



جامعة سيدي محمد بن عبد الله بفاس
ⵜⴰⵎⴰⵎⴰⵔⵜ ⵏ ⵓⵔⴰⵎⴰⵏ ⵏ ⵓⵎⴰⵎⴰⵔ ⵏ ⵓⵎⴰⵎⴰⵔ

Université Sidi Mohamed Ben Abdellah
Faculté des sciences Dhar El Mehraz Fès



Filière : Sciences Mathématiques et Informatique

Projet tutoré Semestre 6

Mémoire

Intitulé : **Collaborative filtering approach in recommender systems**

Présenté par :

EL YAGOUBY Mohammed Amine

TAGHZOUTI Othmane

Encadrant : **Pr. NFAOUI** El Habib

Soutenu le : 28/05/2022

Jury : Examineur 1 : Pr. **BOUMHIDI** Jaouad

Examineur 2 : Pr. **LOQMAN** Chakir

Encadrant : Pr. **NFAOUI** El Habib

Année universitaire : 2021/2022

Acknowledgements

Artificial Intelligence is an absorbing topic. It has massively impacted our lives in the last few decades. But out of its many branches, Recommender Systems are extremely interesting. We owe working on this subject to Professor El Habib Nfaoui, who introduced the topic to us, and supervised us through the project. So, we would like to thank him.

We also sincerely address our appreciation towards all the members of the jury for having devoted their precious time to read and evaluate this work. Our thanks are undoubtedly addressed to our families as well, for their constant support. Finally, we are thanking all those who have contributed to the success of this work.

Abstract

Recommender Systems have been around for the last few decades. It is a technology implemented in numerous commercial applications. Nowadays, consumers are presented with many choices. So, matching them with the most appropriate products is essential to improving their satisfaction and loyalty. The whole experience of the online shopping would be nowhere near what it is today without the contribution of recommender systems.

This report represents the work we accomplished in our final study year project over a period of two months. We studied some of the different approaches that can be used in order to build a recommender system, including the one discovered in the 2006 Netflix competition. We also implemented these solutions in a web application that recommends numerous types of items. We called this application “SmartRecommender”.

Table of contents

ACKNOWLEDGEMENTS	2
ABSTRACT	3
TABLE OF CONTENTS	4
LIST OF FIGURES	7
LIST OF EQUATIONS	8
LIST OF ACRONYMS	9
GENERAL INTRODUCTION	10
CHAPTER I - GENERAL CONTEXT OF THE PROJECT	11
1. Introduction	11
2. Project overview	11
2.1. Context	11
2.2. Definition of the problem	11
2.3. Recommender systems	12
2.3.1. Content based method	12
2.3.2. Collaborative filtering method	13
2.3.3. Adopted method	15
3. Objectives of the project	15
4. Project management	16
4.1. Tasks description	16
4.2. Gantt chart	16
5. Conclusion	18

CHAPTER II - COLLABORATIVE FILTERING RECOMMENDER SYSTEMS	19
1. Introduction	19
2. Data in use	19
3. Memory-based	21
3.1. User-based	22
3.2. Item-based	22
3.3. Similarity measuring	23
3.3.1. Cosine similarity (Vector similarity)	23
3.3.2. MSD similarity	24
3.3.3. Pearson Correlation Coefficient	24
3.4. Matrix Factorization	26
3.4.1. Singular Value Decomposition	26
3.5. Stochastic gradient descent	27
4. Evaluation metrics	28
4.1. Mean absolute error	29
4.2. Root mean squared error	29
5. Conclusion	29
CHAPTER III – WEB APPLICATION “SMARTRECOMMENDER”	31
1. Introduction	31
2. Tools	31
3. Models building	33
3.1. Collecting data	33
3.2. Preparing data	33
3.2.1. Importing	33
3.2.2. Visualizing	34
3.2.3. Splitting ratings data	34
3.3. Choosing the algorithm	34
3.3.1. SVD	34
3.3.2. KNN inspired algorithm	35
3.3.2.1. KNN User based	35
3.3.2.2. KNN Item based	35
3.4. Training the model	36
3.5. Evaluating the model	36
3.6. Making Predictions	37

3.7. Example	37
4. SmartRecommender web application	38
4.1. Description	38
4.2. Use case diagram	39
4.3. SmartRecommender browsing	40
4.3.1. Display data	41
4.3.2. Calculate ratings	42
4.3.3. Get recommendations	43
4.3.4. Check the accuracy	43
4.3.5. Add new user	44
5. Results and discussion	45
5.1. Results	45
5.1.1. SVD	45
5.1.2. KNN (User based)	46
5.1.3. KNN (Item based)	46
5.2. Discussion	47
6. Conclusion	47
GENERAL CONCLUSION AND FUTURE WORK	48
1.1. Conclusion	48
1.2. Future work	48
BIBLIOGRAPHY	50

List of figures

FIGURE 1. CONTENT-BASED APPROACH	12
FIGURE 2. CONTENT-BASED FILTERING	13
FIGURE 3. COLLABORATIVE FILTERING	13
FIGURE 4. “CUSTOMERS WHO LIKED THIS ALSO LIKED THIS”	14
FIGURE 5. COLLABORATIVE FILTERING APPROACH	15
FIGURE 6. TASKS AGAINST TIME	17
FIGURE 7. GANTT CHART	18
FIGURE 8. EXAMPLES OF RATINGS DATASETS	20
FIGURE 9. USER-ITEM MATRIX (NAN ARE UNKNOWN RATINGS)	20
FIGURE 10. MEMORY BASED FILTERING	ERROR! BOOKMARK NOT
DEFINED.	
FIGURE 11. MODEL BASED FILTERING	25
FIGURE 12. SVD REPRESENTATION [12]	27
FIGURE 13. GD SCHEMATIZATION	28
FIGURE 14. SGD SCHEMATIZATION	28
FIGURE 15. DJANGO MVT ARCHITECTURE	39
FIGURE 16. USE CASE DIAGRAM	40

List of equations

(EQ. 1) MEAN-CENTERING (USERS)	21
(EQ. 2) MEAN-CENTERING (ITEMS)	21
(EQ. 7) ESTIMATION OF RATING VALUE BASED ON USERS	22
(EQ. 8) COSINE SIMILARITY (USERS) [10]	23
(EQ. 9) COSINE SIMILARITY (ITEMS) [10]	23
(EQ. 10) MEAN SQUARED DEVIATION (USERS) [11]	24
(EQ. 11) MEAN SQUARED DEVIATION (ITEMS) [11]	24
(EQ. 12) MEAN SQUARED DEVIATION SIMILARITY (USERS) [10]	24
(EQ. 13) MEAN SQUARED DEVIATION SIMILARITY (ITEMS) [10]	24
(EQ. 14) PEARSON SIMILARITY (USERS) [10]	25
(EQ. 15) PEARSON SIMILARITY (ITEMS) [10]	25
(EQ. 16) RATING ESTIMATION	26
(EQ. 17) FUNCTION TO MINIMIZE [14]	27
(EQ. 18) FUNCTION TO MINIMIZE WITH REGULARIZING TERMS [14]	27
(EQ. 19) MEAN ABSOLUTE ERROR	29
(EQ. 20) ROOT MEAN SQUARED ERROR	29

List of acronyms

RS	Recommender systems
CF	Collaborative filtering
SVD	Singular value decomposition
RMSE	Root mean squared error
MAE	Mean absolute error
MSD	Mean squared deviation
SGD	Stochastic gradient descent
GD	Gradient descent
UML	Unified modeling language
UC	Use case
MVT	Model view template

General introduction

Recommender Systems are an emerging research field that has grown fast and become popular. The peak explosion of research in this field occurred with the implementation of collaborative filtering, one of the most widely known recommender system technologies. Since this era, collaborative filtering has become very popular. The most popular competition in recommender systems was held by Netflix. They launched the Netflix Prize in 2006 [1] and the winners of the competition used many approaches based primarily on the collaborative filtering. By now recommender systems are widespread and very popular among users in the middle of huge data quantities. The success of these solutions is indisputable and has irrevocably become part of our lives. In this project, we study the collaborative filtering method. The project report is split into three major chapters:

We start chapter one with a project overview that includes the problematic to resolve and present the viable solutions. Then we discuss the solution we opted for and determine the main objectives of our work. We end the chapter by presenting how we managed to organize the project.

In chapter two, we focus on the collaborative filtering approach, which includes both the memory based and the model based. We present the fundamentals about each one, and explain the logic behind these methods.

Lastly in chapter 3, we implement the algorithms using python language and link them to a web application to show the recommendations results. We will use the famous Django development framework. We compare the results obtained from all the models and datasets and give some personal interpretations.

Chapter I - General context of the project

1. Introduction

In this chapter, we introduce the project overview in which we define the problematic to resolve and present the viable solutions. Then we discuss the solution we opted for and determine the main objectives of our work. We end the chapter by presenting how we managed the organization of the project.

