hw1_code_322479387_207039199_208197814.ipynb

May 18, 2025

0.1 Part A

0.1.1 1

[1]:
```
!pip install MRJob
```

```
Collecting MRJob
  Downloading mrjob-0.7.4-py2.py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.11/dist-
packages (from MRJob) (6.0.2)
Downloading mrjob-0.7.4-py2.py3-none-any.whl (439 kB)
                         0.0/439.6 kB
? eta -:--:--

143.4/439.6 kB 4.0 MB/s eta 0:00:01
                       439.6/439.6 kB
6.2 MB/s eta 0:00:00
Installing collected packages: MRJob
Successfully installed MRJob-0.7.4
```

[2]:
```
%%file hw1_mrA1_322479387_207039199_208197814.py
import csv
from mrjob.job import MRJob
from mrjob.step import MRStep

class MRApproved(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper, reducer=self.reducer)
        ]

    def mapper(self, _, line):
        # parse CSV and skip header
        try:
            row = next(csv.reader([line]))
        except Exception:
            return
        if row[0] == 'title':
```

```
            return

        # extract fields
        try:
            title     = row[0]
            genres    = [g.strip() for g in row[2].split(',') if g.strip()]
            air_date  = row[3]
            raw_time  = row[4]
            air_time  = int(raw_time.replace(':', ''))
        except Exception:
            return

        # apply letter-based criteria (3 & 4)
      #changing for lowercase so we will deal with case insensitive
        tl = title.lower()
        #criteria 4
        if any(c in tl for c in ('a', 'b')):
            return
        #criteria 3
        if sum(tl.count(c) for c in ('p', 'w', 'm')) < 2:
            return

        # apply time-window criterion (1)
        if air_time < 133000 or air_time >= 163000:
            return

        # passed criteria 1,3,4: emit date and genre tags
        approved_set = {'Reality', 'Community', 'Adventure', 'Animated'}
        yield title, ('date', air_date)
        for g in genres:
            yield title, ('all_genre', g)
            if g in approved_set:
                yield title, ('approved_genre', g)

    def reducer(self, title, values):
        all_genres      = set()
        approved_genres = set()
        dates           = set()

        for tag, val in values:
            if tag == 'all_genre':
                all_genres.add(val)
            elif tag == 'approved_genre':
                approved_genres.add(val)
            elif tag == 'date':
                dates.add(val)
```

```
        # enforce criteria 2: at least one approved genre AND at least one date
        #because of the condition we put in the mapper, we will get only the␣
↪dates that answ
        if not approved_genres or not dates:
            return

        # format approved genres into a quoted string
        approved_str = ", ".join(f"'{g}'" for g in sorted(approved_genres))
        # output key and counts
        yield [title, approved_str], [len(dates), len(all_genres)]

if __name__ == '__main__':
    MRApproved.run()
```

