Movie Recommendations in BigQuery ML 2.5

## Overview

BigQuery is Google's fully managed, NoOps, low cost analytics database. With BigQuery you can query terabytes and terabytes of data without having any infrastructure to manage or needing a database administrator. BigQuery uses SQL and can take advantage of the pay-as-you-go model. BigQuery allows you to focus on analyzing data to find meaningful insights.

[BigQuery Machine Learning](https://cloud.google.com/bigquery/docs/bigqueryml-analyst-start) (BigQuery ML) is a feature in BigQuery where data analysts can create, train, evaluate, and predict with machine learning models with minimal coding.

Collaborative filtering provides a way to generate product recommendations for users, or user targeting for products. The starting point is a table with three columns: a user id, an item id, and the rating that the user gave the product. This table can be sparse -- users don’t have to rate all products. Then, based on just the ratings, the technique finds similar users and similar products and determines the rating that a user would give an unseen product. Then, you can recommend the products with the highest predicted ratings to users, or target products at users with the highest predicted ratings.

To illustrate recommender systems in action, you will use the MovieLens dataset. This is a dataset of movie reviews released by [GroupLens](https://grouplens.org/about/what-is-grouplens/" \t "_blank), a research lab in the Department of Computer Science and Engineering at the University of Minnesota, through funding by the US National Science Foundation.

### **Objectives**

In this lab, you learn to perform the following tasks:

* Create a BigQuery dataset to store and load MovieLens data
* Explore the MovieLens dataset
* Use a trained model to make recommendations in BigQuery
* Make product predictions for both single users and batch users

## Task 1: Get MovieLens data

In this task you will use the command line to create a BigQuery dataset to store the MovieLens data. The MovieLens data will then be loaded from a Cloud Storage bucket into the dataset.

### **Start the Cloud Shell Editor**

To create a BigQuery dataset and load the MovieLens data the Cloud Shell is used.

1. In the GCP Console, click **Activate Cloud Shell** (Cloud Shell).
2. If prompted, click **Continue**.

### **Create and Load BigQuery Dataset**

1. Run the following command to create a BigQuery dataset named movies:

bq --location=EU mk --dataset movies

Copied!

content\_copy

1. Run the following commands separately in the Cloud Shell:

bq load --source\_format=CSV \

--location=EU \

--autodetect movies.movielens\_ratings \

gs://dataeng-movielens/ratings.csv

Copied!

content\_copy

bq load --source\_format=CSV \

--location=EU \

--autodetect movies.movielens\_movies\_raw \

gs://dataeng-movielens/movies.csv

Copied!

content\_copy

Click Check my progress to verify the objective.

Get MovieLens Data

Check my progress

## Task 2: Explore the data

In this task you will explore and verify the MovieLens dataset using Query editor.

1. In BigQuery's **Query editor** execute the following query:

SELECT

COUNT(DISTINCT userId) numUsers,

COUNT(DISTINCT movieId) numMovies,

COUNT(\*) totalRatings

FROM

movies.movielens\_ratings

Copied!

content\_copy

You should confirm that the dataset consists of over 138 thousand users, nearly 27 thousand movies, and a little more than 20 million ratings.

1. Examine the first few movies using the query:

SELECT

\*

FROM

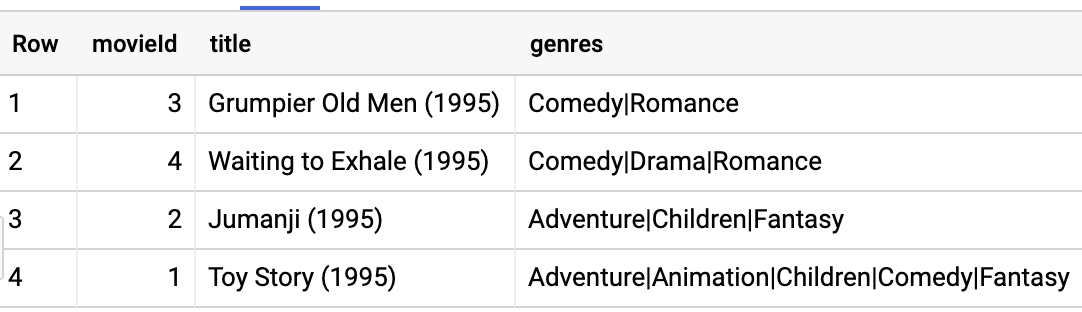
movies.movielens\_movies\_raw

WHERE

movieId < 5

Copied!

