

BI-WVM - Collaborative filtering

Ondřej Perný and Le Thanh Hung

May 12, 2019

1 Description of the project

Project is focused on collaborative filtering on MovieLens data set, that contain user rating of movies. Input is small MovieLens data set with 100 000 records(original has 10 million). Every record contain user id, movie id, movie rating and timestamp. These records are stored in csv file. Another csv file with movie id and movie name is used for mapping original movie names on the movie id's from first file, so we can print real movie names and not just id.

Our output are movies, that can be recommended to user, based on his previous ratings of other movies. The goal is to find most relevant movies, that main user will like. Main user is label for user for whom we are currently recommending movies. To achieve that result we use spearman correlation to find most similar users based on whose ratings we recommend movies.

2 Method of solution

Process of recommending as follows. We gather users that have potential to be relevant (there are at least x movies that rated both this user and main user) to main user so we could use their rating. This group of users is labeled as neighbours. Then series of filters is applied to neighbours. Filters usually come with a thresholds on which they cut off portion of collected users, that seems to be not relevant enough to main user. Core of our recommending as well as main filter is spearman correlation coefficient.

For every neighbor is remembered this coefficient as his relevance to main user. Other filters works mostly as improvements to further optimizations. Filters are parametrized (thresholds) and values those parameters can be alternated, but have default values that empirically proved to have best results. More restrictive those parameters are, better recommendations are achieved but amount of movies that are evaluated to recommendation decline rapidly. When neighbours pass all filters, their rating is used for recommendation.

Firstly is made collection of movies that at least y -amount of neighbours rated, then expected rating for every of those movies is calculated. Expected rating is rating that is expected that main user would give to that specific movie. Calculation of expected rating is average of neighbours ratings weighted by their relevance/similarity to main user. By this value are then movies sorted and recommended. For worst case, where none movies are found to be recommended (for example for new user with no ratings), failsafe mechanism will ensure that at least movies with best ratings from whole data set are recommended.

3 Implementation

3.1 Backend

Whole project is programmed in Python 3.6. Backend doesn't use any special special libraries for calculations. Used libraries are shutil, csv, os and time where all of those are used just for simple task regarding database - csv file.

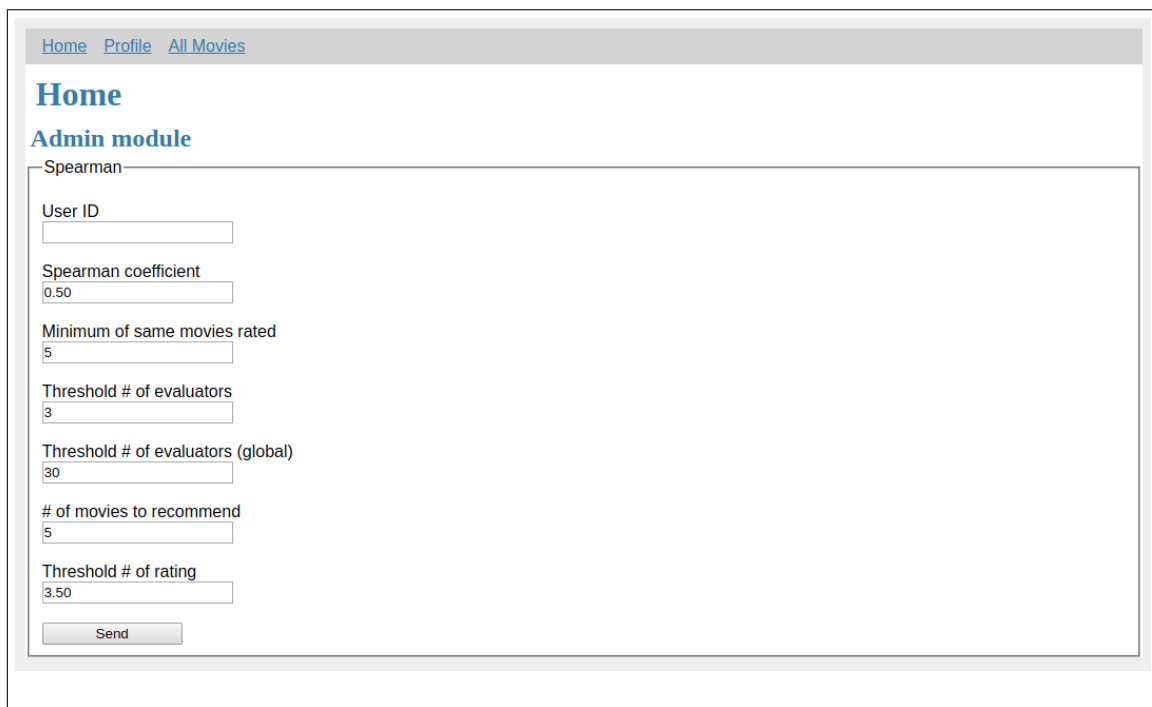
LoadInput.py file contain IOclass that has all functions that maintain work with database(reading, writing, altering, etc.).

