Data analysis

October 9, 2021

#

Final project

Lê Vũ Lợi - Trần Thị Uyên

0.1 1. Introduction:

Nowadays, the use of online services to watch movies has become familiar for people. The development of movies platforms makes it easier for users to access movies. However, too many choices also make it difficult for users to find the right movie. A Recommendation system could solve this problem.

Dataset:

- MovieLens 20M : Contains 20000263 ratings and 465564 tag applications across 27278 movies. Th

- File data used: movies.cvs, ratings.cvs, tags.csv, links.csv

Procedure

- Store data in hadoop File system

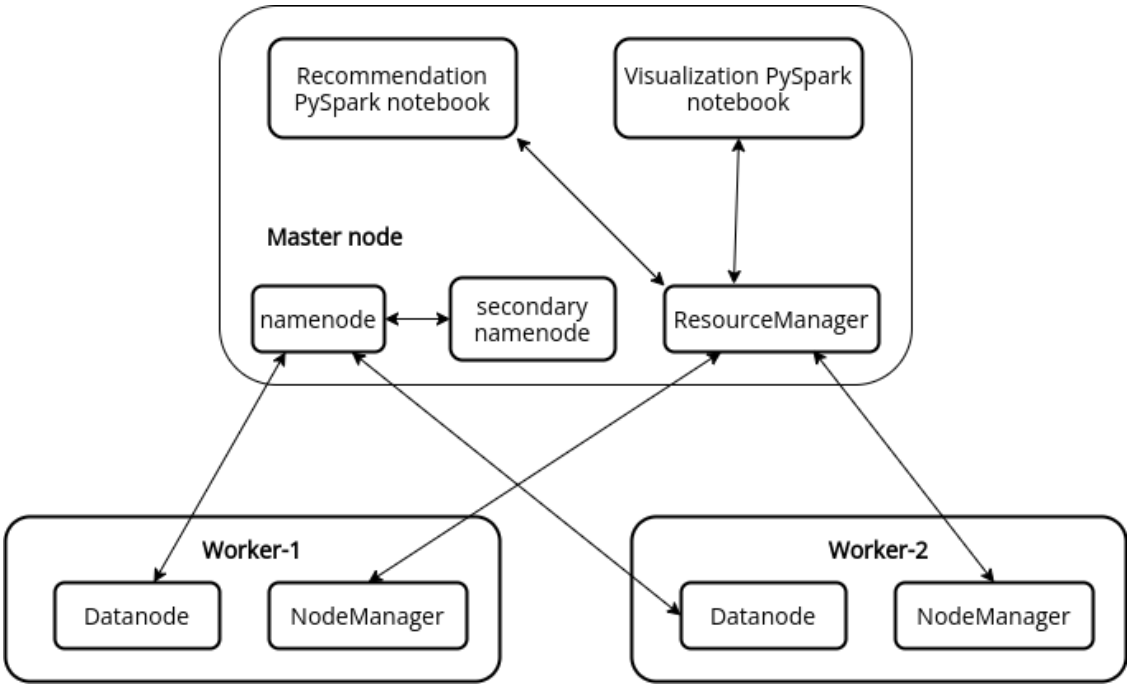
- Utilized PySpark to load movielens data and used SQL to query and analyze data

- Get more movie detail through ImdbPy

- Using PySpark Alternating least squares (ALS) algorithm s to construct a recommendation syste

[1]:
```
from IPython.display import Image
Image("system.png", height=400, width=600)
```