2. Project overview

2.1. Context

During the last few decades, with the rise of Netflix, Amazon, YouTube and many other such web services, recommender systems have become more important in our lives. From suggesting to a buyer many articles that could interest him (e-commerce), to suggesting the right contents and matching his preferences (online advertisement), recommender systems are today a necessary part of our daily online journeys. To have an idea on how impactful these systems became, it is estimated that 60% of watch time on YouTube comes from recommendations.

2.2. Definition of the problem

So how can we build a recommender system? And how do we evaluate metrically the recommendation results? Is it possible to determine which recommender system is best suited to a given field?

2.3. Recommender systems

A recommender system is an algorithm that tries to predict the rating or preference a user would give to an item [2]. Recommender systems determine similar items to the items you like (based on your history or on the preferences you filled), or finds users with similar taste and then serves up a recommendation from the items that these similar users liked. In a very general way, there are two major usable methods:

2.3.1. Content based method

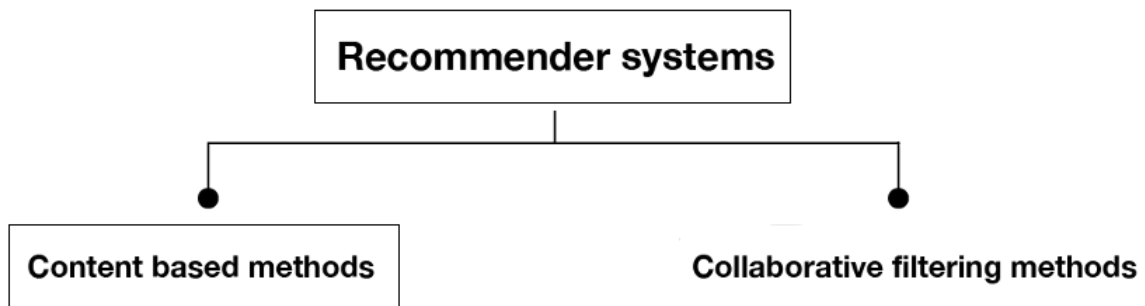


Figure 1. Content-based approach

According to Wikipedia, Content-based filtering methods are “based on a description of the item and a profile of the user's preferences. These methods are best suited to situations where there is known data on an item (name, location, description, etc.), but not on the user. Content-based recommenders treat recommendation as a user classification problem and learn a classifier for the user's likes and dislikes based on an item's” feature. [3]

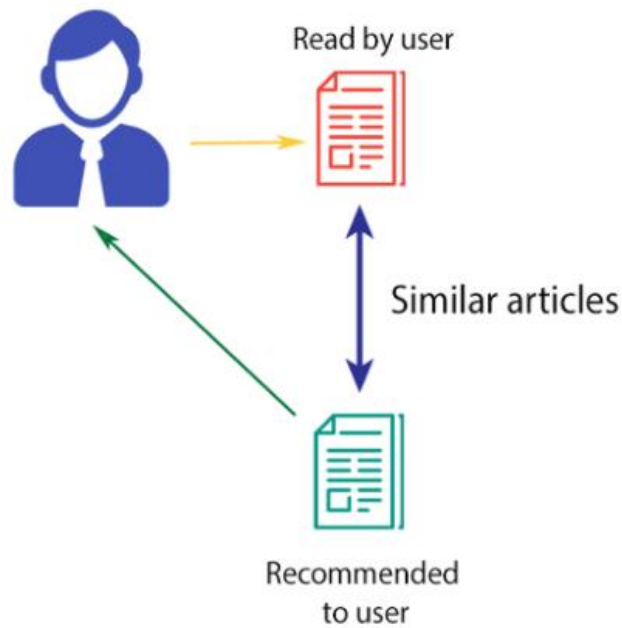


Figure 2. Content-based filtering

2.3.2. Collaborative filtering method

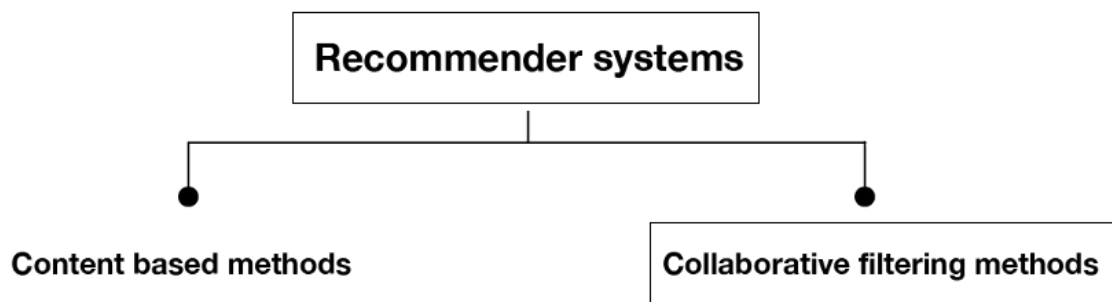


Figure 3. Collaborative filtering

Another widely used approach is the collaborative filtering one. Collaborative filtering is based on the assumption that people who liked the same items in the past will like the same items in the future. For example: some websites recommend new products saying “Customers who liked this also liked this”. [4]

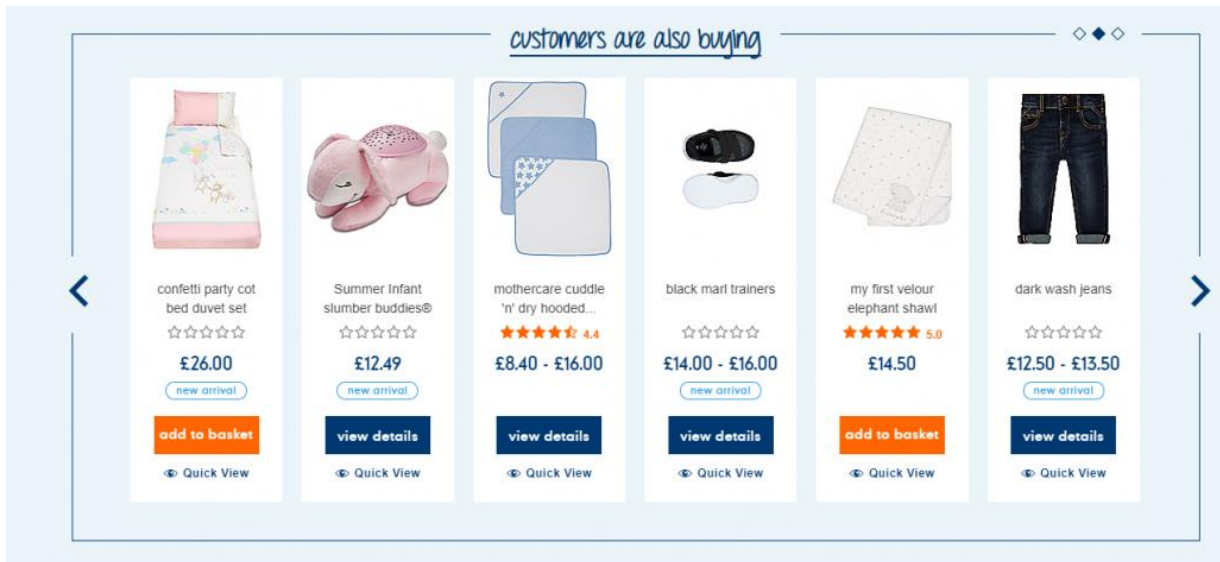


Figure 4. “Customers who liked this also liked this”

In fact, collaborative filtering addresses some of the limitations of content-based filtering, because it uses similarities between users and items at the same time to provide recommendations. It enables the users to get serendipitous recommendations; “occurring or discovered by chance in a happy or beneficial way” [5].

Furthermore, the similarities can be learned automatically, without relying on hand-engineering of features. This is a big advantage of the collaborative filtering approach; it does not rely on analyzing items content and therefore it is capable of recommending complex items like movies or books without requiring a background knowledge about the item itself. [6]

Collaborative filtering methods are classified as memory-based and model-based, which we will study in the next chapter.