Writing hw1_mrA1_322479387_207039199_208197814.py

[3]:
```
!python hw1_mrA1_322479387_207039199_208197814.py 440k_data.csv -q
```

```
["600 Pound Mom", "'Reality'"]  [1, 3]
["Chopped", "'Reality'"]        [6, 2]
["Community Stories", "'Community'"]    [5, 1]
["Computerwise", "'Community'"] [2, 1]
["El Show de Tom y Jerry", "'Animated'"]        [1, 3]
["Empowered: Keys to Unlocking", "'Community'"] [3, 2]
["Extreme Couponing", "'Reality'"]      [4, 1]
["Fixer Upper", "'Reality'"]    [2, 3]
["Flip or Flop", "'Reality'"]   [3, 3]
["Flipping Out", "'Reality'"]   [3, 3]
["GC Perspectives", "'Community'"]      [1, 1]
["House Environment Committee", "'Community'"]  [1, 1]
["Jiggijump", "'Adventure'"]    [20, 3]
["KUOW's Week In Review Summer Tour", "'Community'"]    [1, 1]
["Life's Funniest Moments", "'Reality'"]        [1, 2]
["Littlest Pet Shop", "'Adventure', 'Animated'"]        [10, 4]
["Lord of the Rings: Fellowship of Ring", "'Adventure'"]        [1, 2]
["Love & Hip Hop", "'Reality'"] [2, 2]
["Love & Hip Hop: Hollywood", "'Reality'"]      [4, 1]
["McMorris & McMorris", "'Reality'"]    [1, 2]
["Mickey Mouse", "'Adventure', 'Animated'"]     [8, 4]
["Mission M:25", "'Community'"] [1, 2]
["Mission Menu", "'Reality'"]   [2, 2]
["Missouri Viewpoints", "'Community'"]  [1, 1]
["Mummers TV", "'Community'"]   [2, 1]
["My Time With Jesus", "'Animated'"]    [3, 2]
["New Jersey Now", "'Community'"]       [2, 2]
["New Mexico True TV", "'Community'"]   [1, 1]
["Osiyo: Voices of the Cherokee People", "'Community'"] [5, 1]
```

```
["PPIC Survey Series", "'Community'"]   [1, 1]
["Pok & Mok", "'Animated'"]     [1, 3]
["Pok\u00e9mon 4Ever", "'Adventure', 'Animated'"]       [1, 3]
["Pok\u00e9mon: XY", "'Animated'"]      [16, 4]
["Pompeii", "'Adventure'"]      [6, 3]
["Port Protection", "'Reality'"]        [1, 3]
["Pot Cops", "'Reality'"]       [1, 3]
["Pound Puppies", "'Animated'"] [8, 3]
["Pressing Into His Presence", "'Community'"]   [4, 2]
["Semper Ride", "'Community'"]  [1, 1]
["Shipwrecked", "'Adventure'"]  [1, 1]
["Snooki & JWOWW", "'Reality'"] [2, 1]
["Style Unzipped", "'Reality'"] [1, 2]
["Super Eruption", "'Adventure'"]       [1, 2]
["Super Why!", "'Animated'"]    [119, 3]
["Super Wings", "'Adventure', 'Animated'"]      [14, 4]
["Swim Week", "'Community'"]    [4, 3]
["T.U.F.F. Puppy", "'Adventure', 'Animated'"]   [4, 4]
["Teen Mom 2", "'Reality'"]     [2, 1]
["Teen Mom", "'Reality'"]       [4, 1]
["The Kings of Summer", "'Adventure'"]  [3, 2]
["The People's Couch", "'Reality'"]     [1, 2]
["The People's Court", "'Reality'"]     [53, 2]
["The Powerpuff Girls", "'Adventure', 'Animated'"]      [365, 4]
["The Sylvester & Tweety Mysteries", "'Animated'"]      [6, 3]
["The Wonder Pets!", "'Animated'"]      [16, 4]
["Top 20 Most Shocking", "'Reality'"]   [3, 2]
["Twin Tiers Weekly", "'Community'"]    [4, 1]
["Two More Eggs", "'Animated'"] [8, 3]
["UW View", "'Community'"]      [3, 2]
["Who Rocks New Mexico", "'Community'"] [1, 1]
["Who's on Top?", "'Reality'"]  [2, 1]
["Wild Spirits", "'Adventure'"] [1, 3]
["WordWorld", "'Animated'"]     [31, 3]
```

0.1.2 2

[4]:
```
%%file hw1_mrA2_322479387_207039199_208197814.py
import csv
from mrjob.job import MRJob
from mrjob.step import MRStep

class MRApproved(MRJob):

    def steps(self):
        return [
            MRStep(mapper=self.mapper, reducer=self.reducer),
```

```
            MRStep(mapper=self.score_mapper, reducer=self.best_score)
        ]

    def mapper(self, _, line):
        # parse CSV and skip header
        try:
            row = next(csv.reader([line]))
        except Exception:
            return
        if row[0] == 'title':
            return

        # extract fields
        try:
            title     = row[0]
            genres    = [g.strip() for g in row[2].split(',') if g.strip()]
            air_date  = row[3]
            raw_time  = row[4]
            air_time  = int(raw_time.replace(':', ''))
        except Exception:
            return

        # apply letter-based criteria (3 & 4)
      #changing for lowercase so we will deal with case insensitive
        tl = title.lower()
        #criteria 4
        if any(c in tl for c in ('a', 'b')):
            return
        #criteria 3
        if sum(tl.count(c) for c in ('p', 'w', 'm')) < 2:
            return

        # apply time-window criterion (1)
        if air_time < 133000 or air_time >= 163000:
            return

        # passed criteria 1,3,4: emit date and genre tags
        approved_set = {'Reality', 'Community', 'Adventure', 'Animated'}
        yield title, ('date', air_date)
        for g in genres:
            yield title, ('all_genre', g)
            if g in approved_set:
                yield title, ('approved_genre', g)

    def reducer(self, title, values):
        all_genres      = set()
        approved_genres = set()
```

```
        dates           = set()

        for tag, val in values:
            if tag == 'all_genre':
                all_genres.add(val)
            elif tag == 'approved_genre':
                approved_genres.add(val)
            elif tag == 'date':
                dates.add(val)

        # enforce criteria 2: at least one approved genre AND at least one date
        #because of the condition we put in the mapper, we will get only the␣
↪dates that answ
        if not approved_genres or not dates:
            return

        # format approved genres into a quoted string
        approved_str = ", ".join(f"'{g}'" for g in sorted(approved_genres))
        # output key and counts
        yield [title, approved_str], [len(dates), len(all_genres)]

    #I get from my first reducer a tuple of title and approved genres(in short␣
↪i wrote appgenre)
    #also i get counts of dates and genres as written in q1
    def score_mapper(self, title_appgerne ,counts):
      date_counts, genre_counts = counts
      total_score = date_counts + genre_counts
      #i don't want to get back the approved genre and obly the title itself
      yield None ,(title_appgerne[0], total_score)

    def best_score(self, _, title_scores):
      best_title, best_score = max(title_scores, key=lambda x: x[1])
      yield best_title, best_score




if __name__ == '__main__':
    MRApproved.run()
```

```
Writing hw1_mrA2_322479387_207039199_208197814.py
```

