UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

Eigenclusters for predicting users preferences

Máster en Inteligencia Artificial Avanzada

Autor: **Pedro José Corral Bondia**

Directores: **Víctor Fresno Fernández**
Roberto Centeno Sánchez

September 2019

Table of Contents

Figures Index

Index of Tables

1. Introduction

With the normalization of use of on-line platforms for a huge quantity of different purposes [1], the interaction between the user and these has become a crucial factor for their popularity and success. Nowadays, the total amount of information, services or products available to the user can even be overwhelming and sometimes impracticable to check it completely [2] [3]. The search engines embedded in those on-line platforms can help for narrowing down the final objective seek by the user when it is clear or at least more or less identifiable under some parameters. There are, however, cases where what is wanted by the users is not even known by them, just something that is liked or interesting. This is a very frequent situation when searching content for entertainment or recreation where, in general, every time something is consumed it uses to be new, no matters if it is a game, film [4], book, song, comic, etc. That does not mean some people does not like to repeat some of their past experiences enjoying these more than one, but this case is more the exception than the rule. In the world of entertainment the main trend for users is selecting something not previously experienced. The quest for this different content is not necessarily a radical change; in fact, there are some patterns about the preference of consumers. That latter covers from food to interests in life. So, basically, what a person wants, enjoys or is interested in does not use to be a random selection, but can be grouped in sets of similar content. It is not even unusual the kind of entertainment selected by one person has even little variation.

1.1. Recommendation problem

With the arrival of globalization there is a humongous amount of offer to be consumed, greater than the time to consume it. Just as a sample, YouTube generates around 300 hours of video per minute [2]. That extrapolated to other cases like video games or books, means that it is important to make the right selection, trying to guess which of those are going to satisfy the most their consumers. In those specific cases, it is also an additional factor like the cost of decision, so basically the most accurate in selecting the least waist of time and money. Historically, those decisions have been made gathering information from friends and known people who already have consumed them providing a direct feedback, but from some time ago this task has tried to be automatized. This is how the **recommendation problem** is introduced. It can be defined as the way of giving the most accurate prediction of a user's preferences or probability of liking some specific items such as videos, music, books, hotels, restaurants, online dating and any imaginable service that a platform can provide. In this automatization process is where AI is introduced in a natural way because those have become intelligent decisions.

1.2. Recommender systems

Recommender systems [5] is the term that covers all the automatized methods and their implementations for predicting the interests, preferences or ratings that users would apply to items if they were consuming them. It can be said that the way those systems act is prescriptive, trying to determine what is likely to happen and what not, and then suggesting

the best option among all the available ones. These systems are very popular nowadays for its extensive use in several online platforms related to entertainment, e-commerce and others like Netflix, Amazon, Uber, YouTube, Spotify and many more. Those systems provide competitive value for the companies using them. A prove of it is the *Netflix Prize* [6] for the best collaborative filtering algorithm focused on users' predictions for films. The accuracy of these recommender systems has become an important concern for the companies using them because of the economic impact that these have in the quality service to their customers. Recommendation systems try to solve the recommendation problem using data from enormous quantities of users, these systems identify patterns and trends from different sources to have the best predictions. This concept was firstly formally introduced in 1990 by the Swedish computational linguistic Jussi Karlgren [7].

Depending of the nature of the platform where those systems are implemented, sometimes there can be only a few items to be offered, but in general it is easy to find lots of them. When the first case is true, the system only has to provide a list of the available options that would match the desired request from users, but even in these cases it is interesting to have some kind of prioritization criterion. That is specially important in the cases having lots of positive recommendations. That leads to the situation where despite those recommender systems are used in a prescriptive way, they are based in a predictive foundation. That is, those systems try to predict the likelihood of a good reception of an option by a user applying some kind of evaluable metric, more concretely sortable. This sortable metric is then used for having the right order of preferences and presenting to the user only the most promising candidates. This not only reduces the risk of offering something that could be rejected, but also simplifies the total offer to the user.

At the actual literature it can be found several ways of generating recommendations, and these can be grouped in different categories depending on the selected strategy for solving the problem. Independently from the evaluating method used for knowing the users' preferences, in general it can always be used an algorithm that somehow evaluates users' scores for every item that may be offered to them. After all this sequence of evaluations, it is trivial to sort them prioritizing the best scored in order to offer the first ones as the most likelihood alternatives.

