Weighting Strategies for a Recommender System Using Item Clustering Based on Genres[⋆]

Sébastien Frémal[a], Fabian Lecron[a,*]

[a] *University of Mons, Faculty of Engineering, 20 place du parc, 7000 Mons, Belgium*

Abstract

Recommender Systems are effective to identify items that could interest clients on e-commerce web sites or predict evaluations that people could give to items such as movies. In this context, clustering can be used to improve predictions or to reduce computational time. In this paper, we present a clustering approach based on item metadata informations. Evaluations are clustered according to item genre. As items can have several genres, evaluations can be placed in several clusters. Each cluster provides its own rating prediction and weighting strategies are then used to combine these results in one evaluation. Coupled with an existing collaborative filtering recommender system and applied on Yahoo! and MovieLens datasets, our method improves the MAE between 0.3 and 1.8%, and the RMSE between 4.7 and 9.8%.

Keywords: Recommender System, Clustering, Weighting Strategies

1. Introduction

Recommender Systems are powerful tools to predict which products will interest clients according to their habits and preferences (Leskovec et al., 2014; Adomavicius & Tuzhilin, 2005; Breese et al., 1998). These systems are mainly used on e-commerce websites as these sites generate huge data volumes. For example, in 2015, the Amazon catalog provided 480 million distinct products and presented an average growth of 485 000 new products per day[1]. Each month, the website has veen visited in average by 188 million users[2]. These users create data either implicitly by browsing item pages or buying products, either explicitly by creating desired item lists or by evaluating items. With so

[*] Corresponding author. Tel.: +32 65 37 40 67

Email addresses: `sebastien.fremal@umons.ac.be` (Sébastien Frémal), `fabian.lecron@umons.ac.be` (Fabian Lecron)

[1] https://export-x.com/2013/05/01/how-big-is-amazon/ Accessed 29.09.16

[2] http://www.statista.com/statistics/271450/monthly-unique-visitors-to-us-retail-websites/ Accessed 29.09.16

many items and without a good recommender system, clients could miss items that would really interest them, leading to a shortfall for companies. Therefore, expert systems are needed to analyze all these data and infer predictions.

Many researches have been led on recommender systems since the 90s. The first objective was to build systems able to predict evaluations or to recommend items. Then, a second objective was to reduce the computational time for providing recommendations. To do so, a much explored method is data clustering (Kuzelewska, 2014; DuBois et al., 2009; Sarwar et al., 2002; Altingovde et al., 2013; Sarwar et al., 2001; Rashid et al., 2006). As data are really disparate, similar items or users are clustered together to reduce the dataset on which recommendation algorithms are applied. First clustering algorithms were designed only to reduce the computational time without precision loss, but some are now used to improve recommendations (Selamat & Moghaddam, 2012; Wei et al., 2012; Pitsilis et al., 2011; Quan et al., 2006; Gong, 2010; Ma et al., 2014; Pham et al., 2011; Banerjee et al., 2007; Xue et al., 2005; O'Donovan & Smyth, 2005; Kohrs & Merialdo, 1999; Nathanson et al., 2007). Clustering can refine data to keep only useful informations and therefore avoid noising effect. Many efficient clustering algorithms have already been investigated, but as they become more complex, they also become more time-consuming.

Our research is focused on implementing a clustering approach based on metadata informations which does not require much computational time and which improves prediction results. Some sophisticated website databases classify their products, also known as *items*, in different categories. For example, IMDb [3] and MovieLens (Harper & Konstan, 2015) classify their movies in about twenty categories: Adventure, Romance, Fantasy, Children... A movie can belong to several categories such as Fantasy/Action/Children. As people have generally a tendency to prefer some genres above others, it is judicious to cluster items, i.e. movies, according to their categories. Unlike other clustering techniques, an item can belong to several clusters and have different evaluations. Therefore, there is a need to reduce these opinions in one final evaluation. Several weighting strategies have been experimented to get the most accurate evaluations.

The rest of the paper is organized as follows :

- Section 2 introduces the problem and presents related works concerning Collaborative Filtering and clustering methods.

- Section 3 describes our contributions. Firstly, it details our clustering method based on item genre. Secondly, it presents our weighting strategies used to combine clusters evaluations and provide predictions.

- Section 4 describes the evaluation environment and protocol. It provides experiment results demonstrating the utility of our weighting strategies. Our contributions improve the MAE between 0.3 and 1,8%, and the RMSE between 4.7 and 9.8%.

[3] IMDB. Internet Movie DataBase. (1990) http://www.imdb.com/ Accessed 29.06.16

- Section 5 presents our conclusions and our future works.

2. Contributions and Related Works

Recommender systems are used to help identifying items that should interest clients (Leskovec et al., 2014). Nowadays, it is mostly used in the field of web applications where data regarding products and clients are generated everyday. These data are used by recommender systems to identify similarities between products or consumers and then to extrapolate client preferences. Most standard systems consist of a set of users $U = \{u_1, u_2, ..., u_m\}$, a set of items $I = \{i_1, i_2, ..., i_n\}$ and a set of evaluations. Among possible evaluations, there are ratings given by users. A rating given by a user i for an item j is denoted $R(i, j)$. Ratings constitute a $m \times n$ matrix R, called *utility matrix*. This matrix is used to extrapolate data of two kinds (Selamat & Moghaddam, 2012):

1. Prediction: the objective is to compute an unknown element $R(i, j)$ of the utility matrix, R, to propose an evaluation of the j^{th} item to the i^{th} user.
2. Recommendation: the objective is to compute a list of the most interesting, and not yet purchased, products for a user.

There are different ways for recommender systems to achieve these objectives. These systems are classified in three broad groups (Adomavicius & Tuzhilin, 2005):

1. Content-based systems: it exploits item properties to find items similar to those previously liked by the active user. For example, if a user watched jazz music video on Youtube, the website will recommend videos tagged as *music* and *jazz*.
2. Collaborative Filtering (CF): it exploits similarities between users and/or items to extrapolate new utility measures (e.g. ratings). These systems can be of two types (Breese et al., 1998):
   (a) Memory-based: it uses previous ratings or user interactions to anticipate future behaviour.
   (b) Model-based: a model is computed based on available data and is then used to generate new data.
3. Hybrid approaches: it exploits a combination of content-based and CF methods.

