MOVIE RECOMMENDATION SYSTEM

Group Members

Aasav Badera 18075001

Deepesh Tank 18075017

Dishant Chaurasia 18075018

Divyansh Chaturvedi 18074005

INTRODUCTION

- A Movie Recommendation System seeks to predict or filter movie preferences according to the user's choices.
- Recommendation System typically speed up searches and make it easier for users to access content they're interested in and surprise them with movies based on their likes that they would have never searched for.
- The main goal of this data mining project is to build a movie recommendation engine that predicts the rating that a user would give to a movie that he has not yet rated and to minimize the difference between the actual rating and predicted rating.
- Companies like Netflix , Amazon , Youtube etc uses their own complex recommendation systems to enhance their user's experiences as well as attracts new users.

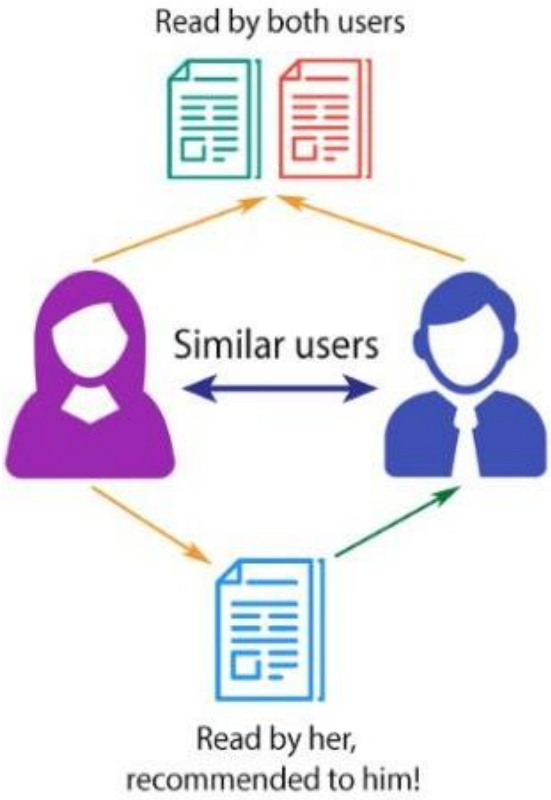
WHY RECOMMENDATIONS SYSTEMS ARE IMPORTANT ?

- 35% of the purchases on Amazon are the result of their recommender system.
- Recommendations are responsible for 70% of the time people spend watching videos on youtube.
- 75% of what people are watching on Netflix comes from recommendations
- Employing a recommender system enables Netflix to save around $1 billion each year.

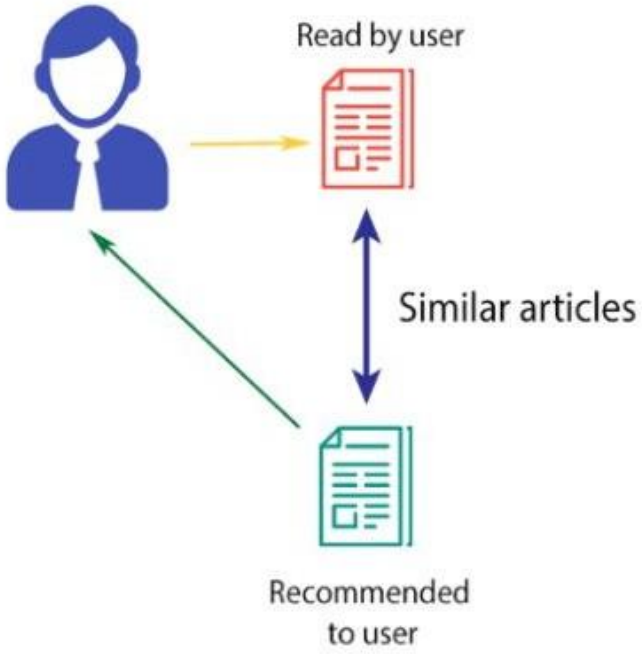
TYPES OF RECOMMENDATION ENGINES

- There are basically three types of recommendation engines:
  a. Collaborative filtering :
     This method is usually based on collecting and analyzing information on user's behaviors, their activities or preferences and predicting what they will like based on the similarity with other users.
  a. Content-Based Filtering:
     This method is based on the description of an item and a profile of the user's preferred choices
  a. Hybrid Recommendation Systems:
     This method is implemented by making content-based and collaborative-based predictions separately and then combining them.