content\_copy



1. You can see that the genres column is a formatted string. Parse the genres into an array and rewrite the results into a table named movielens\_movies.

CREATE OR REPLACE TABLE

movies.movielens\_movies AS

SELECT

\* REPLACE(SPLIT(genres, "|") AS genres)

FROM

movies.movielens\_movies\_raw

Copied!

content\_copy

Feel free to perform additional queries until you are comfortable with the dataset.

Click Check my progress to verify the objective.

Explore the Data

Check my progress

## Task 3: Evaluate a trained model created using collaborative filtering

In this task you will view the metrics for a trained model which was generated using matrix factorization.

Matrix factorization is a collaborative filtering technique that relies on two vectors called the user factors and the item factors. The user factors is a low-dimensional representation of a user\_id and the item factors similarly represents an item\_id.

To perform a matrix factorization of our data, you use the typical BigQuery ML syntax except that the model\_type is matrix\_factorization and you have to identify which columns play what roles in the collaborative filtering setup.

In order to apply matrix factorization to the movie ratings data, the BigQuery ML query needs to be executed to create the model. However, creation of this model type can take up to 40 minutes and requires a Google Cloud project with reservation-oriented resources -- which is unlike those offered by the Qwiklabs environment.

A model has been created in the Cloud Training project's **cloud-training-prod-bucket** BigQuery dataset for use in the rest of the lab.

**NOTE:** The query below is for reference only. Please **DO NOT EXECUTE** this query in your project.

CREATE OR REPLACE MODEL movies.movie\_recommender

OPTIONS (model\_type='matrix\_factorization', user\_col='userId', item\_col='movieId', rating\_col='rating', l2\_reg=0.2, num\_factors=16) AS

SELECT userId, movieId, rating

FROM movies.movielens\_ratings

Note, the num\_factors and l2\_reg options have been selected after much experimentation to speed up training of the model.

1. To view metrics for the trained model, run the following query:

SELECT \* FROM ML.EVALUATE(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`)

Copied!

content\_copy

## Task 4: Make Recommendations

In this task you will use the trained model to provide recommendations.

1. Let’s find the best comedy movies to recommend to the user whose userId is 903. Enter the query below:

SELECT

\*

FROM

ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`,

(

SELECT

movieId,

title,

903 AS userId

FROM

`movies.movielens\_movies`,

UNNEST(genres) g

WHERE

g = 'Comedy' ))

ORDER BY

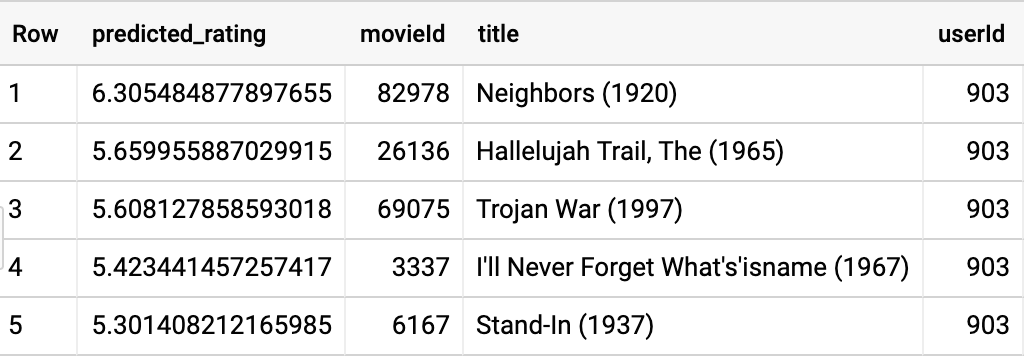
predicted\_rating DESC

LIMIT

5

Copied!