There is a common basic characteristic for the several methods that are going to be explained. All of them use historical information about the user related to the prediction of items or rates. Basically this is using information equivalent to the one to be predicted. This information is obtained directly from the user where the prediction is going to be made, or sometimes from other similar users whom behaviour is considered (somehow) close to this user. When using data from other users, their similarity to the one to be evaluated is based on similar interests, rating of known items or other attributes that can be justified as correlated to what is predicted.

1.3. Cold start

This is the given name to a well known problem associated with the recommender systems and describe these situations where it cannot infer properly or at all because the lack of enough information [8]. When this happens it is caused by the algorithm or the model facing a situation with a new user, item or community.

1.3.1. New user

This is maybe the most common case in open communities; users join these and initially there is virtually no information about the user rather what is required for the signing up. Despite this situation of lack of needed information, users want to access and enjoy the benefits of the recommendation system. That leads to one dilemma where the are two clear options to consider:

1. The systems has not enough data for providing quality recommendations but must provide them.
2. It is established that the system avoids to offer any recommendation until a minimum threshold of data is available for guarantee enough accuracy.

Both scenarios lead to a risky situation. If the first case happens, it is likely that the user will judge the overall quality of the recommender just by those inaccurate suggestions. That is because not all the users will assume that the more their interact with the system the more it will learn and improve future recommendations. That perception of poor quality service can even finish in a churn of the user with its economic potential consequences. On another hand, when the second decision is taken and no recommendation is given to novel users during some time, the risk is not offering a valuable service and that may also lead to a bad global perception of the platform.

A common solution applied in order to solve these issues is to query the new users about some of their preferences. Obviously, that increases the signing process and in the worst cases it could even produce the abandon of the process, so finding the minimum valuable information that ensures the right balance between the accuracy of the suggestions and the time and actions needed for joining the system is the key of this approach. As mentioned earlier, the total amount of data needed for achieving this purpose is strongly related to the selected model.

1.3.2. New item

Similarly to the new user situation, when a new item is include to be evaluated by users in order to recommend it or not, it uses to happen a lack of information about the item; this is because there are no previous (or enough) interactions. In these situations of little or no interactions the recommender systems can only but provide poor quality suggestions or none at all.

The dilemma with new items is similar to the case of new users, but if the full catalogue is big enough, the impact should not be as bigger as one user with unknown preferences. The risk is not concentrated in just one user, but spread among the different users where the item could be recommended.

1.3.3. New community

This is the worst situation to be faced because there are not enough users for providing the critical mass of data needed for quality predictions. In communities where the content is also generated by the users, or somehow related to them, it can even happens that there also no items or just few of them. That case is specifically complicated because it cannot be applied the solutions for the previous situations.

1.3.4. Popularity bias

This situation [9] happens when there is not a flat distribution of interactions for all the items. In other words, not all the items have an equal proportional fraction of the interactions. This unequally distributed attention from the users is explained by the items' popularities; it is easy to think about some books, films, video games, etc. that have become so popular that lots of people have formed an opinion about them. This also affects in the opposite way, all the popularity some items have is translated in lack of interest about other less known. Just as a sample of this non-uniform distribution, the **figure 1** shows the distribution of interactions per item at the Movielens dataset [10].

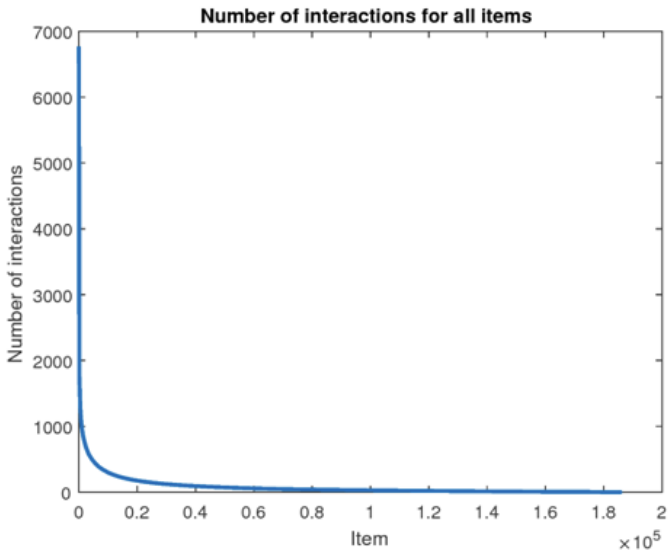

Figure 1. Distribution of users interactions per item

Different popularities lead to different evolution of the data associated to these items. In one hand the popular items, with lots of data, allow to refine predictions related to them more accurately, increasing their probability of being recommended. This is not a bad situation at all but give some competitive advantage to unpopular items that face the opposite situation; because of the lack of interactions their evaluation is known to be less accurate than the popular and their probability of being offered is smaller. Despite this logic depicted, this bias has more or less impact depending on the algorithm selected for generating the recommendations.

