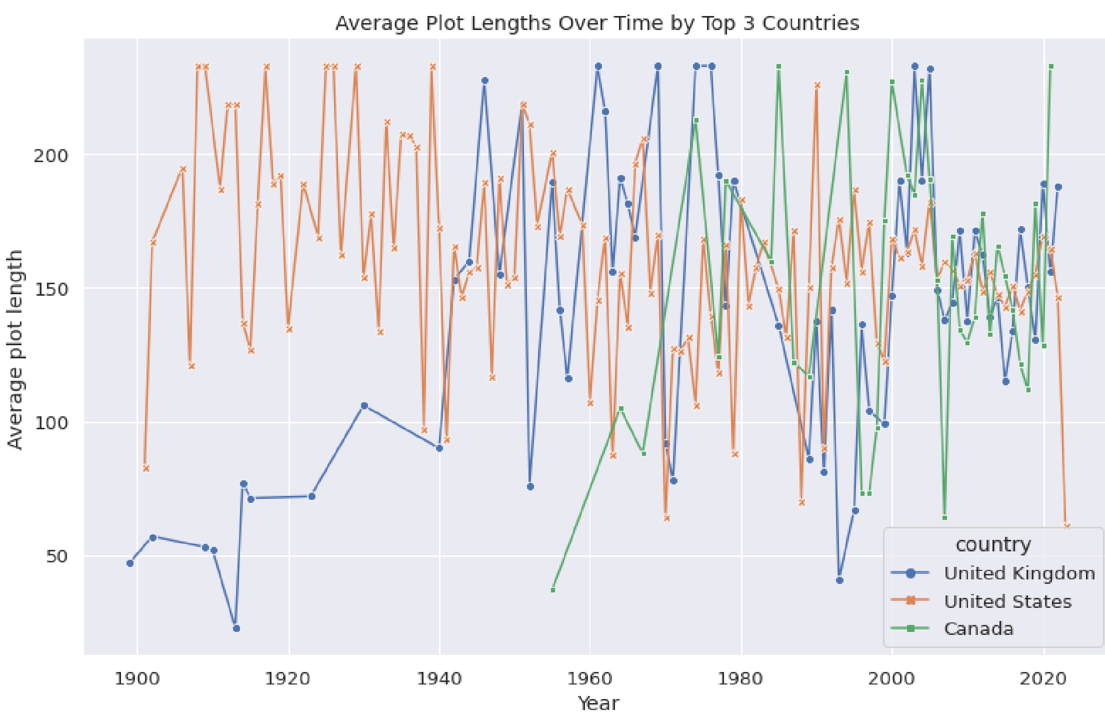
Interpretation

Both plots reveal that the lengths of the movie plots, for Japan has a higher median number of characters than the rest, otherwise all other countries seem to be closely distributed around the same length for movie plots.

4.5 5. Visualizing Movie Plots Over Time

[13]:
```
# Average plot lengths over
plots_avg = movies_info.groupby(['year'])['plot_length'].mean().reset_index()


# time series plot of average plot length overall
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.lineplot(data=plots_avg, x="year", y="plot_length")
plt.title("Average Plot Lengths Over Time (Overall)")
plt.xlabel("Year")
plt.ylabel("Average plot length")
plt.show()


# Average plot lengths over time by countries
plt.figure(figsize=(13,8))
plots_over_time = movies_info.groupby(['year','country'])['plot_length'].mean().
 ↪reset_index()
where = plots_over_time['country'].isin(['United States', 'United␣
 ↪Kingdom','Canada'])
plots_over_time = plots_over_time[where]

# time series plot of average plot length by countries
sns.lineplot(data=plots_over_time, x="year", y="plot_length", hue="country",␣
 ↪style = "country",
    markers=True, dashes=False)
plt.title("Average Plot Lengths Over Time by Top 3 Countries")
plt.xlabel("Year")
plt.ylabel("Average plot length")
plt.show()
```





Intepretation

The timeline reflects the number of average movie plot lengths over the years with all the countries, then a second plot aggregating by the top 3 countries with the most movies avaliable in this sample, which is the United States, United Kingdom, and Canada. Overall, movies across all countries in this sample have created shorter movie plot descriptions over time, whereas in the top 3 countries, any real pattern is hard to distinguish as there are a lot of variations.

5 Preparing Texual Data for Statistics and Modeling

- Remove Punctuation
- Remove extra white space
- Tokenize on white space pattern
- Fold to lowercase
- Remove stopwords

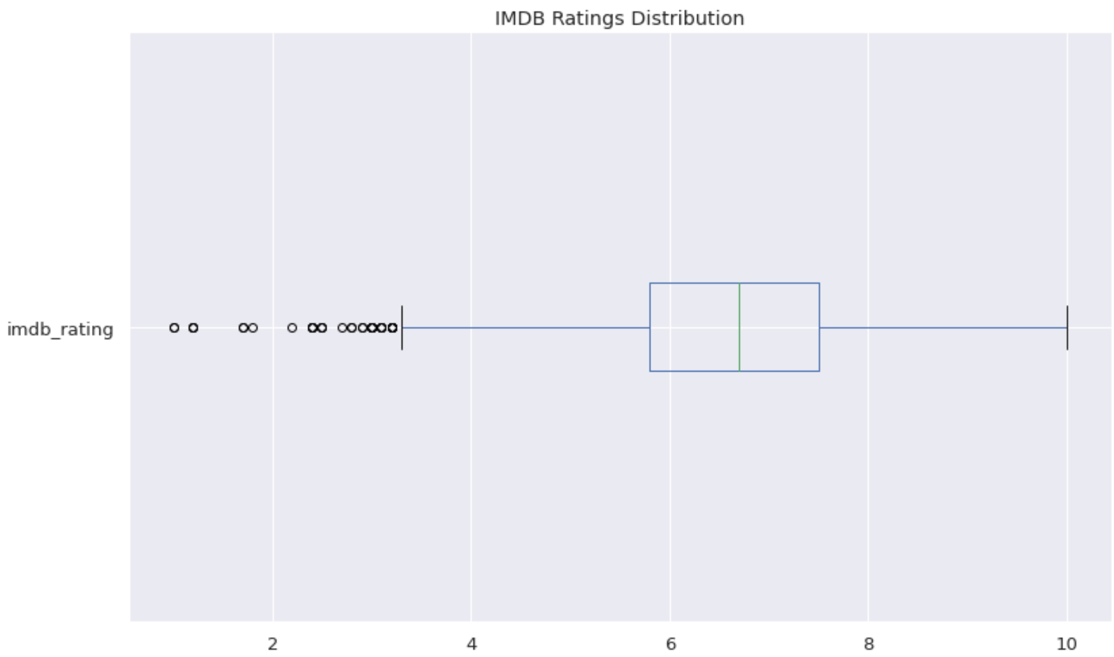
[14]:
```
punctuation = set(punctuation)
# Text cleaning function
def clean_text_data(column):
    new_description = []
    for description in column:
        update_desc = description
        # Remove the punctuation from each description
        for i in description:
            if i in punctuation:
                update_desc = update_desc.replace(i, "")
        # Remove extra white space
        update_desc = re.sub(r'\s+', ' ', update_desc)
        # Split on whitespace
        update_desc = update_desc.split()
        # Fold to lowercase
        for i in range(len(update_desc)):
            update_desc[i] = update_desc[i].lower()
        # Remove stopwords
        update_desc = [i for i in update_desc if i not in sw]
        new_description.append(update_desc)
    return new_description
```

[15]:
```
#Remove empty lists from cleaned_genre
movies_info = movies_info[movies_info['genre'] != ' ']

# Clean the plot description and genre text
movies_info['cleaned_plot'] = clean_text_data(movies_info['plot'])
movies_info['cleaned_genre'] = clean_text_data(movies_info['genre'])

# Keep only the first word in the cleaned genre lists
movies_info['first_genre'] = [i[0] for i in movies_info['cleaned_genre']]

# Create new df with only the first_genre and cleaned_plot columns
```

```
cleaned_df = movies_info[['title', 'first_genre', 'cleaned_plot',␣
↪'imdb_rating']]
cleaned_df.sample(15)
```