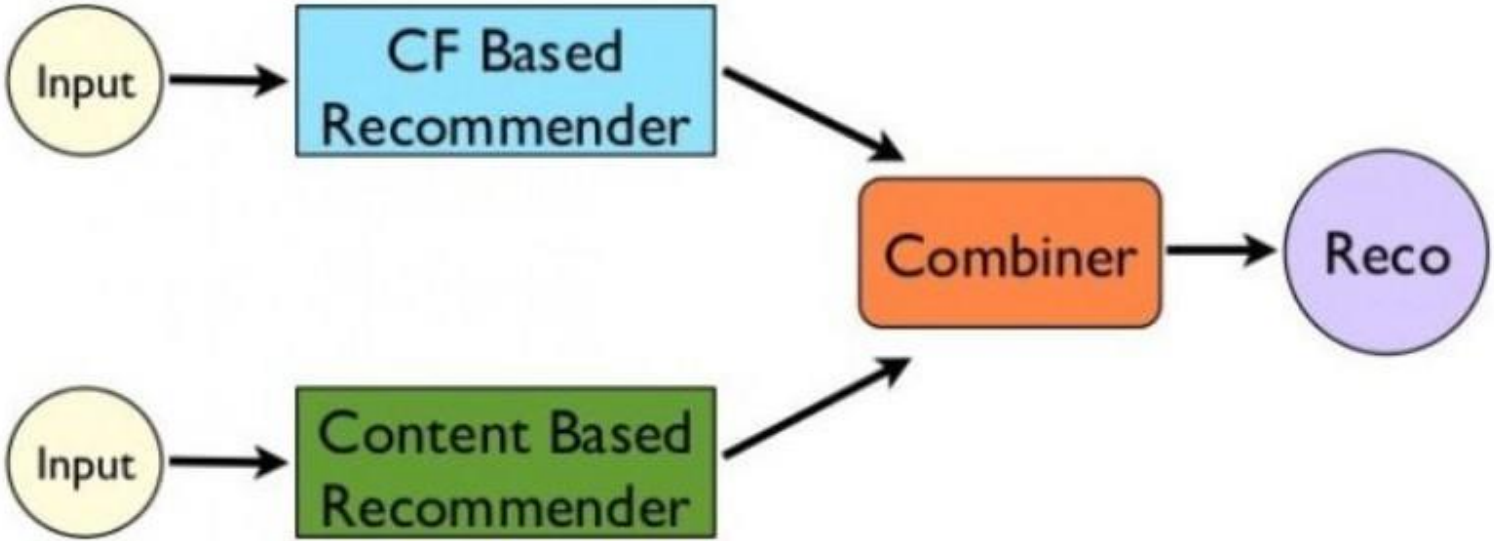
COLLABORATIVE FILTERING



CONTENT-BASED FILTERING



Hybrid Recommendations



IMPLEMENTATIO N

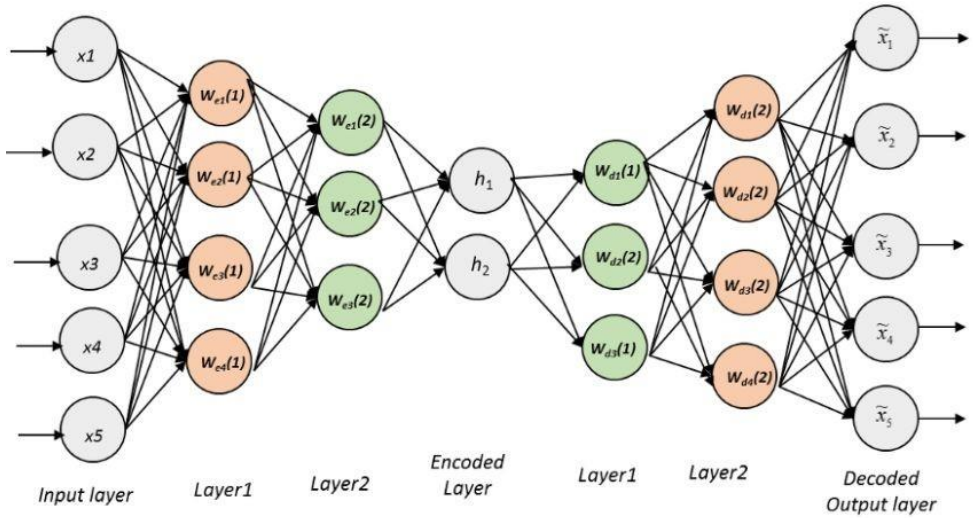
- In our project we use collaborative filtering for our movie recommendation system.Collaborative filtering is based on the assumption that people who agreed in the past will agree in the future, and that they will like similar kinds of items as they liked in the past.
- For example, if a person A likes item 1, 2, 3 and B like 2,3,4 then they have similar interests and A should like item 4 and B should like item 1.
- We use Artificial Neural Network (ANN) consisting of stacked autoencoders for implementing the project.