1.4. Problems to be considered

Like any other tasks where AI is introduced, the way to solve the problem is using the maximum valuable amount of information in the most efficient manner. That is basically the combination of two different but related dimensions. On one hand the information available and on another note how well performs the use of this information. Despite it is not absurd thinking that one can compensate the other, a completely lack of information makes impossible to apply any insight at all. The problem does not have to be in such radical situation, but it is sensible considering that there are situations where the available information is pretty close to

nothing. In such cases, the methods for obtaining insights from almost no information are not as frequent as when there is a lot of information.

The problem of obtaining information can also be also considered from two different angles about the available data. The first, earlier mentioned, it is just the availability; the second one is the nature of this data. This nature also affects the way the information is extracted from pure data and how it is articulated in order of synthesize valuable insights for predicting the users' preferences. The available data can be structured, like rates provided by users, or it can be unstructured like textual reviews about similar items somehow related to the ones to be evaluated. There can be also other kinds of structured information like geolocation or personal characteristics like age or gender among several other. All this means that depending on the obtained data, the methods for processing and combining these can be very different; so there is no one single best solution for any environment where the recommendation problem is wanted to be solved. With that mindset it can be justified new ways of facing the problem of the recommendations, not necessarily arguing a direct comparison of performance with other actual methods, but rather because the suitability of using one or another approach depending on the circumstances drawn by the available data.

1.5. Scope of this proposal

The approach proposed in this work is specially related to the problem of the cold start, more concretely to the situation where there is little information or none at all known about the user in order to make good predictions. As it will be detailed in next sections, the proposal suggest a series of variants with the aim of extracting as much information from the user as possible. That does not mean it is strictly reduced to this area, because the methodology exposed is robust enough for providing reasonable good enough predictions in a wide range of different availabilities of data. But on another hand, despite it is able to be used as a general method, it has been introduced an extra effort in dealing and obtaining the best results on this reduced area of very little knowledge from a user.

2. State of the art

As mentioned during the introduction, recommendation systems are not a virgin area and several ideas has been developed, implemented and proven to be really efficient. This section is intended to be a compilation of the most important approaches actually available for solving the recommendation problem. Those are grouped in several categories depending on the conceptual approach used for obtaining a solution; these are not necessarily applied in an isolated way; it is not unusual to have a combination of several of them within a hybrid system. That is for having a good performance but also for compensating the different advantages and drawback from the methods used. Just as a sample of those, the **cold start** [8] problem has different impacts depending on the method used.

There is not a natural differentiation of the actual methods presented in this section in terms of the quantity of the available data. Those approaches are general and designed to provide predictions in a whole spectrum of circumstances independently of the system's knowledge about its users. A completely different thing is their performance in these different situations of having more or less data so, in general, they provide the best results when reasonable amounts of data are available. This quality is then affected when the information for feeding the models becomes reduced, even facing situations where the predictions show considerable error. It is because of this situation that this section covers several of those methods, because they are not specifically designed to deal with small data cases but it also falls within their scope to a greater o lesser extent.

The rational behind the categories and subgroups presented can be summarized as:

- **Collaborative filtering** as a way of predicting user's interests with the data obtained from other users similar enough to this one.
- **Content-based filtering** bases the user's predictions about items on the data coming from other previously related items.
- **Hybrid recommender systems** use combinations of pre-existing methods in order to have better results than the combined methods alone.
- **Location-based recommendation** has the characteristic of giving a special weight to the geolocation information about the user, using it for filtering and prioritizing the results.
- **Risk-aware** takes very seriously into consideration the possible negative impact of bad recommendations.

Finally, at the end of the section, there is a zoom on some experiments performed using different approaches and an evaluation of their average error. This information will be used later for a better understanding of the results of the proposed experiments for validating the propossed approach.