As CF is a popular and efficient strategy (Linden et al., 2003; Balabanović & Shoham, 1997; Cöster & Svensson, 2002), our research relies on these systems. A model-based example of CF is based on the Singular Value Decomposition (SVD) model. SVD is a really efficient model which has already proved its worth in ratings prediction (Sarwar et al., 2000; Paterek, 2007; Lee et al., 2014). The idea is to find two matrices V, of size $m \times f$, and D, of size $f \times n$, such as $V \times D = R$, with f a factor of dimensionality chosen by the user. This equation is verified on known values of R, the utility matrix. It is then supposed that if the equation is true for known values, then it will be near of the truth for unknown values. The rating prediction of the j^{th} item by the i^{th} user is then computed by multiplying the i^{th} row of V by the j^{th} column of D. The open-source recommender system library MyMediaLite (Gantner et al., 2011) is used in our research to achieve this matrix factorization. To compute V and D, the library begins by initializing these two matrices with normally distributed random numbers. Then, for each known value, the error is computed with the formulas $E(i,j) = R(i,j) - D_j^T V_i$, with V_i, the i^{th} row of V, and D_j, the j^{th} column of D. These errors are used to correct V and D matrices according to the stochastic gradient descent method (Koren et al., 2009). Once V and D matrices are computed for each cluster, clusters collaborate to predict evaluations. Firstly, a prediction is computed for each active genres (genres at which the active item belongs to). Then, these local predictions are reduced to compute the global prediction.

This approach have some main issues. Firstly, as users rate only a small percentage of the entire item collection, under 1% for most systems (Sarwar et al., 2000), the utility matrix has mainly unknown data. This sparsity impacts negatively the accuracy of predictions.

Secondly, computation of collaborative filtering algorithms grows with the number of users and items. As web-based recommenders have a sustain growth of both these factors, their systems could encounter serious problems of scalability if they are not handled properly.

Thirdly, users can have similar opinions for a subset of items, but divergent opinions for another subset. For example, on a website like MovieLens, two users could share the same opinion for *action* movies but they could disagree on *romance* movies. These differences could mask their similarities and informations about the subset on which both users agree would not be used while computing predictions.

To improve these three points, different clustering methods have been studied. These methods mainly cluster users. The most often used clustering methods are adjusted cosine, k-nearest neighbors and k-means algorithms (Kuzelewska, 2014; Sarwar et al., 2001). Each user is represented by a vector of item ratings and users with similar vectors are grouped together. Early researches mainly aimed to reduce the computational time of recommendations or predictions. Most recommender systems are used in website and must provide a real-time service with a short response time. Therefore, clustering was used to improve sparsity and scalability at the cost of results accuracy (DuBois et al., 2009; Sarwar et al., 2002; Altingovde et al., 2013; Sarwar et al., 2001). Then, researchers developed more complex clustering algorithms to also get results improvements. For example, explicit trust information and social networks have been used to propagate affinity (Pitsilis et al., 2011; Ma et al., 2014; Pham et al., 2011; O'Donovan & Smyth, 2005), demographic data such as sex, age or location were used to identify similar users (Selamat & Moghaddam, 2012)... Clustering items before searching similar users (Wei et al., 2012; Xue et al., 2005) and biclustering (items and users clustering) (Gong, 2010; Banerjee et al., 2007; Kohrs & Merialdo, 1999; Nathanson et al., 2007; Konstan et al., 1997; Cho et al., 2004) also improve results.

In the literature, works proposing strategies to aggregate recommendations from multiple clusters are always based on multi-view clustering, where each data source represents a view. For instance, authors presented a framework which considers a trust-based similarity clustering, a similarity-based user clustering and a similarity-based item clustering (Ma et al., 2016). In another context, social relationships were incorporated into a clustering-based method (Guo et al., 2015). The two views considered are the user similarity and the user trust. Therefore, one user can belong to two different clusters and used support vector regression (SVR) in order to handle the situation where two predictions are generated from two clusters. The problem with SVR is the choice of the kernel and the need to perform cross-validation in order to fix parameters.

In (He et al., 2014), the authors detailed co-regularized nonnegative matrix factorization (CoNMF), which is an extension of NMF for multi-view clustering by jointly factorizing multiple matrices. As a consequence, multiple views are fused during the clustering process. Experiments were performed in the context of comment-based multi-view clustering.

Heterogeneous networks were also aggregated in order to improve trust prediction performance in social networks (Zhang et al., 2016). These networks are actually an explicit trust graph and a rating graph used at the end to combine explicit and implicit similarities using a linear combination method implying a parameter to be fixed by the user.

Unlike the aforementioned approaches, the present work cannot be seen as a multi-view clustering problem. Clustering is only performed by considering one view: the item genre. Our research is focused on setting up an original hybrid method, mixing content-based clustering and CF, based on the exploitation of multi-genre items. Our objective is to improve movie rating predictions with a clustering method less time consuming than the examples introduced in the previous paragraph. Such as (Quan et al., 2006), we suppose that people ratings can depend on the item category. For example, someone can really love *War* movies and give many useful ratings for this genre. At the contrary, the same person could be less attracted by *Romance* movies and would rate less movies of this category. In (Quan et al., 2006), authors defend that some users could disagree about movies inside a category and they therefore use a complex mechanism to identify and cluster similar movies. However, our goal is to study the impact of a less time consuming clustering method.

In the next section, we introduce how clustering is computed and different methods to reduce predictions.

3. Item Clustering and Weighting Strategies

3.1. Clustering According to Item Genre

The item content-based clustering is executed during the ratings reading. The process creates a cluster for each movie genre. Then, ratings are put in one or several clusters depending on the active genres of the related movie. For example, the movie “Avatar” belongs to three movie genres: Action, Adventure and Fantasy. Ratings concerning these movies will therefore be put in three clusters: the Action cluster, the Adventure cluster and the Fantasy cluster. By doing so, we create overlapping groups of ratings related to movies sharing at least one common genre.