[15]:
```
                                   title  first_genre  \
4856              And Then There Was One        short
494             One Day You'll Understand        drama
436                    One in a Thousand        drama
1787            Fifty People One Question  documentary
4293                       One Year Later        short
4318                   One Pillow One Soul       family
2998                    I Am the Other One  documentary
3758              One Minute to Midnight        short
488       Sam Smith: I'm Not the Only One        short
1540                       One Small Step        short
3961                Laal Vaali (The Red One)        short
1435               I'm Your Number One Fan  documentary
4114                 One Heart: One Spirit  documentary
2330   One Hot Rotting, Zombie Love Song        short
4246                      One and the Same        short

                                            cleaned_plot  imdb_rating
4856  [woman, discovers, shes, pregnant, finds, husb…          NaN
494   [man, endeavors, collect, memories, grandparen…          5.7
436   [iris, expelled, school, spends, days, cousins…          6.3
1787  [one, town, 50, different, people, one, diffic…          5.8
4293  [one, year, later, spooky, halloween, fairytal…          NaN
4318  [story, tomb, removal, story, conflict, family…          NaN
2998  [luca, longs, lost, love, thalles, name, chang…          NaN
3758  [friendship, two, teenage, boys, imminent, nuc…          NaN
488    [music, video, sam, smiths, song, know, im, one]          8.2
1540  [dasani, 9, attempts, juggle, responsibilities…          8.8
3961  [comedy, errors, situation, arises, lata, find…          NaN
1435  [professor, paul, mullen, looks, way, admirati…          7.2
4114  [aboriginal, australian, native, american, doc…          NaN
2330  [jonas, kindhearted, zombie, sick, tired, kill…          5.4
4246  [still, photographer, sound, recordist, indepe…          NaN
```

[16]:
```
# Plotting Value Distribtuions
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
cleaned_df['imdb_rating'].plot(kind='box', vert=False)
plt.title('IMDB Ratings Distribution')
plt.show()
```



[17]:
```
# Count instances of each genre
cleaned_df['first_genre'].value_counts()
```

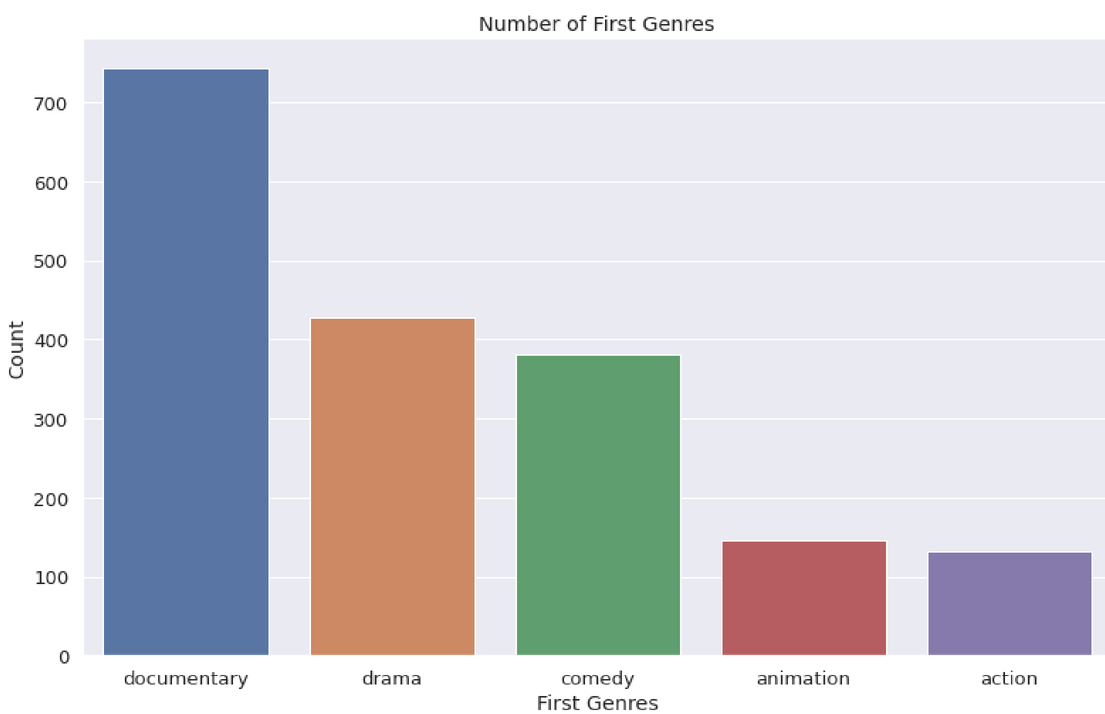
[17]:
```
short          1342
documentary     743
drama           428
comedy          380
animation       145
action          132
crime            71
horror           40
adventure        35
thriller         32
music            32
biography        20
family           19
western          17
romance          15
scifi            15
sport             9
fantasy           9
mystery           9
musical           5
history           5
realitytv         3
talkshow          3
```

```
news                  1
Name: first_genre, dtype: int64
```

5.1 Only use top 5 first genres

- Include descriptive statistics on final clean data set
- Frequency Diagram
- Word cloud

[18]:
```
# Include only the top 5 genres, excluding shorts and na
cleaned_df = cleaned_df.loc[cleaned_df['first_genre'].isin(['documentary',␣
 ↪'drama', 'comedy', 'animation', 'action'])]

# plot top 5 genres
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.countplot(x="first_genre", data=cleaned_df,
order = cleaned_df['first_genre'].value_counts().index)
plt.title("Number of First Genres")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```



5.2 Descriptive Statistics

[19]:
```
def descriptive_stats(tokens, common_tokens = 5, verbose=True) :
    """
        Given a list of tokens, print number of tokens, number of unique␣
    ↪tokens,
        number of characters, lexical diversity (https://en.wikipedia.org/wiki/
    ↪Lexical_diversity),
        and num_tokens most common tokens. Return a list with the number of␣
    ↪tokens, number
        of unique tokens, lexical diversity, and number of characters.

    """

    # Fill in the correct values here.
    num_tokens = len(tokens)
    num_unique_tokens = len(set(tokens))
    lexical_diversity = num_unique_tokens/num_tokens
    num_characters = sum([len(i) for i in tokens])

    if verbose :
        print(f"There are {num_tokens} tokens in the data.")
        print(f"There are {num_unique_tokens} unique tokens in the data.")
        print(f"There are {num_characters} characters in the data.")
        print(f"The lexical diversity is {lexical_diversity:.3f} in the data.")

        # print the five most common tokens
        counter = Counter(tokens)
        counter_list = counter.most_common(common_tokens)
        print(f"Most {common_tokens} common words: {counter_list}\n")
```

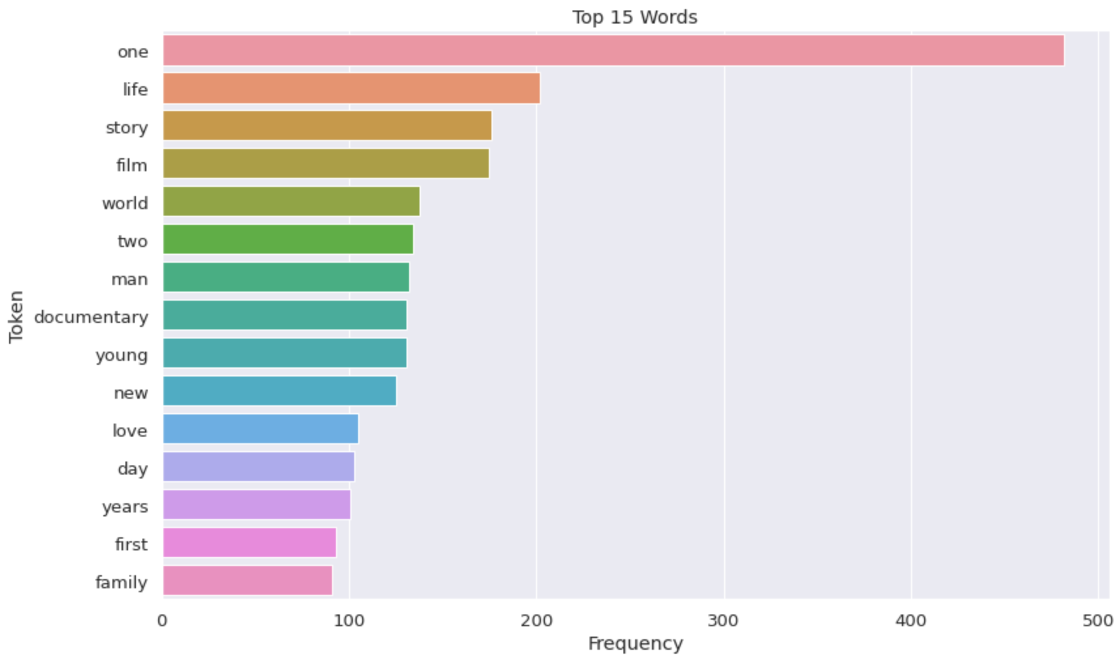
[20]:
```
movie_plot_tokens = cleaned_df.apply(lambda x: pd.
    ↪Series(x['cleaned_plot']),axis=1).stack().reset_index(level=1, drop=True).
    ↪tolist()
print("\t Movies Plot Descriptions:")
descriptive_stats(movie_plot_tokens)
```