2.1. Collaborative filtering

Collaborative filtering (**CF**) [11] is one of the most popular categories for classifying the recommendation systems. There are two main interpretations [12] of CF depending on applying a general overview or entering in some of the hypotheses applied:

- **General definition**. CF refers to the use of a combination of different methods that requires collaboration from them; those methods involve different (in content and nature) sources of information, metrics of evaluation for users, viewpoints and others.
- **Detailed definition**. CF is the method that allows to predict preferences and interests for a user using information from similar users rather than from random ones.

Going deeper into the detailed definition, CF is based on the time immutability of the user's preferences, so knowing these preferences from the past, they are expected to be valid also for the future. The way of evaluating and categorizing those preferences (users' interests representation) can be very different depending on the approach, but must be consistent for all the users that form the knowledge domain. The data extracted from many users is then used for predicting the interests, preferences and tastes about items they have not been related before.

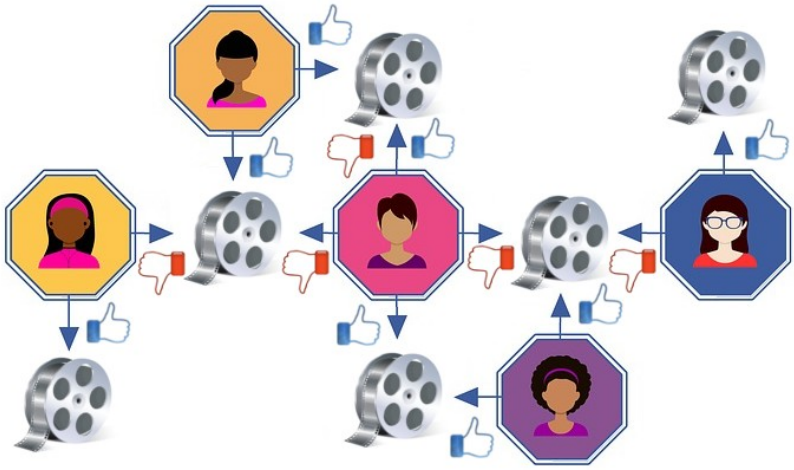

Figure 2. Conceptual representation of a rating network. Network formed by users and items (specific case of films). Users share films in common but also films have users in common. Every user contributes to the films reviewed with some kind of metric related to identify their preferences towards some of the films.

The second big assumption for CFs is based on the existence of patterns among the users that can be exploited for segmenting them or for defining similarities among them. When two users (u and v) belong to a same segment, by definition they share some degree of preferences/interests/tastes in comparison to users belonging to other groups. The known information from one user can be used for his/her segmentation or for finding similar users. This similar user v from the same segment provide new information about preferences unknown for the user u. This preferences from v are used to predicted some of u's own

unknown ones with a expected greater accuracy compared to just using information from users with lesser similarity to u.

Summarizing, the two common assumptions for CFs are:

1. **Immutability**. Tastes, likes, dislikes, preferences, motivations, etc. do not change over time, or at least it happens so slowly that these can be considered constant during the time range to be evaluated. This time range includes the pre-existing data from users and when any evaluation of users' interests are performed.
2. **Similarity**. Tastes, preferences, etc. are not randomly distributed among the users, those follow some aggregation patterns so given two users u and v, if they share some of those preferences, it will be more probable to have similar preferences in other matters compared to random users who do not have common points of interest.

Despite the described methodology is based in a user-centric conception, there is a variant known as **item-based collaborative filtering** that applies equivalent principles to the items and their similarity instead of the users. This method [14] was invented and successfully applied since 1998 by Amazon.

The following sections focus on the user-centric approach of CFs and their main variants, but many of those user-based collaborative filtering (UBCF) cases have their equivalent item-based collaborative filtering (IBCF). The common nexus (but also their difference) for those methods is how the similarity between users is evaluated. These sections do not pretend to be a completely exhaustive work of every approach available but try to provide a comprehensive explanation and categorization of them.