To do so, the number of genres, denoted G, is identified using metadata informations and a rating set R'_k is created for each of them. Then, while reading the rating database, each rating is put in the clusters corresponding to its active movie genres. The G clusters are then used to create utility matrices for each genre. The matrix dimensions are adapted to the number of users and items belonging to each cluster. There are more matrices, but they are smaller and denser than the original utility matrix. To achieve this, each item and user get new local identifiers for each cluster they belong to. Dictionaries are used to make the correspondence between global and local identifiers. This is depicted in Figure 1. This Figure introduces 5 movies (“Grumpier Old Men”, “Waiting to Exhale”, “Sense and Sensibility”, “Wings of Courage” and “White Squall”) belonging to 4 different genres (Comedy, Romance, Drama and Adventure). The first movie of the list is “Grumpier Old Men”, with an identifier equal to 3. As it has 2 genres, it will be associated to 2 clusters: Comedy and Romance. As it is the first movie to be associated with both these clusters, it has 0 as local identifier in these two clusters. The dictionaries corresponding to these clusters retain that the film with the identifier 3 has 0 as local identifier. Then, the second film, “Waiting to Exhale”, with an identifier equato to 4, has 2 movie genres: “Comedy” and “Drama”. As the “Comedy” cluster already possesses one movie, the local identifier of “Waiting to Exhale” for this cluster is 1. As it is the first “Drama” movie, the film gets the 0 local identifier for this cluster. The clustering method repeats these steps for each film. It is not depicted in Figure 1, but a dictionnary associating global movie identifiers and movie genres, represented by a binary mask, is computed to represent movie-clusters association. When reading ratings, these dictionaries are used to know in which cluster ratings must be put and which is the local identifier of the active movie.

The open-source recommender system library MyMediaLite (Gantner et al., 2011) is then used to compute V and D matrices for each cluster of ratings. As these clusters are independent, matrices are factorized in parallel by several cores.

3.2. Weighting Strategies

After the item content-based clustering, V_k and D_k are used to compute local predictions $R'_k(i,j)$ for each $k \in K$, where K is the set of active genres. It is then necessary to reduce each contribution to get the global prediction $R'(i,j)$. If the active item belongs to only one genre, the local prediction is also the global prediction and it becomes a standard case of CF.

The predictions computation is depicted at Figure 2. Figure 2 takes for example the movie “White Squall” introduced in Figure 1. The movie identifier is 86. When looking at the local movie identifier dictionaries associated to Drama and Adventure clusters, we find respectively 2 and 1 as local identifiers.

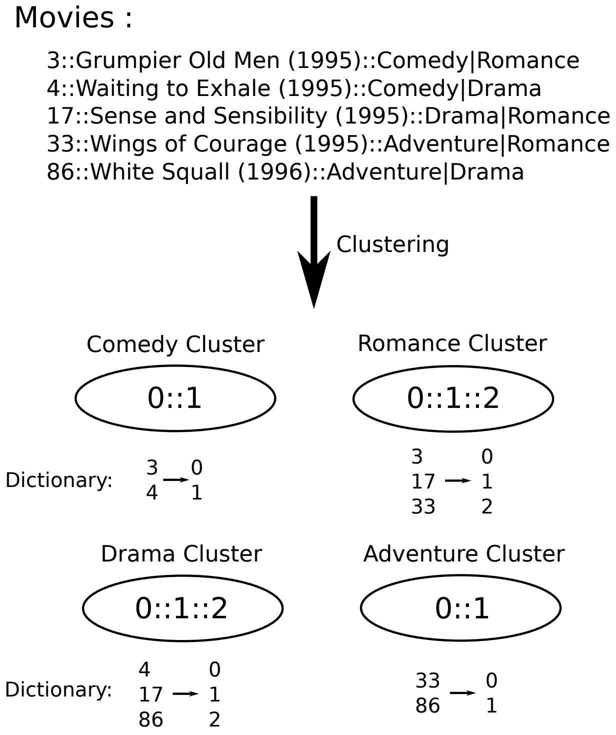

Movies :
3::Grumpier Old Men (1995)::Comedy|Romance
4::Waiting to Exhale (1995)::Comedy|Drama
17::Sense and Sensibility (1995)::Drama|Romance
33::Wings of Courage (1995)::Adventure|Romance
86::White Squall (1996)::Adventure|Drama
Clustering
Comedy Cluster
0::1
Romance Cluster
0::1::2
Dictionary:
3 4 → 0 1
3 17 33 → 0 1 2
Drama Cluster
0::1::2
Adventure Cluster
0::1
Dictionary:
4 17 86 → 0 1 2
33 86 → 0 1

Figure 1: The clustering process firstly checks which genres are associated to the movies so it can later know in which cluster ratings must be put. Movies get local identifiers for each cluster they are associated to and a dictionary is assigned to each cluster to retrieve internal identifiers from global identifiers. With these informations, the clustering method can properly put each rating in the clusters associated to their active genres.

It was not depicted in these Figures, but users also get local identifiers for each cluster just like movies did. We consider here that the local identifier of the user with the identifier 3 is 0 for both clusters. Therefore, to get the prediction of the Drama cluster, the first row of V_2 is multiplied by the third column of D_2. To get the prediction of the Adventure cluster, the first row of V_3 is multiplied by the second column of D_3. The two predictions, 3.9 and 3.3, must then be reduced to obtain the global prediction. Figure 2 computes the average of these two predictions to get the global prediction.

Several reduction strategies were experimented to get the best results. Some are well-known approaches and other are strategies based on the conclusions of our observations. Most of these strategies are focused on training cluster weights to compute weighted averages. To do so, a subset of known data is used as a training set accordingly to the following strategies :

- Average: local predictions $R'_k(i,j)$ are computed for each active genre and are then used to compute a standard average (N is the number of active

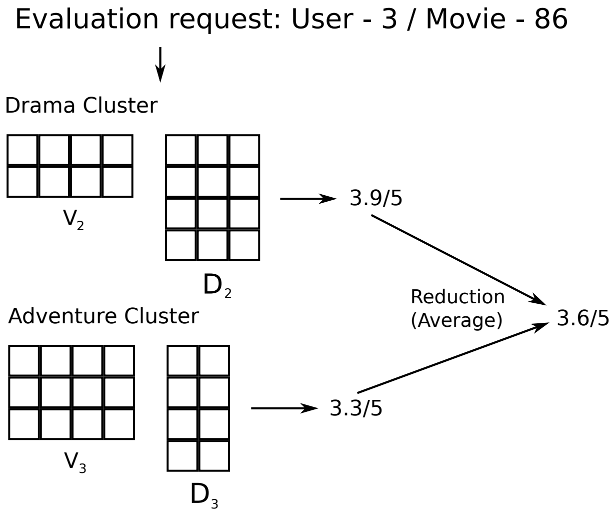


Figure 2: Once genres of a movie have been identified, user and movie local identifiers are used to determine respectively which row of V_k and which column of D_k must be used to compute cluster predictions. These predictions are then reduced accordingly to the chosen weighting strategy (i.e. Average).

