Project Overview

Introduction

Recommender Systems are used in a variety of ways by many different companies across all industries. YouTube, Amazon, Netflix, Spotify, among many other companies use recommender systems to suggest movies, products, videos, and music to customers. Think about all the 'For You' suggestions on practically every website that you visit. That content is most likely generated by a recommender system.

In 2006, Netflix hosted a competition to create a recommender system that could beat their current system. Contestants were given about 100 million ratings to use as a training set to try and beat the RMSE of about 0.95 by 10% in order to claim a 1 million dollar prize. After about 3 years and several awards handed out for significant improvements along the way, a final winner was announced. You can read more about the Netflix Competition here and here.

This project is modeled after the Netflix Competition and asks you to create a recommender system on the MovieLens data set to predict movie ratings. The data is made available and maintained by GroupLens Research from the University of Michigan. There are a couple of different data splits, but to keep things computationally feasible, you will use the 100,000 ratings from 943 users on 1682 movies located here.

Data

You can read more about the data on the provided README, but what is important to know for this project is that the data is split up into several files containing the user and movie data that you will be using. The ratings can be downloaded as a single file containing all the ratings, but Grouplens kindly created several splits in the data labeled u1.base - u5.base and correspondingly named test sets u1.test - u5.test. **The ratings data that you will be working with should come from u1.base. Use u1.base to create your own train/test/validation split. The file u1.test should be used as a final test. Don't even bother opening it until you have completed all your models. Until then, pretend like it doesn't exist.** The other files in this folder such as u.item and u.user, containing the more detailed information on each movie and user, can be used in their entirety.

Movielens Data

You do not need to submit the data from Movielens. The following is a sample piece of code to load u1.base into a data frame:

```
import pandas as pd
```

```
df = pd.read_csv("http://files.grouplens.org/datasets/movielens/ml-100k/u1.base",sep='\t',header=None, names=["userId", "movieId", "rating", "timestamp"])
```

Other Data

There are a number of ways that you can incorporate other bits of data into this project for your hybrid models (see below). One interesting idea is to incorporate the descriptions of each movie into the Recommender. This data can be found on these kaggle dataset

```
!kaggle datasets download rounakbanik/the-movies-dataset --unzip
```

```
!kaggle datasets download stefanoleone992/imdb-extensive-dataset --unzip
```

Unfortunately, the movie id's in the MovieLens dataset don't match up to the imdb movie ids in the second dataset. In another bit of misfortune, trying to do a join on the movie titles and even the combined title-year key (which explains some duplicates, but not all) still leaves you with about 200 titles (of 1681 - there is a movie titled 'unknown' which you might want to throw out as it doesn't have any associated information on the movie) that you need to search for manually. This might seem annoying for you to do, **therefore you may collaborate for only this particular aspect of the project, i.e. to split up the missing titles and find their corresponding imdbIDs as well as other preprocessing associated with this task.** Otherwise, you may discuss ideas but not share code. Let me repeat: DO NOT SHARE CODE. Do not copy code from the web. This project is your work.

Here is a link to a notebook that can help get you started. There are some issues with it, the main one being that it does not account for duplicate movies in the Movielens dataset. I would suggest you filter those first before attempting to perform any joins as this will likely reduce the overall number of imdb ids you will need to search for manually. The main reason why a number of these got missed in the join is due to the way the title is parsed out of the 'items' dataframe. The movie title can differ in the other dataset in addition to some (I'm thinking of one in particular, but there are probably more) titles that are transliterated in one dataset and translated in another. If you can find a way of joining the datasets more efficiently, ***please*** share on Slack.

Additionally, if you do find another dataset that you would like to incorporate into the project, that would be great, but please get approval to use the dataset first.

Recommender Systems

Introduction

There are several ways to go about building a Recommender System. The two most common approaches are Content Based Filtering and Collaborative Filtering. Here are some links that you should read to help you get started.

Google Crash Course to Recommender Systems
Introduction to Recommender Systems

ML for Recommender Systems **-** part 1
ML for Recommender Systems - part 2 (Read if you have time.)

Recommender Systems in Practice

Getting Started

A good idea for any Machine Learning project is to define a baseline model before working on using some of the ML tools in your arsenal to make predictions. This is done to make sure that you are building models that can do better than just randomly guessing. For the recommender system several ideas for a baseline model might be to use the mean/median of ratings. This statistic could be based on all the ratings across all movies and users, on a per movie basis, or on a per user basis. Either way, these are just baseline models, so don't spend that much time on them. These are simply different ways of going about it.

The metric being used to test all models is RMSE.

Building Models and Making Predictions

Simple Model

For your simple model, use one of the collaborative filtering methods to make predictions. Here is a helpful link I found
Explicit Matrix Factorization: ALS, SGD, and All That Jazz

Here are some more links to help you get started:

Guide to Recommender Engines

Hybrid Models

For your hybrid model, consider using an autoencoder to create embeddings using the description data on each movie, or take the learned embeddings from Matrix Factorization and feed it through DNN. There are many ways to do things, and depending on how you implemented the first part of the project this may or may not work for you, but here are some links for hybrid Recommenders.

Deep Autoencoders for Collaborative Filtering

Hybrid Recommender Using Deep Learning

Or you could try a weighted hybrid
Weighted Hybrid Recommender

Other Links:
Hybrid Recommenders

Note: If you have a printout in your notebook, *please* keep it to a reasonable length (i.e. don't print out 500 epochs worth of metrics.)

Submission

The purpose of this project is to give you experience researching a topic in Machine Learning and applying it to a Real World™ situation. As part of the project submission, include a report that describes the different algorithms used for Recommender Systems as well as their advantages and disadvantages. Make sure to include a discussion on topics relating to Recommender Systems (some ideas, not an exhaustive list. similarity measures, time, space, sparsity, cold start, interpretability) for the algorithms as appropriate in addition to any other issues that may be present in live Recommender Systems.
Based on your research, provide an explanation for why you chose a particular algorithm/method for your Recommender System.

In your Github create a RecommenderSystem folder for your submission. This is where all project files should be submitted to. Alternatively, if you've done everything in Colab, you can submit the Colab by sharing with edit permissions. Your write-up should be in Google Docs.

Recommender

Have a notebook and python file in your folder named *Recommender.ipynb/.py*. Your submission should, at least, contain sections on:

1. Data Exploration and Visualization
    a. Are there missing values, outliers, what do the distributions of ratings look like…..

2. Baseline Model
3. Simple Model
4. Hybrid Model
5. Evaluation
    a. For each Model, excluding the Baseline, make sure to include the metrics on training and testing as well as a final evaluation on u1.test.

Report

Create a Google Doc for your write-up. The first section should contain links to relevant code (e.g., colab notebook). Then your report should be next. Format the report so that the different sections are clearly delineated. It should include:

1. A section analyzing the different methods of building a Recommender System as detailed above.
2. A section, primarily based on your research, on why you chose the particular algorithm for your basic and hybrid Recommender
3. A section that details your progress on the project (in the event that things stall out, we can at least have a record of your effort, making the awarding of partial credit easier) as well as a final analysis on your Recommenders performance.

Deadline

The project is due on May 20th, 2024 at the end of the day (by midnight). There **will not** be any extensions provided under any circumstances: you have over a month to work on this project, so allocate your time wisely - don't wait until the last minute! Whatever is in your Github repository at that time will be considered your final submission for the project.

A half-letter grade will be subtracted from the overall final grade of the project for each day the submission is late. Additionally, a bonus will not be awarded for late submissions.

Grading

The grading for this project will be done on a scale of 100 points (with a potential 10 point bonus) where the break down will be as follows:

- Report - 20%
    - Recommender Systems overview - 55%
    - Reasoning for choice of algorithm - 25%
    - Project journaling and analysis on implementation - 20%
- Recommender System - 80%
    - Simple Recommender - 65% (graded based on correctness of implementation)
    - Hybrid Recommender - 20% (see above)
    - Data Exploration - 5%

    - Clarity of code, organization, and commenting - 10%
- Bonus - 10 pts possible
    - A bonus of up to 10 pts. (equivalent to a full letter grade on the project) will be assessed and granted based on exceptional work in one or more of the following categories:
        - Recommender Systems overview in report
        - Low RMSE
        - Well thought out Hybrid Architecture Model (i.e. incorporating contextual data from other datasets, good feature engineering, incorporating other ML models in an ensemble or bagging)

This is not a weekend project.

Begin working on it as soon as possible.

Post all questions on the Google Chat.

Good Luck!