2.1.1. Memory-based CF

This can be used for both user-based and item-based CFs and bases the similarity on the rates from users to items. When an unknown rate from user u to item i is wanted to be predicted, this is done using some aggregation function over the rest of available ratings. This aggregation function can be expressed in the following ways depending on if its a user-based (1) or item-based (2) CF.

$$r_{u,i} = r(u, i, U) \tag{1}$$

$$r_{u,i} = r(u, i, I) \tag{2}$$

For UBCFs the expression (1) is used and U identifies the set of closest users to u in terms of similarity. For the IBCFs the variant (2) is applied where it is considered the I set of the most similar items of i. In a general way, both (1) and (2) use the user u and item i for evaluating $r_{u,i}$ (the rating from u to i). The final way $r_{u,i}$ is applied can vary. In its simplest form it can just be the average of the ratings of the elements of U or I.

$$r_{u,i} = \frac{1}{|U|} \cdot \sum_{v \in U} r_{v,i} \tag{3}$$

$$r_{u,i} = \frac{1}{|I|} \cdot \sum_{j \in I} r_{u,j} \tag{4}$$

Once again, it is be used (3) for UBCF or (4) for IBCF. These averages are calculated on the cardinality of $|U|$ or $|I|$ and the rates from other users v to the same item i, denoted as $r_{v,i}$ (3) or the rates $r_{u,j}$ (4) of the same user u to similar items j.

The next step in sophistication for this approach is introducing some kind of coefficient that weights the contribution of every rate to the average. In those weighted averages one common option for evaluating every one is a direct comparison of the similarity; the greater this similarity is, the greater this weight is going to be. For the UBCFs case, it takes the following form:

$$r_{u,i} = \sum_{v \in U} w_v \cdot r_{v,i} \tag{5}$$

$$w_v = \frac{\text{simil}(u,v)}{\sum_{v \in U} |\text{simil}(u,v)|} \tag{6}$$

The relation (6) is introduced for guarantee that the weights w_v are normalized. That can be demonstrated by developing their sum as:

$$\sum_{u' \in U} w_{u'} = \sum_{u' \in U} \frac{\text{simil}(u,u')}{\sum_{v \in U} |\text{simil}(u,v)|} = \frac{\sum_{u' \in U} \text{simil}(u,u')}{\sum_{v \in U} |\text{simil}(u,v)|} = 1 \tag{7}$$

In order to apply the relation (7), it is required that any value of similarity is always positive or zero. The equivalent to equations (5) and (6) for IBCFs are:

$$r_{u,i} = \sum_{j \in I} w_u \cdot r_{u,j} \tag{8}$$

$$w_j = \frac{\text{simil}(i,j)}{\sum_{\iota \in I} |\text{simil}(i,\iota)|} \tag{9}$$

As can be seen in the previous equations (8) and (9), the criteria for evaluating the similarity is a key element for these methods. The same way the ratings can be calculated using different approaches, the same happens for the evaluation of similarity between pairs of users or items. In the next sections it is be covered a wide collection of different computation techniques for obtaining the similarity.

2.1.1.1. Correlation-based similarity

This is the case that covers any use of correlation for evaluating the similarity. One frequent election for that is the **Pearson correlation** through the known rates of the users or items. A general definition of the Pearson correlation coefficient for two variables is describing it as the covariance of these two variables divided by the products of their standard deviations. When applying this similarity to two users u and v it is used their known rates population; that is every user is going to be defined by their rates. In this case the *population Pearson correlation coefficient* of these tow users is named as $\rho_{u,v}$ and takes the form:

$$\rho_{u,v}=\frac{\sum_{i\in I}(r_{u,i}-\bar{r}_u)(r_{v,i}-\bar{r}_v)}{\sqrt{\sum_{i\in I}(r_{u,i}-\bar{r}_u)^2}\cdot\sqrt{\sum_{i\in I}(r_{v,i}-\bar{r}_v)^2}} \quad (10)$$

The equation (10) is evaluated trough the set I of common items (intersection) reviewed by them both; this is important because the similarity is then evaluated using a common ground. The variable $\bar{r}_u$ represents the average rates of user u to the common items in I, and $\bar{r}_v$ is the equivalent for user v. When dealing with the item-based approach, the similarities are applied between two items i and j; in this case the common ground for them is the population of common users U that have reviewed them both. The equivalent version of the equation (10) is then:

$$\rho_{i,j}=\frac{\sum_{u\in U}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U}(r_{u,i}-\bar{r}_i)^2}\cdot\sqrt{\sum_{u\in U}(r_{u,j}-\bar{r}_j)^2}} \quad (11)$$

In the equation (11), the variable $\bar{r}_i$ represents the average rates to the item i from the common users, and $\bar{r}_j$ is the equivalent for the item j.