genres):

$$R'(i,j) = \frac{1}{N} \sum_{k \in K} R'_k(i,j).$$

- Democracy: with Average, each active genre has the same weight while balancing out predictions. With Democracy, a larger cluster is considered as smarter than the other clusters. If there are more contributions to compute a local prediction, it should be more accurate. Each local prediction is therefore weighted by its cluster size s_k:

$$R'(i,j) = \frac{\sum_{k \in K} s_k R'_k(i,j)}{\sum_{k \in K} s_k}.$$

- Smart Weights I (SW I): this strategy is developped to work with more precision. For each combination of genres, SW I associates a weight to each genre. The hypothesis is that each genre of a combination of genre has its own importance, depending on the combination it belongs to. There are therefore $G * 2^G$ weights w_{kc}, where c is the combination of genres. For example, Figure 1 displays 5 different genres combinations (each film has its own genres combination). Ratings sharing the same movie genre combination, like "Comedy—Romance", are group and then this strategy computes weights specific to this genre combination for the Comedy and the Romance cluster. To do so, predictions $R'_k(i,j)$ are computed $\forall k \in K$ and $w_{k'c'}$ is increased by one, where c' is the active genre combination and

k' is the cluster providing the closest result of the known rating. Then, $R'(i,j)$ is found with the formula:

$$R'(i,j) = \frac{\sum_{k \in K} w_{kc'} R'_k(i,j)}{\sum_{k \in K} w_{kc'}}.$$

- Smart Weights II (SW II): the previous idea creates a huge partition of films for the weight training. If the MovieLens dataset (Harper & Konstan, 2015), collected by the GroupLens Research lab, and its 19 movie categories are taken for example, the strategy creates 524 288 genre combinations. As these datasets have between 100 000 and 22 million ratings, some weights are not properly trained. Therefore SW II uses only one weight w_k per genre. For each evaluation of the training set the weight associated to the category the closest of the solution is increased by one. This formula is then used to compute global predictions:

  $$R'(i,j) = \frac{\sum_{k \in K} w_k R'_k(i,j)}{\sum_{k \in K} w_k}.$$

- Best Category: it exploits weights of SW I. Instead of computing a weighted average, it chooses the prediction of the category having the highest SW I weight:

  $$R'(i,j) = R'_k(i,j).$$

  where $w_{k'c'} > w_{kc'}\ \forall k \in K,\ k \neq k'$. The hypothesis is that there is one best prediction and that the other predictions will just deteriorate this best prediction. Only one prediction is picked to avoid this deterioration.

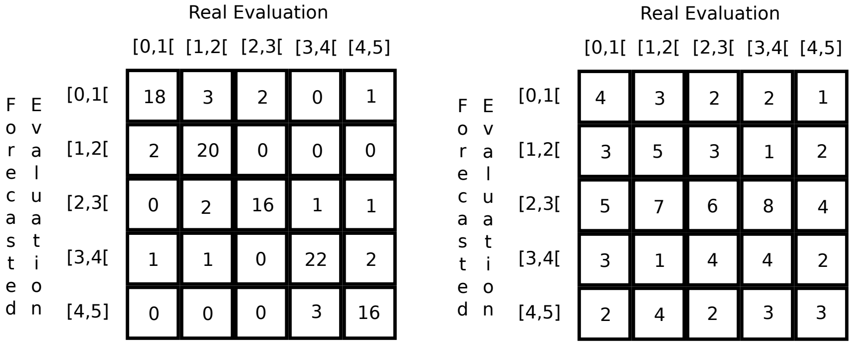
- Multiple Linear Regression: it is based on the same hypothesis than SW I. For each combination of categories, evaluations are modeled by a linear equation:

  $$R'(i,j) = \beta_{c'0} + \sum_{k \in K} \beta_{kc'} * R'_k(i,j).$$

  Evaluations of the training dataset are used to form systems of equations for each combination of categories. Then, the GNU Scientific Library [4] is used to solve the system and find the β parameters.

- Confusion Matrices I (CM I): confusion matrices check if forecasted values fit with the reality. Lines represent forecasted predictions values and columns represent known evaluations values (see Figure 3). If the diagonal of a matrix is more populated than the rest of the matrix, it means that the model fits the reality and that predictions are near of the correct value. I.e., the matrix pictured at Figure 4.(a) will provide results near of

[4] Free Software Foundation. GNU Scientific Library. (2009). https://www.gnu.org/software/gsl/ Accessed 29.09.16



Real Evaluation

Forecasted Evaluation	[0,1[	[1,2[	[2,3[	[3,4[	[4,5]
[0,1[	18	3	2	0	1
[1,2[	2	20	0	0	0
[2,3[	0	2	16	1	1
[3,4[	1	1	0	22	2
[4,5]	0	0	0	3	16

(a) Most forecasted evalutations match real evaluations.
The cluster prediction is valuable.

Real Evaluation

Forecasted Evaluation	[0,1[	[1,2[	[2,3[	[3,4[	[4,5]
[0,1[	4	3	2	2	1
[1,2[	3	5	3	1	2
[2,3[	5	7	6	8	4
[3,4[	3	1	4	4	2
[4,5]	2	4	2	3	3

(b) Forecasted evaluations does not match real evaluations.
The cluster prediction is not valuable.

Figure 3: Example of confusion matrices with evaluation intervals of 1. If the diagonal of a matrix is more populated than the rest, it means that the model fits the reality and that predictions are near of the correct value.

the correct value, while the matrix pictured at Figure 4.(b) will provide poor results. One confusion matrix per cluster is used in our method. The result spectrum is divided in 10 intervals. As evaluations we analyzed are between 0 and 5, each interval is 0.5 (while matrices of Figure 4 use an interval of 1). These matrices need training. For each known evaluation, clusters compute their predictions. For each evaluation the matrix element located at the intersection of the forecasted prediction and the real evaluation is incremented by 1. To compute cluster weights, a first strategy is to select a line of the corresponding matrix according to the interval in which falls the cluster prediction and to divide the diagonal element by the sum of all the line elements:

$$w_{ky} = \frac{M_k(y,y)}{\sum_n M_k(y,n)}.$$

$$R'(i,j) = \frac{\sum_{k \in K} w_{ky} R'_k(i,j)}{\sum_{k \in K} .w_{ky}}$$

with M_k the confusion matrix corresponding at the k^{th} cluster and y the interval identifier in which the prediction is located.

