DESIGN DOCUMENT

CS F469 Movie Recommender System

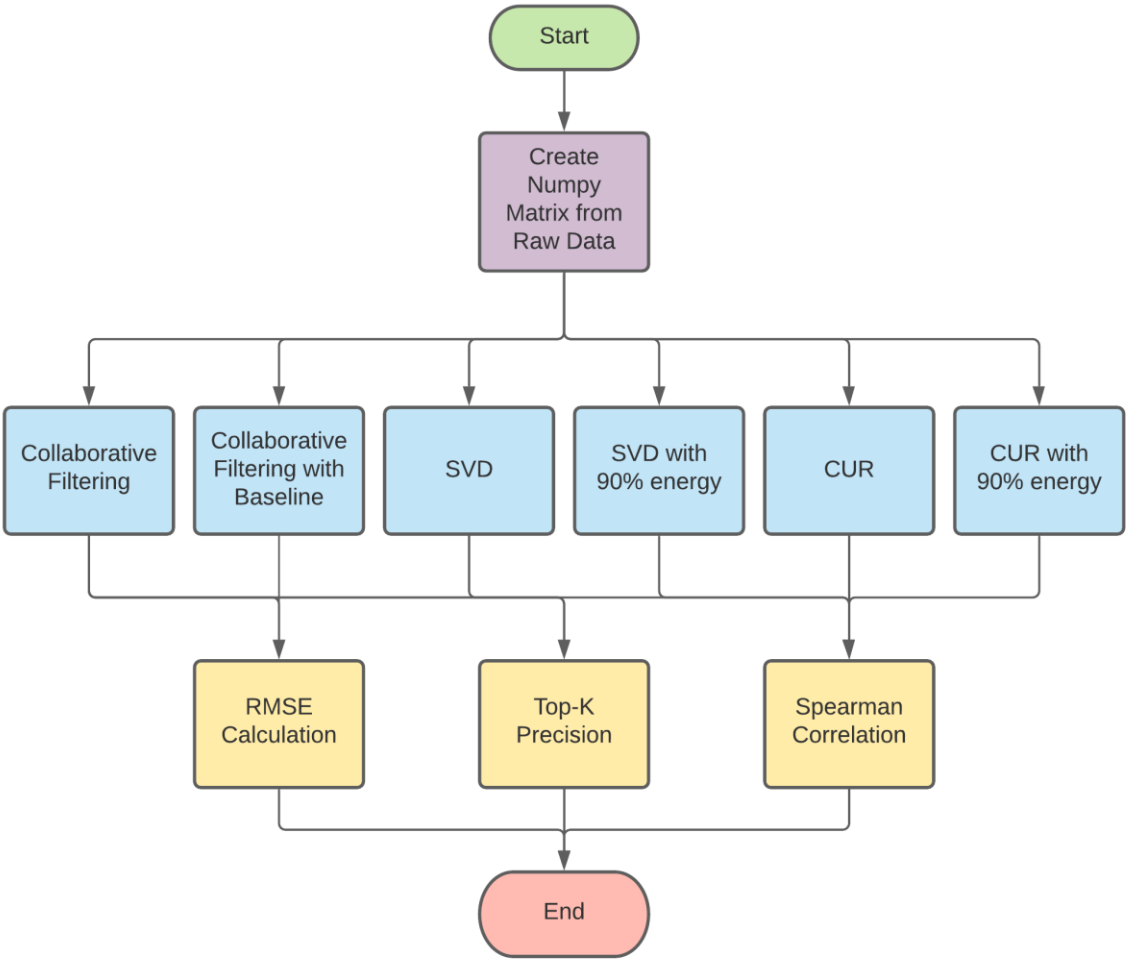
This project is a recommender system based on the MovieLens dataset built using

1. Collaborative filtering
2. SVD - Singular Value Decomposition
3. CUR

Requirements

1. Numpy
2. Pandas

Flowchart



Collaborative Filtering (CF)

The implemented algorithm is an approach to recommendation systems where we estimate the rating a user might give to an item based on the ratings the user has given to other items. This is done by computing the similarity between two items and then using the score as weights to the ratings already present.

Pearson Correlation Coefficient

`method similarities.pearson_sim`

The similarity metric being used here is the Pearson Correlation Coefficient, given by the Cosine similarity between two standardized vectors (i.e., rows correspoinding to items).

$\mathrm{sim}(x, y) = \Sigma(r_{xs}-\mu_x)\ (r_{ys}-\mu_y)\ /\ \sqrt{[\Sigma(r_{xs}-\mu_x)^2 \text{ x } \Sigma(r_{ys}-\mu_y)^2]}$

Item-Item CF

`class collaborate.Collaborate`

The actual rating is estimated here and the input matrix is filled.

To estimate rating for a given (user, item) pair:

$r_{xi} = \Sigma \mathrm{sim}(i,j).r_{xj}\ /\ \Sigma \mathrm{sim}(i,j)$

A common practice to get a better estimate is to account for baseline offset.

$r_{xi} = b + \Sigma \mathrm{sim}(i,j).(r_{xj} - b_x)\ /\ \Sigma \mathrm{sim}(i,j)$

where b is given by, $b = \mu + b_x + b_i$

μ: Overall Mean
b_x: Deviation for user *x*
b_i: Deviation for item *i*

Evaluation Measures

Contents

1. RMSE - Root Mean Square Error
2. Precision in Top K
3. Spearman Correlation

RMSE - Root Mean Square Error

Formula: `(sum((predicted - actual) ** 2) / n) ^ 0.5`

Precision in Top K

Gives an estimate of how many of the predicted ratings are present in the top K ratings of the user since only the good one's count in the error measure.

Spearman Correlation

Formula: `1 - [sum(diff(predicted - actual)^2) / n((n^2)-1)]`

Preprocessing

Dataset: `http://files.grouplens.org/datasets/movielens/ml-100k.zip`

1. Column 1 - User ID
2. Column 2 - Movie ID
3. Column 3 - Rating

Other columns are ignored since they are irrelevant to the implemented algorithms.

Output: Numpy array of shape `963 * 1682`

SVD - Singular Value Decomposition

Theory:

`A = U * Sig * V'`

A = Original Data Matrix (Users * Items)

U = Users to Concept matrix

V = Items to Concept matrix

Sig = Concept Strength matrix containing Eigen values in decreasing order

CUR

Theory:

`A = C * U * R`

A = Original Data Matrix (Users * Items)

C = Columns from original matrix

R = Rows from original matrix

U = Intersection of rows and columns from R and C matrices

Authors

- Kumar Pranjal - 2018A7PS0163H
- Sukrit - 2018A7PS0205H
- Mridul Kumar Rai - 2018AAPS0359H
- Sri Satya Aditya Vithanala - 2018A7PS0175H