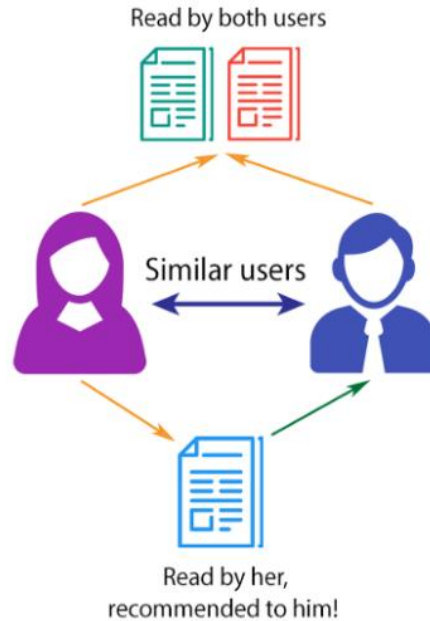


Figure 5. Collaborative filtering approach

2.3.3. Adopted method

In the content-based method, feature representation of items is hand-engineered to some extent. So, this technique requires a lot of domain knowledge. On the other hand, there is no need for domain knowledge in the collaborative filtering approach. Hence, we decided to go for it in our project. Furthermore, it has other perks; it works well even if the data is small. It has a very interesting advantage too; it helps the users to discover a new interest in a given item even if they never showed interest in it, and we can guess that because similar users are interested in that item. [7] [8]

3. Objectives of the project

During this project, our main aim is to achieve the following goals:

- Implementing the previously chosen solutions (Collaborative filtering techniques) into machine learning models
- Analyzing and comparing the results
- Creation of a web application whose back end is based on the implemented models

4. Project management

4.1. Tasks description

- Building the background knowledge:

Studying the machine learning and data mining basics and training on some well-known models

- Learning about recommender systems:

Understanding the idea behind their functioning and their different types

- Elaborating the algorithms:

Implementing the code using a new programming language

- Developing the web application:

Linking the back end (studied models) to the front end

- Reaction of the final year project report:

Writing of the report based on what was learned in the module “Projet tutoré” in this semester

4.2. Gantt chart

A Gantt chart is used in project management. As stated in Gantt’s website, “it is one of the most popular and useful ways of showing tasks displayed against time. On the left of the chart is a list of the activities. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity”. [9] This allows to see:

- The various tasks we realized
- When each activity began and ended
- How long each activity lasted
- The start and end date of the whole project

To summarize, the following Gantt chart shows the jury what was done in our project, with respect to time:

Task Name	Durati	Start	Finish	Prede
♣ Building background knowledge	21 days	28/02/22	21/03/22	
Discovery of artificial intelligence and machine learning	7 days	28/02/22	07/03/22	
Studying of machine learning basic algorithms	10 days	07/03/22	17/03/22	2
Learning basics of Python	2 days	17/03/22	19/03/22	3
Familiarization with Google Colab and Jupyter	2 days	19/03/22	21/03/22	4
♣ Learning about recommender systems	12 days	21/03/22	02/04/22	
Recommender Systems approaches	12 days	21/03/22	02/04/22	5
Singular Value Decomposition mathematical aspects	5 days	21/03/22	26/03/22	5
♣ Elaborating the algorithms	22 days	02/04/22	24/04/22	
Elaboration of some model based approaches	10 days	02/04/22	12/04/22	8;7
Elaboration of some memory based approaches	10 days	12/04/22	22/04/22	10
Testing multiple datasets	2 days	22/04/22	24/04/22	11;10
Crossvalidating results	1 day	22/04/22	23/04/22	10;11
Defining the use-case diagram	1 day	24/04/22	25/04/22	9
Developping the web application	16 days	24/04/22	10/05/22	9
♣ Preparing the report	16 days	24/04/22	10/05/22	9
Writing the report	11 days	24/04/22	05/05/22	
Designing the images	2 days	24/04/22	26/04/22	
Checking and validating	5 days	05/05/22	10/05/22	17
♣ Final steps	7 days	10/05/22	17/05/22	19
Reviewing the application	7 days	10/05/22	17/05/22	
Preparing the presentation	5 days	10/05/22	15/05/22	

Figure 6. Tasks against time

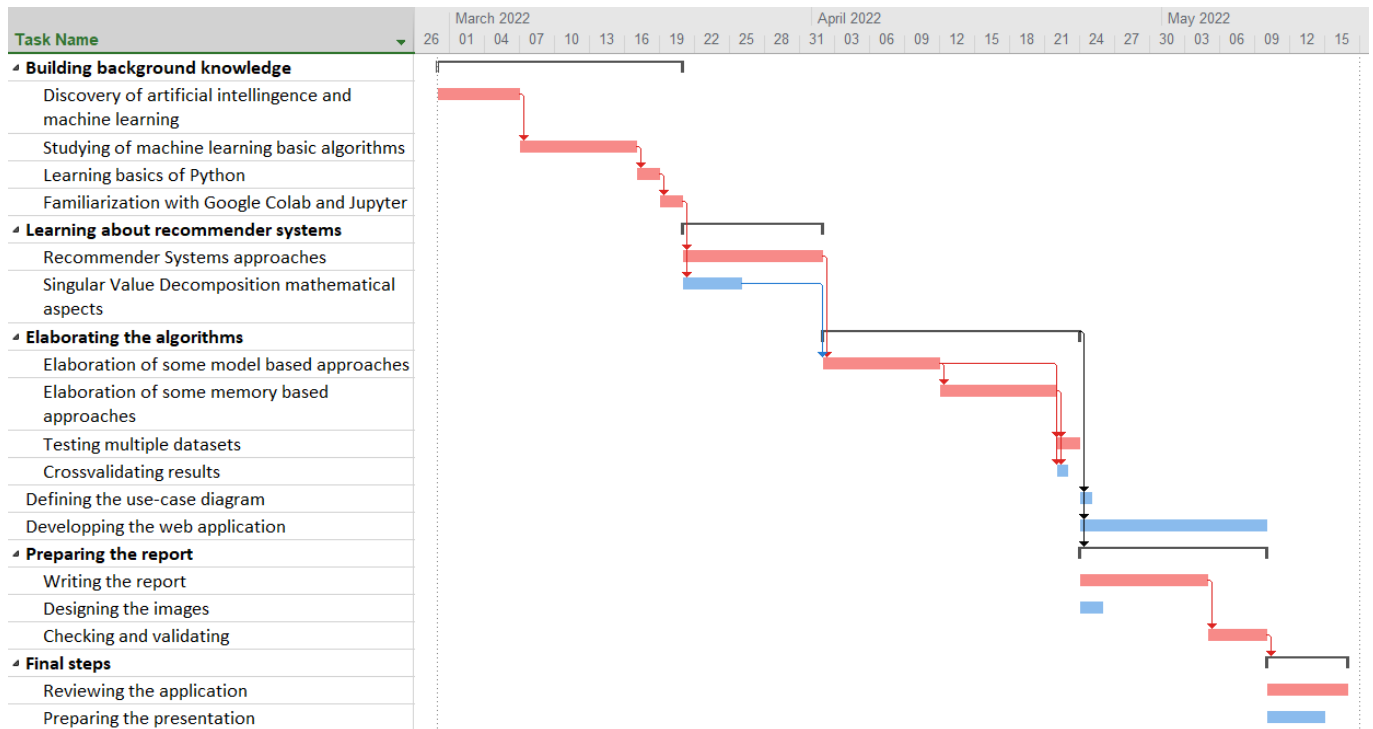


Figure 7. Gantt chart

5. Conclusion

We started this chapter by defining the problematic to resolve and presented the viable solutions. Then we discussed the solution we opted for and precised the main objectives of our work. We ended the chapter by presenting a Gantt chart that describes the tasks achieved during the project.

Chapter II - Collaborative filtering recommender systems

1. Introduction

As it has been presented, two families of Recommender Systems exist, the Content Based and the Collaborative Filtering approaches. In this project, we study Collaborative Filtering, including its different models and we present differences between them and show the different recommendation results. Collaborative Filtering algorithms that compute similarities among neighbors in-memory, are called Memory Based. In this approach, the dataset is loaded into memory and used directly to produce recommendations. On the other hand, there is the Model Based approach, in which a model is built and used to compute the recommendation. In this chapter, we study these two approaches separately and discuss the ways of validating the recommendations.

2. Data in use

A good way for users to express how much they liked or disliked a given item is by giving a numerical value (from 1 to 5 for instance). This integer scale is the basis for similarity measures. It gives a good interval to express taste compared to unary (good or purchased) or binary scale (like or dislike), as it gives more space. [10]

User	Item	Rating	User	Item	Rating	User	Item	Rating
196	242	3	1	1193	5	314	1	5
186	302	3	1	661	3	439	1	3
22	377	1	1	914	3	588	1	5
244	51	2	1	3408	4	1169	1	4
166	346	1	1	2355	5	1185	1	4
...
880	476	3	6040	1091	1	48386	10000	5
716	204	5	6040	1094	5	49007	10000	4
276	1090	1	6040	562	5	49383	10000	5
13	225	2	6040	1096	4	50124	10000	5
12	203	3	6040	1097	4	51328	10000	1
100000 rows × 2 columns			1000209 rows × 2 columns			981756 rows × 2 columns		

Figure 8. Examples of ratings datasets

We use this data to create user-item matrix, which is the most important input for our models. It contains users in rows, items in columns and ratings are the matrix elements.