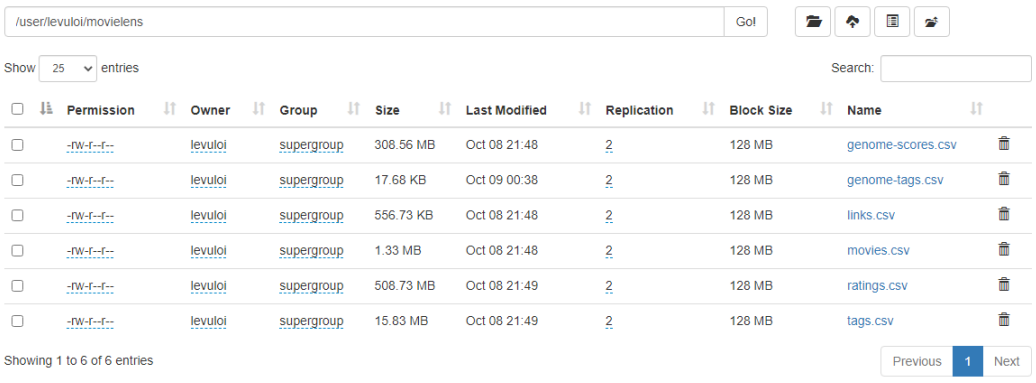
[1]:



Data in hadoop File system

[2]:
```
from IPython.display import Image
Image("hdfs.PNG")
```

[2]:



import numpy as np import pandas as pd import seaborn as sns import matplotlib.pyplot as plt import math %matplotlib inline

[3]:
```
import findspark
findspark.init()
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[*]").getOrCreate()
```

```
# Test the spark
df = spark.createDataFrame([{"hello": "world"} for x in range(1000)])
df.show(3, False)
```

```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform
(file:/home/levuloi/spark-3.1.2-bin-hadoop3.2/jars/spark-unsafe_2.12-3.1.2.jar)
to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of
org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal
reflective access operations
WARNING: All illegal access operations will be denied in a future release
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use
setLogLevel(newLevel).
[Stage 0:>                                                          (0 + 1) / 1]
```

```
+-----+
|hello|
+-----+
|world|
|world|
|world|
+-----+
only showing top 3 rows
```

[4]:
```
import os
os.environ["PYSPARK_PYTHON"] = "python3"
```

0.2 2. Read data and example

[5]:
```
from pyspark.sql import SparkSession
spark = SparkSession \
    .builder \
    .appName("Movie analysis") \
    .getOrCreate()
# if using google colab, using this
root = "movielens/"
# if using databrick
# databrick_root = "/FileStore/tables/movielen_small/
```

[6]:
```
movies_df = spark.read.csv(root+'/movies.csv',header=True,)
movies_df.printSchema()
movies_df.show(5)
```

```
root
 |-- movieId: string (nullable = true)
 |-- title: string (nullable = true)
 |-- genres: string (nullable = true)

+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Adventure|Animati…|
|      2|      Jumanji (1995)|Adventure|Childre…|
|      3|Grumpier Old Men …|      Comedy|Romance|
|      4|Waiting to Exhale…|Comedy|Drama|Romance|
|      5|Father of the Bri…|              Comedy|
+-------+--------------------+--------------------+
only showing top 5 rows
```

[7]:
```
ratings_df = spark.read.csv(root+'ratings.csv',header=True,)
ratings_df.printSchema()
ratings_df.show(5)
```

```
root
 |-- userId: string (nullable = true)
 |-- movieId: string (nullable = true)
 |-- rating: string (nullable = true)
 |-- timestamp: string (nullable = true)

+------+-------+------+----------+
|userId|movieId|rating| timestamp|
+------+-------+------+----------+
|     1|      2|   3.5|1112486027|
|     1|     29|   3.5|1112484676|
|     1|     32|   3.5|1112484819|
|     1|     47|   3.5|1112484727|
|     1|     50|   3.5|1112484580|
+------+-------+------+----------+
only showing top 5 rows
```

[8]:
```
links_df = spark.read.csv(root+'links.csv',header=True,)
links_df.printSchema()
links_df.show(5)
```

```
root
 |-- movieId: string (nullable = true)
 |-- imdbId: string (nullable = true)
 |-- tmdbId: string (nullable = true)
```

```
+-------+-------+------+
|movieId| imdbId|tmdbId|
+-------+-------+------+
|      1|0114709|   862|
|      2|0113497|  8844|
|      3|0113228| 15602|
|      4|0114885| 31357|
|      5|0113041| 11862|
+-------+-------+------+
only showing top 5 rows
```