- Confusion Matrices II (CM II): the problem of the previous method is that it requires many known evaluations to train and fill matrices. Even with 20 millions of ratings, many intervals of the confusion matrices have no inputs. Therefore, some w_{yk} stays at 0 and some $R'(i,j)$ cannot be computed as the denominator of the formula is 0. Instead of measuring the reliability of prediction intervals for a cluster, the reliability of the entire cluster is measured. For each cluster, the sum of the diagonal elements is divided by the sum of all the matrix elements. Once computed, these weights do not change:

$$w_k = \frac{\sum_y M_k(y,y)}{\sum_y \sum_z M_k(y,z)}.$$

$$R'(i,j) = \frac{\sum_{k \in K} w_k R'_k(i,j)}{\sum_{k \in K} .w_k}$$

- Global Best Category: as for Best Category, the hypothesis is that there is always a category which is the nearest of the real evaluation and the others will just deteriorate the global result. Each previous method measures the reliability of the clusters involved in the prediction computation. For Global Best Category, each previous method is considered as a voter naming the best cluster according to its criteria. The cluster having the more votes is used to provide the prediction.

All these different weighting strategies were evaluated with different datasets. The experimental protocol and its results are showed and discussed in the next section.

4. Experiments

4.1. Data

For this research, metadata with detailed informations about item genre are necessary. Most available datasets only assign one genre to items. However, there are, to our knowledge, two interesting dataset sources. MovieLens datasets (Harper & Konstan, 2015) are more descriptive and movies can have several genres. The Yahoo! movie dataset [5] does not have such a description, but it has a correspondence table which maps some Yahoo! movie identifiers to MovieLens movie identifiers. It is then possible to attribute several genres to some Yahoo! movies. Datasets from these two sources were used in our experiments.

Besides these two sources, the Amazon product dataset (McAuley et al., 2015b,a) also has detailed metadata, but it has hierarchical categories rather than a flat genre set. For example, these hierarchizations can be found: [“Electronics”, “eBook Readers & Accessories”, “Power Adapters”], [“Electronics”, “eBook Readers & Accessories”, “Bundles”], [“Electronics”, “Computers & Accessories”, “Touch Screen Tablet Accessories”, “Bundles”] ... With these hierarchies of categories, we could compute predictions more and more specialized. For example, the classification [“Electronics”, “eBook Readers & Accessories”, “Power Adapters”] would give three clusters : “Electronics”, “eBook Readers

[5] Yahoo! R4 - Yahoo! Movies User Ratings and Descriptive Content Information, v.1.0 (23 MB) (2006) https://webscope.sandbox.yahoo.com/catalog.php?datatype=r Accessed 29.09.16

& Accessories" and "Power Adapters". The "Electronics" cluster would give a general prediction based on opinions about all electronic devices (such as musical instruments for example) ; the "eBook Readers" cluster will give a prediction based on opinions concerning eBook Readers (including those of people who never bought a power adapters) and the "Power Adapters" will provide a specialized opinion based on other evaluations of power adapters made by power adapters consumers. It would be very interesting to investigate the prediction accuracy of clusters depending on their level of specialization, but it would be a different study. In this study, we are seeking for:

- clusters producing specialized predictions for each movie genre.
- different opinions for a same item such as, if a proper weighting strategy is used, the reduced prediction is more accurate than a prediction based on the entire item set.

Thus, we did not use the Amazon dataset in our experiments.

Four sets were selected to experiment the different weighting strategies: the Yahoo! Webscope movies dataset and three MovieLens movies datasets. Here are their description:

- Yahoo! Webscope Dataset R4: it contains 221,367 ratings from 7,642 users on 11,915 movies (with 43.56% of one-genre movies).
- MovieLens 1M Dataset: it contains 1 million ratings from 6000 users on 4000 movies (with 52.15% of one-genre movies).
- MovieLens 10M Dataset: it contains 10 million ratings from 72,000 users on 10,000 movies (with 37.50% of one-genre movies).
- MovieLens 20M Dataset: it contains 20 million ratings from 138,000 users on 27,000 movies (with 36.69% of one-genre movies).

4.2. Evaluation Environment and Protocol

Evaluations were executed on a 6-core AMD Phenom(tm) II X6 1090T Processor with 8GB of RAM. The operating system used is Linux Ubuntu 3.19.0-25-generic.

In the following experiments, the datasets were randomly divided into n subsets, with $n = 10$, and the algorithms were executed n times, following a n-fold cross-validation. In each run, one of the n subsets, containing about 10% of the ratings, was used as the test set while the other n-1 subsets were merged into a training set. Then, the train set is used to train the model, that means computing D_k and V_k matrices. After that, the test set is used to estimate differences between the model and the reality. To estimate these differences, Mean Absolute Error (MAE) and the Root-Mean-Square Error (RMSE) were used:

$$MAE = \frac{1}{N} \sum_{(i,j) \in \tau} |r_{ij} - r'_{ij}|.$$

$$RMSE = \sqrt{\frac{1}{N} \sum_{(i,j)\in\tau} (r_{ij} - r'_{ij})^2}.$$

where τ is the test set of size N, r_{ij} are known ratings and r'_{ij} are predicted values. MAE is a measure of the efficiency of predictions. The more it is near of 0, the more accurate predictions are. RMSE is a measure of the stability of predictions. As it squares errors, its value grows faster than MAE when predictions are far from correct values. If a dataset has a small MAE but a high RMSE, it means that even if predictions are generally near of correct values, some results are strongly incorrect.

As the file mapping Yahoo! identifier on MovieLens identifier does not map all movies, only 35% of Yahoo! movies have been used.

The evaluation program is structured in two parts. Firstly, the MyMediaLite Recommender System Library is used to compute a matrix factorization of the entire utility matrix to get a baseline. Some parts of the library, originally written in C#, were translated in C to improve performances. When evaluating performances on a dataset of 90 570 ratings, the rewriting reduced the computational time from 2.33s to 1.15s. Some other code optimizations were brought and reduced the computational time to 0.58s. Finally, the use of the optimization flag -O3 reduced the computational time to 0.259s.

Secondly, the evaluation program clusters items and ratings while reading data files. Then, a matrix factorization is executed for each cluster. As clusters are independent, these factorizations are computed in parallel. After these operations, weights are trained and, finally, MAE and RMSE are computed for each weighting strategies.

In the following, we first display and comment efficiency of weighting strategies and then we display detailed computational times for the original and the clustered version of the recommender system algorithm.

4.3. Weighting Strategies

MAEs and RMSE are presented in Table 1. This table displays means and standard deviations computed from results of the 10-fold cross-validation. Best results are bolded and second-best results are underlined. Most standard deviations represent 0.05% of means. In the Yahoo! dataset case, the worst case for standard deviations, it represents 3% of means. Results are therefore stable. The Baseline MAE line of Table 1 displays results of the original algorithm, which is applied on the entire dataset. The next two lines present best and worst achievable results. Best MAE results are obtained by selecting, among cluster predictions, the nearest evaluation of the true rating. It is the ideal goal to reach. Worst MAE results are obtained by selecting poorest evaluations. These figures set boundaries between which results of weighting strategies are located. The other lines of Table 1 displays the results of our weighting strategies. Here are our conclusions based on the analysis of this table :

- Average, which is the simplest solution, is also the worst solution for this kind of reduction (in average, it is 3.9% worse than the best solution).

Weighting strategies are necessary to get better results.

- Weighting strategies taking only one genre into account, like Best Category and Global Best Category, provide better results than Average, but they are less efficient than strategies combining results of several clusters.
- Among multi-genre weighting strategies, SW II does not deliver interesting results. Its results are near of CM II results, there is less than 1% of difference, but CM II is always better than SW II.
- So far, interesting strategies are SW I, MLR, CM I, CM II and Democracy. They all offer at least one result which is the first or the second-best result. Within these strategies, CM I and Democracy are the least effective. CM I offers the best MAE result for the Yahoo dataset, but it is the only

Table 1: Presentation of the Mean Absolute Error and the Root-Mean-Square Error for the different weighting strategies. This table displays means and standard deviations (put in brackets) computed from results of the 10-fold cross-validation. The best result is bolded. The second-best result is underlined. Demo. = Democracy, Best Cat. = Best Category, Global Best Cat. = Global Best Category

	ML – 20M	ML – 10M	ML – 1M	Yahoo!
Baseline MAE	0.7112 (0.0003)	0.7185 (0.0005)	0.7641 (0.0019)	0.6763 (0.0095)
Best MAE	0.6169 (0.0005)	0.6282 (0.0006)	0.6931 (0.0019)	0.5545 (0.0077)
Worst MAE	0.8131 (0.0005)	0.8165 (0.0005)	0.8431 (0.0016)	0.7986 (0.0086)
Average MAE	0.7139 (0.0004)	0.7212 (0.0004)	0.7674 (0.0016)	0.7307 (0.1157)
Demo. MAE	0.7081 (0.0004)	0.7153 (0.0004)	0.7619 (0.0018)	0.6681 (0.0074)
SW I MAE	0.7054 (0.0004)	0.7136 (0.0005)	0.7616 (0.0017)	0.6643 (0.0080)
SW II MAE	0.7079 (0.0004)	0.7152 (0.0004)	0.7619 (0.0018)	0.6681 (0.0073)
Best Cat. MAE	0.7103 (0.0004)	0.7193 (0.0006)	0.7669 (0.0017)	0.6719 (0.0089)
MLR MAE	**0.7027 (0.0004)**	**0.7106 (0.0006)**	**0.7611 (0.0019)**	0.7157 (0.0119)
CM I MAE	0.7073 (0.0004)	0.7152 (0.0005)	0.7659 (0.0016)	**0.6569 (0.0086)**
CM II MAE	0.7064 (0.0004)	0.7141 (0.0005)	0.7615 (0.0017)	0.6679 (0.0077)
Global Best Cat.	0.7105 (0.0005)	0.7192 (0.0006)	0.7663 (0.0016)	0.6719 (0.0089)
Baseline RMSE	0.9096 (0.0005)	0.9151 (0.0006)	0.9542 (0.0026)	0.9728 (0.0153)
Best RMSE	0.6849 (0.0010)	0.6983 (0.0011)	0.7999 (0.0051)	0.7413 (0.0260)
Worst RMSE	1.0125 (0.0014)	1.0125 (0.0014)	1.0588 (0.0055)	1.1404 (0.0314)
Average RMSE	0.8412 (0.0010)	0.8356 (0.0382)	0.9245 (0.0052)	0.9319 (0.0279)
Demo. RMSE	0.8241 (0.0010)	0.8311 (0.0011)	**0.9091 (0.0052)**	**0.8997 (0.0267)**
SW I RMSE	**0.8207 (0.0010)**	**0.8296 (0.0013)**	0.9096 (0.0051)	**0.8992 (0.0283)**
SW II RMSE	0.8240 (0.0010)	0.8310 (0.0011)	0.9092 (0.0053)	0.9009 (0.0265)
Best Cat. RMSE	0.8413 (0.0010)	0.8507 (0.0015)	0.9293 (0.0056)	0.9439 (0.0310)
MLR RMSE	0.8309 (0.0010)	0.8374 (0.0015)	0.9212 (0.0052)	1.1412 (0.0468)
CM I RMSE	0.8244 (0.0010)	0.8329 (0.0011)	0.9192 (0.0053)	0.9205 (0.0286)
CM II RMSE	0.8216 (0.0010)	**0.8296 (0.0012)**	**0.9086 (0.0051)**	0.9008 (0.0275)
Global Best Cat.	0.8390 (0.0010)	0.8466 (0.0019)	0.9255 (0.0049)	0.9453 (0.0306)

achievement of this strategy. For other datasets, CM II povide better results. Democracy provides two second-best RMSE results, but RMSE SW I and CM II results are near (less than 0.5%) and often better than Democracy RMSE results.

- MLR supplies 3 best MAE results but no best or second-best RMSE results. However, there is only a slight difference of about 1% between MLR and SW I RMSE results. Also, the Yahoo! dataset reveals that MLR is the worst strategy for small datasets. MLR is therefore one of the best strategies for datasets which size is at least 1M but is not recommended for small datasets.

- SW I delivers most of the second-best MAE results (and it is only at about 0.3% of the best results) and most of the best RMSE results. It is also effective for all sizes. It is therefore a balanced and effective solution.

- CM II delivers some best and second-best results, but, even if it is not the best, it stays really close of SW I results. It is therefore a good alternative to SW I.

- Finally, comparing SW I results to the baseline, we observe an improvement of 0.9% in average (between 0.3 and 1.8%) of MAE results and an improvement of 7.8% in average (between 4.7 and 9.8%) of RMSE results.

4.4. Computational Times

Computational times of the different algorithms and their steps have been observed. Computational times of the original algorithm are displayed at Table 2 and those of the clustering algorithm are displayed at Table 3. The clustering is executed when reading data files and doubles the duration of this operation. It takes between 2 and 3% of the entire computational time for MovieLens datasets and about 15% of the computational time for the Yahoo! dataset. Clustering costs are therefore not high. Moreover, clustering creates independent clusters which are then factorized simultaneously by several cores. As most ratings are put in different clusters, there is an increase of ratings, but the factorization duration decreases thanks to the parallelization. With 6 cores, there is a decrease of between 22 and 47% of this operation computational time compared with the original version. Finally, some weight strategies need to train weights, which is an additional step that was not part of the original algorithm. This step adds between 15 and 20% of the computational time of the original algorithm. Globally, computational times of the two methods are similar. This can be seen on Figure 4. Therefore, clustering improves slightly results by stabilizing evaluations without costing much additional time if enough resources are used.