Item	1	2	3	4	5	6	7	8	9	10	...	3943	3944	3945	3946	3947	3948	3949	3950	3951	3952
User																					
1	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...
6036	NaN	NaN	NaN	2.0	NaN	3.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6037	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6038	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6039	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6040	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6040 rows × 3706 columns																					

Figure 9. User-Item matrix (NaN are unknown ratings)

However, one of the problems faced with using ratings as a means of representing taste, is that each user has personal interpretation of the scale. While one user might give high marks to movies he enjoyed moderately, another user might give the high grades

to exceptional movies only. There is a widely used system to compensate for differences in the rating approach by users, it is called mean-centering. [10]

The mean-centering algorithm re-maps a user's rating by subtracting the mean value of all his ratings, effectively signaling if the particular rating is positive or negative when compared to the mean. Positive values represent above-average ratings, negative results represent below-average ratings, and zero represents an average rating. The formulation is given by:

$$h(r_{ui}) = r_{ui} - \bar{r}_u$$

(Eq. 1) Mean-centering (users)

where $h(r_{ui})$ represents the mean-centered rating of user u for item i , r_{ui} represents the actual rating of user u for item i , and \bar{r}_u represents the average rating of user u across all rated items by him.

$$h(r_{ui}) = r_{ui} - \bar{r}_i$$

(Eq. 2) Mean-centering (items)

3. Memory-based

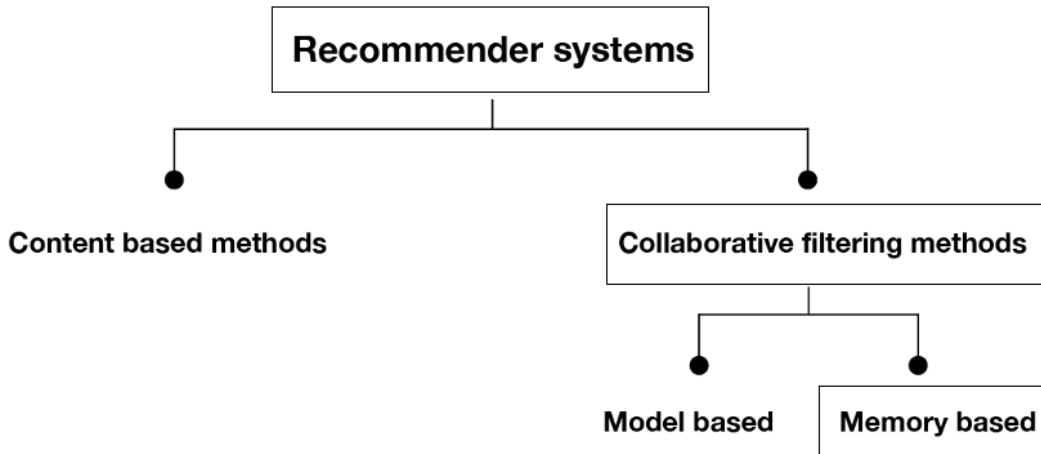


Figure 10. Memory based filtering

As reported by Wikipedia, the memory-based approach “uses user rating data to compute the similarity between users or items. Typical examples of this approach are

neighborhood-based CF and item-based/user-based top-N recommendations. For example, in user-based approaches, the value of ratings user u gives to item i is calculated as an aggregation of some similar users' rating of the item" [4]:

$$\hat{r}_{u,i} = \text{aggr}_{u' \in U} r_{u',i}$$

(Eq. 3) Estimation of rating value based on users

U denotes the set of top N users that are most similar to user u who rated item i .

3.1. User-based

Rating of the item is done using the rating of neighboring users. In simple words, it is based on the notion of users' similarity. An approximative algorithm for the standard user-based filtering looks as follows [10]:

1. "Input: User-Item Rating matrix R
2. Output: Recommendation lists of size l
3. Const l : Number of items to recommended to user u
4. Const v : Maximum number of users in $N(u)$, the neighbors of user u
5. For each user u Do
6. Set $N(u)$ to the v users most similar to user u
7. For each item i that user u has not rated Do
8. Combine ratings given to item i by neighbors $N_i(u)$
9. End
10. Recommend to user u the top- l items with the highest predicted rating $\hat{r}_{u,i}$
11. End" [10]

3.2. Item-based

The rating of the item is predicted using the user's own rating on neighboring items. In simple words, it is based on the notion of item similarity. An approximative algorithm for the standard item-based filtering looks as follows [10]:

1. "Input: User-Item Rating matrix R
2. Output: Recommendation lists of size l
3. Const l : Number of items to recommended to user u
4. Const j : Maximum number of items in $N(i)$, the neighbors of item i
5. For each item i Do
6. Set $N(i)$ to the j items most similar to item i

7. For each user u that has no rating for item i Do
8. Combine ratings of user u in neighbors $Nu(i)$
9. End
10. Recommend to user u the top- l items with the highest predicted rating $\hat{r}_{u,i}$
11. End” [10]

3.3. Similarity measuring

3.3.1. Cosine similarity (Vector similarity)

This way of comparing two things presents a user as a vector of ratings rated by himself and an item as a vector of ratings rated by the set of users. The cosine between two vectors representing two users (or items) indicates the similarity value between each other. A value near one represents a strong relation between the two variables. A value near 0 represents no relation (independent variables). The following formula presents how this similarity is calculated:

$$\cos similarity(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 \sum_{i \in I_v} r_{vi}^2}}$$

(Eq. 4) Cosine similarity (users) [11]

r_{ui} and r_{vi} are the ratings values on item i given by users u and v , respectively. I_u and I_v represent the sets of items rated by users u and v , respectively, and I_{uv} represents the set of items commonly rated by both u and v .

$$\cos similarity(i, j) = \frac{\sum_{u \in U_{ij}} r_{iu} r_{ju}}{\sqrt{\sum_{u \in U_i} r_{iu}^2 \sum_{u \in U_j} r_{ju}^2}}$$

(Eq. 5) Cosine similarity (items) [11]

r_{iu} and r_{ju} are the rating values assigned by the same user u on items i and j , respectively. U_i and U_j represent the sets of users who rated the items i and j , respectively, and U_{ij} represents the set of users who rated both items i and j .

3.3.2. MSD similarity

MSD is also used to compare differences between two things [12]. The mean squared deviation similarity is defined as:

$$msd(u, v) = \frac{1}{|I_{uv}|} \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2$$

(Eq. 6) Mean squared deviation (users) [12]

$msd(u, v)$ denotes the similarity of users u and v , across all items commonly rated. In the case of Item Based Recommender Systems, the formulation becomes:

$$msd(i, j) = \frac{1}{|U_{ij}|} \sum_{u \in U_{ij}} (r_{ui} - r_{uj})^2$$

(Eq. 7) Mean squared deviation (items) [12]

Here $msd(i, j)$ represents the similarity of items i and j , across all users that rated such items.

The MSD-similarity is then defined as:

$$msd \text{ similarity}(u, v) = \frac{1}{msd(u, v) + 1}$$

(Eq. 8) Mean squared deviation similarity (users) [11]

$$msd \text{ similarity}(i, j) = \frac{1}{msd(i, j) + 1}$$

(Eq. 9) Mean squared deviation similarity (items) [11]

The + 1 term is here to just avoid dividing by zero. [11]

3.3.3. Pearson Correlation Coefficient

PCC formula returns a value between -1 and 1, where: 1 indicates a strong positive correlation, -1 indicates a strong negative correlation and 0 indicates no correlation at all. The following formula calculates the similarity between two users u and v :

$$pearson\ similarity(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - u_u)(r_{vi} - u_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - u_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - u_v)^2}}$$

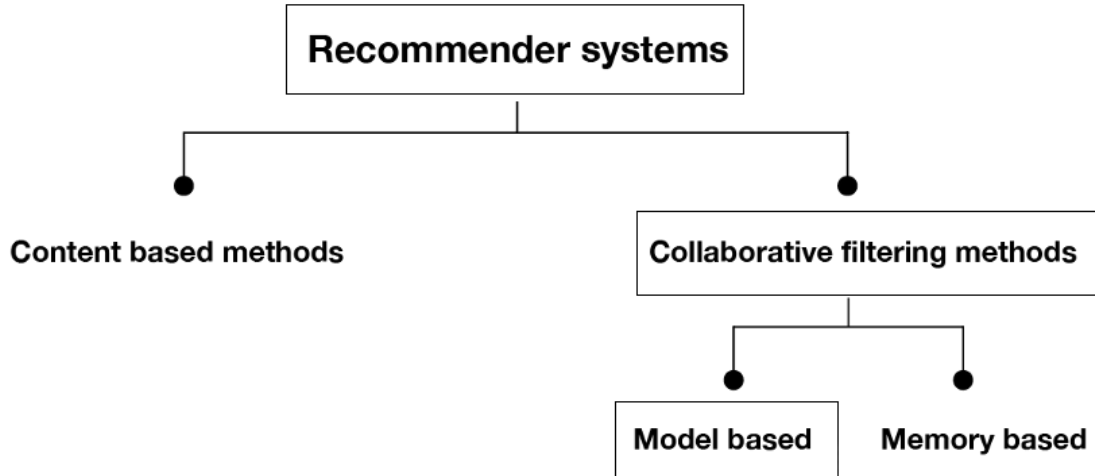
(Eq. 10) Pearson similarity (users) [11]