AUTOENCODERS

- An autoencoder neural network is an Unsupervised Machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.
- Application of autoencoders :
  - Data Reduction
  - Image Denoising
  - Recommendation System
  - Feature Extraction

STACKED AUTOENCODERS

- A stacked autoencoder is a neural network consist several layers of autoencoders where output of each hidden layer is connected to the input of the successive hidden layer.

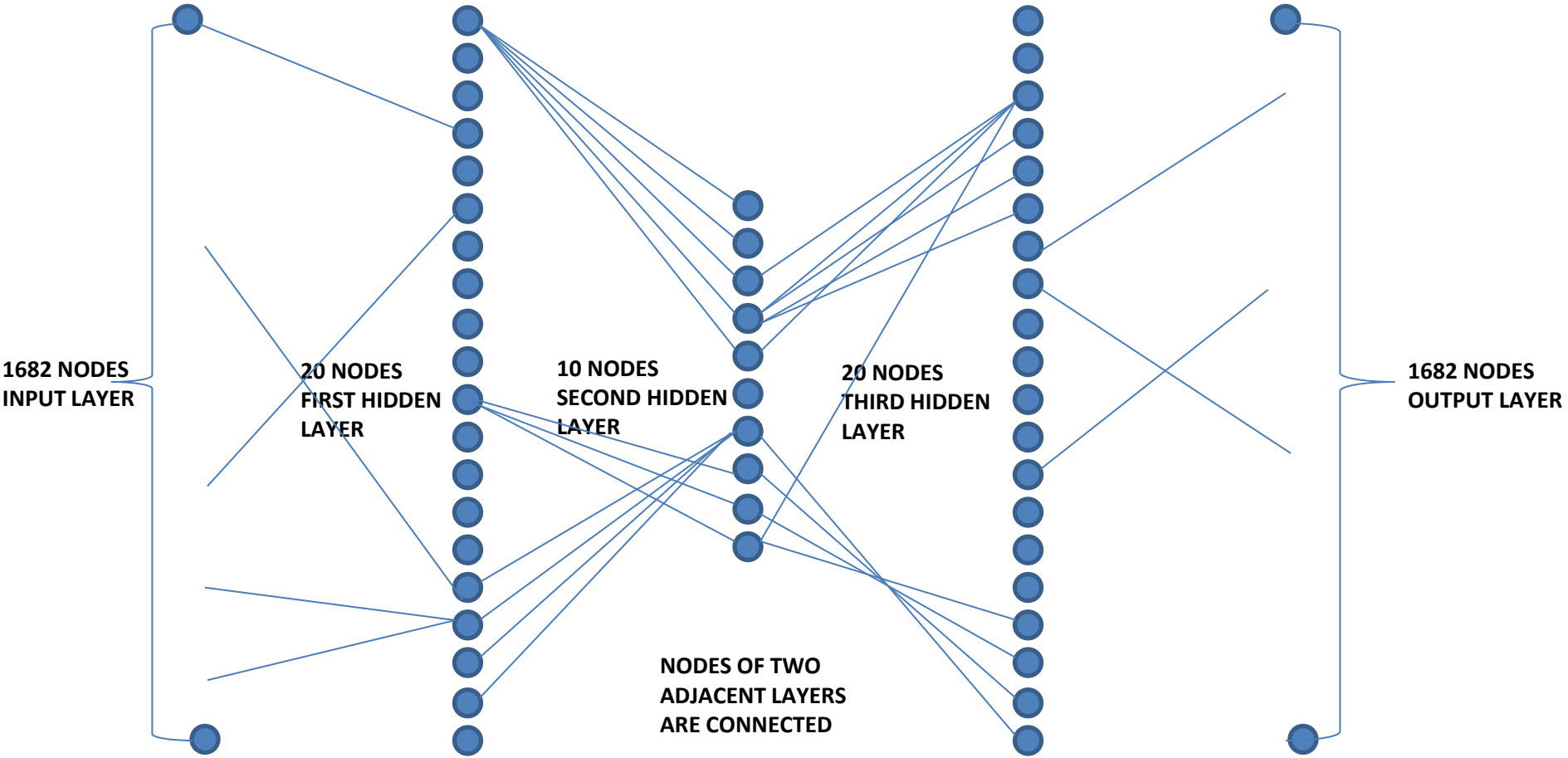


User-Item Matrix

- We will create an user-item Matrix which will have users vs movies as it's parameters and the data inside each cell will be rating given by a user of that row to that movie from 1 to 5 or 0 if not watched.