```
         Movies Plot Descriptions:
There are 31048 tokens in the data.
There are 9494 unique tokens in the data.
There are 191378 characters in the data.
The lexical diversity is 0.306 in the data.
Most 5 common words: [('one', 482), ('life', 202), ('story', 176), ('film',
175), ('world', 138)]
```

5.3 Creating a Frequency Diagram

[21]:
```
def count_words(cleaned_df, column='cleaned_plot', preprocess=None, min_freq=2):
    # process tokens and update counter
    def update(doc):
        tokens = doc if preprocess is None else preprocess(doc)
        counter.update(tokens)

    # create counter and run through all data
    counter = Counter()
    cleaned_df[column].map(update)

    # transform counter into a DataFrame
    freq_df = pd.DataFrame.from_dict(counter, orient='index', columns=['freq'])
    freq_df = freq_df.query('freq >= @min_freq')
    freq_df.index.name = 'token'

    return freq_df.sort_values('freq', ascending=False)


freq_df = count_words(cleaned_df).reset_index()
```

[22]:
```
# Plot Frequency Diagram
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.barplot(x="freq", y="token", data=freq_df.head(15), orient = "h")
#ax.invert_yaxis()
plt.title("Top 15 Words")
plt.xlabel("Frequency")
plt.ylabel("Token")
plt.show()
```



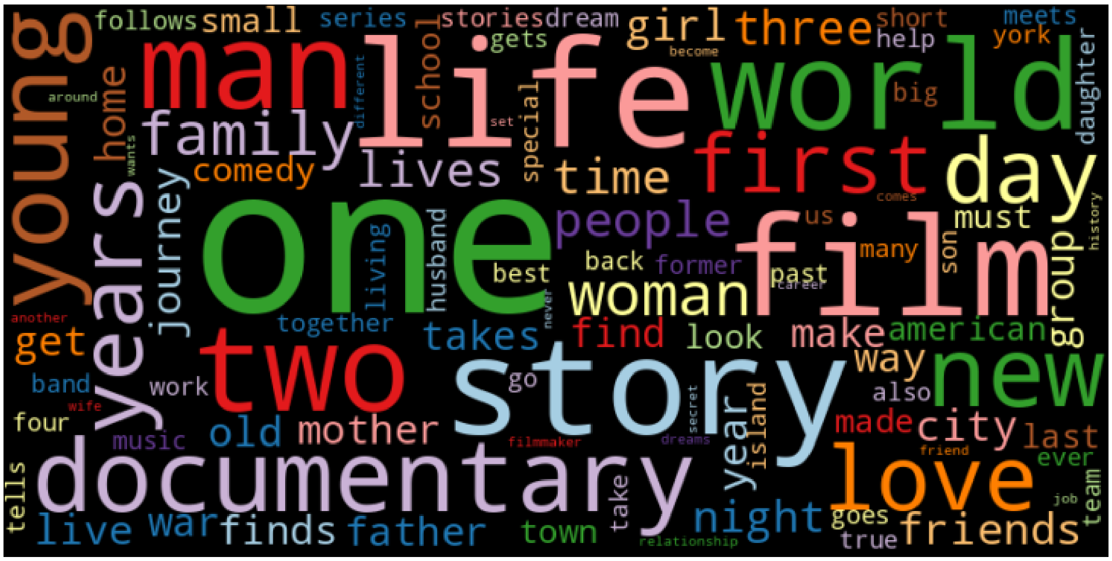
5.4 Word Cloud

[23]:
```
def wordcloud(word_freq, title=None, max_words=200, stopwords=None):
    # Create word cloud
    wc = WordCloud(width=800, height=400,
        background_color= "black", colormap="Paired",
        max_font_size=150, max_words=max_words)

    # convert DataFrame into dict
    if type(word_freq) == pd.Series:
        counter = Counter(word_freq.fillna(0).to_dict())
    else:
        counter = word_freq
    # filter stop words in frequency counter
    if stopwords is not None:
        counter = {token:freq for (token, freq) in counter.items() if token not␣
↪in stopwords}

    wc.generate_from_frequencies(counter)
    plt.title(title)
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
```

[24]:
```
# Plot Word Cloud
plt.figure(figsize=(13,8))
```

```python
freq_df = count_words(cleaned_df)
wordcloud(freq_df['freq'], max_words=100)
plt.title("Top 100 Words")
plt.show()
```

Top 100 Words



5.5 Export Final Clean Dataset

[25]:
```python
# Write to a csv
cleaned_df.to_csv('Cleaned Plot Data.csv', index = False)
```

Created in Deepnote

Classification Models

June 26, 2022

1 Final Project - Movie Reviews Analysis

- **Team 13: Jimmy Nguyen, Dallin Munger, Tyler Wolff**

2 Packages

[1]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from spacy.lang.en.stop_words import STOP_WORDS as stopwords
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
from sklearn.dummy import DummyClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('ignore')
import pickle
import pandas as pd
import numpy as np
import re
import matplotlib.pyplot as plt
import seaborn as sns
```

```
/shared-libs/python3.7/py/lib/python3.7/site-packages/tqdm/auto.py:22:
TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

3 Step 1: Data Preparation

- Loading Data set for Modeling
- Un-tokenize previous columns that was converted into tokens for descriptive statistics

[2]:
```
# load cleaned data set from previous notebook
df = pd.read_csv("Cleaned Plot Data.csv")
```

```python
df.sample(10)
```

[2]:
```
                                                  title  first_genre  \
201   One Piece: 3D2Y - Overcome Ace's Death! Luffy'…    animation
655                                    One Day of Betty        drama
138   Gabriel &quot;Fluffy&quot; Iglesias: One Show …       comedy
49                 The Tall Blond Man with One Black Shoe       comedy
1544  Marilyn Waring on Politics, Local & Global, Sh…  documentary
1216                                  This Man Is the One  documentary
395                                         The Quiet One  documentary
643                                     SWAT: Warhead One       action
972       One Bad Cat: The Reverend Albert Wagner Story  documentary
463                                         Just One Look       comedy