[5]:
```
!python hw1_mrA2_322479387_207039199_208197814.py 440k_data.csv -q
```

```
"The Powerpuff Girls"   369
```

0.2 Part B

0.2.1 Initialization

[7]:
```
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
!update-alternatives --set java /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java
!java -version
```

```
update-alternatives: using /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java to
provide /usr/bin/java (java) in manual mode
openjdk version "1.8.0_452"
OpenJDK Runtime Environment (build 1.8.0_452-8u452-ga~us1-0ubuntu1~22.04-b09)
OpenJDK 64-Bit Server VM (build 25.452-b09, mixed mode)
```

[8]:
```
!pip install --force-reinstall pyspark==3.4.0 -q
!pip install findspark -q
```

```
                              310.8/310.8
MB 4.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
                             200.5/200.5 kB
14.3 MB/s eta 0:00:00
  Building wheel for pyspark (setup.py) … done
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
dataproc-spark-connect 0.7.3 requires pyspark[connect]>=3.5, but you have
pyspark 3.4.0 which is incompatible.
```

Initializing a Spark Session

[9]:
```
import pyspark
from pyspark.sql import SparkSession
from pyspark.sql.types import *

spark = SparkSession.builder\
  .appName('PySpark Data Processing')\
  .config('spark.ui.port', '4050').getOrCreate()
sc = spark.sparkContext
```

0.2.2 Understanding and cleaning the data

read csv into dataframe, using structype for skipping the costly "infer Scfhema"

[10]:
```
from pyspark.sql.types import StructField, StructType, StringType, DoubleType,␣
↪IntegerType
from pyspark.sql.functions import*
schema_tut = StructType([StructField('title', StringType(), True),
                     StructField('prog_code', StringType(), True),
                     StructField('genre', StringType(), True),
                     StructField('air_date', StringType(), True),
                     StructField('air_time', StringType(), True),
                     StructField('Duration', DoubleType(), True)])

dataPath = './440k_data.csv'
shows = spark.read.format('csv')\
                .option('header','true')\
                .schema(schema_tut)\
                .load(dataPath)
```

Converting genre from string to list for simpler use

[11]:
```
shows = shows.withColumn('genre', split('genre',',\\s*'))
```

Adding days_of_week column (help to filter Thursdays)

[12]:
```
shows = shows.withColumn('date', to_date(col('air_date'), 'yyyyMMdd'))
shows_with_days = shows.withColumn('days_of_week', date_format('date', 'EEEE'))
```

Removing shows airing Thursdays 13:30 - 15:30

[13]:
```
# Adding end_time to check boundries
shows_with_days = shows_with_days \
        .withColumn("air_time_pad", lpad(col("air_time"),  6, "0")) \
        .withColumn('air_hour', col('air_time_pad').substr(1, 2).cast("int")) \
        .withColumn('air_min', col('air_time_pad').substr(3, 2).cast("int")) \
        .withColumn('Dur_hour', floor(col('Duration') / 60)) \
        .withColumn('Dur_min', floor(col('Duration') % 60)) \
        .withColumn('end_min_raw', col('air_min') + col('Dur_min')) \
        .withColumn('end_hour', col('air_hour') + col('Dur_hour')
                              + floor(col('end_min_raw') / 60)) \
        .withColumn('end_min', col('end_min_raw') % 60) \
        .withColumn('start_time', col('air_hour') * 100 + col('air_min')) \
        .withColumn('end_time', col('end_hour') * 100 + col('end_min') ) \
        .drop('air_hour', 'air_min', 'Dur_hour', 'Dur_min',
              'end_min_raw', 'end_hour', 'end_min')

# Line 1: Padding air_time to 6 characters with leading zeros
# Line 2+3: Extract hours and minutes from air_time then cast to int
# Line 4+5: Extract hours and minutes from Duration
# Line 6: Compute end time in minutes
# Line 7+8: Compute end time in hours and adding hour if end_min >= 60
```

```
# Line 9: Subtract 60 minutes (if we added hour)
# Line 10+11: Converting to the original format and keep necessary cols


# Filtering bad shows (airring in forbidden time)
bad_shows = shows_with_days.filter((
              (col('days_of_week') == 'Thursday') &
              (col('start_time') <= 1530) &
              (col('end_time') >= 1330)))

# Removing bad shows from the data
good_shows = shows_with_days.join(bad_shows, on='title', how='left_anti')
```

