In [1]:

```
import os
# cd H:\tecky-academy\c17-bad-project-01-tw\data_src
os.chdir(r"D:\tecky-academy\c17-bad-project-01-tw\data_src")
```

In [2]:

```
from typing import Reversible
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.metrics import mean_squared_error
```

In [3]:

```
# read review
df = pd.read_csv('meta_Movies_and_TV.csv')
df.head()
```

```
C:\Users\lau\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWar
ning: Columns (1,13) have mixed types.Specify dtype option on import or set low_memory=F
alse.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[3]:

| | category | tech1 | description | fit | title | also_buy | tech2 | brand | feature | ra |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | ['Movies & TV', 'Movies'] | NaN | [] | NaN | Understanding Seizures and Epilepsy | [] | NaN | NaN | [] | 886, Mov & T |
| **1** | ['Movies & TV', 'Movies'] | NaN | [] | NaN | Spirit Led&mdash;Moving By Grace In The Holy S... | [] | NaN | NaN | [] | 342, Mov & T |
| **2** | ['Movies & TV', 'Movies'] | NaN | ['Disc 1: Flour Power (Scones; Shortcakes; Sou... | NaN | My Fair Pastry (Good Eats Vol. 9) | [] | NaN | Alton Brown | [] | 370, Mov & T |
| **3** | ['Movies & TV', 'Movies'] | NaN | ['Barefoot Contessa Volume 2: On these three d... | NaN | Barefoot Contessa (with Ina Garten), Entertain... | ['B002I5GNW4', 'B005WXPVMM', 'B009UY3W8O', 'B0... | NaN | Ina Garten | [] | 342, Mov & T |
| **4** | ['Movies & TV', 'Movies'] | NaN | ['Rise and Swine (Good Eats Vol. 7) includes b... | NaN | Rise and Swine (Good Eats Vol. 7) | ['B000P1CKES', 'B000NR4CRM'] | NaN | Alton Brown | [] | 351, Mov & T |

In [4]:

```
# read meta
df_movies = pd.read_csv('Movies_and_TV_1.0.csv')
df_movies.head()
```

```
C:\Users\lau\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWar
```

```
ning: Columns (10) have mixed types.Specify dtype option on import or set low_memory=Fal
se.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[4]:

| | overall | verified | reviewTime | reviewerID | asin | style | reviewerName | reviewTe |
|---|---|---|---|---|---|---|---|---|
| **0** | 5.0 | True | 03 11, 2013 | A3478QRKQDOPQ2 | 0001527665 | {'Format:': ' VHS Tape'} | jacki | really happ they g evangelisec spoiler a |
| **1** | 5.0 | True | 02 18, 2013 | A2VHSG6TZHU1OB | 0001527665 | {'Format:': ' Amazon Video'} | Ken P | Having lived West Ne Guinea (Papu during |
| **2** | 5.0 | False | 01 17, 2013 | A23EJWOW1TLENE | 0001527665 | {'Format:': ' Amazon Video'} | Reina Berumen | Excellent loo in contextualizir the Gospe |
| **3** | 5.0 | True | 01 10, 2013 | A1KM9FNEJ8Q171 | 0001527665 | {'Format:': ' Amazon Video'} | N Coyle | More tha anything, I'v bee challenged f |
| **4** | 4.0 | True | 12 26, 2012 | A38LY2SSHVHRYB | 0001527665 | {'Format:': ' Amazon Video'} | Jodie Vesely | This is a gre movie for missiona going |

In [5]:

```
df_join = pd.merge(df_movies, df, how='inner', on='asin')
df_join.head()
```

Out[5]:

| | overall | verified | reviewTime | reviewerID | asin | style | reviewerName | reviewTe |
|---|---|---|---|---|---|---|---|---|
| **0** | 5.0 | True | 03 11, 2013 | A3478QRKQDOPQ2 | 0001527665 | {'Format:': ' VHS Tape'} | jacki | really happ they g evangelisec spoiler a |
| **1** | 5.0 | True | 02 18, 2013 | A2VHSG6TZHU1OB | 0001527665 | {'Format:': ' Amazon Video'} | Ken P | Having lived West Ne Guinea (Papu during |
| **2** | 5.0 | False | 01 17, 2013 | A23EJWOW1TLENE | 0001527665 | {'Format:': ' Amazon Video'} | Reina Berumen | Excellent loo in contextualizir the Gospe |
| **3** | 5.0 | True | 01 10, 2013 | A1KM9FNEJ8Q171 | 0001527665 | {'Format:': ' Amazon Video'} | N Coyle | More tha anything, I'v bee challenged f |

| | overall | verified | reviewTime | reviewerID | asin | style | reviewerName | reviewTe |
|---|---|---|---|---|---|---|---|---|
| **4** | 4.0 | True | 12 26, 2012 | A38LY2SSHVHRYB | 0001527665 | {'Format:': ' Amazon Video'} | Jodie Vesely | This is a gre movie for missiona going |