content\_copy



1. This result includes movies the user has already seen and rated in the past. Let’s remove them:

SELECT

\*

FROM

ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`,

(

WITH

seen AS (

SELECT

ARRAY\_AGG(movieId) AS movies

FROM

movies.movielens\_ratings

WHERE

userId = 903 )

SELECT

movieId,

title,

903 AS userId

FROM

movies.movielens\_movies,

UNNEST(genres) g,

seen

WHERE

g = 'Comedy'

AND movieId NOT IN UNNEST(seen.movies) ))

ORDER BY

predicted\_rating DESC

LIMIT

5

Copied!

content\_copy

For this user, this happens to yield the same set of movies -- the top predicted ratings didn’t include any of the movies the user has already seen.

Click Check my progress to verify the objective.

Making Recommendations

Check my progress

## Task 5:Apply customer targeting

In this task you will look at how to identify the top-rated movies for a specific user. Sometimes, you have a product and have to find the customers who are likely to appreciate it.

1. You wish to get more reviews for movieId=96481 which has only one rating and you wish to send coupons to the 100 users who are likely to rate it the highest. Identify those users using:

SELECT

\*

FROM

ML.PREDICT(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`,

(

WITH

allUsers AS (

SELECT

DISTINCT userId

FROM

movies.movielens\_ratings )

SELECT

96481 AS movieId,

(

SELECT

title

FROM

movies.movielens\_movies

WHERE

movieId=96481) title,

userId

FROM

allUsers ))

ORDER BY

predicted\_rating DESC

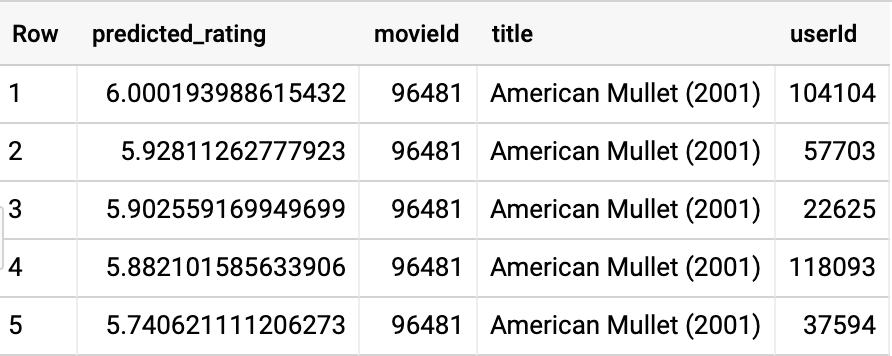
LIMIT

100

Copied!

content\_copy

The result gives us 100 users to target, the top 5 of whom are:



Click Check my progress to verify the objective.

Customer Targeting

Check my progress

## Task 6: Perform Batch predictions for all users and movies

In this task you will perform a query to obtain batch predictions for users and movies.

What if you wish to carry out predictions for every user and movie combination? Instead of having to pull distinct users and movies as in the previous query, a convenience function is provided to carry out batch predictions for all movieId and userId encountered during training.

1. Enter the following query to obtain batch predictions:

SELECT

\*

FROM

ML.RECOMMEND(MODEL `cloud-training-prod-bucket.movies.movie\_recommender`)

LIMIT

100000

Copied!

content\_copy

Without the LIMIT command the results would be too large to return given the default settings. But the output provides you a sense of the type of predictions that can be made with this model.

As seen in a section above, it is possible to filter out movies the user has already seen and rated in the past. The reason already seen movies aren’t filtered out by default is that there are situations (think of restaurant recommendations, for example) where it is perfectly expected that you would need to recommend restaurants the user has liked in the past.

## End your lab