[9]:
```
user_detail_df = spark.read.csv(root+'user-synthetic.csv',header=True,)
user_detail_df.printSchema()
user_detail_df.show(5)
```

```
root
 |-- userId: string (nullable = true)
 |-- age: string (nullable = true)
 |-- sex: string (nullable = true)
 |-- occupation: string (nullable = true)
 |-- zipcode: string (nullable = true)

+------+---+---+----------+-------+
|userId|age|sex|occupation|zipcode|
+------+---+---+----------+-------+
|     1| 24|  M|technician|  85711|
|     2| 53|  F|     other|  94043|
|     3| 23|  M|    writer|  32067|
|     4| 24|  M|technician|  43537|
|     5| 33|  F|     other|  15213|
+------+---+---+----------+-------+
only showing top 5 rows
```

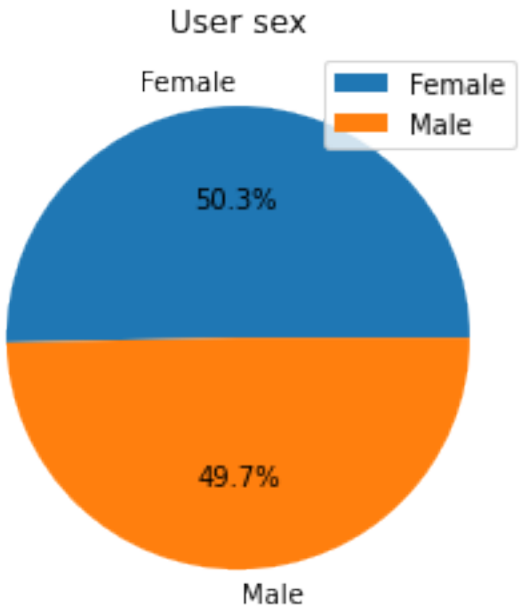
1 3. User detail

[10]:
```
df = user_detail_df.groupBy("sex").count()
df.show()
```

```
[Stage 18:=================================================>     (67 + 2) / 75]

+---+-----+
|sex|count|
+---+-----+
|  F|69040|
|  M|69453|
+---+-----+
```

[16]:
```
import matplotlib.pyplot as plt
import numpy as np

sex_value = [df.collect()[1][1], df.collect()[0][1]]
print(sex_value)
mylables = ["Female", "Male"]
plt.pie(np.array(sex_value), labels = mylables, autopct='%1.1f%%')
plt.title('User sex')
plt.legend()
plt.show()
```

```
[6672, 6584]
```

