Collaborative Movie Filtering with Website and Database

by

Mridu Shukla (18BCE1179)
Saswat Panda(18BCE1281)

A project report submitted to

Prof. Dr. Radhika Selvamani

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

in partial fulfillment of the requirements for the course of

CSE3013 – ARTIFICIAL INTELLIGENCE

in

B.Tech. COMPUTER SCIENCE AND ENGINEERING

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

APRIL 2020

BONAFIDE CERTIFICATE

Certified that this project report entitled "Collaborative Movie Filtering with Website and Database" is a bonafide work of Mridu Shukla(18BCE1179) and Saswat Panda(18BCE1281) who carried out the project work under my supervision and guidance for **CSE3013 – ARTIFICIAL INTELLIGENCE.**

Prof. Dr. Radhika Selvamani V

Assistant Professor(Senior Grade 1)

School of Computer Science and Engineering (SCOPE),

VIT Chennai

Chennai – 600 127.

ABSTRACT

With the rapid development of Internet technology, today's society has entered the era of Web 2, information overload has become a reality. How to find the required information in the mass of data has become a hot research topic. The movie is one of the main spiritual entertainment, also has the problem of information overload. In order to solve this problem, we put forward a proposal for a personalized movie recommendation system.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Dr. Radhika Selvamani,** Assistant Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Prof. Dr. Jagadeesh Kannan R**, Dean of the School of Computer Science and Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Prof. Dr. Justus S for** his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

(This page is intentionally left blank)

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

The objective of this project is to design and implement a movie recommendation system prototype combined with the actual needs of movie recommendation through researching of KNN algorithm and Matrix-Factorization algorithm which is a class of collaborative filtering algorithm.

1.2 BENEFITS

An enjoyable game that utilizes assembly language concepts.

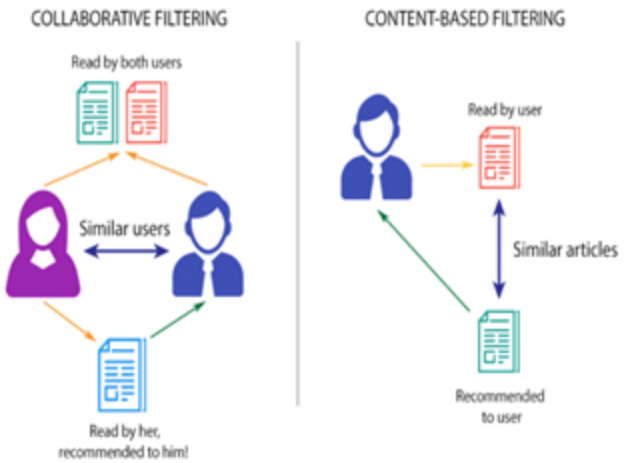
1.3 FEATURES

- Interactive and user-friendly website
- Matrix-Factorization algorithm used to provide proper recommendations.
- User data saved in the database.
- If no more moves are possible, a draw is declared.