Moreover, Table 3 displays the computational time needed to train all strategy weights together. If only one strategy is used, which is the most probable case, the computational time falls. Details of weights computational times are displayed at Table 4. Weights I are used in SW I, Best Category and Global Best Category strategies. Weights II are used in SW II strategy. Confusion matrices

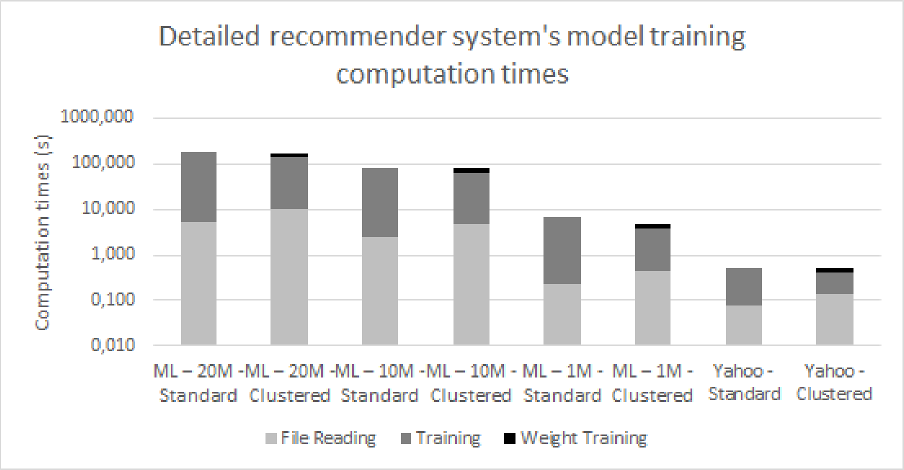


Figure 4: Algorithms detailed computational times

are used for both CM and regression parameters are used for MLR. Computing one set of weights takes approximately half the computational time taken to compute all weights at once. Regression parameters are the most costly weights to be computed as they take between 15 and 60% more time than Weights I. Other weights are computed with slightly different computational times, but the difference is of maximum 18%.

4.5. Experiments synthesis

In section 4.3, SW I, MLR and CM II were identified as efficient strategies. In section 4.4, we have seen that MLR weights require the biggest computational time, CM II saves 22.25% of that computational time and SW I 31.1%. SW I had already been identified as one of the best solution for the results accuracy, it is also the solution requiring the less computing time to train weights.

Democracy is in average 0.45% less accurate than other solutions, but it does not require weight training. It is therefore a good and cheaper alternative to SW I, MLR and CM II.

	File Reading	Matrix Factorization	Total
ML – 20M	5.207	170.702	175.909
ML – 10M	2.442	77.324	79.766
ML – 1M	0.233	6.235	6.469
Yahoo!	0.075	0.433	0.508

Table 2: Measure of the computational time of the original algorithm (s).

	File Reading	Matrix Factorization	Weight Training	Total
ML – 20M	10.019	133.916	28.817	172.752
ML – 10M	4.633	58.677	13.041	76.351
ML – 1M	0.435	3.354	0.997	4.787
Yahoo!	0.144	0.271	0.105	0.520

Table 3: Measure of the computational time of the clustering algorithm (s).

	Weights I	Weights II	Confusion Matrices	Regression Parameters
ML – 20M	12.776	13.985	15.174	17.103
ML – 10M	6.069	6.724	7.111	8.186
ML – 1M	0.494	0.560	0.549	0.571
Yahoo!	0.048	0.048	0.048	0.119

Table 4: Measure of our weights computational time (s).

5. Conclusions

Recommender systems are useful tools for online activities such as e-commerce or recommendation websites such as IMDb. This paper described a clustering approach based on item metadata informations aiming at a result improvement. Items are clustered according to the genre(s) they belong to. As an item can belong to several genres, it can be placed in several clusters. Therefore, several evaluations may be available when requesting a prediction for an item and a reduction of clusters evaluations is required. Several reduction strategies were experimented on movie rating databases. The most useful ones are Democracy, Smart Weights I (SW I) and Multiple Linear Regression (MLR). MLR performs really well for large datasets, but becomes inefficient for small datasets. For small datasets, SW I becomes more interesting (its MAE is 7.2% better than MLR MAE and its RMSE is 21.3% better). Democracy is not as good as the two first strategies (it is in average 0.45% less accurate) but it does not require weight computation and it is therefore the fastest strategy offering an improvement compared to the original algorithm.

If MAE is generally only slightly improved, with a maximum of 1.8% of the baseline, RMSE has a better improvement as it reaches between 4.7 and 9.8% of the baseline for SW I strategy. A smallest RMSE means that inaccurate evaluations are closer to correct solutions than before.

Computational times were also analyzed. The clustering allows a code parallelization as each cluster is computed independently, but weighting strategies require weight training, which adds some computational time. Globally, our method is better than the baseline method. SW I weights training is the cheapest training (31.1% faster than MLR). Then comes the Confusion Matrices (CM) weights training (22.25% faster than MLR) and then the MLR parameters computation (17.103 s for the 20M dataset). Democracy does not need weight training and is consequently the fastest solution. SW I is a good candidate for general recommender systems. For more specific systems, the weighting strategy should be chosen accordingly to its size and objectives.

For future work, it would be interesting to test these weights on other datasets, with a clustering on something else than the genre. For example, if a dataset contains metadata on users about their hobbies, it would be possible to experiment the methodology on a user-based clustering. Moreover, results of our weighting strategies are halfway between the best and the worst MAE, it would be interesting to find new strategies bringing results closer to the best MAE. Finally, we used matrix factorization to proof our clustering method advantages with an existing recommender system algorithm, but it does not proof that every recommender systems will bring the same results. It would be interesting to test our approach with other recommender systems to verify if other algorithms would bring the same results.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, *17*, 734–749. doi:10.1109/TKDE.2005.99.

Altingovde, I. S., Subakan, O. N., & Ulusoy, O. (2013). Cluster searching strategies for collaborative recommendation systems. *Inf. Process. Manage.*, *49*, 688–697. doi:10.1016/j.ipm.2012.07.008.

Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Commun. ACM*, *40*, 66–72. doi:10.1145/245108.245124.