                                           cleaned_plot  imdb_rating
201   ['special', 'takes', 'place', 'two', 'year', '…          7.8
655   ['free', 'entry', 'adventurous', 'journey', 'a…          5.7
138   ['gabriel', 'fluffy', 'iglesias', 'discusses',…          7.2
49    ['hapless', 'orchestra', 'player', 'becomes', …          7.2
1544  ['economist', 'marilyn', 'waring', 'uses', 'ex…          NaN
1216  ['retrospective', 'adam', 'adamant', 'lives', …          6.9
395   ['quiet', 'one', 'offers', 'unique', 'never', …          7.1
643   ['los', 'angeles', 'special', 'police', 'force…          2.5
972   ['one', 'bad', 'cat', 'transformative', 'role'…          8.4
463   ['best', 'friends', 'start', 'kung', 'fu', 'le…          6.8
```

[3]:
```python
# untokenize plot descriptions
df['cleaned_plot'] = df['cleaned_plot'].str.replace(r'[^\w\s]', '', regex=True).str.strip()
df['cleaned_plot'].sample(10)
```

[3]:
```
7       mentally unstable photo developer targets uppe…
13      struggling recover emotionally brutal assault …
443     brilliant corporate lawyer luther simmonds acc…
1542    twenty years reunification germany still divid…
700     story anakin skywalker central character star …
448     contemporary story set perth western australia…
1570    alan parker lives enviable life one day januar…
1704    origin story worlds greatest heromemoirist sir…
88      making killing fen returns china rich man seek…
973     motley gang characters includes movie star rep…
Name: cleaned_plot, dtype: object
```

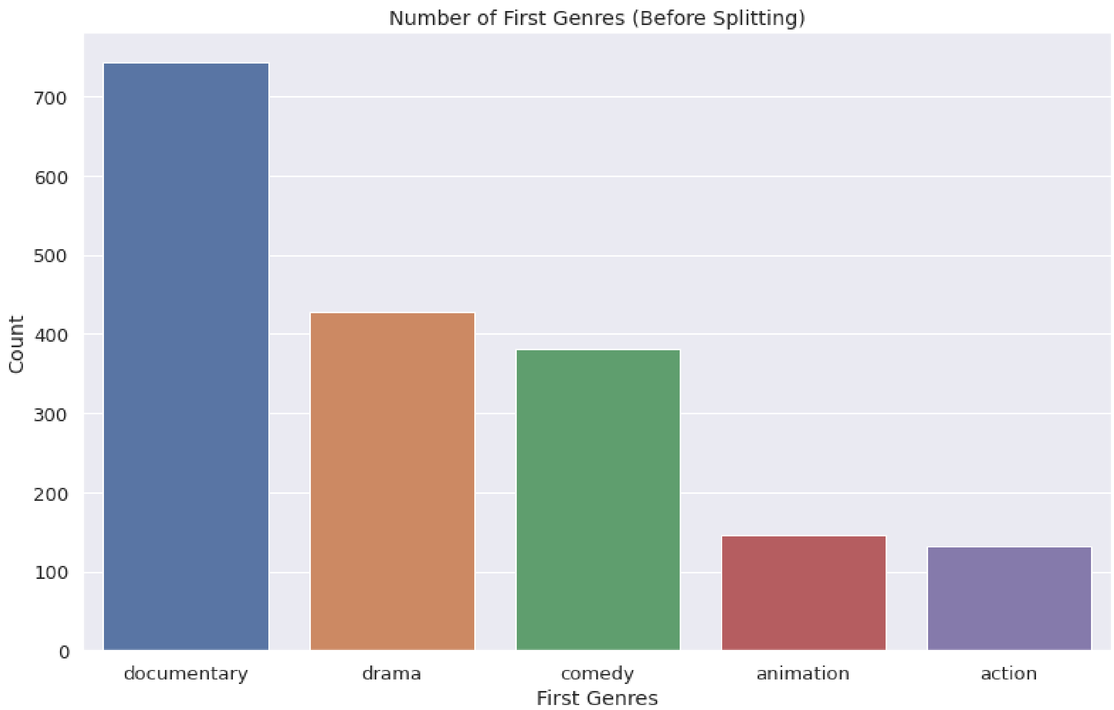
4 Step 2: Train-Test Split

- Checking for Class Imbalance
- Downsampling the Majority class

- Split Data on balanced data set

4.1 Checking for Class Imbalance

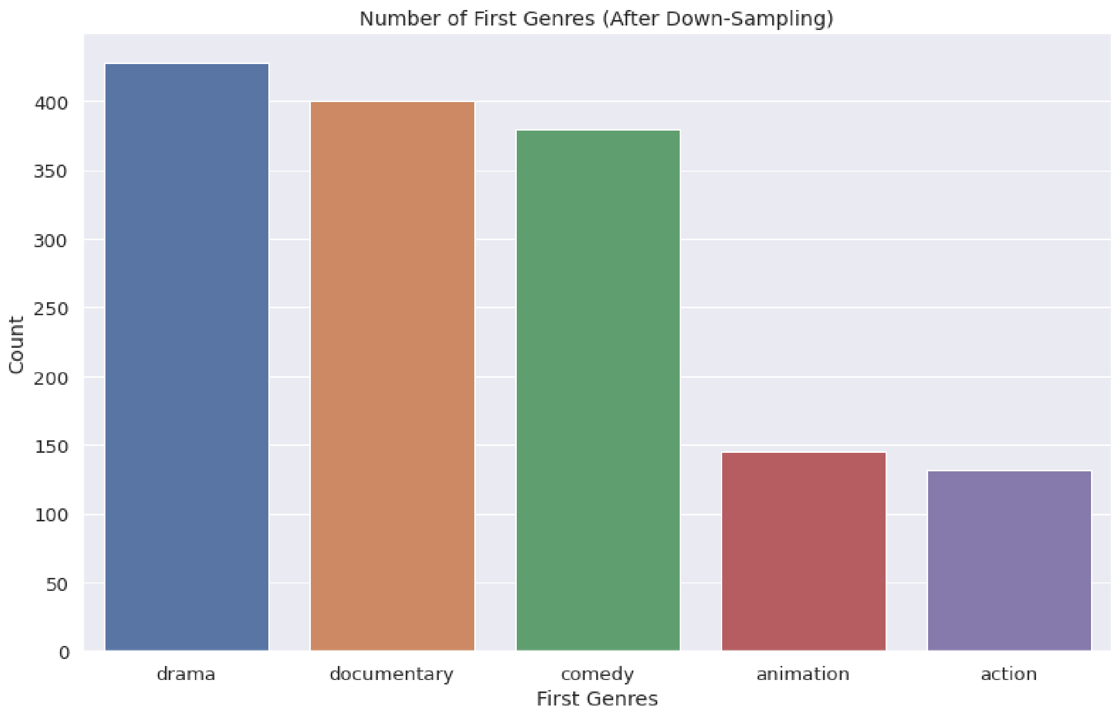
[4]:
```
# plot original target variable classes
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.countplot(x="first_genre", data=df,
order = df['first_genre'].value_counts().index)
plt.title("Number of First Genres (Before Splitting)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```



