Date: 11 December 2019

Name: Hadi Saputro

Subject: Movie Recommender System

Movie recommender system is filtering system to predict users' movie preferences. In this project, the system was built using Python 3.7 based on Hybrid filtering algorithm. The type of hybrid algorithm used here was weighted algorithm, which the predicted rating from Collaborative Filtering (CF) and Content Based Filtering (CBF) were combined in to some weighted.

Firstly, data train provided on Kaggle was used to generate the recommender system. Using Pandas package to read the csv datasets into dataframe. The data were on file train-PDA2019.csv for user rating data and content-PDA2019.csv for movies detail. First, before deciding which algorithm used for CF, KFold from sklearn package was used to identify which algorithm was the best between SVD, NormalPredictor, KNNBaseline, KNNBasic, KNNWithZScore, BaselineOnly and CoClustering. Then, SVD was chosen since it has the least value of RMSE. Furthermore, GridSearchCV was used to find the best param for SVD, and the param consist of lr_all=0.001 and reg_all=0.01 was the best.

File of train-PDA2019.csv was used as full train and by calling method build_anti_testset(), the predicted rating for movies that had not been rated by users was generated. Using this result, the top 10 movies with highest predicted rating for each user was used as the recommendation.

The CBF algorithm was started by cleaning the data from content-PDA2019.csv. the value of column genres was split by '|', then the result was combined to value of column tag and saved to ne new column of descr. Each of this value was an array of word. Then each word that has space was transformed without space because I want word "Johny Depp" was different with "Johny" or "Johny Volta". Then it was lemmatize so word "actors" and "actor" would be the same word. After that, I used tf-idf to build vector content based on words. This method same as counting words but the value was normalized.

Using this movie content, user preference was generated based on the movies they had rated. The users' preference has same dimension with item preference. $U = \frac{\sum_{i=1}^{N} w_i R_i}{\sum R}$. U was user preference vector, w_i was vector movie i and R_i was given rate of movie i. By calculating

cosine-similarity between user preference and movies preference vector, the top 10 recommended movies were predicted by taking the 10 highest score of cosine similarity.

Finally, both methods were combined. Since the prediction value between CF and CBF had different range, (CF had range 1-5 and CBF had range 0-1), I used MinMaxScaler to make both values had same range. Then, after experimenting adding weight for each value, I found 7:3 of CF:CBF produced better recommendation. So, the value from CF was multiplied by 0.7 and CBF was 0.3, then the results were added. After that, the final result was sorted from the biggest to the lowest, and the top 10 movies were recommended to user.

Using the data test from Kaggle, this hybrid method ran for around 1 hour 5 minutes and the score was around 0.04896. This experiment was ran on Macbook Pro 2018, 2,2 GHz 6-Core Intel Core i7, 16 GB RAM using Python 3.7 with packages: numpy, pandas, surprise, sklearn, scipy and nltk. To run this program, find the Anaconda notebook file in src folder, run each cell code sequentially from top to the end. To use different datasets, place the new one on pda2019 folder and change the filename in notebook file.