2.1.1.2. Vector cosine-based similarity

This is the case where the similarity is evaluated as the cosine of the angle that two different entities form when those are represented as vectors (**figure 3**). Let's consider the items as the entities so the matrix $\mathbf{R}$ of users U and items I has the range $|U|\times|I|$. The two items i and j to be considered are represented by the vectors obtained from the i-th and the j-th columns in matrix $\mathbf{R}$. Having those two vectors represented as $\vec{i}$ and $\vec{j}$, their cosine (similarity) is then calculated as:

$$\cos(\vec{i},\vec{j})=\frac{\vec{i}\bullet\vec{j}}{\|\vec{i}\|\cdot\|\vec{j}\|} \quad (12)$$

In the relation (12), the operator "•" is the scalar product of the two vectors and the used notation $\|\vec{i}\|$ indicates the length of this vector.

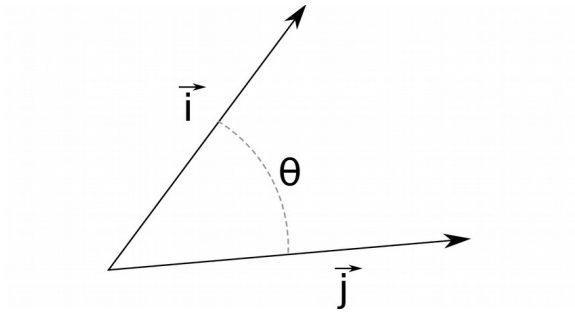

Figure 3. Graphical representation of the cosine similarity based on the angle formed by two vectors (users).

2.1.2. Model-based CF

The main difference of this variant is the direct use of mathematical models for predicting the unknown 3-tuples user-score-item. Instead of looking for a specific way of evaluating according to the closest ones, the logic for predicting user scores is into the model itself. Those models are fitted using the available data for users and items. The main benefit of this approach is there are several generic models that can be applied according to what is more convenient for every case; in general, what is more convenient uses to be defined as what is more accurate in terms of predictions. The models can be **data-mining** algorithms or the nowadays popular **machine learning** [13] ones.

The data-mining approach covers different variants that are pretty close to the dataset. Those can be based on specific patterns, that means that the patterns are found in the different datasets and situations that are wanted to be solved or these use to happen on a reduced collection of datasets or just only one. Because of that, those methods are not necessarily easy to generalize in different environments. Maybe a very specific behaviour can be found in one concrete dataset and it is very accurate; that offers a good solution to this concrete problem, but if it cannot be extended to other scopes then it is less useful than other more general methods. Independently of that, it can be used some statistical models, like linear or non-linear regressions that use known attributes from the users (and maybe also the items) for predicting the ratings. Such solution is still bounded to the dataset structure, the available attributes for users and items depend on the present information, but it is a higher level of abstraction in the way of proposing a methodology. The problem domain is then reduced to properly identify and measure the available attributes. There exists other general solutions [15] covered by the the data-mining category like Bayesian *networks*, *clustering algorithms*, *regression-based* and some others; those are introduced in the next sections.

2.1.2.1. Bayesian belief networks

Those are directed acyclic graphs (DAG) where nodes represent probabilities and the edges define their conditional dependencies [16]. The probabilities contained into the nodes are Bayesian variables that can represent from observable metrics to assumptions in the form of hypotheses or unknown variables. When there are no nodes between nodes, the variables they represent are conditionally independent [17]. These Bayesian belief networks (BNs) can be used as classificatory models. The strengths of these models are the following:

- Inferring unobserved variables can be performed and used for solving probabilistic queries.
- These networks can learn their parameters applying the principle of maximum entropy under some circumstances.
- Can be used to perform inference.

2.1.2.2. Simple Bayesian CF

This model uses a naïve Bayes (NB) algorithm and its predictions to be for supporting CF. The NB prediction is the one with highest probability among all the different classes evaluated; these probabilities are calculated under the assumption that probabilities are independent for every class.

NB is a conditional probability model [18] that assigns a probability to every possible outcome (class) c_k (from class set C) considering the features vector that are a collection of independent variables $\mathbf{x}=(x_1,x_2,...,x_n)$. This probability is denoted as $p(c_k \mid x_1,x_2,...,x_n)$. The simple Bayesian model deals with the problem of incomplete data calculating the probabilities from observed data, and then selecting as prediction the class (outcome) with higher probability. That is formally described as:

$$c_k=\arg\max_{j\in C} p(c_j)\prod_{o\in O} P(X_o=x_o|c_j) \tag{13}$$

In order to avoid conditional probabilities with value 0, it is used additive smoothing [19] setting the specific smoothing parameter to $\alpha=1$.