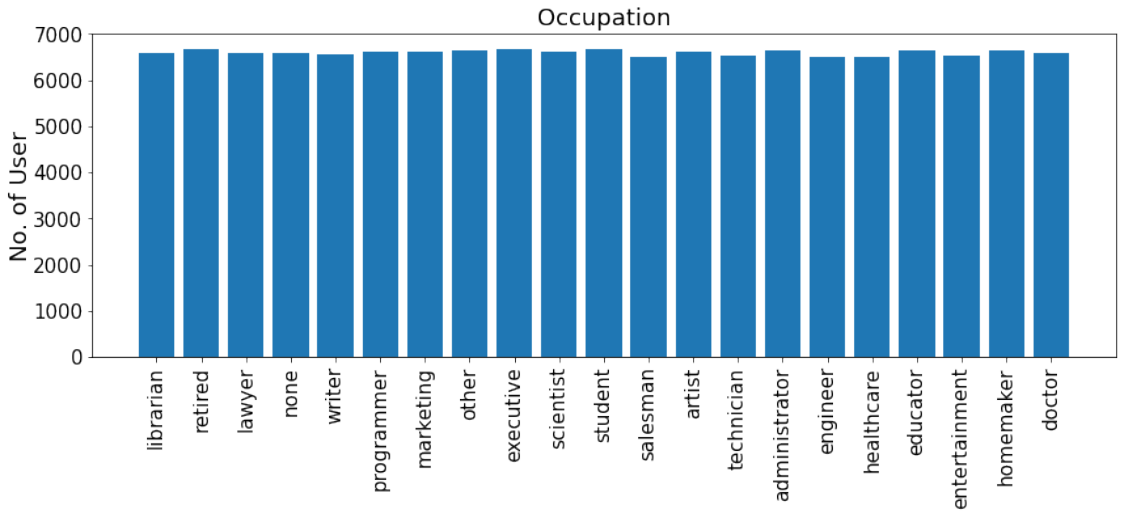


[17]:
```
df = user_detail_df.groupBy("occupation").count()
df.show(5)
```

```
+----------+-----+
|occupation|count|
+----------+-----+
| librarian| 6584|
|   retired| 6672|
|    lawyer| 6575|
```

```
|      none| 6576|
|    writer| 6563|
+----------+-----+
only showing top 5 rows
```

[21]:
```
numOccupation = df.count()

import matplotlib.pyplot as plt
fig = plt.figure(1, figsize=(14,10))
ax2 = fig.add_subplot(2,1,2)
y_axis = [df.collect()[i][1] for i in range(numOccupation)]
x_axis = [i for i in range(numOccupation)]
x_label = [df.collect()[i][0] for i in range(numOccupation)]
plt.xticks(rotation = 90, fontsize = 15)
plt.yticks(fontsize = 15)
plt.xticks(x_axis, x_label)
plt.ylabel("No. of User", fontsize = 18, labelpad = 0)
ax2.bar(x_axis, y_axis, align = 'center')
plt.title("Occupation",fontsize = 18)
plt.show()
```



2 4. Rating Distribution

[22]:
```
from pyspark.sql.functions import col,isnan,when,count
```

2.0.1 Check if have null data

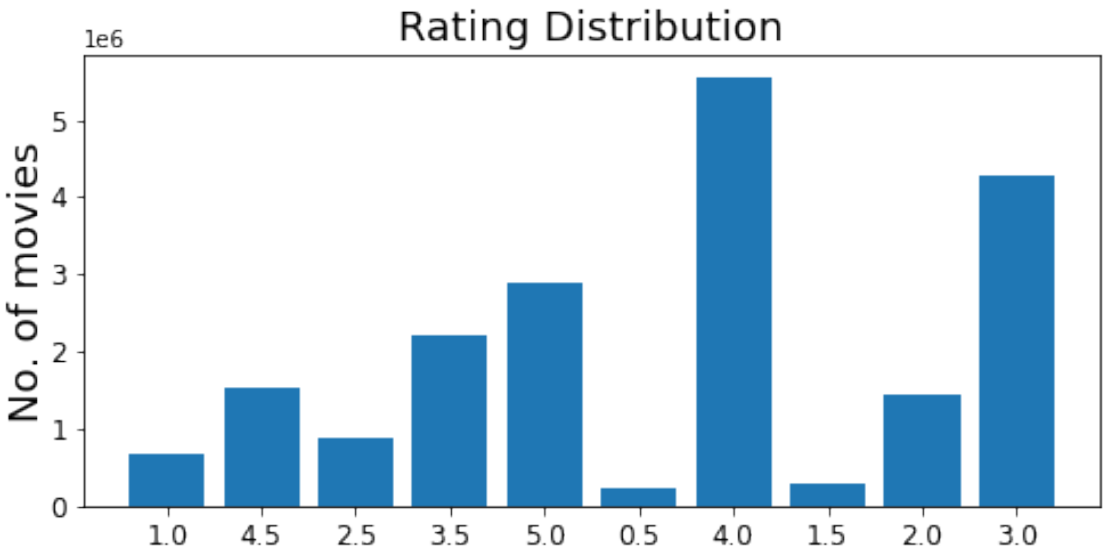
[23]:
```
col_df = ["userId", "movieId", "rating", "timestamp"]
ratings_df.select([count(when(isnan(c) | col(c).isNull(), c)).alias(c) for c in␣
↪col_df]).show()
```

```
[Stage 249:=============================================>             (3 + 1) / 4]

+------+-------+------+---------+
|userId|movieId|rating|timestamp|
+------+-------+------+---------+
|     0|      0|     0|        0|
+------+-------+------+---------+
```