4.2 Downsampling the Majority class

[5]:
```
# Filter for documentaries and sample 400 rows from it
df_sample = df[df['first_genre'] == 'documentary'].sample(n=400)
# Create a separate DataFrame containing all other genres
df_sampleRest = df[df['first_genre'] != 'documentary']
# Concatenate the two DataFrame to create the new balanced bug reports dataset
df_balanced = pd.concat([df_sampleRest, df_sample])
```

```python
# plot new downsampled target variable classes
plt.figure(figsize=(13,8))
sns.set(font_scale = 1.2)
sns.countplot(x="first_genre", data=df_balanced,
order = df_balanced['first_genre'].value_counts().index)
plt.title("Number of First Genres (After Down-Sampling)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```



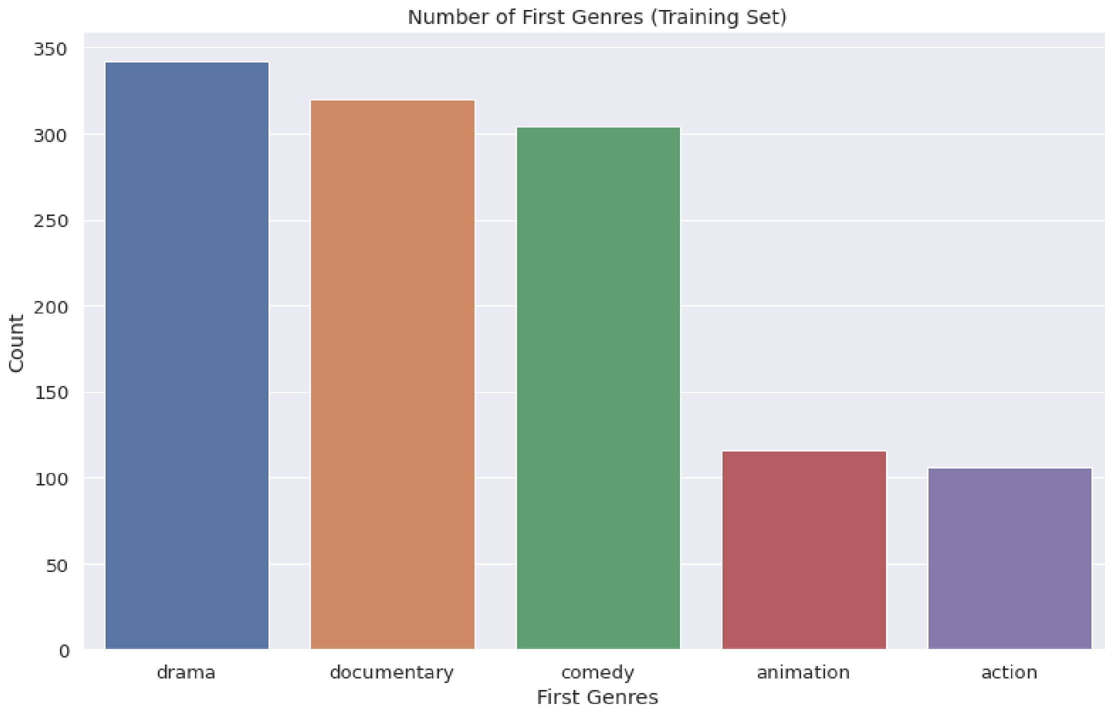
4.3 Split data on balanced data set

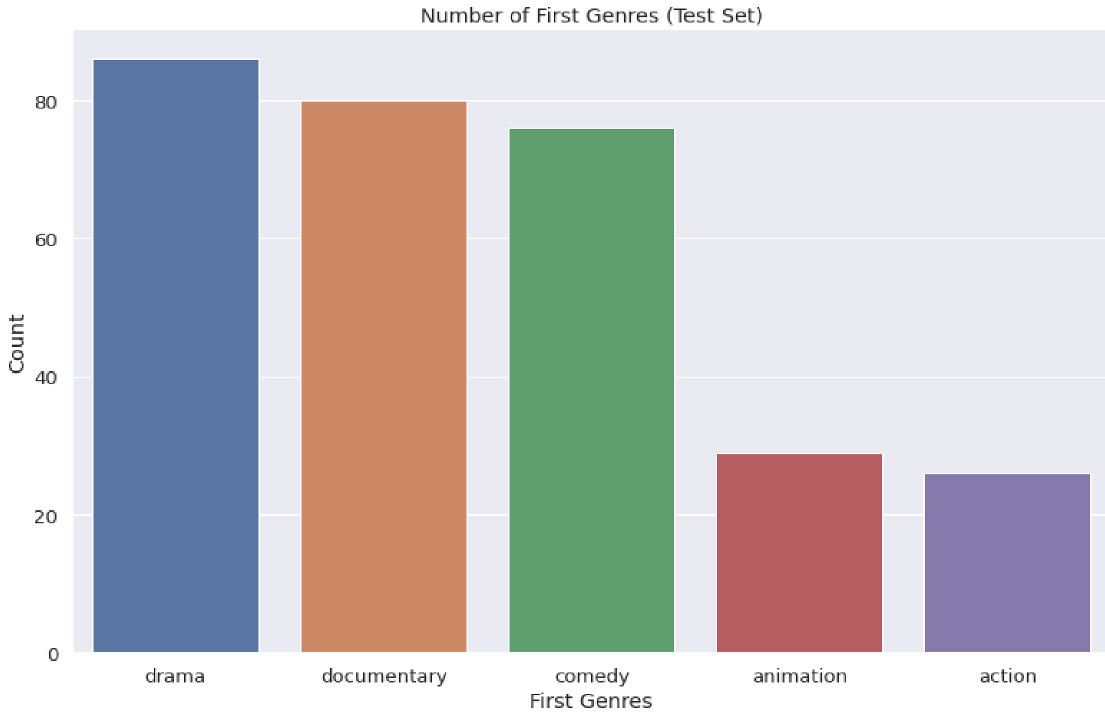
[6]:
```python
# Split data set on 80% training and 20% test and keep target variable classes␣
↪balanced
X_train, X_test, Y_train, Y_test =␣
↪train_test_split(df_balanced['cleaned_plot'],\
df_balanced['first_genre'], test_size=0.2, random_state=42,\
stratify=df_balanced['first_genre'])

# Print shapes of training and testing data
print('Size of Training Data ', X_train.shape[0])
print('Size of Test Data ', X_test.shape[0])
```

```
Size of Training Data  1188
```

```
Size of Test Data  297
```

[7]:
```
# plot original target variable classes
plt.figure(figsize=(13,8))
training_y = pd.DataFrame(Y_train)
sns.set(font_scale = 1.2)
sns.countplot(x = "first_genre", data=training_y,
order = training_y['first_genre'].value_counts().index)
plt.title("Number of First Genres (Training Set)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()

# plot original target variable classes
plt.figure(figsize=(13,8))
test_y = pd.DataFrame(Y_test)
sns.set(font_scale = 1.2)
sns.countplot(x = "first_genre", data=test_y,
order = test_y['first_genre'].value_counts().index)
plt.title("Number of First Genres (Test Set)")
plt.xlabel("First Genres")
plt.ylabel("Count")
plt.show()
```





5 Step 3: Training the Machine Learning Model

- Use TFIDF transformer for words to features
- Model 1: Linear SVC

5.1 Use TFIDF transformer for words to features

[8]:
```
tfidf = TfidfVectorizer(min_df = 10, stop_words=stopwords)
X_train_tf = tfidf.fit_transform(X_train)
```

5.2 Model 1: Linear SVC

[9]:
```
model1 = LinearSVC(random_state=0, tol=1e-5)
model1.fit(X_train_tf, Y_train)
```

[9]:
```
LinearSVC(random_state=0, tol=1e-05)
```

6 Step 4: Model Evaluation

- Model 1 Evaluation
- Baseline model evaluation
- Using Cross-Validation to Estimate Accuracy

6.1 Model 1 Evaluation

[10]:
```
# Model 1 evaluation
X_test_tf = tfidf.transform(X_test)
Y_pred = model1.predict(X_test_tf)
print ('Accuracy Score - ', accuracy_score(Y_test, Y_pred))
```

```
Accuracy Score -  0.5353535353535354
```

6.2 Baseline Model Evaluation

[11]:
```
clf = DummyClassifier(strategy='most_frequent')
clf.fit(X_train, Y_train)
Y_pred_baseline = clf.predict(X_test)
print ('Accuracy Score - ', accuracy_score(Y_test, Y_pred_baseline))
```

```
Accuracy Score -  0.2895622895622896
```

6.3 Cross-validation to Estimate Accuracy

[12]:
```
tfidf = TfidfVectorizer(min_df = 10, stop_words=stopwords)
df_tf = tfidf.fit_transform(df_balanced['cleaned_plot']).toarray()

# Cross Validation with 5 folds
scores = cross_val_score(estimator=model1, X=df_tf,␣
 ↪y=df_balanced['first_genre'], cv=5)
print ("Validation scores from each iteration of the cross validation ", scores)
print ("Mean value across of validation scores ", scores.mean())
print ("Standard deviation of validation scores ", scores.std())
```

```
Validation scores from each iteration of the cross validation  [0.45791246
0.48821549 0.44781145 0.40740741 0.48821549]
Mean value across of validation scores  0.457912457912457944
Standard deviation of validation scores  0.029964438331699653
```

7 Performing Hyperparameter Tuning with Grid Search

- Set up parameters pipeline
- Select best hyperparameters
- Model evaluation after best parameters selected

7.1 Set up hyper-parameters pipeline and train

[13]:
```
# Add parameters in pipeline
training_pipeline = Pipeline(
steps=[('tfidf', TfidfVectorizer(stop_words=stopwords)),
('model', LinearSVC(random_state=42, tol=1e-5))])
grid_param = [{
```

```
'tfidf__min_df': [5, 10],
'tfidf__ngram_range': [(1, 3), (1, 6)],
'model__penalty': ['l2'],
'model__loss': ['hinge'],
'model__max_iter': [10000]
}, {
'tfidf__min_df': [5, 10],
'tfidf__ngram_range': [(1, 3), (1, 6)],
'model__C': [1, 10],
'model__tol': [1e-2, 1e-3]
}]