Where r_{ui} and r_{vi} are ratings of users u and v on the same item i . I_{uv} represents the set of items commonly rated by both u and v . r_u and r_v denote the average ratings of the users u and v on item i in I_{uv} , respectively. The same formula is used to calculate similarity between items:

$$pearson\ similarity(i, j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r}_i)^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r}_j)^2}}$$

(Eq. 11) Pearson similarity (items) [11]

Where r_{ui} and r_{uj} are ratings of user u on items i and j , respectively. U_{ij} represents the set of users who rated both items i and j ; \bar{r}_i and \bar{r}_j denote the average ratings on i and j in U_{ij} , respectively. Model-based

**Figure 11. Model based filtering**

Model-based recommender systems use machine learning algorithms to predict users rating of unrated items. There are many model-based CF algorithms, the most commonly used are matrix factorization models such as to applying a singular value decomposition or non-negative matrix factorization to reconstruct the rating matrix.

3.4. Matrix Factorization

Matrix factorization models map allow to find latent factors. User-item interactions are represented by inner products in that space. Accordingly, each item i is associated with a vector q_i , and each user u is associated with a vector p_u . It is stated in the work “Matrix factorization techniques for recommender systems” that “for a given item i , the elements of q_i measure the extent to which the item possesses those factors, positive or negative. For a given user u , the elements of p_u measure the extent of interest the user has in items that are high on the corresponding factors, again, positive or negative. The resulting dot product, $q_i^T p_u$ represents the interaction between user u and item i ; the user’s overall interest in the item’s characteristics in other words” [13]. This approximates user u ’s rating of item i , which is denoted by \hat{r}_{ui} :

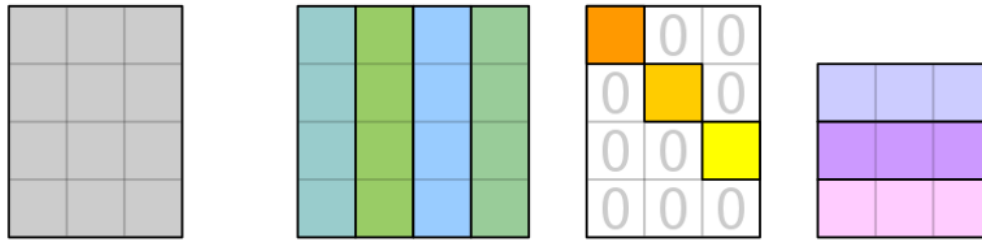
$$\hat{r}_{ui} = q_i^T p_u$$

(Eq. 12) Rating estimation

Here comes the big challenge, how do we calculate q_i for each item and p_u for each user. Such a model is closely related to singular value decomposition (SVD), a known technique for identifying latent semantic factors.

3.4.1. Singular Value Decomposition

In linear algebra, singular value decomposition (SVD) is a matrix factorization technique. This is a very interesting method, because it allows decomposing any existing matrix and its results can be interpreted.



$$\begin{matrix} \mathbf{M} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^* \\ m \times n & & m \times m & m \times n & n \times n \end{matrix}$$

Figure 12. SVD representation [14]

In fact, the columns (called singular vectors), represent latent features. This is extremely meaningful; in a movie dataset for example, applying SVD to the user-item matrix will produce vectors that group a set of features corresponding to the movie, it can be different movies genres for example.

However, our user-item matrix is missing most of its values. Conventional SVD is only defined when knowledge about the matrix is complete. So instead of using the pure SVD technique, we use machine learning to directly factorize the matrix based on the filled values only [13]. The system minimizes the regularized squared error on the set of known ratings:

$$\min_{p,q} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2$$

(Eq. 13) Function to minimize [15]

Here, K is the set of the (u, i) pairs for which r_{ui} is known (the training set). The system learns the model by minimizing the objective function. However, the goal is to build a model in order to predict unknown ratings. Hence, we should avoid overfitting the model by regularizing the learned parameters, whose magnitudes are penalized. “The constant λ controls the extent of regularization and is determined by cross-validation” [13].

$$\min_{p,q} \sum_{(u,i) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

(Eq. 14) Function to minimize with regularizing terms [15]

3.5. Stochastic gradient descent

Stochastic gradient descent can be regarded as an approximation of gradient descent optimization. But what is the gradient descent?

Gradient descent is “an iterative optimization algorithm for finding the local minimum of a function. To find the local minimum of a function using gradient descent, we must take steps proportional to the negative of the gradient (move away from the

gradient) of the function at the current point” [16]. However, computations to complete the algorithm are so numerous. Hence gradient descent is so slow on huge data.

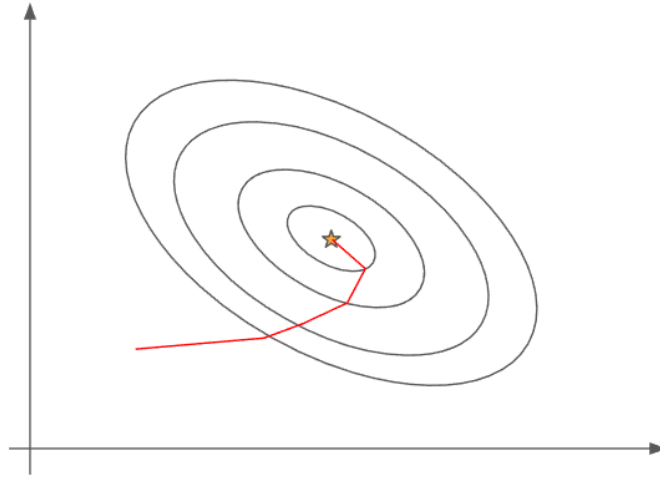


Figure 13. GD schematization

Stochastic gradient descent solves the problem because it replaces the actual gradient (calculated from the entire data set) by an estimated gradient (calculated from a randomly selected subset of the data). Especially in high-dimensional optimization problems this reduces the very high computational burden, achieving faster iterations in trade for a lower convergence rate. [16]

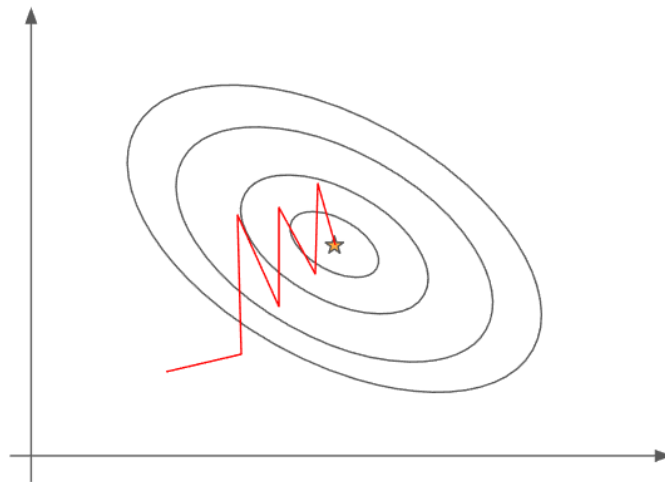


Figure 14. SGD schematization

4. Evaluation metrics

An evaluation metric tries to measure the performance of the algorithm. It quantifies how the estimations produced by the strategy deviate from their known values. The

smaller the error, the better the Recommender System works. Two types of Accuracy measures are in wide spread use; they are the Mean Absolute Error (MAE), and the Root Mean Squared Error (RMSE).

4.1. Mean absolute error

It is simply the average absolute difference between actual ratings and predicted ratings. Each error contributes to MAE in proportion to the absolute value of the error, meaning that the size of the error is the same at the high and the low ends of the similarity. [17]

$$MAE = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

(Eq. 15) Mean absolute error

4.2. Root mean squared error

According to Wikipedia, RMSE is “the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error; thus, larger errors have a disproportionately large effect on RMSD” [18]. Consequently, in contrast to MAE, the RMSE penalizes more the larger errors, which makes sense in the context of Recommender Systems.

$$RMSE = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

(Eq. 16) Root mean squared error

5. Conclusion

In collaborative filtering, multiple models are used in order to calculate predictions for recommender systems. Memory based algorithms calculate it directly from data, while model-based algorithms begin by creating a model, then reconstruct user-item matrix to calculate the predictions. Model based algorithms take substantially longer to compute and needs to be computed anew if the matrix of data changes, which happens every time a new user enters a rating. So generally, the model needs to be re-trained.