Removing shows with forbidden names in title with rlike() function where: * (?i) - makes it case-insensitive * _ ensures word boundary (i.e, don't match "newspaper" accidentally)

[14]:
```
# forbidden names in title
forbidden_names = "(?i).*\\b(friends|bang|breaking|montana|doctor|fox|news)\\b.
 ↪*"

# Filter forbidden names
good_shows = good_shows.filter(~col("title").rlike(forbidden_names))
```

###Computing the score

[15]:
```
shows_score = good_shows \
      .withColumn("one_genre", size(col("genre")) == 1) \
      .withColumn("dur_div_5", col("Duration") / 5) \
      .withColumn("Adv / Ani", array_contains("genre","Adventure") |
                  array_contains("genre", "Animated")) \
      .withColumn("girls_in_title", col("title").rlike("(?i).*\\b(girls)\\b.
 ↪*")) \
      .withColumn("score",
                  col("one_genre").cast("Double")*10
                  + col("dur_div_5").cast("Double")
                  + col("Adv / Ani").cast("Double")*90
                  + col("girls_in_title").cast("Double")*100 ) \
      .groupBy("title","genre").sum("score") \
      .withColumnRenamed("sum(score)", "total score") \
      .orderBy(col("total score").desc()) \
      .select(["title", "genre" ,"total score"])

# Line 1: bool for one genre
# Line 2: dividing the duration time
# Line 3-4: bool for having Adventure or Animated genre
# Line 5: bool for 'girls' in title
# Line 6-10: Compute score
```

```python
# Line 11-12: Sum up all the scores over all the viewing
# Line 13: Ordering by score (descenfing)
# Line 14: Keep necessary cols

Top_20_pairs = shows_score.limit(20)
Top_20_pairs.show(truncate=False)
```

```
+-----------------------+---------------------------------------------+--------
----------+
|title                  |genre                                        |total
score       |
+-----------------------+---------------------------------------------+--------
----------+
|The Simpsons           |[Sitcom, Animated]                           |74880.4
|
|2 Broke Girls          |[Sitcom]                                     |41762.6
|
|Up to the Minute       |[News]
|23767.200000000103|
|Futurama               |[Sitcom, Science fiction, Animated]          |22672.2
|
|Mike & Molly           |[Sitcom]
|19903.799999999996|
|The Fairly OddParents  |[Children, Comedy, Animated]                 |19434.0
|
|Globe Trekker          |[Travel, Adventure]                          |17848.0
|
|Modern Family          |[Sitcom]
|17040.800000000003|
|Peppa Pig              |[Children, Adventure, Entertainment,
Animated]|13875.000000000004|
|Archer                 |[Comedy, Animated]
|12690.999999999993|
|NCIS: Los Angeles      |[Crime drama, Action, Adventure, Mystery]
|12545.599999999999|
|Bob's Burgers          |[Sitcom, Animated]                           |11517.2
|
|Robot Chicken          |[Comedy, Animated]                           |11442.0
|
|Little Einsteins       |[Children, Educational, Art, Music, Animated] |9881.6
|
|Ruff-Ruff, Tweet & Dave|[Children, Educational, Game show, Animated]
|9645.000000000004 |
|JAG                    |[Crime drama, Action, Adventure, Law]        |8772.0
|
|Maya & Miguel          |[Children, Educational, Animated]            |8736.0
|
```

```
|Rugrats                |[Children, Fantasy, Animated]
|8718.599999999999 |
|Hey Arnold!            |[Children, Comedy, Animated]                  |8514.4
|
|Sanjay and Craig       |[Children, Adventure, Fantasy, Animated]      |8241.0
|
+-----------------------+----------------------------------------------+--------
----------+
```

[ ]:
```
!apt-get install texlive texlive-xetex texlive-latex-extra pandoc
!pip install pypandoc
```

[ ]:
```
!jupyter nbconvert --to PDF "/content/labX_ID.ipynb"

!jupyter nbconvert --to html "/content/labX_ID.ipynb"
```