# grid search to find best parameters
gridSearchProcessor = GridSearchCV(estimator=training_pipeline,
param_grid=grid_param,
cv=5)
gridSearchProcessor.fit(df_balanced['cleaned_plot'],
df_balanced['first_genre'])

# best parameters
best_params = gridSearchProcessor.best_params_

# best model
best_model = gridSearchProcessor.best_estimator_
```

7.2 Select best hyper-parameters

[14]:
```
# print out best parameters
print("Best alpha parameter identified by grid search ", best_params)
best_result = gridSearchProcessor.best_score_
print("Best result identified by grid search ", best_result)
```

```
Best alpha parameter identified by grid search  {'model__C': 1, 'model__tol':
0.001, 'tfidf__min_df': 5, 'tfidf__ngram_range': (1, 3)}
Best result identified by grid search  0.503030303030303
```

[15]:
```
# see other parameter results
gridsearch_results = pd.DataFrame(gridSearchProcessor.cv_results_)
gridsearch_results[['rank_test_score', 'mean_test_score',
'params']].sort_values(by=['rank_test_score'])[:5]
```

[15]:
```
   rank_test_score  mean_test_score  \
9                1         0.503030
8                1         0.503030
5                3         0.502357
4                3         0.502357
0                5         0.501010
```

```
                                              params
9  {'model__C': 1, 'model__tol': 0.001, 'tfidf__m…
8  {'model__C': 1, 'model__tol': 0.001, 'tfidf__m…
5  {'model__C': 1, 'model__tol': 0.01, 'tfidf__mi…
4  {'model__C': 1, 'model__tol': 0.01, 'tfidf__mi…
0  {'model__loss': 'hinge', 'model__max_iter': 10…
```

7.3 Model Evaluation After Hyper-parameter Tuning

[16]:
```python
# Step 4 - Model Evaluation
Y_pred = best_model.predict(X_test)
print('Accuracy Score - ', accuracy_score(Y_test, Y_pred))
print(classification_report(Y_test, Y_pred))
```

```
Accuracy Score -  0.9494949494949495
              precision    recall  f1-score   support

      action       1.00      0.96      0.98        26
   animation       0.96      0.93      0.95        29
      comedy       0.96      0.89      0.93        76
 documentary       0.96      0.99      0.98        80
       drama       0.91      0.97      0.94        86

    accuracy                           0.95       297
   macro avg       0.96      0.95      0.95       297
weighted avg       0.95      0.95      0.95       297
```

8 Export Model for Deployment

[17]:
```python
# save the model to disk
filename = '/work/Movies_Reviews_Analysis/models/final_classification_model.pkl'
pickle.dump(best_model, open(filename, 'wb'))

# some time later...

# load the model from disk
#final_model = pickle.load(open(filename, 'rb'))
```

Created in Deepnote

Topic Models

June 26, 2022

1 Final Project - Movie Reviews Analysis

- **Team 13: Jimmy Nguyen, Dallin Munger, Tyler Wolff**

2 Packages

[1]:
```
import numpy as np
import pandas as pd
from tqdm.auto import tqdm
import pyLDAvis
import pyLDAvis.sklearn
import pyLDAvis.gensim_models
import spacy
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, TruncatedSVD, LatentDirichletAllocation
from spacy.lang.en.stop_words import STOP_WORDS as stopwords
from collections import Counter, defaultdict
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation
from wordcloud import WordCloud
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)


# Define function to display topics for the topic models (directly from BTAP␣
↪repo)
def display_topics(model, features, no_top_words=5):
    for topic, words in enumerate(model.components_):
        total = words.sum()
        largest = words.argsort()[::-1] # invert sort order
        print("\nTopic %02d" % topic)
        for i in range(0, no_top_words):
            print("  %s (%2.2f)" % (features[largest[i]],␣
↪abs(words[largest[i]]*100.0/total)))
```

```
/home/jimmynguyen/anaconda3/envs/ads509/lib/python3.9/site-
packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

3 Prepare Data

[2]:
```
# Read in the cleaned plot data
plot_data = pd.read_csv('Cleaned Plot Data.csv')
# View dataframe
plot_data.sample(5)
```

[2]:
```
                                title  first_genre  \
394                   The Perfect One        drama
115            Father There Is Only One       comedy
1656  One Wish for Iran, Love Israel  documentary
1817                One Hundred Steps  documentary
913                     The Lucky One       comedy

                                           cleaned_plot  imdb_rating
394   ['new', 'mother', 'dealing', 'postpartum', 'de…          5.6
115   ['family', 'father', 'discovers', 'hard', 'car…          6.0
1656  ['one', 'wish', 'iran', 'love', 'israel', 'hig…          NaN
1817  ['forty', 'years', 'frank', 'garfunkel', 'taug…          NaN
913   ['countryman', 'arrives', 'athens', 'make', 'b…          5.0
```

[3]:
```
# Untokenize plot descriptions
plot_data['cleaned_plot'] = plot_data['cleaned_plot'].str.replace(r'[^\w\s]',␣
↪'', regex= True).str.strip()
plot_data['cleaned_plot'].sample(5)
```

[3]:
```
1806    sensitive portrait psychotic glamorous drugadd…
496     gold lion shiki offered alliance gol roger lat…
1734    modern day parable setting mail package proces…
903     early 1980s beginning would become 12yearlong …
1620    wal junior meg get selected go france study la…
Name: cleaned_plot, dtype: object
```

[4]:
```
# TF-IDF text vectorization
tfidf_text_vectorizer = TfidfVectorizer(stop_words=stopwords, min_df=3)
plot_data_tfidf = tfidf_text_vectorizer.fit_transform(plot_data["cleaned_plot"])
plot_data_tfidf.shape
```

[4]:
```
(1828, 2414)
```

4 Topic Modeling

4.0.1 Non-Negative Matrix Factorization

[5]:
```
# Non-Negative Matrix Factorization Model
plot_nmf_model = NMF(n_components=5, random_state=42)
W_text_matrix = plot_nmf_model.fit_transform(plot_data_tfidf)
H_text_matrix = plot_nmf_model.components_

# Show results of the topic model
display_topics(plot_nmf_model, tfidf_text_vectorizer.get_feature_names())
```

```
Topic 00
  young (2.01)
  man (1.94)
  woman (1.49)
  love (1.49)
  girl (0.91)

Topic 01
  documentary (3.88)
  film (3.86)
  short (1.53)
  new (1.04)
  making (0.77)

Topic 02
  story (7.19)
  tells (1.62)
  based (1.29)
  journey (1.11)
  true (0.98)

Topic 03
  life (7.42)
  years (1.29)
  art (0.85)
  musician (0.70)
  jokes (0.70)

Topic 04
  day (8.09)
  lives (1.51)
  night (1.07)
  peace (0.96)
  city (0.86)
```

[6]:
```
# Create document-topic dataframe and add genre column
def genre_by_topic(df):
    document_topic = pd.DataFrame(df)
    topic_genre = pd.concat([document_topic.idxmax(axis=1),
 ↪plot_data['first_genre']], axis=1)
    topic_genre.columns = ['topic', 'genre']
    return topic_genre.groupby(['topic', 'genre']).size()
genre_by_topic(W_text_matrix)
```

[6]:
```
topic  genre
0      action          68
       animation       52
       comedy         210
       documentary     92
       drama          210
1      action          20
       animation       28
       comedy          59
       documentary    313
       drama           31
2      action          28
       animation       31
       comedy          34
       documentary    124
       drama           68
3      action          11
       animation       16
       comedy          45
       documentary    120
       drama           71
4      action           5
       animation       18
       comedy          32
       documentary     94
       drama           48
dtype: int64
```