Algorithm.py file contain Recommendation class, that is core of the whole application. All calculating and filtering functions are here. Constructor for this class takes as parameters thresholds that can alternate results of recommending(changing these thresholds is accessible from frontend administrator module). Recommendation class is meant to be recommendation system for one specific user and so user_id is one of constructor parameters and all calculation inside class are only this user related.

3.2 Frontend

Frontend was written in python microframework Flask which is based on Werkzeug and Jinja 2. For user input are used Forms from Flask WTF which connects Flask and WTForms.

On the homepage one can select the parameters for the collaborative filtering as well as the ID of the user that the movies are recommended to.



The screenshot shows a web application interface. At the top, there is a navigation bar with links: [Home](#), [Profile](#), and [All Movies](#). Below this, the page title is "Home". Underneath, there is a section titled "Admin module". Inside this section, there is a form with the following fields and values:

- Spearman** (header)
- User ID**:
- Spearman coefficient**:
- Minimum of same movies rated**:
- Threshold # of evaluators**:
- Threshold # of evaluators (global)**:
- # of movies to recommend**:
- Threshold # of rating**:
- Send** button

Figure 1

Then you are redirected to the profile page where you can see the recommended movies and list of all movies the the user had rated.

User is also able to edit the rating of already rated movies.

[Home](#) [Profile](#) [All Movies](#)

Profile

Recommended movies

ID	Expected Rating	Name
1148	4.8596621254947845	Wallace & Gromit: The Wrong Trousers (1993)
168252	4.8437028754382006	Logan (2017)
5618	4.770836332452973	Spirited Away (Sen to Chihiro no kamikakushi) (2001)
838	4.7282818618297915	Emma (1996)
596	4.720726828847569	Pinocchio (1940)

Rated movies

[Edit Rating](#)

ID	Rating	Name
1	4.0	Toy Story (1995)
3	4.0	Grumpier Old Men (1995)
6	4.0	Heat (1995)
47	5.0	Seven (a.k.a. Se7en) (1995)
50	5.0	Usual Suspects, The (1995)
70	3.0	From Dusk Till Dawn (1996)
101	5.0	Bottle Rocket (1996)
110	4.0	Braveheart (1995)
151	5.0	Rob Roy (1995)

Figure 2

On the last page you can see list of all the movies.

List of all movies

ID	Name
1	Toy Story (1995)
2	Jumanji (1995)
3	Grumpier Old Men (1995)
4	Waiting to Exhale (1995)
5	Father of the Bride Part II (1995)
6	Heat (1995)
7	Sabrina (1995)
8	Tom and Huck (1995)
9	Sudden Death (1995)
10	GoldenEye (1995)
11	American President, The (1995)
12	Dracula: Dead and Loving It (1995)
13	Balto (1995)
14	Nixon (1995)
15	Cutthroat Island (1995)
16	Casino (1995)
17	Sense and Sensibility (1995)
18	Four Rooms (1995)
19	Ace Ventura: When Nature Calls (1995)
20	Money Train (1995)