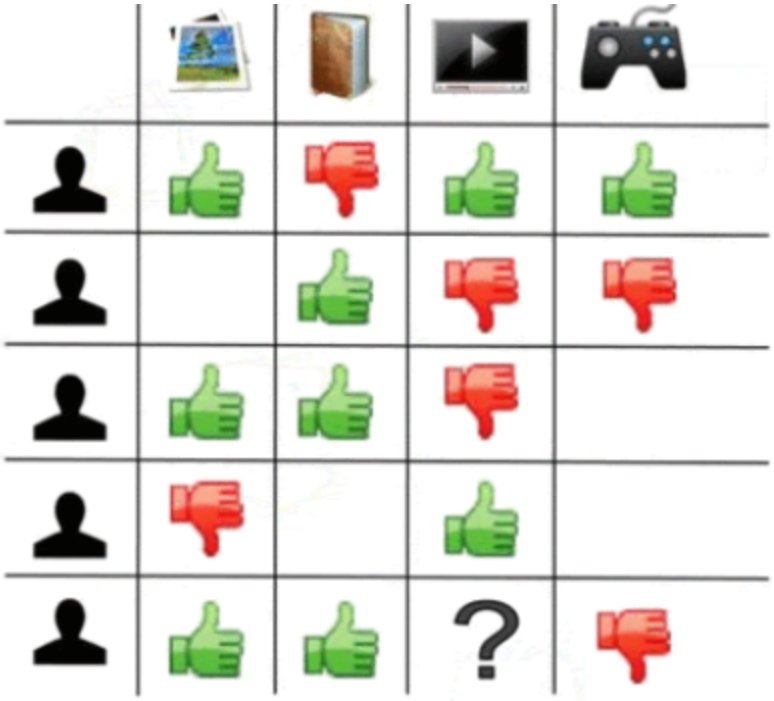
2 About Collaborative filtering

2.1 BLOCK DIAGRAM

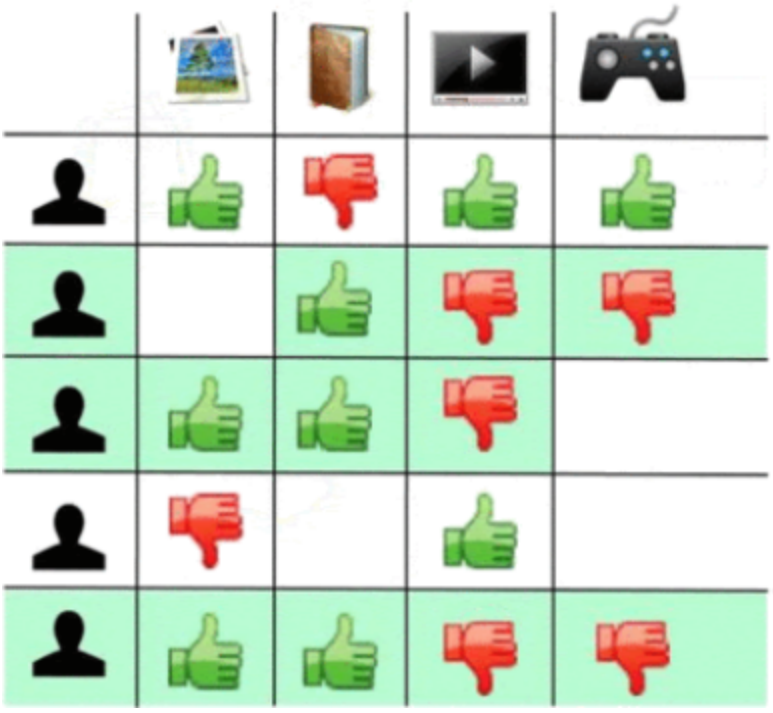
Collaborative filtering involves collecting information from many users and then making predictions based on some similarity measures between users and between items. This can be classified into user-based and item-based models.



Matrix factorization is a class of **collaborative filtering** algorithms used in recommender systems. This family of methods became widely known during the **Netflix prize** challenge due to its effectiveness.



The missing value of the last user of the above image will be predicted by collaborative filtering based on the below images.



In **matrix factorization**, the above image(users and attributes in the image) is converted to a matrix. We can see in the just above image that based on the first two users the last users' choice is predicted.

3. SOFTWARE IMPLEMENTATION

3.1 LANGUAGE & TOOLS

Languages Required:

- Python
- HTML
- CSS
- JavaScript

Tools Required:

- Jupyter Notebook
- Django Framework
- Movie Dataset

3.2 Algorithm

In this section algorithm of **matrix factorization**, which is a class of collaborative filtering is discussed.

	D1	D2	D3	D4
U1	5	3	-	1
U2	4	-	-	1
U3	1	1	-	5
U4	1	-	-	4
U5	-	1	5	4

The task of predicting the missing ratings can be considered as filling in the blanks (the hyphens in the matrix) such that the values would be consistent with the existing ratings in the matrix.

3.3 Python Implementation

- Dataset cleaning and reading

```
import os
import pandas as pd


# configure file path
data_path = os.path.join(os.environ['DATA_PATH'], 'MovieLens')
movies_filename = 'movies.csv'
ratings_filename = 'ratings.csv'


# read data
df_movies = pd.read_csv(
    os.path.join(data_path, movies_filename),
    usecols=['movieId', 'title'],
    dtype={'movieId': 'int32', 'title': 'str'})

df_ratings = pd.read_csv(
    os.path.join(data_path, ratings_filename),
    usecols=['userId', 'movieId', 'rating'],
    dtype={'userId': 'int32', 'movieId': 'int32', 'rating':
'float32'})
```

In [7]:
```
df_movies.head()
```

Out[7]:

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

In [8]:
```
df_ratings.head()
```

Out[8]:

	userId	movieId	rating
0	1	307	3.5
1	1	481	3.5
2	1	1091	1.5
3	1	1257	4.5
4	1	1449	4.5

In [31]:
```
df_movie_features.head(5)
```

Out[31]:

userId	4	5	10	14	15	18	19	26	31	34	...	283199	283204	283206	283208	283210	283215	283219	283222	283224	283228
movieId																					
1	4.0	0.0	5.0	4.5	4.0	0.0	0.0	0.0	5.0	0.0	...	5.0	0.0	0.0	4.5	0.0	4.0	4.0	0.0	0.0	4.5
2	4.0	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	4.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	4.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0

- Collaborative Filtering

```
import numpy

def matrix_factorization(R, P, Q, K, steps=5000, alpha=0.0002,
beta=0.02):
    Q = Q.T
    for step in xrange(steps):
        for i in xrange(len(R)):
            for j in xrange(len(R[i])):
                if R[i][j] > 0:
                    eij = R[i][j] - numpy.dot(P[i,:],Q[:,j])
                    for k in xrange(K):
                        P[i][k] = P[i][k] + alpha * (2 * eij * Q[k][j]
- beta * P[i][k])
                        Q[k][j] = Q[k][j] + alpha * (2 * eij * P[i][k]
- beta * Q[k][j])
        eR = numpy.dot(P,Q)
        e = 0
        for i in xrange(len(R)):
            for j in xrange(len(R[i])):
                if R[i][j] > 0:
                    e = e + pow(R[i][j] - numpy.dot(P[i,:],Q[:,j]), 2)
                    for k in xrange(K):
                        e = e + (beta/2) * (pow(P[i][k],2) + pow(Q[k]
[j],2))
        if e < 0.001:
            break
    return P, Q.T
```

The complete code can be found in our GitHub repository which has been open-sourced.