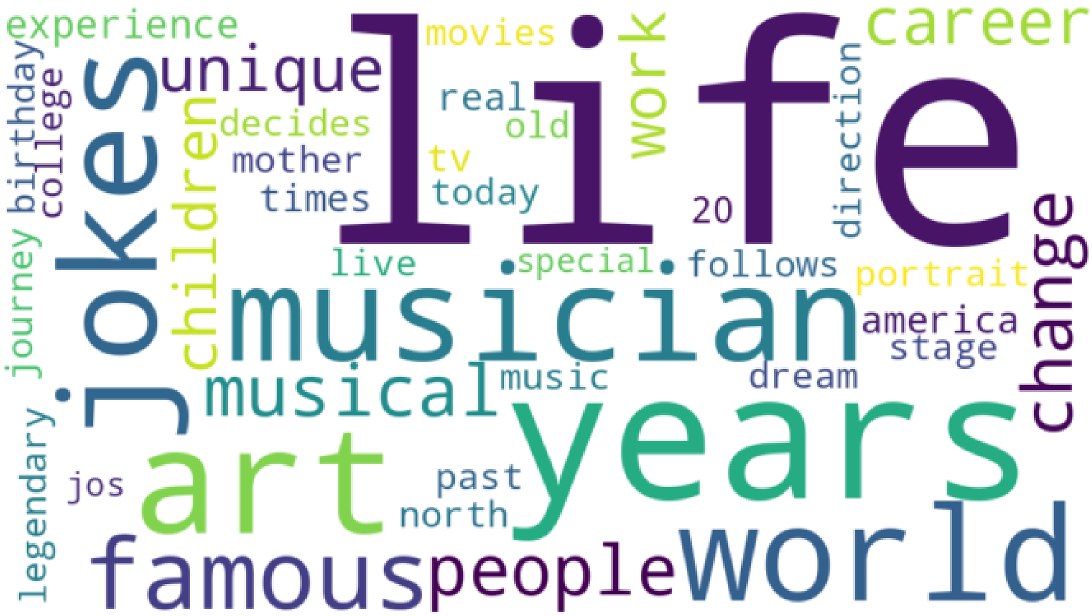
The NMF topic model appears to spread each genre around within each topic. Topic 0 has a higher concentration of drama and comedy. Topic 1 does appear to group documentary primarily, but the last 3 topics appear to just spread the top 3 genres around. It is difficult to determine which topic would match with which genre.

[7]:
```
# display word cloud function
def wordcloud_topics(model, features, no_top_words=40):
    for topic, words in enumerate(model.components_):
        size = {}
        largest = words.argsort()[::-1] # invert sort order
```

```
        for i in range(0, no_top_words):
            size[features[largest[i]]] = abs(words[largest[i]])
        wc = WordCloud(background_color="white", max_words=100,
            width=960, height=540)
        wc.generate_from_frequencies(size)
        plt.figure(figsize=(12,12))
        plt.imshow(wc, interpolation='bilinear')
        plt.axis("off")


#NMF Word Cloud for Review


wordcloud_topics(plot_nmf_model, tfidf_text_vectorizer.get_feature_names())
```


new
film
trio
follows
classic
shorts
way
rare
comedy
making
musical
season
following
featuring
interviews
takes
explores
american
crew
filmed
stories
documentary
feature
history
special
people
short
look
city
cast
footage
scenes
chronicling
voice
music
live
world
york
band
director


approach
wars
little
battle
true
star
old
told
come
friends
trying
war
tells
world
based
worlds
year
years
team
baltic
american
joseph
athlete
discover
eyes
story
group
game
teenager
jokes
different
set
gay
youth
journey
united
drama
screen
100
short


life
years
musician
jokes
art
world
famous
people
musical
change
children
work
career
unique
experience
movies
real
old
decides
mother
tv
times
today
birthday
college
direction
20
live
special
follows
portrait
journey
america
stage
music
dream
legendary
jos
past
north

day
lives
night
city
peace
jeremy
launched
inspired
documented
states
project
gilley
unanimous
de
filmmaker
1999
old
years
race
la
modern
ordinary
place
school
sweet
adoption
stories
mother
people
community
wake
single
new
created
british
un
date
year
member
international

4.0.2 Latent Dirichlet Allocation

[8]:
```python
# Create Topics For LDA
count_para_vectorizer = CountVectorizer(stop_words=stopwords, min_df=3)
count_para_vectors = count_para_vectorizer.
 ↪fit_transform(plot_data['cleaned_plot'])
lda_para_model = LatentDirichletAllocation(n_components=5, random_state=42)
W_lda_para_matrix = lda_para_model.fit_transform(count_para_vectors)
H_lda_para_matrix = lda_para_model.components_

display_topics(lda_para_model, count_para_vectorizer.get_feature_names())
```

```
Topic 00
  film (1.26)
  world (1.23)
  documentary (1.02)
  love (0.92)
  girl (0.66)

Topic 01
  life (2.30)
  film (1.18)
  new (1.14)
  world (0.98)
  story (0.88)

Topic 02
  young (1.25)
  man (1.21)
  night (0.94)
  story (0.68)
  lives (0.63)

Topic 03
  story (1.48)
  man (0.92)
  film (0.92)
  life (0.85)
  years (0.84)

Topic 04
  young (1.19)
  woman (1.03)
  man (0.93)
  love (0.78)
  life (0.74)
```

[9]:
```
# Compare LDA topic modeling to original genres
genre_by_topic(W_lda_para_matrix)
```

[9]:
```
topic  genre
0      action          18
       animation       33
       comedy          53
       documentary    179
       drama           56
1      action          31
       animation       21
       comedy          61
       documentary    221
       drama           83
2      action          28
       animation       49
       comedy         100
       documentary     85
       drama          104
3      action          19
       animation       16
       comedy          78
       documentary    171
       drama           91
4      action          36
       animation       26
       comedy          88
       documentary     87
       drama           94
dtype: int64
```

Once again, it appears the genres are spread across topics. Each genre has one topic where there is a slightly higher concentration, but overall there is a fairly even distribution across groups.

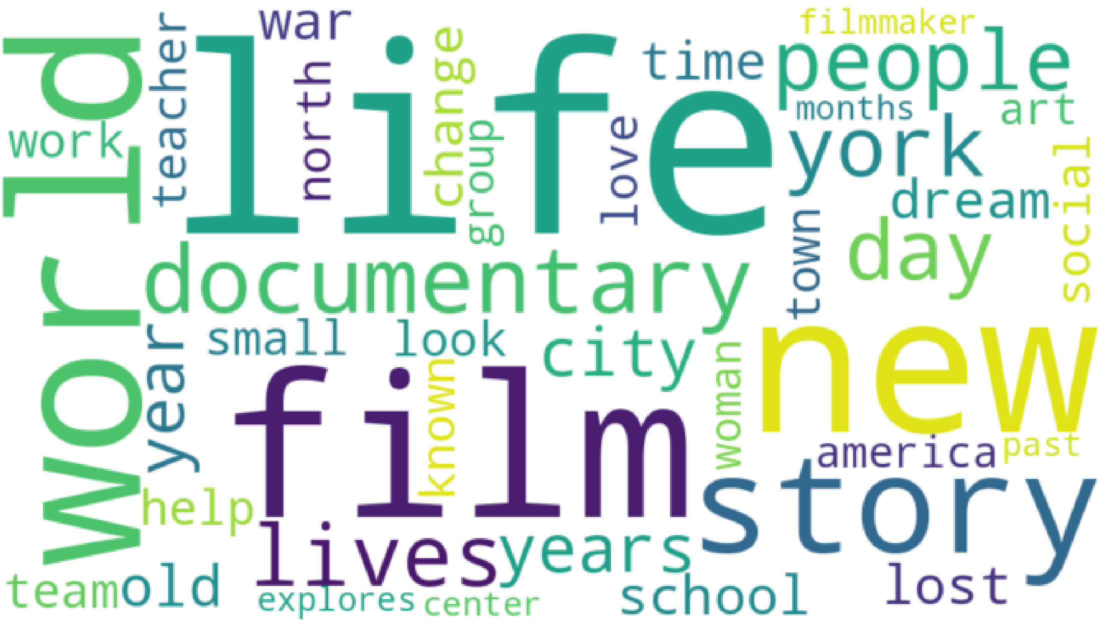
[10]:
```
lda_display = pyLDAvis.sklearn.prepare(lda_para_model, count_para_vectors,
 count_para_vectorizer, sort_topics=False)
pyLDAvis.display(lda_display)
```