[25]:
```
count_rate_df = ratings_df.groupBy("rating").count()
count_rate_df.show(5)
```

```
+------+-------+
|rating|  count|
+------+-------+
|   1.0| 680732|
|   4.5|1534824|
|   2.5| 883398|
|   3.5|2200156|
|   5.0|2898660|
+------+-------+
only showing top 5 rows
```

[26]:
```
numRate = count_rate_df.count()

fig = plt.figure(1, figsize=(8,8))
ax2 = fig.add_subplot(2,1,2)
y_axis = [count_rate_df.collect()[i][1] for i in range(numRate)]
x_axis = [i for i in range(numRate)]
x_label = [count_rate_df.collect()[i][0] for i in range(numRate)]
plt.xticks(fontsize = 12)
plt.yticks(fontsize = 12)
plt.xticks(x_axis, x_label)
plt.ylabel("No. of movies", fontsize = 18, labelpad = 0)
ax2.bar(x_axis, y_axis, align = 'center')
plt.title("Rating Distribution",fontsize = 18)
plt.show()
```



3 5.Top 10 most viewed movies

3.0.1 GroupBy using "movieId" count the number of users who watched a particular movie

Sorting in decreasing order Show movies with name and cover picture - Join movies and views - Join links and views

[13]:
```
views_df = ratings_df.groupBy("movieId").count()
movies_df.join(views_df, views_df.movieId == movies_df.movieId ,"full").
 ↪sort(col("count").desc()).show(10)
```

```
[Stage 62:=================================================>  (192 + 2) / 200]

+-------+--------------------+--------------------+-------+-----+
|movieId|               title|              genres|movieId|count|
+-------+--------------------+--------------------+-------+-----+
|    296| Pulp Fiction (1994)|Comedy|Crime|Dram…|    296|67310|
|    356| Forrest Gump (1994)|Comedy|Drama|Roma…|    356|66172|
|    318|Shawshank Redempt…|         Crime|Drama|    318|63366|
|    593|Silence of the La…|Crime|Horror|Thri…|    593|63299|
|    480|Jurassic Park (1993)|Action|Adventure|…|    480|59715|
|    260|Star Wars: Episod…|Action|Adventure|…|    260|54502|
|    110|   Braveheart (1995)|    Action|Drama|War|    110|53769|
|    589|Terminator 2: Jud…|       Action|Sci-Fi|    589|52244|
|   2571|  Matrix, The (1999)|Action|Sci-Fi|Thr…|   2571|51334|
|    527|Schindler's List …|           Drama|War|    527|50054|
```

```
+-------+--------------------+--------------------+-------+-----+
only showing top 10 rows
```

3.0.2 Get imdbId of top 5 movie

[14]:
```
views_links_df = links_df.join(views_df,links_df.movieId == views_df.
 ↪movieId,"full").sort(col("count").desc())
views_links = [ row.imdbId for row in views_links_df.head(5)]
print(views_links)
```

```
[Stage 65:=====================================================> (195 + 2) / 200]
['0110912', '0109830', '0111161', '0102926', '0107290']
```

3.0.3 Get cover imagine of movies

[22]:
```
import imdb
from IPython.display import  display, Image
from IPython.core.display import HTML
from io import BytesIO

ia = imdb.IMDb()
list_img = []
for code in views_links:
    # getting information
    series = ia.get_movie(code)
    cover = series.data['cover url']
    list_img.append(Image(url= cover))

display(list_img[0],list_img[1], list_img[2], list_img[3], list_img[4])
```