Figure 3

3.3 Project structure

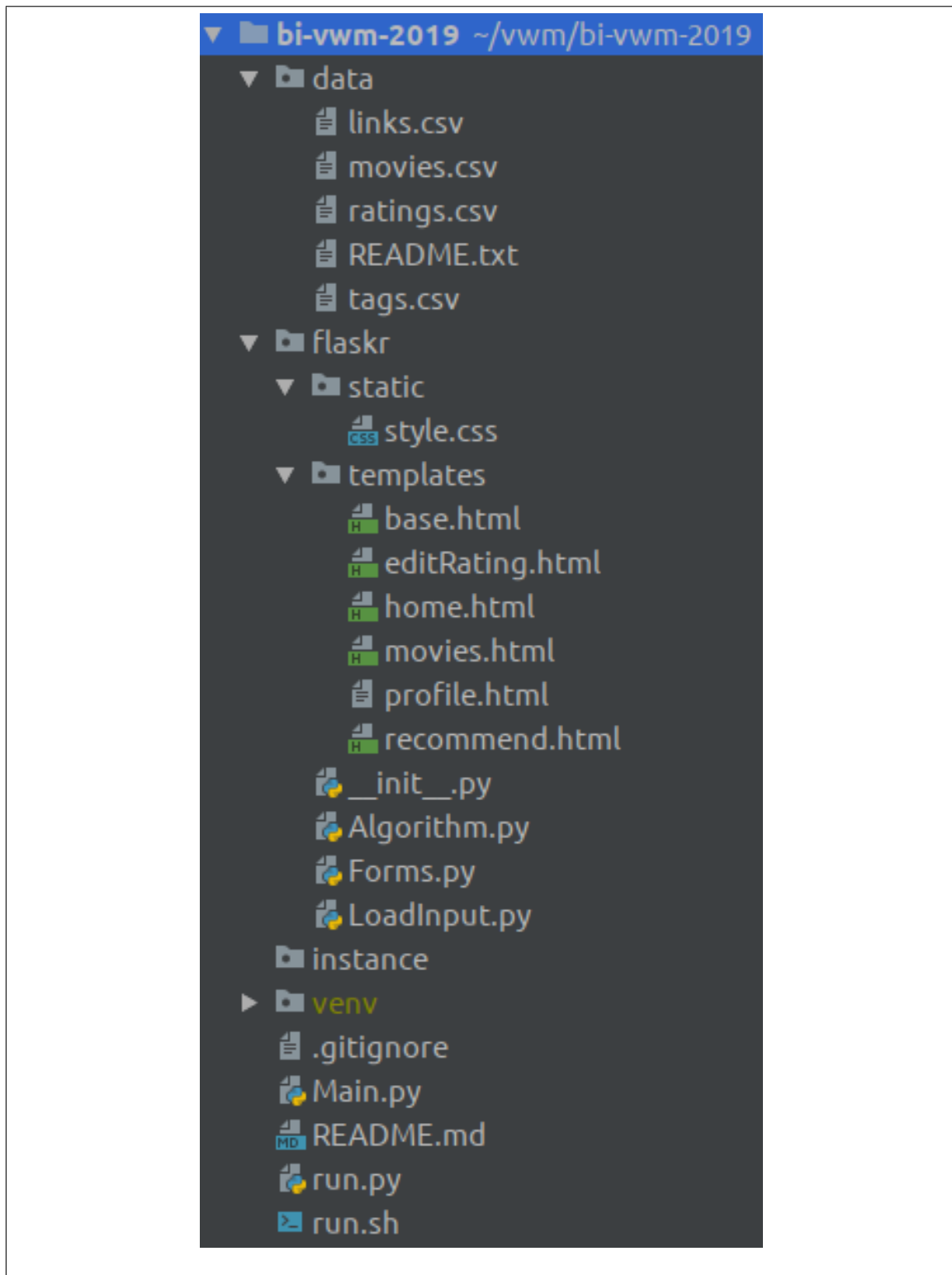


Figure 4

In folder data we have databases of movies and ratings in csv format. For backend there are file LoadInput.py which handles reading input from csv files and file Al-