There are some approaches [20] where the multi-class is converted to binary-class data in order to evaluate boolean variables associated with the rating matrix. Despite this simplifies the application of algorithm, it faces scalability problems because of the class reduction loss of information. This is a real problem when tackling the usual scenario of multi-class data; in those cases some results [21] show that using all the classes has better scalability than Pearson correlation-based CF but worse accuracy than it.

2.1.2.3. NB-ELR and TAN-ELR

NB-ELR stands for naïve Bayes (NB) optimized by extended logistic regression (ELR), whilst TAN-ELR means tree augmented naïve Bayes (TAN) optimized by ELR. Those are advanced algorithms developed with the aim to face situations having incomplete data. Even when this incompleteness of data is not the case, these two models use to provide better accuracy when classifying [22], [23]. More concretely, there is a clear improvement compared to simpler Bayesian algorithms, and this is even greater when this comparison includes other memory-based methods. On another note, the drawback comes from the computational effort introduced as result of a more sophisticated algorithm, this is directly translated into consuming longer times for training the algorithms.

2.1.2.4. Bayesian belief networks with nested decision trees

This is a direct improvement in the inference capacity for Bayesian networks. It expands the probability approach gathered in every of the network's node introducing the logic of

decision trees. More concretely, every of those nodes is associated to one item, and the different buckets that the trees can use for predicting is one of the reachable states where a different rate makes the difference between them.

The empirical analysis [24] of these algorithms proves better results in comparison to vector cosine and clustering algorithms, but equivalent to memory-based when using the Pearson correlation as similarity function.

2.1.2.5. Clustering

These methods cover the different clustering options used for CF. The clusters can be defined as sets of elements where their internal similarities (every element vs other elements in the same cluster) are statistically different to others clusters similarities. That ensures the different clusters define groups of similar elements, no matter if those are users or items. Once again, there are several candidates for becoming the similarity function and it is not unusual to use the Pearson correlation or the Minkowski distance for that. This latter metric can be applied to any n-dimensional hyperspace representing users or items, and can be interpreted as the generalized transition between the Manhattan and the euclidean distance. Its mathematical expression for two n-dimensional vectors $\vec{u}$ and $\vec{v}$ is:

$$d(\vec{u},\vec{v}) = \left(\sum_{i=1}^{n} \left| u_i - v_i \right|^p \right)^{\frac{1}{p}} \tag{14}$$

The order p in the equation (14) is what modulates the Minkowski distance from the Manhattan one (p=1) to the euclidean distance (p=2) but it is not limited to just those values. In general, those clustering techniques are not completely exhaustive in the sense of looking for the absolute optimal cluster distribution that minimizes a cost function. That is consequence of the enormous quantity of configurations to be evaluated, so these methods use to apply some kind of heuristic approach that leads to good enough solutions in the sense of a right balance between computational effort and the quality of results.

Clustering (**figure 4**) is a popular practice in several data-mining and analytics domains, so it has some degree of development and maturity. The clustering methods can be divided into different categories depending on the strategy that their algorithms apply; these are:

- **Hierarchical clustering**, also known as connectivity-based clustering because it considers that some objects are closely related to nearby neighbours rather than to others further away. The way that distances between objects are computed changes depending on the algorithm. It also depends on the criterion for linkage. These methods do not provide a single result but a set of alternatives where the user can choose the appropriate clusters. The main problem with this approach is it does not deal properly with outliers.

- **K-means clustering** [25] is a centroid-based clustering where every cluster has a centroid which is a central vector formed from all the vectors of the objects the cluster contains. It applies a strategy for finding the cluster centres and assigning the different objects to the closest one. In general, the total number of clusters denoted by k is a parameter chosen initially by the user.

- **Distribution-based clustering** is a way of generating clusters identified by some kind of distribution model that is applied to all the cluster members. Despite those methods are able to generate complex clusters capturing how the objects' attributes depend and are correlated, it is not always easy to find a distribution model for an empirical distribution. On another hand, those also easily overfitted.

- **Density-based clustering** tries to identify and differentiate the areas with specific densities; these areas can have different shapes as far as they can be distinguished from the neighbour clusters. The main advantages of this method are that the complexity for solving it is linear related to the number of elements and the solutions are mainly deterministic except for the border points.