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

4 6. Number of movies are there in each genres

4.0.1 Count the Number of movies are there in each genres

[16]:
```
from pyspark.sql import functions as F
import pandas as pd

df_movies_genres = movies_df.select('movieId',F.explode(F.split(movies_df.
 ↪genres,'[|]')).alias('genre'))
df = df_movies_genres.groupBy("genre").count()
df.show(5)
```

```
+---------+-----+
|    genre|count|
+---------+-----+
|    Crime| 2939|
|  Romance| 4127|
| Thriller| 4178|
|Adventure| 2329|
|    Drama|13344|
+---------+-----+
only showing top 5 rows
```

df(type): pandas dataframe

4.0.2 Data Visualization

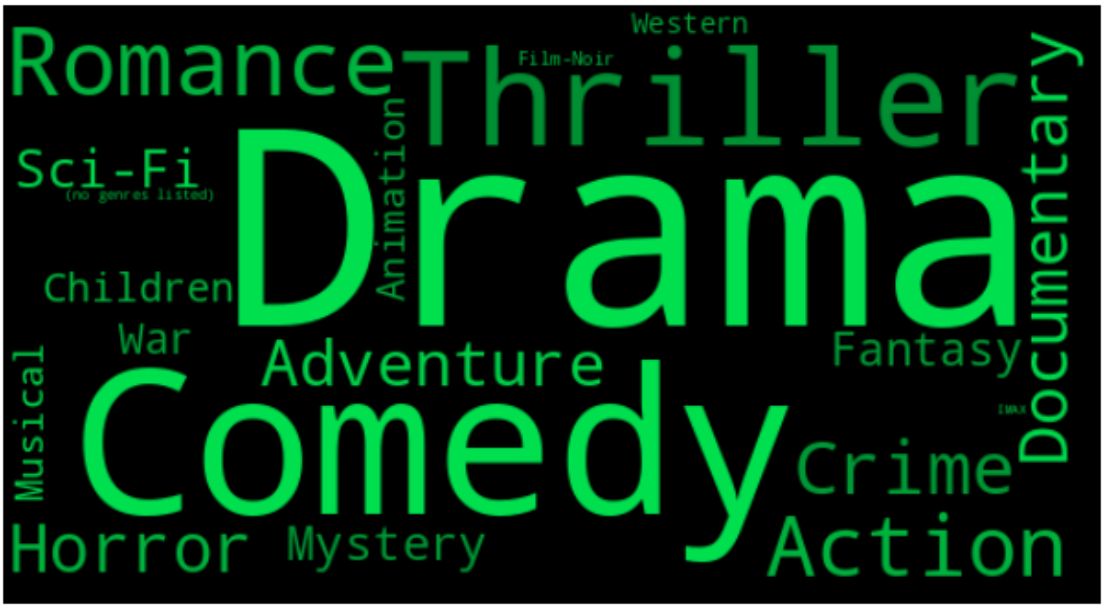
[20]:
```
import numpy as np #provides numerical arrays and functions to manipulate the␣
 ↪arrays efficiently
import random
import matplotlib.pyplot as plt # data visualization library
from wordcloud import WordCloud, STOPWORDS #used to generate world clou
```

[19]:
```
numGenre = df.count()
print(numGenre)
words = dict()
for i in range(numGenre):
    words[df.collect()[i][0]] = df.collect()[i][1]
print(words)
```