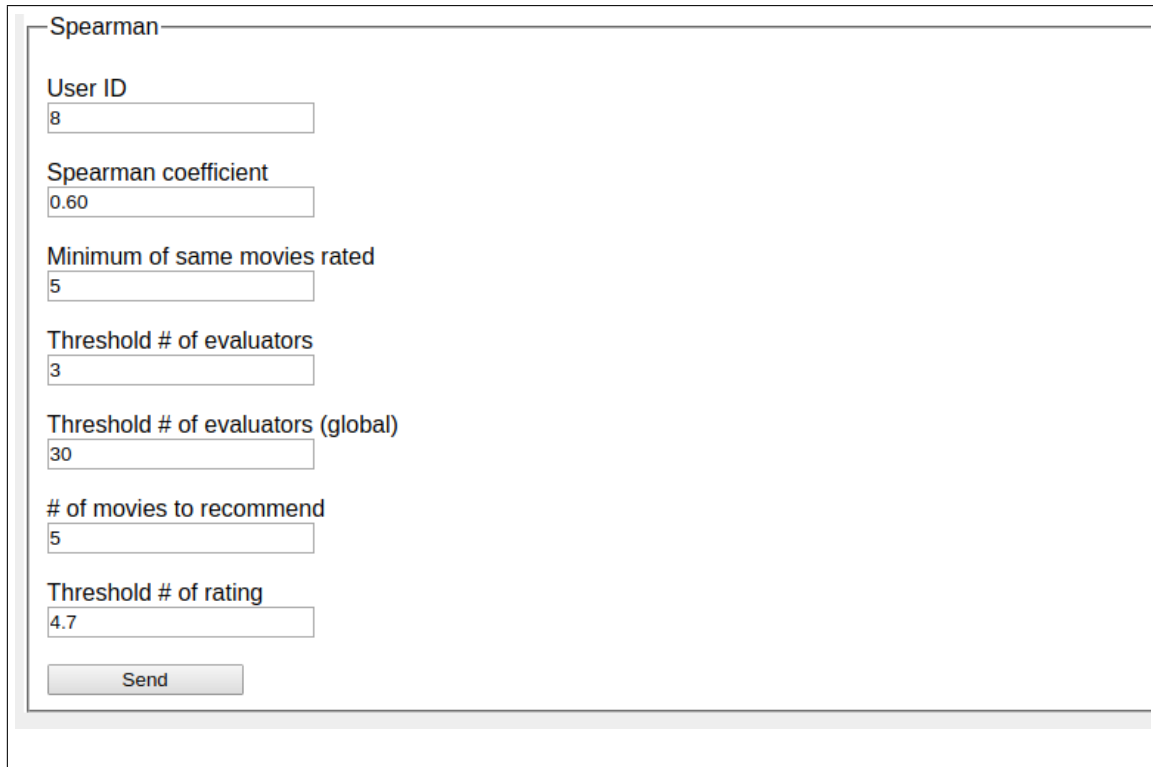
gorithm.py for backbone of our application.

File `__init__.py` is responsible for calling different endpoints of the website and connecting backend with the frontend. The endpoints are rendered with html files in the templates folder.

All templates inherit from base template `base.html` and the file `style.css` in static folder handles the styling of the webpage.

4 Examples of output

4.1 Input



The image shows a web-based input form titled "Spearman". It contains several labeled input fields and a "Send" button. The fields are arranged vertically on the left side of the form. The values entered in the fields are: User ID (8), Spearman coefficient (0.60), Minimum of same movies rated (5), Threshold # of evaluators (3), Threshold # of evaluators (global) (30), # of movies to recommend (5), and Threshold # of rating (4.7). The "Send" button is located at the bottom of the form.

Field Label	Value
User ID	8
Spearman coefficient	0.60
Minimum of same movies rated	5
Threshold # of evaluators	3
Threshold # of evaluators (global)	30
# of movies to recommend	5
Threshold # of rating	4.7

Figure 5

More about the parameters in next chapter (Experimental section) but here briefly:

User ID: ID of a user (1 to 610)

Spearman coefficient:

Minimum of same movies rated: minimal amount of movies that rated both main user and other user

Threshold # of evaluators: minimal amount of users that must have rated specific movie

Threshold # of evaluators (global)

of movies to recommend: Number of recommended movies that will be shown

Threshold # of rating: Minimum expected rating that we want to be shown

4.2 Output

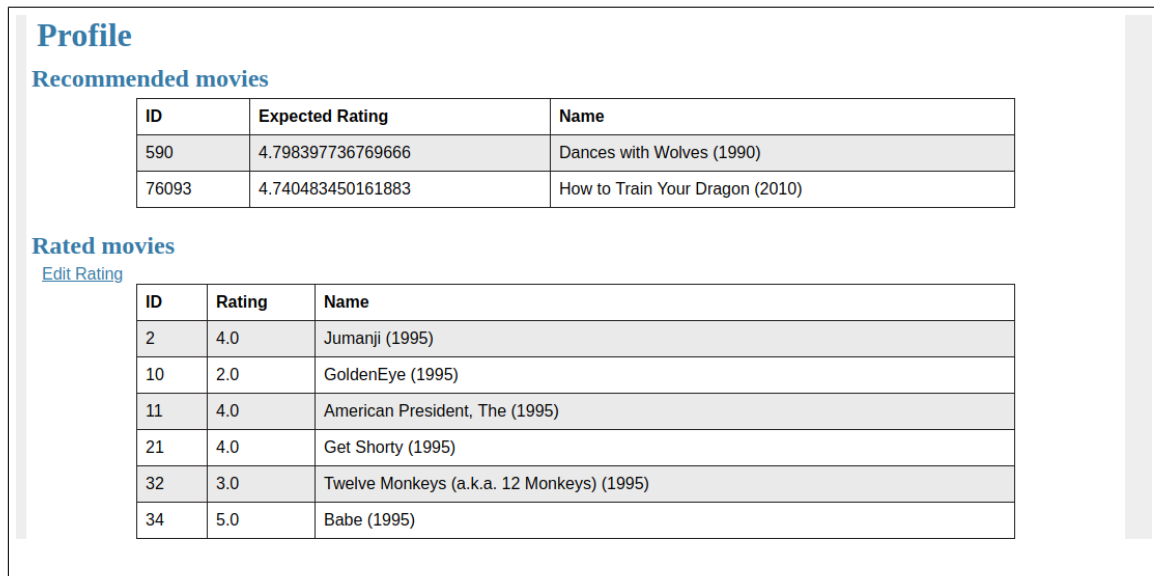


Figure 6

First table shows list of recommended movies calculated according the parameters in the form. Second table shows list of all movies that were rated by user.

5 Experimental section

We experienced with different setting of thresholds that affect things like how similar users must be, to be accepted as relevant to main user, how high must be their spearman correlation coefficient and others. Administrator module contain 6 parameters that works as those thresholds for filtering functions in backend and can alter result greatly.

First spearman_coefficient. Users correlation coefficient must be at least at high as this threshold otherwise is cut off as irrelevant. Empirically we set this default value to be 0.5. Examples showed us that our system can work very well even with higher coefficient, however we are using small MovieLens which is reduced from original MovieLens. This smaller dataset have advantage of quite higher density(about 4% density) of matrix user/movie rating than original(about 0.5% density). Let's say we would want to use original dataset with higher spearman coefficient than 0.5, we would have left enough neighbours, but we would have trouble finding many movies that were rated by more different user, which would lead to weak results. So coefficient 0.5 proved to be good enough on this data set and as well avoid overfitting to this data set in case we choose to use different.

Other thresholds are:

min_same_rated_movies - minimal amount of movies that rated both main user and other user (so the other user can be accepted as possibly relevant).

Threshold_number_of_evaluators - minimal amount of users that must have rated specific movie, so the movie can be recommended

Threshold_number_of_evaluators_global - minimal amount of users that must have rated specific movie, so the movie can be recommended when recommending from global best movies (not user specific) Number_to_recommend - maximum amount of movies that are recommended to user (can be less if not enough movies satisfy conditions)

Threshold_rating - minimal expected rating that movie must have, so it can be recommended.

6 Discussion

6.1 Backend

In terms of backend, biggest improvements could be in efficiency and file structure. During writing backend, many things were changed during process which lead to file structure that I still believe is easy to read, but could been divided in more files and classes to improve readability even further.

Efficiency was no problem in any point of our application, but some parts of code could possibly be written more efficient in exchange for more programming work, which in this case was unnecessary. In case of deploying this application for real assignment, to work on even bigger data sets, some efficiency optimizations should be applied (or even program to be rewritten in more efficient language as C++).

6.2 Frontend

From the endpoint function you could't return data from the form to another function/endpoint so we had to adjust our program so some of the parts are on the same page.

The workaround around this would be using global variables.

7 Conclusion

In collaboration we created recommender system with clear web page frontend and variable thresholds, for recommending movies to users based on similarity expressed by spearman correlation between users. Plus, different optimization techniques were applied to reach empirically plausible results. System was written as proof-of-concept so effectivity(in terms of speed) was not primary goal, but on our data set is recommendation time response negligible. System is prepared for further possible optimizations in future.