Memory Based approaches do not suffer from this problem because the similarity is computed every single time a recommendation is sought.

Chapter III – Web Application

“SmartRecommender”

1. Introduction

Recently, Python became one of the most popular programming languages for the task of machine learning, and it has replaced many languages in the industry. One of its big advantages is its vast collection of libraries (Scikit-learn, TensorFlow, Numpy...).

In this chapter, the major tools used during our project will be briefly presented. The codes' implementation will be explained step by step. And we will finish by presenting a web application “SmartRecommender” that displays the results, so that users will be able to note the difference in recommendation between the different models.

2. Tools



Python is a high-level, interpreted, general-purpose programming language. Python's large libraries system, commonly cited as one of its greatest strengths, provides tools suited to many tasks, including machine learning used in our project.



Surprise is a Python scikit based library, used for building and analyzing recommender systems. It is perfectly suited for our project.



Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.



NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.



Jupyter Notebook is an interactive computational environment for creating notebook documents. It enables the execution of code cells separately.



Django is a Python-based free and open-source web framework that follows the model–template–views architectural pattern. We used it during the creation of our web application.



Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and JavaScript-based design templates.

3. Models building

3.1. Collecting data

When searching for datasets that respect the user item rating architecture, all of our findings were movies or books related datasets. For instance, people vote a music track by liking it or disliking it, not by giving it a rating. Hence, we couldn't find datasets related to different themes (music...), so we picked multiple movie datasets (100K, 1M, 10M) and a books dataset. In the following steps, we show the results for the MovieLens-1M dataset.

3.2. Preparing data

3.2.1. Importing

We start by importing ratings data with the help of Surprise library importer. This data is sufficient to make predictions, but we import items data too to get movies names and genres for the display. We used Pandas importer for it.

```
# Importing data

# Ratings data
file_path = os.path.expanduser('Datasets/ml-100k/u.data')
reader = Reader(line_format='user item rating timestamp', sep='\t')
data = Dataset.load_from_file(file_path, reader=reader)

# Movies data
movies_data_columns = ['movieId', 'title']
movies_data = pd.read_csv('Datasets/ml-100k/u.item', sep='|', names = movies_data_columns,
                          usecols = range(2), encoding = "ISO-8859-1", engine = 'python')
movies_data = movies_data.set_index('movieId')
```

3.2.2. Visualizing

```
data.head()
```

userId	movieId	ratings
196	242	3
186	302	3
22	377	1
244	51	2
166	346	1

```
movies_data.head()
```

movieId	title
1	Toy Story (1995)
2	GoldenEye (1995)
3	Four Rooms (1995)
4	Get Shorty (1995)
5	Copycat (1995)

3.2.3. Splitting ratings data

We used the standard ratio: 80% for training and 20% for testing. The training set is the set that our model will learn from. The testing set will be used to check the accuracy of our model after training.

```
# Splitting data

# test set is made of 20% of the ratings.
trainset, testset = train_test_split(data, test_size=.20)
```

3.3. Choosing the algorithm

We are basically using 2 algorithms from surprise library to implement the 3 models discussed in chapter 2:

3.3.1. SVD

For the model based, we went for the famous SVD algorithm as popularized by Simon Funk during the Netflix Prize. We have to pick the values for the parameters [15]:

N: number of features (embeddings). Generally, values around 50 are used.

Regularization term: Regularization is a process that changes the result answer to be "simpler". It is used to prevent overfitting.

Learning rate: it determines the step size at each iteration while moving toward a minimum of the function. It metaphorically represents the speed at which a machine learning model "learns".

Number of iterations: The number of iterations of the SGD procedure.

```
# Picking the algorithm

# We'll use the famous SVD algorithm.
algo = SVD(n_factors=50,
          reg_all=0.02,
          lr_all=0.005,
          n_epochs=20)
```

3.3.2. KNN inspired algorithm

This is an algorithm directly derived from a basic nearest neighbors approach. [19]

K: It is the number of neighbors to take into account for aggregation. Values around 40 are fine.

Similarity options: We just have to select the options for the similarity measure (Cosine similarity, MSD similarity or Pearson similarity)

3.3.2.1. KNN User based

The similarities configuration is set first. Here we pick cosine similarity measuring. Then we precise if it's user based or item based by setting a boolean value.

```
# Picking the algorithm
configuration = {'name': 'cosine',
                'user_based': True
                }

# The KNN algorithm
algo = KNNBasic(k=40,
               sim_options=configuration)
```

3.3.2.2. KNN Item based

Same work done in the user based, except we set 'user_based' to False. We also change the similarity measuring to MSD for the example.

```
# Picking the algorithm
configuration = {'name': 'MSD',
                'user_based': False
                }

# The KNN algorithm
algo = KNNBasic(k=40,
               sim_options=configuration)
```

3.4. Training the model

“Training is the most important step in machine learning. We just have to pass the prepared data to our machine learning model to find patterns and make predictions. It results in the model learning from the data so that it can accomplish the task set. Over time, with training, the model gets better at predicting.” [20]

```
# Training the algorithm on the trainset

algo.fit(trainset)
```

3.5. Evaluating the model

“After training the model, we check it to see how it’s performing. This is done by testing the performance of the model on previously unseen data. The unseen data is the testing set that we split our data into earlier” [20]. Hence, we calculate predictions for the test set.

```
# Predicting ratings for the testset

predictions = algo.test(testset)
```

Then we calculate the RMSE and MAE for the predictions made on the test set.

```
# Computing accuracy

accuracy_rmse = accuracy.rmse(predictions)
accuracy_mae = accuracy.mae(predictions)

RMSE: 0.9800
MAE: 0.7713
```

3.6. Making Predictions

In order to make predictions for a specific user and item, we just use the function `predict`. For instance, the user 196 has rated 3 the item 242. The prediction is 3,56.

```
# getting a prediction for a specific user and item

pred196_242 = algo.predict('196', '242', 3)
pred196_242

Prediction(uid='196', iid='242', r_ui=3, est=3.564471632913397)
```

This is used to make top recommendations for a user, we just predict ratings for all the unrated items using the function `predict`, then we sort the results and return the items with the highest scores.

3.7. Example

We pick randomly user with `Id=5`. We display its ten top rated movies he watched with the function `topRated`.

```
# Displaying rated items for a user

def topRated(user,merged):

    return merged.loc[user].sort_values('rating',ascending=False).reset_index(drop=True).head(10)

topRated(3,merged)
```

id	rating	title	genre
2355	5	Bug's Life, A (1998)	Animation Children's Comedy
1304	5	Butch Cassidy and the Sundance Kid (1969)	Action Comedy Western
1615	5	Edge, The (1997)	Adventure Thriller
2167	5	Blade (1998)	Action Adventure Horror
1266	5	Unforgiven (1992)	Western
733	5	Rock, The (1996)	Action Adventure Thriller
1198	5	Raiders of the Lost Ark (1981)	Action Adventure
1378	5	Young Guns (1988)	Action Comedy Western
1197	5	Princess Bride, The (1987)	Action Adventure Comedy Romance
3552	5	Caddyshack (1980)	Comedy

It is obvious that this user enjoys action, adventure and western movies. Same goes for recommended movies, they are mostly action and adventure movies, which makes sense.

id	title	genre
110	Braveheart (1995)	Action Drama War
1221	Godfather: Part II, The (1974)	Action Crime Drama
858	Godfather, The (1972)	Action Crime Drama
1204	Lawrence of Arabia (1962)	Adventure War
3470	Dersu Uzala (1974)	Adventure Drama
3196	Stalag 17 (1953)	Drama War
2019	Seven Samurai (The Magnificent Seven) (Shichin...	Action Drama
2905	Sanjuro (1962)	Action Adventure
670	World of Apu, The (Apu Sansar) (1959)	Drama
1207	To Kill a Mockingbird (1962)	Drama

4. SmartRecommender web application

4.1. Description

We were tasked with realizing a web application that presents the previously achieved work. It gives the user the possibility of selecting different prediction approaches, using the collaborative filtering with 2 different datasets, one for movies and the other for books.

We used Django, so we respected its model-view-template architecture. It is a software design pattern; a collection of three important components Model View and Template. [21]

- The Model helps to handle database. It is a data access layer which handles the data.
- The View is the user interface, it renders a template
- The Template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

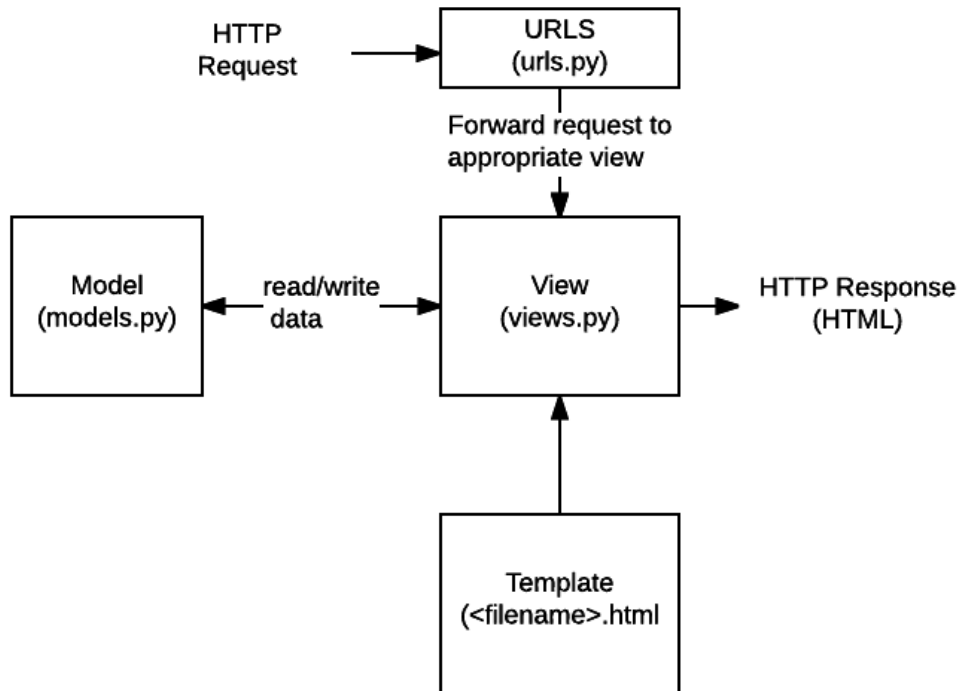


Figure 15. Django MVT architecture

4.2. Use case diagram

In the Unified Modeling Language, a use case diagram can summarize the details of actors and their interactions with the system.

A use case diagram doesn't go into details. Instead, a proper diagram depicts a high-level overview of the relationship between use cases and actors.

Concretely, our web application has one actor which is the webapp user with 3 use cases;

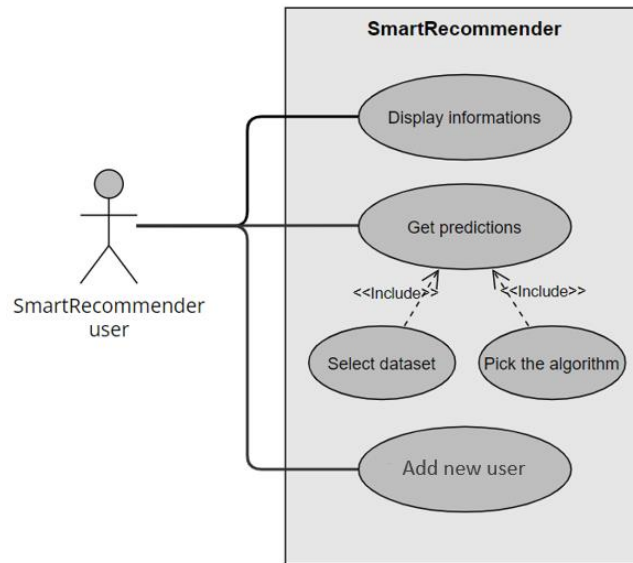


Figure 16. Use case diagram

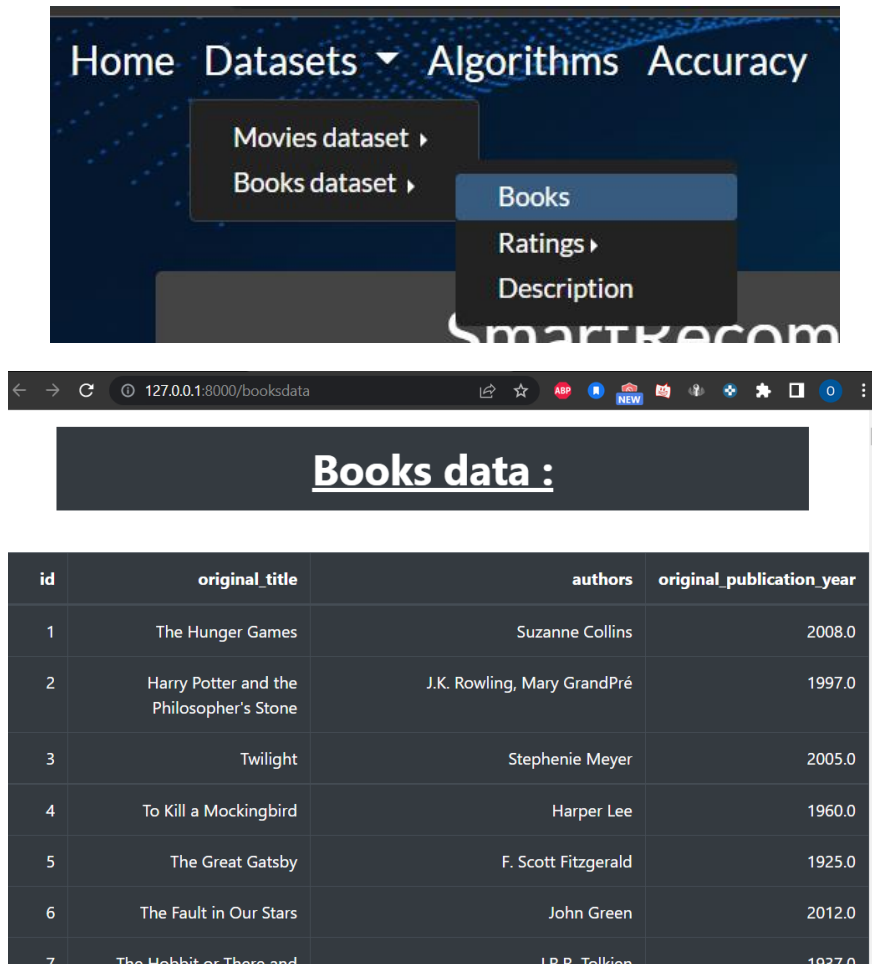
4.3. SmartRecommender browsing

The screenshot shows the SmartRecommender web application interface. The browser address bar displays the URL: `C:/Users/intel/PycharmProjects/pandasprojectv3/app-home/index...`. The page has a dark blue background with a network graph pattern. At the top, there is a navigation bar with links: 'Home', 'Datasets', and 'About'. The main content area features a large dark grey box with the text 'SmartRecommender'. Below this, there is a form with several input fields and buttons. The form includes three dropdown menus for selecting the dataset ('Books dataset'), the algorithm ('MSD'), and the choice for the KNN algorithm ('Item based'). There are also text input fields for 'Enter the user Id:', 'Enter the item Id:', and 'Enter the number of recommendations:'. At the bottom of the form, there are four buttons: 'Calculate rating', 'Get recommendations', 'Check the accuracy', and 'Add new user'.

4.3.1. Display data

Example 1:

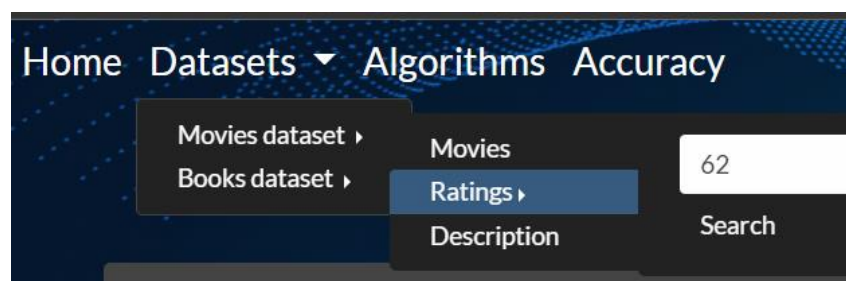
Books data:



id	original_title	authors	original_publication_year
1	The Hunger Games	Suzanne Collins	2008.0
2	Harry Potter and the Philosopher's Stone	J.K. Rowling, Mary GrandPré	1997.0
3	Twilight	Stephenie Meyer	2005.0
4	To Kill a Mockingbird	Harper Lee	1960.0
5	The Great Gatsby	F. Scott Fitzgerald	1925.0
6	The Fault in Our Stars	John Green	2012.0
7	The Hobbit or There and	J.R.R. Tolkien	1937.0

Example 2:

Ratings of user 62 in the movies' dataset:



Ratings :

user	item	rating
62	3789	5
62	2987	4
62	1249	3
62	1177	3
62	2124	4

4.3.2. Calculate ratings

First, the user has to select the wanted options.

Select the dataset :

Movies dataset
Movies dataset
Books dataset

Select the algorithm :

SVD
SVD
KNN
Cosine
MSD
Pearson

Then he has to enter the user Id and the item Id and hit

Calculate rating

Select the dataset :

Movies dataset

Select the algorithm :

MSD

Select the choice for the KNN algorithm:

Item based

Enter the user Id :

3

Enter the item Id :

5

In this example, we selected movies dataset, the KNN (Item based) algorithm with MSD similarity for user 3 and item 5.