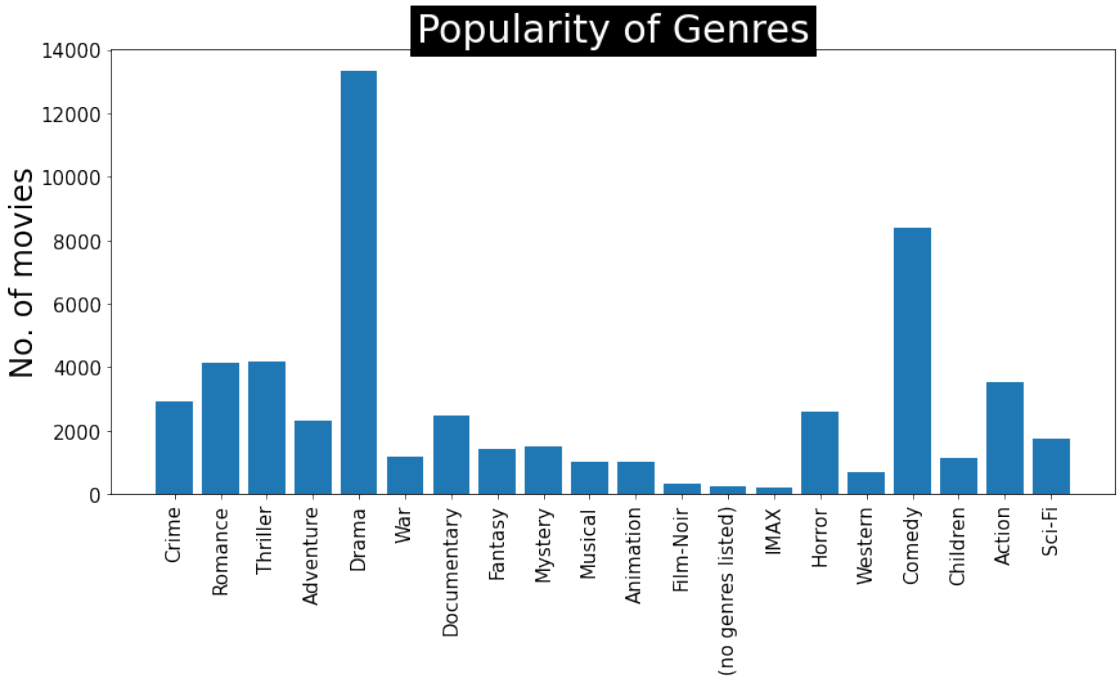
```
20
[Stage 156:=================================================>  (190 + 2) / 200]
{'Crime': 2939, 'Romance': 4127, 'Thriller': 4178, 'Adventure': 2329, 'Drama':
13344, 'War': 1194, 'Documentary': 2471, 'Fantasy': 1412, 'Mystery': 1514,
```

```
'Musical': 1036, 'Animation': 1027, 'Film-Noir': 330, '(no genres listed)': 246,
'IMAX': 196, 'Horror': 2611, 'Western': 676, 'Comedy': 8374, 'Children': 1139,
'Action': 3520, 'Sci-Fi': 1743}
```

[21]:
```
# Function that control the color of the words
def random_color_func(word=None, font_size=None, position=None,
                      orientation=None, font_path=None, random_state=None):
    h = int(360.0 * tone / 255.0)
    s = int(100.0 * 255.0 / 255.0)
    l = int(100.0 * float(random_state.randint(70, 120)) / 255.0)
    return "hsl({}, {}%, {}%)".format(h, s, l)


#Finally, the result is shown as a wordcloud:
tone = 100 # define the color of the words
f, ax = plt.subplots(figsize=(14, 6))
wordcloud = WordCloud(width=550,height=300, background_color='black',
                      max_words=1628,relative_scaling=0.7,
                      color_func = random_color_func,
                      normalize_plurals=False)
wordcloud.generate_from_frequencies(words)
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```



[23]:
```
# lets display the same result in the histogram
fig = plt.figure(1, figsize=(14,14))
ax2 = fig.add_subplot(2,1,2)
y_axis = [df.collect()[i][1] for i in range(numGenre)]
x_axis = [i for i in range(numGenre)]
x_label = [df.collect()[i][0] for i in range(numGenre)]
plt.xticks(rotation=90, fontsize = 15)
plt.yticks(fontsize = 15)
plt.xticks(x_axis, x_label)
plt.ylabel("No. of movies", fontsize = 24, labelpad = 0)
ax2.bar(x_axis, y_axis, align = 'center')
plt.title("Popularity of Genres",bbox={'facecolor':'k', 'pad':
 ↪5},color='w',fontsize = 30)
plt.show()
```



[ ]: