CSE427S

FINAL PROJECT #1: COLLABORATIVE FILTERING USING THE NETFLIX DATA

M. Neumann

Due: FRI 13 DEC 2019 (6PM) – **NO EXTENSION**

Project Goal

In this project your group will predict 100,000 movie ratings for users in a subset of the original NETFLIX data issued for the NETFLIX Prize. This challenge aimed at substantially improving the accuracy of predictions about how much someone is going to enjoy a movie based on their movie preferences. It was issued by the Netflix company and on September 21, 2009 a $1mio Grand Prize was awarded to the winning team.[1]

Goal: Analyze the NETFLIX data using PIG (or SPARK) and, based on the outcomes of this analysis, develop a feasible and efficient implementation of the collaborative filtering algorithm in **MAPREDUCE** or **SPARK**. **Execute your program on Amazon EMR to get the rating predictions and evaluate those ratings by comparing them to the provided true ratings (*gold standard*).** Part of the grade for the results (10%) will be assigned according to a ranking of the number and quality of your team's predictions compared to the other teams' predictions!

Getting Started

Update your SVN repository, you will find additional materials for the *final project* in the folder `final_project/netflix`. You will have to agree to the following usage conditions to be able to do this project:

[1] You can read more about it here: `http://www.netflixprize.com/`.

Usage Agreement

By using the dataset from the Netflix Prize for you agree as follows:

- I agree to the terms specified in Netflix Prize Rules (cf. README file provided in your SVN repo).
- I agree to **delete** this dataset once the project has been completed.
- I will not redistribute this data in any form.

If you agree to the usage agreement above, download the data from:

```
https://classes.cec.wustl.edu/cse427/netflix_subset.zip
```

Indicating Group Work and Submission Repository

Groups cannot be larger than **four** students and must be registered at the **milestone 1** sign-up with the instructor or TA. The wustlkey of the SVN repository used for submission must also be registered at this time. It is your responsibility to add and commit all your final project submissions to exactly **this repository**.

Problem 1: Collaborative Filtering Approach

This problem is part of **milestone 2** (*problem understanding*) and **milestone 3** (*model parameters, model choices, and evaluation*) .

Revise the collaborative filtering approach discussed in the lecture. Make sure you **understand the basic approach** and get a **firm grasp of its variable components**. In your approach you are free to take any combination of the following options/ values for the following parameters:

- similarity measure (JACCARD, COSINE, CORRELATION, other?)
- number of similar users k
- prediction method (weighted or un-weighted average)
- you may threshold the ratings
- you may normalize the ratings or not (cf. Alg 1 vs. Alg 2 discussed in the lecture)
- user-user model or item-item model

Read up on the descriptions of collaborative filtering in the literature (MMDS Chapter 9 and maybe this blog is helpful `http://blog.echen.me/2011/10/24/winning-the-netflix-prize-a-summary/`; feel free to search for more sources online).
To design an efficient and powerful implementation you will need to analyze the given data first.

Problem 2: Analyzing the Netflix Data

This problem defines **milestone 2** (*data preparation*).

In this problem you will **analyze the input data to plan out an efficient implementation** of your approach. The format of the dataset is described in the `description.txt` file. For (a) and (b) you can use any analysis tool of your choice (MAPREDUCE or SPARK (or even HIVE, PIG, or IMPALA)). Remember to develop and test your implementations on a small subsample of the data!

(a) In order to be able to evaluate your approach the goal of your project is to predict the ratings for all user-item pairs in the test set (`TestingRatings.txt`). How many distinct items and how many distinct users are there in the test set (`TestingRatings.txt`)?

(b) The collaborative filtering approaches lives from finding **many similar users** (for a user-user model) or **many similar items** (item-item model):

- user similarities are measured by *overlapping items*
- item similarities are measured by *overlapping users*

The better the similarities the better the rating predictions, and hence, the recommendations. So, we are interested in how well we can estimate user or item similarity respectively. To get an idea on which similarity can be estimated more accurately, pick a user from the test set, extract the items this user has rated in the training set and compute how many other users in the training set have rated the same items. Let's call this quantity *overlap of an item for a given user*. To get a comparison statistic you can average over all the items of that user and also perform this computation for multiple users from the test set to get an *estimated average overlap of items for users*. Then do the same for items instead of users to get the *estimated average overlap of users* for items. Which number do you expect to be higher:

- average overlap of items rated by the users in the training set for users in the test set or
- average overlap of users that rated items in the training set for items appearing in the test set?

You should get those estimates from a small amount of users and also a small amount of items; no need to use the whole dataset.

(c) Use the statistics computed in the two previous parts to decide which way of implementing the collaborative filtering approach is best for the given evaluation task. Consider both *efficiency* and prediction *quality*. Explain/justify your choices in the milestone 2 demo and in your project report.

(d) If you plan to use normalized ratings implement the pre-processing job and run it. If you plan to use Jaccard similarity produce a version of both the training and the testing set where you filter out all movies with low ratings.

Discuss your findings and the approach you want to implement in the milestone 2 demo with the TA or instructor. Include your findings and a description of your approach in the

written project report in your final submission. Submit your implementations by adding all scripts or java files and classes to the `milestone2` folder. **Do NOT add any data!**

Add the new files/folders to your SVN repo before committing:

```
$ svn add milestone2/your_files
$ svn commit -m 'milestone 2 submission' .
```

Problem 3: Collaborative Filtering Implementation

Now, you are ready to implement your approach. The goal of this final project is to predict as many of the ratings for the 100,000 user-movie pairs in the `TestingRatings.txt` file as possible. The true ratings (*gold standard*) are given in that file as well, make sure you only use those to compute the evaluation measure on how accurate your approach predicts the ratings!

Detailed implementation requirements/specifications:
Implement a collaborative filtering algorithm using the 3.25 million ratings provided in the `TrainingRatings.txt` file as given utility matrix. Finding the k-most similar items or users should be implemented in MAPREDUCE or SPARK . Part of the grade for the results (10%) will be assigned according to a ranking of the number and quality of your team's predictions compared to the other teams' results (the more ratings you predict the better; the lower the error on your predictions the better)!

Step 1: Implementation

Implement your approach. Make sure you discuss this step with your team members responsible for Cloud Execution, so that your implementation (ALG 1 (or 2) vs ALG 3 (or 4)) fits their needs!

Step 2: Execution and Evaluation

(a) Run your implementation to predict the ratings for all (or as many as possible) user-item pairs in `TestingRatings.txt`. The first step (finding the most similar items or users) won't complete on the pseudo-cluster in your VM (*Trust me, it won't finish or run out of memory!*). See **Problem 4** for more information on how to get this to work.

(b) For evaluation, compute the **Mean Absolute Error** (as defined here: `https://en.wikipedia.org/wiki/Mean_absolute_error`) and the **Root Mean Squared Error** (as defined here: `https://en.wikipedia.org/wiki/Root-mean-square_deviation`) for your predictions.

Step 3: Does your approach work for your own preferences?

Add yourself as a new user to the data set. To do this, you will need to create a new, unique user ID for yourself. Select some movies that you have seen among those in the training set, and add your ratings for those. Use your approach to output predictions for the movies you haven't rated, and rank those in decreasing order of the rankings. Do you

agree with the predictions of your system? Include a description of this experiment in the project report. Then, check out some of the top ranked movies that you haven't seen (but only **after** you have finished your work on the project!).

Step 4: Documentation of Approach and Results (Report)

Write the project report documenting and discussing your collaborative filtering approach, your implementation, and the obtained evaluation results (number of predictions computed and average error measures). This report should be readable for an informed outsider and it should not require the reader to look at or run any code.

Problem 4: Cloud Execution

Execute your MAPREDUCE program or SPARK application on the full data on Amazon EMR. Be aware that execution times on the cloud might still be pretty slow and you might have to resort to only computing the predicted ratings for a subset of the user-movie pairs in testingRatings (ALGO 3 and 4 provided in the appendix of the RS I lecture slides will be useful to achieve this).
Challenge: If you are looking for an even bigger dataset for your EMR execution, check out the Amazon Product Data (`http://jmcauley.ucsd.edu/data/amazon/`).
Document your cloud execution approach and describe your findings including runtimes in your final project report.

Final Submission Instructions

Submit your report including **documentation**, as well as, **results** as `project_report.pdf` by adding it to the `final_project/netflix` folder in your SVN repository. Submit your implementation by adding your implementations to the `final_project/netflix/src` folder in your SVN repository. **Do NOT add any data!**

Add the new files/folders to your SVN repo before committing:

```
$ svn add src/*
$ svn add project_report.pdf
$ svn commit -m 'final project submission' .
```

Copyrights

Problems are adapted and data is taken from Pedro Domingos' class on Data Mining/Machine Learning at University of Washington, 2012.