5 rows × 30 columns

In [6]:
```
n_dims = 10
```

In [7]:
```
def get_ratings_matrix(df, train_size=0.75):
    user_to_row = {}
    movie_to_column = {}
    df_values = df.values
    parameters = {}

    uniq_users = np.unique(df['reviewerID'])
    uniq_movies = np.unique(df['asin'])

    # mapping raw reviewerID and asin to new id in rating matrix
    for i, user_id in enumerate(uniq_users):
        user_to_row[user_id] = i

    for j, movie_id in enumerate(uniq_movies):
        movie_to_column[movie_id] = j

    n_users = len(uniq_users)
    n_movies = len(uniq_movies)

    R = np.zeros((n_users, n_movies))

    df_copy = df.copy()
    train_set = df_copy.sample(frac=train_size, random_state=0)
    test_set = df_copy.drop(train_set.index)

    for index, row in train_set.iterrows():
        i = user_to_row[row.reviewerID]
        j = movie_to_column[row.asin]
        R[i, j] = row.overall

    return R, train_set, test_set, n_users, n_movies, user_to_row, movie_to_column

R, train_set, test_set, n_users, n_movies, user_to_row, movie_to_column = get_ratings_m
print(R)
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

In [8]:
```
np.sum(R)
```

Out[8]:
```
280882.0
```

In [9]:
```
parameters = {}
```

In [10]:
```
def initialize_parameters(lambda_U, lambda_V):
    U = np.zeros((n_dims, n_users), dtype=np.float64)
    V = np.random.normal(0.0, 1.0 / lambda_V, (n_dims, n_movies))

    parameters['U'] = U
    parameters['V'] = V
    parameters['lambda_U'] = lambda_U
    parameters['lambda_V'] = lambda_V
```

In [11]:
```
def update_parameters():
    U = parameters['U']
    V = parameters['V']
    lambda_U = parameters['lambda_U']
    lambda_V = parameters['lambda_V']

    for i in range(n_users):
        V_j = V[:, R[i, :] > 0]
        U[:, i] = np.dot(np.linalg.inv(np.dot(V_j, V_j.T) + lambda_U * np.identity(n_di

    for j in range(n_movies):
        U_i = U[:, R[:, j] > 0]
        V[:, j] = np.dot(np.linalg.inv(np.dot(U_i, U_i.T) + lambda_V * np.identity(n_di

    parameters['U'] = U
    parameters['V'] = V
```

In [12]:
```
def log_a_posteriori():
    lambda_U = parameters['lambda_U']
    lambda_V = parameters['lambda_V']
    U = parameters['U']
    V = parameters['V']

    UV = np.dot(U.T, V)
    R_UV = (R[R > 0] - UV[R > 0])

    return -0.5 * (np.sum(np.dot(R_UV, R_UV.T)) + lambda_U * np.trace(np.dot(U, U.T)) +

def predict(user_id, movie_id):
    U = parameters['U']
    V = parameters['V']

    r_ij = U[:, user_to_row[user_id]].T.reshape(1, -1) @ V[:, movie_to_column[movie_id]

    max_rating = parameters['max_rating']
    min_rating = parameters['min_rating']

    return 0 if max_rating == min_rating else ((r_ij[0][0] - min_rating) / (max_rating
def evaluate(dataset):
    ground_truths = []
    predictions = []

    for index, row in dataset.iterrows():
        ground_truths.append(row.loc['overall'])
        predictions.append(predict(row.loc['reviewerID'], row.loc['asin']))

    return mean_squared_error(ground_truths, predictions, squared=False)

def update_max_min_ratings():
    U = parameters['U']
    V = parameters['V']

    R = U.T @ V
    min_rating = np.min(R)
    max_rating = np.max(R)

    parameters['min_rating'] = min_rating
    parameters['max_rating'] = max_rating

def train(n_epochs):
    initialize_parameters(0.3, 0.3)
    log_aps = []
    rmse_train = []
    rmse_test = []

    update_max_min_ratings()
    rmse_train.append(evaluate(train_set))
    rmse_test.append(evaluate(test_set))

    for k in range(n_epochs):
        update_parameters()
        log_ap = log_a_posteriori()
        log_aps.append(log_ap)

        if (k + 1) % 10 == 0:
            update_max_min_ratings()

            rmse_train.append(evaluate(train_set))
            rmse_test.append(evaluate(test_set))
            print('Log p a-posteriori at iteration', k + 1, ':', log_ap)

    update_max_min_ratings()

    return log_aps, rmse_train, rmse_test
```

In [13]:

```
log_ps, rmse_train, rmse_test = train(150)
```

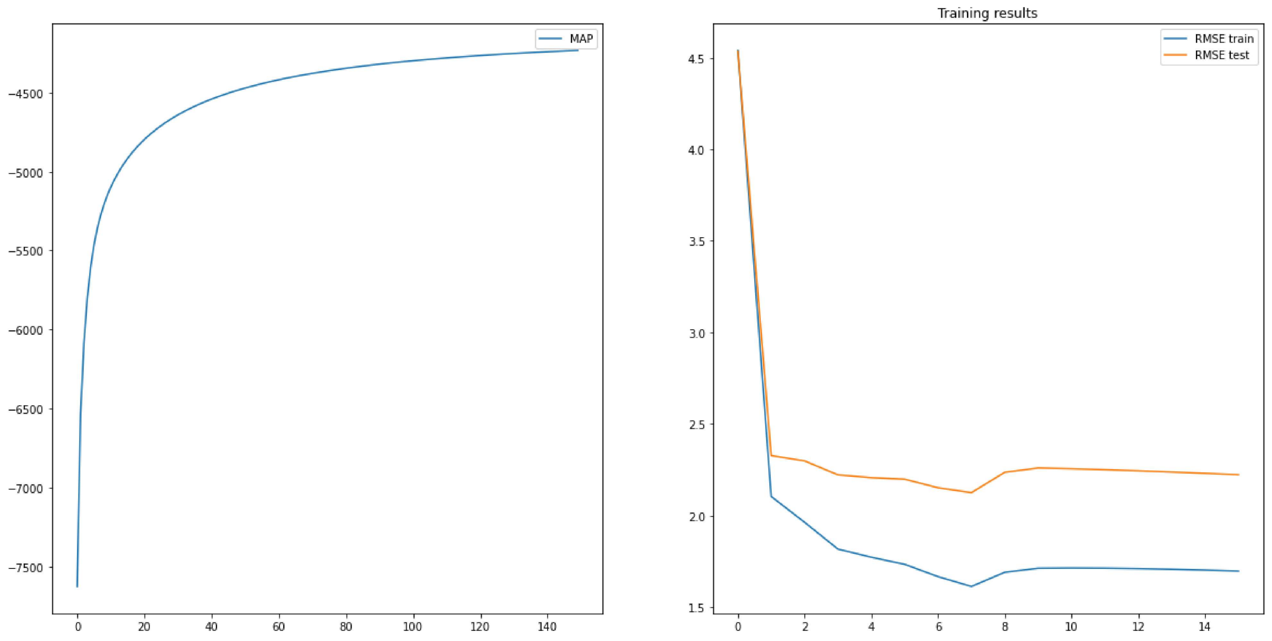
```
Log p a-posteriori at iteration 10 : -5148.93924747985
Log p a-posteriori at iteration 20 : -4818.483627502603
Log p a-posteriori at iteration 30 : -4654.176254294873
Log p a-posteriori at iteration 40 : -4550.599476900297
Log p a-posteriori at iteration 50 : -4478.267589809166
Log p a-posteriori at iteration 60 : -4424.590909359267
Log p a-posteriori at iteration 70 : -4383.104120064552
Log p a-posteriori at iteration 80 : -4350.194380496072
Log p a-posteriori at iteration 90 : -4323.579530303969
Log p a-posteriori at iteration 100 : -4301.689296834488
Log p a-posteriori at iteration 110 : -4283.4413594137095
Log p a-posteriori at iteration 120 : -4268.066868938015
Log p a-posteriori at iteration 130 : -4254.998983986519
Log p a-posteriori at iteration 140 : -4243.808438604406
Log p a-posteriori at iteration 150 : -4234.164060200418
```

In [16]:

```
_, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
plt.title('Training results')
ax1.plot(np.arange(len(log_ps)), log_ps, label='MAP')
ax1.legend()

ax2.plot(np.arange(len(rmse_train)), rmse_train, label='RMSE train')
ax2.plot(np.arange(len(rmse_test)), rmse_test, label='RMSE test')
ax2.legend()

plt.show()
```



In [17]:

```
print('RMSE of training set:', evaluate(train_set))
print('RMSE of testing set:', evaluate(test_set))
```

```
RMSE of training set: 1.6954606872483509
RMSE of testing set: 2.2217341386495733
```

In [18]:

```
user_id = "A3478QRKQDOPQ2"
df_join[df_join['reviewerID'] == user_id].sort_values(by=['overall'], ascending=False).
df_join[df_join['reviewerID'] == user_id].sort_values(by=['overall']).head(10)
```

Out[18]:

| | overall | verified | reviewTime | reviewerID | asin | style | reviewerName | reviewT |
|---|---|---|---|---|---|---|---|---|
| **54331** | 4.0 | True | 03 4, 2014 | A3478QRKQDOPQ2 | 0783225911 | {'Format:': ' VHS Tape'} | jacki | i think trie convert and fo |

| | overall | verified | reviewTime | reviewerID | asin | style | reviewerName | reviewT |
|---|---|---|---|---|---|---|---|---|
| **0** | 5.0 | True | 03 11, 2013 | A3478QRKQDOPQ2 | 0001527665 | {'Format:': ' VHS Tape'} | jacki | re happy t evangel .. spoile |

2 rows × 30 columns

In [19]:
```python
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [20]:
```python
# Look up most likely preferences
predictions = np.zeros((n_movies, 1))
movie_to_column_items = np.array(list(movie_to_column.items()))
df_result = pd.DataFrame(columns=['reviewerID','asin','title','prediction'])

for i, movie in enumerate(movie_to_column_items):
    predictions[i] = predict(user_id, movie[0])

indices = np.argsort(-predictions, axis=0)

for j in range(10):
    movie_id = movie_to_column_items[np.where(movie_to_column_items[:, 1] == str(indice
    df_row = pd.DataFrame({
        'reviewerID': user_id,
        'asin': movie_id,
        'title': df_join[df_join['asin'] == movie_id].iloc[0]['title'],
        'prediction': predictions[indices[j]][0][0]
    }, index=[j])
    df_result = df_result.append(df_row, sort=False)

df_result
```

Out[20]:

| | reviewerID | asin | title | prediction |
|---|---|---|---|---|
| **0** | A3478QRKQDOPQ2 | 0767819462 | Stepmom VHS | 3.896583 |
| **1** | A3478QRKQDOPQ2 | 0783245130 | Creature From the Black Lagoon VHS | 3.700583 |
| **2** | A3478QRKQDOPQ2 | 0783227272 | Amistad VHS | 3.675690 |
| **3** | A3478QRKQDOPQ2 | 0767802799 | Age of Innocence VHS | 3.652337 |
| **4** | A3478QRKQDOPQ2 | 0788821075 | Pretty Woman VHS | 3.599523 |
| **5** | A3478QRKQDOPQ2 | 0767821408 | Bottle Rocket | 3.592665 |
| **6** | A3478QRKQDOPQ2 | 0767817486 | Midnight Express | 3.577857 |
| **7** | A3478QRKQDOPQ2 | 0782010040 | Sands of Iwo Jima | 3.533862 |
| **8** | A3478QRKQDOPQ2 | 0767837398 | SLC Punk | 3.521178 |
| **9** | A3478QRKQDOPQ2 | 0767808673 | Spice World | 3.508803 |

In [21]:
```
# Look up least likely preferences
df_result = pd.DataFrame(columns=['reviewerID','asin','title','prediction'])
indices = np.argsort(predictions, axis=0)

for j in range(10):
    movie_id = movie_to_column_items[np.where(movie_to_column_items[:, 1] == str(indice
    df_row = pd.DataFrame({
        'reviewerID': user_id,
        'asin': movie_id,
        'title': df_join[df_join['asin'] == movie_id].iloc[0]['title'],
        'prediction': predictions[indices[j]][0][0]
    }, index=[j])
    df_result = df_result.append(df_row, sort=False)

df_result
```

Out[21]:

| | reviewerID | asin | title | prediction |
|---|---|---|---|---|
| 0 | A3478QRKQDOPQ2 | 0783112750 | When Trumpets Fade VHS | 1.571157 |
| 1 | A3478QRKQDOPQ2 | 0784017808 | Denise Austin - Hit the Spot:Sizzler VHS | 1.981228 |
| 2 | A3478QRKQDOPQ2 | 0005019281 | An American Christmas Carol VHS | 1.989331 |
| 3 | A3478QRKQDOPQ2 | 0578047861 | The ADVENTISTS | 2.021174 |
| 4 | A3478QRKQDOPQ2 | 076780192X | Close Encounters of the Third Kind VHS | 2.023957 |
| 5 | A3478QRKQDOPQ2 | 0738920061 | Sesame Street - Let's Eat VHS | 2.037258 |
| 6 | A3478QRKQDOPQ2 | 000503860X | Chapter X Live [VHS] | 2.069641 |
| 7 | A3478QRKQDOPQ2 | 0767827759 | The Grudge | 2.083677 |
| 8 | A3478QRKQDOPQ2 | 0767020731 | Monty Python's Flying Circus - Season 2 VHS | 2.098475 |
| 9 | A3478QRKQDOPQ2 | 0005419263 | Steve Green: Hide 'em in Your Heart Volume 2: ... | 2.107519 |