# **HW 4 (584-Rangwala): Recommender Systems**

# Pankaj Kumar Jatav – G10338769 Prajwal Arvind Desai– G01326781

**Goal:**

Develop a Recommender System that Uses the Rating Information (Matrix) and Side-Information (i.e., Additional Content).

**Approach:**

We follow the below diagram step to solve this assignment.



**Pre-Processing of Data:**

Below are the steps taken to pre-process that data.

1. Choose the unique value for reach user, movies and context based data to reduict the dimensionality

np.unique(users)

1. Create dictionary for user, movies and each features so that lookup for movie in run time can take less time

users = np.unique(np.concatenate((train\_array.userID.values, test\_array.userID.values)))  
users = dict(zip(users, np.arange(users.size)))

1. As there were no cold start users and cold start movies the training file was good to go for further preprocessing also there are movies present in the features file which were not present in the movies file, to overcome the issue of missing movie data we biased the feature files by removing the movies which were not present

**Model Training:**

Below are the steps taken to create model from training data.

1. Convert each realtion to sparse matrix.
2. Create a function to find the k-nearest neighbours for user, movie and slected features.

rec\_knn(users, movies, sparse\_train, test\_array, feature\_similarities, k=20)

1. Find the means for user, movies and each selected features. Then caculated the means for all to predict the user rating.

user = users[line[0]]  
movie = movies[line[1]]  
user\_neighbor\_indices = user\_cos\_similarity[user].todense()  
user\_neighbor\_indices = np.array(user\_neighbor\_indices).flatten()  
user\_neighbor\_indices = user\_neighbor\_indices.argpartition(-k - 1)[-k - 1:]  
user\_neighbor\_indices = user\_neighbor\_indices[np.where(user\_neighbor\_indices != user)]  
  
user\_ratings = []  
for userID in user\_neighbor\_indices:  
 user\_ratings.append(sparse\_train[userID, movie])  
  
user\_ratings = np.array(user\_ratings)  
user\_ratings = user\_ratings[np.nonzero(user\_ratings)]  
  
if user\_ratings.size == 0:  
 user\_ratings = sparse\_train.tocsc()[user, :].data  
 try:  
 user\_rating = user\_ratings.mean()  
 except:  
 user\_rating = 0  
else:  
 user\_rating = user\_ratings.mean()

**Models:**

We have used the below clustering technique to find the best for this assignment.

1. KNN without features (User based recomandation)
2. KNN with features (Context base recomandation )

**Cross Validation:**

We split the training data into 80:20 ratio and then performend cross validation on the split data and got the following results with different features selected

1. Tag

0.732126149290946

2. Genre

0.7353198535141031

3. Actor

0.7404877668692535

4. Director

0.7431614461586411

5. Tag-Genre

0.7771532647654668

6. Actor-Director

0.7971260713729156

7. All

0.8472695184665733

**Step To Run the program:**

Run main.py file

>>python3 main.py

It will give 5 options as bellow.

Please select a options

1. Run Cross Validation

2. Run Submission

3. Run Both

4. Save Data Graph

5. Exit

Enter your choice: 4

File save as fig.jpeg

Please select a options

1. Run Cross Validation

2. Run Submission

3. Run Both

4. Save Data Graph

5. Exit

Enter your choice: 5

**Plots:**

**Part-1 Training data**