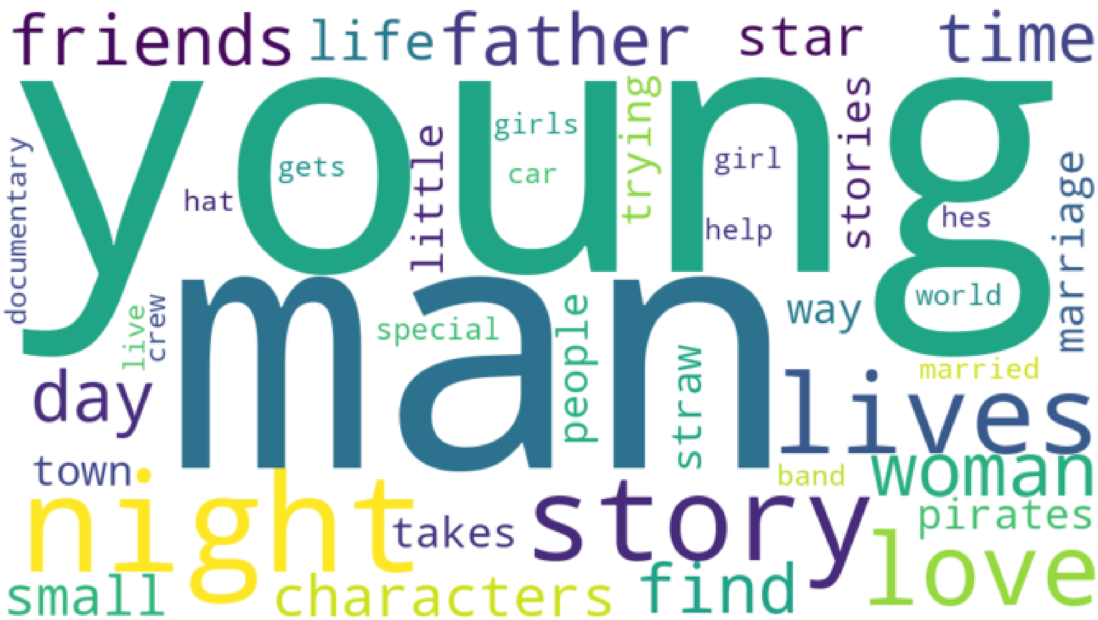
```
/home/jimmynguyen/anaconda3/envs/ads509/lib/python3.9/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/home/jimmynguyen/anaconda3/envs/ads509/lib/python3.9/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
```

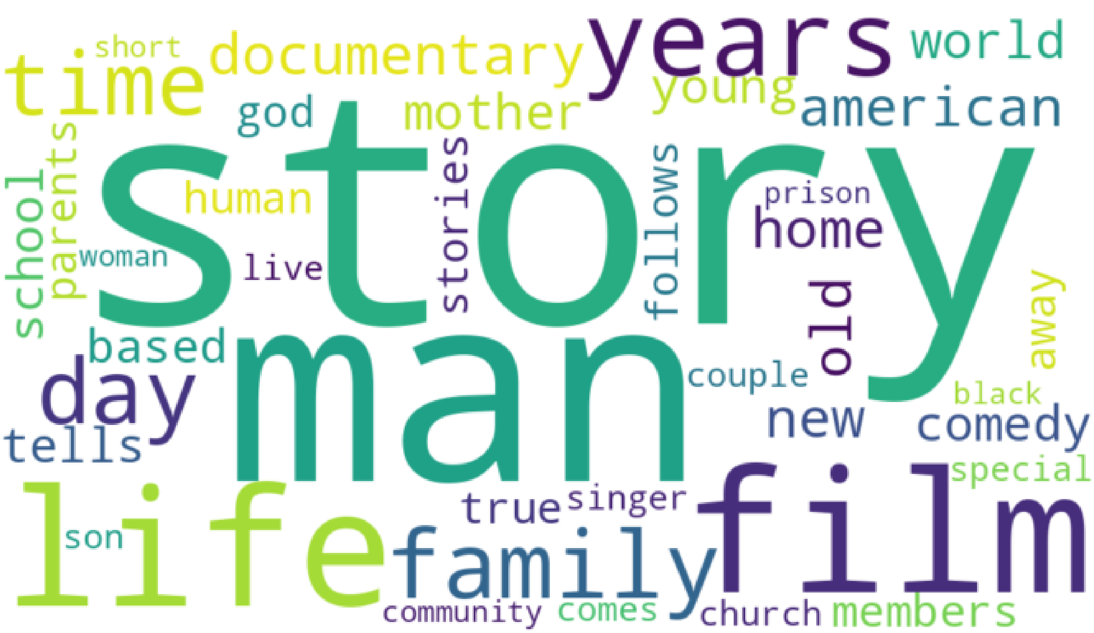
```
/home/jimmynguyen/anaconda3/envs/ads509/lib/python3.9/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
/home/jimmynguyen/anaconda3/envs/ads509/lib/python3.9/site-
packages/past/builtins/misc.py:45: DeprecationWarning: the imp module is
deprecated in favour of importlib; see the module's documentation for
alternative uses
  from imp import reload
```

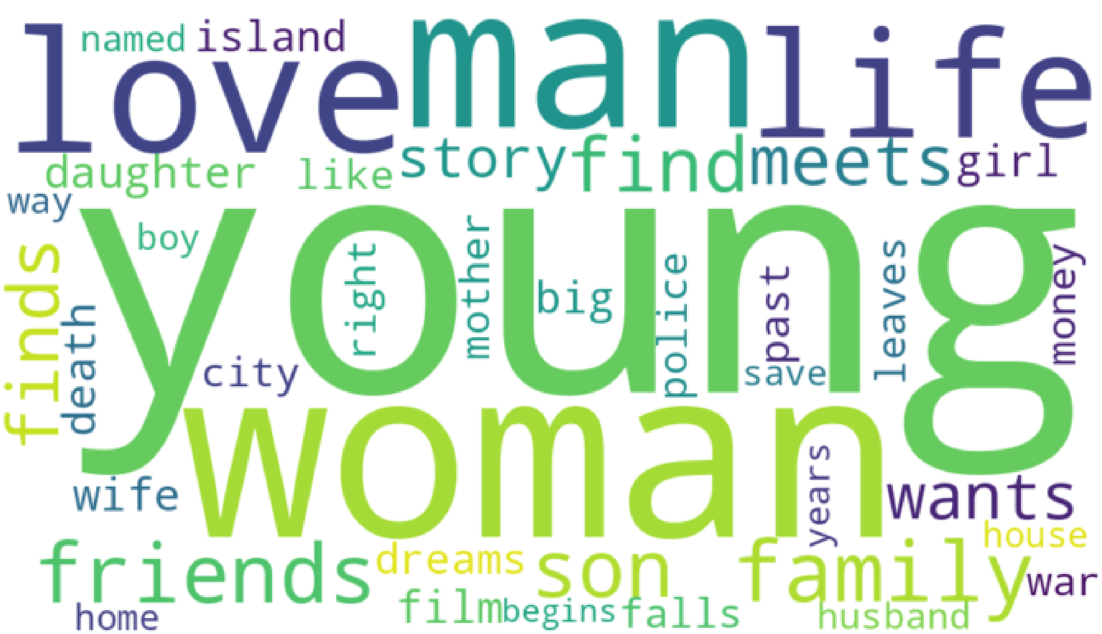
[10]:
```
<IPython.core.display.HTML object>
```

[11]:
```python
#LDA Word Cloud for Review
wordcloud_topics(lda_para_model, count_para_vectorizer.get_feature_names())
```


life
world
new
film
story
documentary
people
york
day
city
lives
years
time
war
teacher
north
change
group
love
town
dream
social
art
months
filmmaker
work
year
small
look
known
woman
america
past
help
team
old
explores
center
school
lost


young
man
friends
life
father
star
time
night
story
lives
woman
love
day
find
characters
small
takes
pirates
town
band
married
marriage
stories
trying
little
girls
gets
car
girl
hat
help
hes
world
way
special
people
straw
documentary
live
crew





5 Recommendations

1. Add more genres in the future (Where it can pull multiple genres if the plot fits in more than one)
2. More in depth design with the application to make it more attractive for the user

3. Train the models on shorter descriptions (Currently accuracy with less words is down)
4. Create a model that can work for longer plot descriptions that way if a movie writer was unsure of the genre of their future film they can put the entire plot into the application and then it will give them an accurate genre

Created in Deepnote