Banerjee, A., Dhillon, I., Ghosh, J., Merugu, S., & Modha, D. S. (2007). A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *J. Mach. Learn. Res.*, *8*, 1919–1986.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* UAI'98 (pp. 43–52). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Cho, H., Dhillon, I. S., Guan, Y., & Sra, S. (2004). Minimum sum-squared residue based co-clustering of gene expression data. *Proceedings of the Fourth SIAM International Conference on Data Mining (SDM)*, (pp. 114–125).

Cöster, R., & Svensson, M. (2002). Inverted file search algorithms for collaborative filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '02 (pp. 246–252). New York, NY, USA: ACM. doi:10.1145/564376.564420.

DuBois, T., Golbeck, J., Kleint, J., & Srinivasan, A. (2009). Improving recommendation accuracy by clustering social networks with trust. In *RecSys workshop on Recommender Systems and the Social Web*. New York, NY USA.

Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Mymedialite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems* RecSys '11 (pp. 305–308). New York, NY, USA: ACM. doi:`10.1145/2043932.2043989`.

Gong, S. (2010). A collaborative filtering recommendation algorithm based on user clustering and item clustering. In *Journal of Software* (pp. 745–752). Academy Publisher volume 5.

Guo, G., Zhang, J., & Yorke-Smith, N. (2015). Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *Knowledge-Based Systems*, *74*, 14 – 27.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, *5*, 19:1–19:19. doi:`10.1145/2827872`.

He, X., Kan, M.-Y., Xie, P., & Chen, X. (2014). Comment-based multi-view clustering of web 2.0 items. In *Proceedings of the 23rd International Conference on World Wide Web* WWW '14 (pp. 771–782). ACM.

Kohrs, A., & Merialdo, B. (1999). Clustering for collaborative filtering applications. In *Proceedings of International Conference on Computational Intelligence for Modeling, control and automation*. Vienna, Austria: IOS Press.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, *40*, 77–87. URL: `http://doi.acm.org/10.1145/245108.245126`. doi:`10.1145/245108.245126`.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*, 30–37. doi:`10.1109/MC.2009.263`.

Kuzelewska, U. (2014). Clustering algorithms in hybrid recommender system on movielens data. In *Studies in Logic, Grammar and Rhetoric* (pp. 125–139). volume 37.

Lee, J., Bengio, S., Kim, S., Lebanon, G., & Singer, Y. (2014). Local collaborative ranking. In *Proceedings of the 23rd International Conference on World Wide Web* WWW '14 (pp. 85–96). New York, NY, USA: ACM. doi:`10.1145/2566486.2567970`.

Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). *Mining of Massive Datasets*. (2nd ed.). Cambridge: Cambridge University Press.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, *7*, 76–80. doi:10.1109/MIC.2003.1167344.

Ma, X., Lu, H., Gan, Z., & Ma, Y. (2014). Improving recommendation accuracy with clustering-based social regularization. In *Web Technologies and Applications* (pp. 177–188). Springer International Publishing.

Ma, X., Lu, H., Gan, Z., & Zhao, Q. (2016). An exploration of improving prediction accuracy by constructing a multi-type clustering based recommendation framework. *Neurocomputing*, *191*, 388 – 397.

McAuley, J., Pandey, R., & Leskovec, J. (2015a). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '15 (pp. 785–794). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2783258.2783381. doi:10.1145/2783258.2783381.

McAuley, J., Targett, C., Shi, Q., & van den Hengel, A. (2015b). Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '15 (pp. 43–52). New York, NY, USA: ACM. URL: http://doi.acm.org/10.1145/2766462.2767755. doi:10.1145/2766462.2767755.

Nathanson, T., Bitton, E., & Goldberg, K. (2007). Eigentaste 5.0: Constant-time adaptability in a recommender system using item clustering. In *Proceedings of the 2007 ACM Conference on Recommender Systems* RecSys '07 (pp. 149–152). New York, NY, USA: ACM. doi:10.1145/1297231.1297258.

O'Donovan, J., & Smyth, B. (2005). Trust in recommender systems. In *Proceedings of the 10th International Conference on Intelligent User Interfaces* IUI '05 (pp. 167–174). New York, NY, USA: ACM. doi:10.1145/1040830.1040870.

Paterek, A. (2007). Improving regularized singular value decomposition for collaborative filtering. In *Proceedings KDD Cup Workshop at SIGKDD'07, 13th ACM International Conference on Knowledge Discovery and Data Mining* (pp. 39–42). San Jose, California, USA.

Pham, M. C., Cao, Y., Klamma, R., & Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science*, *17*, 583–604.

Pitsilis, G., Zhang, X., & Wang, W. (2011). Clustering recommenders in collaborative filtering using explicit trust information. In *IFIP Advances in Information and Communication Technology (Vol. 358)*.

Quan, T. K., Fuyuki, I., & Shinichi, H. (2006). Improving accuracy of recommender system by clustering items based on stability of user similarity. In *Proceedings of the International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce* CIMCA '06 (pp. 61–68). Washington, DC, USA: IEEE Computer Society. doi:`10.1109/CIMCA.2006.123`.

Rashid, A. M., Lam, S. K., Karypis, G., & Riedl, J. (2006). Clustknn: A highly scalable hybrid model- and memory-based cf algorithm. In *Proceedings of WebKDD-06, KDD Workshop on Web Mining and Web Usage Analysis, at 12 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* WWW '01 (pp. 285–295). New York, NY, USA: ACM. doi:`10.1145/371920.372071`.

Sarwar, B., Karypis, G., Konstan, J. A., & Riedl, J. T. (2000). Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*.

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *5th International Conference on Computer and Information Technology*.

Selamat, A., & Moghaddam, S. G. (2012). Improved collaborative filtering on recommender based systems using smoothing density-based user clustering. In *International Journal of Advancements in Computing Technology* (pp. 352–359). ACM volume 4.

Wei, S., Ye, N., Zhang, S., & Huang, X. (2012). Collaborative filtering recommendation algorithm based on item clustering and global similarity. In *Business Intelligence and Financial Engineering (BIFE), 2012 Fifth International Conference on*. Lanzhou: IEEE.

Xue, G.-R., Lin, C., Yang, Q., Xi, W., Zeng, H.-J., Yu, Y., & Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* SIGIR '05 (pp. 114–121). New York, NY, USA: ACM. doi:`10.1145/1076034.1076056`.

Zhang, W., Wu, B., & Liu, Y. (2016). Cluster-level trust prediction based on multi-modal social networks. *Neurocomputing*, *210*, 206 – 216.