Users	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	...
User1	0	0	4	0	1	0	...
User2	2	5	2	0	0	2	...
User3	0	0	5	3	2	4	...
User4	1	0	0	4	0	0	...
User5	2	3	0	0	0	0	...
...	...	...	...	...	...	...	...

MODEL ARCHITECTURE



Algorithm :

Step 1: Each cell (u,i) of user-item matrix contains the rating (from 1 to 5, 0 if no rating) of the movie i by the user u.

Step 2: The input vector for first user x = (r1, r2,.....,rm) contains all its ratings for all the movies.It is then fed into the network.

Step 3: The input vector x is encoded into a vector z of lower dimensions using a mapping function f (here Sigmoid function).

$$z = f(Wx + b)$$ where W is the vector of input weights and b is the bias

Step 4: z is then decoded into the output vector y of same dimensions as x.The aim is to replicate x(input vector).

Step 5: Then we compute reconstruction error d(x,y) = ||x-y||. Our goal is to minimize the reconstruction error.

Step 6: This step is essentially gradient descent.The error is back propagated from right to left. The weights are updated according to how much they are responsible for the error. The learning rate decides by how much we are updating the weights.

Step 7: The steps 1-6 are repeated for each user.

Step 8: When the whole training set passes through the network, that makes an epoch. We redo more epochs to improve the accuracy of the model.

K-Fold Cross Validation

- Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.
- Motivation is to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.
- Parameter ‘k‘ refers to the number of groups that a given data sample is to be split into.
- We have taken value of k =5, thus making it 5-fold cross validation.

Procedure:

The procedure is as follow:

1. Shuffle the dataset randomly.
2. Split the dataset into k(in our case 5) groups.
3. For each unique group:
    1. Take the group as a testing data set
    2. Take the remaining groups as a training data set
    3. Train a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model.
4. Summarize the skill of the model using the evaluation scores.

Dataset splitting:

- We have splitted the dataset into 5 different ratios of training and test data.

- They are as follows:
  - 50-50 for training dataset and testing dataset.
  - 60-40 for training dataset and testing dataset.
  - 70-30 for training dataset and testing dataset.
  - 80-20 for training dataset and testing dataset.
  - 90-10 for training dataset and testing dataset.

Parameters & Hyperparameters Tuning :

- Parameter:
  Previously we had 10 - 8 - 10 nodes in hidden layers, now we 20 - 10 - 20 nodes in the hidden layers.

- Activation Function :
  Changed to sigmoid for the best results.

- Learning rate:
  Learning rate has been tuned to 0.01 .

RESULTS :

Previous RMSE Loss = 1.0243

Ratio of Training and Test Dataset

RMSE Loss	90-10	80-20	70-30	60-40	50-50
Set 1	0.9309	0.9670	1.0031	1.0248	0.9848
Set 2	0.9186	0.9642	0.9772	0.9957	0.9835
Set 3	0.9275	0.9762	1.0117	1.0115	0.9847
Set 4	0.9181	0.9686	0.9940	0.9973	0.9858
Set 5	0.9308	0.9579	0.9943	1.0800	0.9841
RMSE Loss after 5-fold cross validation	0.9252	0.9668	0.9960	1.0218	0.9846

Note: Total epochs on each set is 100

Dataset Source :
https://grouplens.org/datasets/movielens/

Comparison with other standard recommendation system :

	Netflix Recommendation System (2009) Matrix Factorisation (a.k.a. SVD) and *Restricted Boltzmann Machines* (RBM)	**MovieLens Recommendation using Matrix Factorization by Jakob Ivarsson**	**Top N recommendation System by Susan Li**	**MovieLens Recommender System Collaborative filtering**	**Our Project based on Movielens Dataset (100k)**	*Netflix Recommendation System (2007) CineMatch*	**MovieLens Recommender System Content-based recommendation**
Loss	0.8567	0.8743	0.89958	0.923	0.9252	0.9525	0.993

THANK YOU