The result is:

Prediction for the user 3 and the movie 5:

User ID	3
Item ID	5
title	Father of the Bride Part II (1995)
predicted rating	3.856086441030154

4.3.3. Get recommendations

If instead, the user wants to calculate top recommendations for a user, all he has to do is enter the number of recommendations after selecting the options then hit the button

Get recommendations

Enter the number of recommendations:

5

The result is:

Top 5 recommendations for the user 3 :

Manny & Lo (1996)
Men With Guns (1997)
Flying Tigers (1942)
Independence Day (ID4) (1996)
Puppet Master (1989)

4.3.4. Check the accuracy

To check the accuracy, the user has to click on

Check the accuracy

 after having selected the right dataset and algorithm.

Example: Books dataset with KNN and Pearson similarity

Accuracy of the algorithm :

RMSE

0.909863738517196

MAE

0.6684053861064472

4.3.5. Add new user

The user is assigned a new Id.

New user Id is: 6041

ratings	id	title	
add rating	1	Toy Story (1995)	Animation Children's Comedy
add rating	2	Jumanji (1995)	Adventure Children's Fantasy
add rating	3	Grumpier Old Men (1995)	Comedy Romance
add rating	4	Waiting to Exhale (1995)	Comedy Drama
add rating	5	Father of the Bride Part II (1995)	Comedy

He can then add ratings to the items by clicking on “add rating”:

add rating	103	Unforgettabl
add rating	104	Happy Gilme
add rating	105	Bridges of M

Then he rates and submits the rating:



When checking the dataset, we can see that the new user was added with the picked ratings:

Ratings :

user	item	rating
6041	6	5
6041	16	5
6041	47	5
6041	104	1

Now we can recommend for this new user just like the users already stored in the dataset by using his new Id: 6041.

5. Results and discussion

5.1. Results

We can change the parameters of an algorithm or the entire algorithm to see how it affects the accuracy. We did this using a class named GridSearchCV, it tries all the combinations of parameters and reports the best parameters for the accuracy measure.

5.1.1. SVD

Parameters tuned:

```
param_grid = { 'n_factors' : [30,60,80,100],
               'reg_all' : [0.02,0.03,0.04],
               'n_epochs': [10,20,30],
               'lr_all': [0.004,0.005,0.006]
               }
```

Result:

	100K	1M	Books	10M
N	60	100	30	80
Regularization term	0.03	0.03	0.02	0.03
Iterations number	30	20	20	20
RMSE	0.9355	0.8664	0.8396	0.8001
MAE	0.7398	0.6820	0.6563	0.6149

Table 1. Best parameters for SVD

5.1.2. KNN (User based)

Parameters tuned:

```
param_grid = {'k': [20,40,60,80],
              'sim_options': {'name': ['msd', 'cosine', 'Pearson'],
                              'user_based': [True]}
              }
```

Result:

	100K	1M	Books	10M
K	20	60		
Similarity	MSD	MSD	Insufficient	Insufficient
RMSE	0.9780	0.9234	memory	memory
MAE	0.7723	0.7286		

Table 2. Best parameters for KNN (user based)

5.1.3. KNN (Item based)

Parameters tuned:

```
param_grid = {'k': [30,40,50,60,90],
              'sim_options': {'name': ['msd', 'cosine', 'Pearson'],
                              'user_based': [False]}
              }
```

Result:

	100K	1M	Books	10M
K	40	30	90	40
Similarity	MSD	MSD	cosine	MSD
RMSE	0.9719	0.9142	0.8887	0.8591
MAE	0.7699	0.7200	0.6706	0.6591

Table 3. Best parameters for KNN (item based)

5.2. Discussion

If we put accuracy results aside, we can easily observe that recommendations change from one model to the other. Even with close RMSE scores, the results of recommendations according to each algorithm are very different. Hence, we can never decide that one model is better than the other, because one model might have a higher RMSE score but give more diversified recommendations for example, and if a user is looking for new types of items, this might suit him better. And vice-versa, maybe another user is only interested in one particular type of items, so this model doesn't suit him. This is somehow close to the serendipity principle, that we mentioned previously when comparing content based to collaborative filtering but we are only using it as an example here. Another example could be recommending popular and unpopular items. As a conclusion, it is not sufficient to compare accuracy scores to deduce the better model, as it is relative to the user's view and intention.

6. Conclusion

We started this chapter by presenting the tools that were used during our project. There is no need to say that nowadays, many instruments are available to work with. From machine learning libraries to suitable working environments, everything is accessible online.

We proposed a way of making recommendations based on the collaborative filtering principle, using python language and its adapted libraries. We compared the results of each model.

Furthermore, we created a web application using Django that displays the recommendations results.

General conclusion and future work

1.1. Conclusion

This work proposed 3 different approaches aimed at building recommender systems. These approaches were tested by modifying different parameters. We then created a simple application using Django to implement and visualize the results.

In these tests, we used the well-researched MovieLens datasets; 100k, 1m, 10m and Goodreads-books dataset.

Concerning the memory-based approaches, we modified K and the similarity modules. For the model-based approach, we modified N (number of features), the learning rate, regularization term and number of iterations. Depending on the datasets, we had different results.

But overall, the model based outperformed the memory based. Nevertheless, it was more time and resource consuming.

In our web application, the user can pick the dataset and the algorithm he wants to test with, so that he can visualize the different results given by each model.

1.2. Future work

For a new user or item, there isn't enough data to make accurate recommendations. A good way of avoiding this, is to implement a content-based model that doesn't suffer from the cold start problem, and to use this model in recommendations until the user has rated enough items.

Another way of improving our application is by transforming the datasets into database tables, so that users can register permanently on the web application and have

their own profile. This was our idea at the start of the application implementation but due to the lack of time, we weren't able to achieve it.

Bibliography

- [1] A. Sandor, "History of recommender systems," [Online]. Available: <https://www.onespire.net/news/history-of-recommender-systems/#:~:text=The%20basics%20of%20recommender%20systems,textual%20content%20with%20each%20other..>
- [2] Wikipedia, "Recommender system," [Online]. Available: https://en.wikipedia.org/wiki/Recommender_system.
- [3] F. Isinkaye, Y. Folajimi and B. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, 2015.
- [4] Wikipedia, "Collaborative filtering," [Online]. Available: https://en.wikipedia.org/wiki/Collaborative_filtering.
- [5] Lexico, "Meaning & definition," Oxford. [Online].
- [6] P. Melville, R. Mooney and R. Nagarajan, "Content-Boosted Collaborative Filtering for Improved Recommendations," Department of Computer Sciences University of Texas, Texas, July 2002.
- [7] Developers google, "Collaborative Filtering Advantages & Disadvantages," Google, [Online]. Available: <https://developers.google.com/machine-learning/recommendation/collaborative/summary>.
- [8] Developers google, "Content-based Filtering Advantages & Disadvantages," Google, [Online]. Available: <https://developers.google.com/machine-learning/recommendation/content-based/summary>.
- [9] Gantt.com, "What is a Gantt Chart?," [Online]. Available: <https://www.gantt.com/>.
- [10] C. Levinas, "An analysis of memory based collaborative filtering recommender systems with improvement proposales," September 2014.

- [11] Surprise documentation, "Similarities module," readthedocs, [Online]. Available: <https://surprise.readthedocs.io/en/stable/similarities.html>.
- [12] Wikipedia, "Mean squared displacement," [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_displacement.
- [13] K. Yehuda, B. Robert and V. Chris, "Matrix factorization techniques for recommender systems," IEEE Computer Society, August 2009.
- [14] Wikipedia, "Singular value decomposition," [Online]. Available: https://en.wikipedia.org/wiki/Singular_value_decomposition.
- [15] Surprise Documentation, "Matrix Factorization-based algorithms," 2015. [Online]. Available: https://surprise.readthedocs.io/en/stable/matrix_factorization.html.
- [16] A. Srinivasan, "Stochastic Gradient Descent — Clearly Explained," September 2019. [Online]. Available: <https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31>.
- [17] Wikipedia, "Mean absolute error," [Online]. Available: https://en.wikipedia.org/wiki/Mean_absolute_error.
- [18] Wikipedia, "Root-mean-square deviation," [Online]. Available: https://en.wikipedia.org/wiki/Root-mean-square_deviation.
- [19] Surprise Documentation, "K-NN inspired algorithms," 2015. [Online]. Available: https://surprise.readthedocs.io/en/stable/knn_inspired.html.
- [20] simplilearn, "The Complete Guide to Machine Learning Steps," [Online]. Available: <https://www.simplilearn.com/tutorials/machine-learning-tutorial/machine-learning-steps>.
- [21] Javatpoint, "Django MVT," [Online]. Available: <https://www.javatpoint.com/django-mvt>.