The clustering methods are not always used as the basis for a CF approach, sometimes those are just used for preliminary studies. Some of their applications [26] are for partitioning the data and then apply a memory-based method over this data. One advanced approach is the flexible mixture model (FMM) that allows users and items to be clustered at the same time. These FMM methods have been found to provide better accuracy [27] than the Pearson correlation CFs among others. In general, clustering methods provide better scalability than other CF methods; that is because they can focus on subsets (the clusters) instead of the whole population.

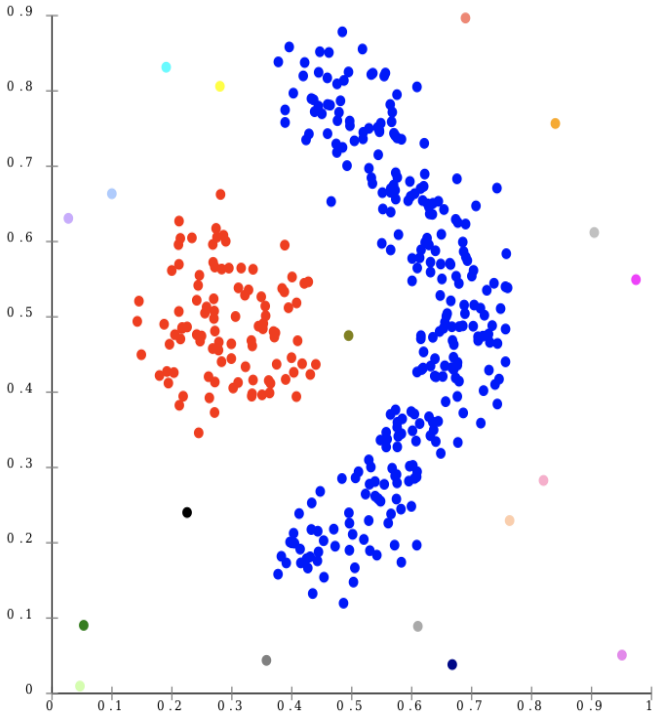

Figure 4. Example of a clustering algorithm extracted from Wikipedia.
Clustering performed with 20 groups represented with different colours. The abscisses and ordinates are normalized and nondimensional as those variables are the result of a dimensional reduction of the different attributes that define every point. As can be observed, some of the clusters are constituted just by one element.

2.1.2.6. Regression-based recommendation algorithms

As it was advanced at the beginning of the Model-based section, the regressions are simple models that can predict continuous values instead of discrete ones. The attributes, preferences or any other kind of information that can translated into a metric, can be used for predicting the rates of the users, but in general what is used is just the known scores from users [29]. Let's define $\mathbf{Y}$ as the matrix of ratings, so every y_{ij} is the rating of user i to item j, so $\mathbf{Y} = (y_{ij}) \in \mathbb{R}^{n \times m}$. On the other hand, $\mathbf{X}$ is the matrix of ratings from m users to k items and $\mathbf{X} = (x_{kj}) \in \mathbb{R}^{k \times m}$. These two matrices of ratings from users can be related as

$$\boldsymbol{Y} = \mathbf{A}\,\boldsymbol{X} + \boldsymbol{\beta} \tag{15}$$

where $\mathbf{A}$ is the coefficients matrix and $\boldsymbol{\beta}$ is the vector related to the noise bias to be corrected in the equation. In the less conventional notation used, the addition of a vector to a matrix is defined as $M_{ij} + \beta_i$, where $\mathbf{M} = \mathbf{AX}$. One frequent problem is that $\mathbf{Y}$ uses to be sparse, and in order to solve that it has been developed a *sparse factor analysis* [28] where the missing elements are replaced with some criteria from the default voting values. This approach provides better scalability than the Pearson correlation-based CFs.

An alternative approach also based on regression was proposed by Vucetic and Obradovic [29]; they evaluated the similarities on the items. For doing that it is needed a series of linear models fitted using the standard regression method of least squares.

2.1.2.7. Markov decision processes

Markov decision processes (MDP) [31] could be described as the set of rules that describe how systems [30] transition from one states into others. This excessive simple but intuitive definition may help to understand the more rigorously speaking, MDP is a 4-tuple (S,A,R,P) system where every of its elements is described as:

- S is a finite set of states that can be reached. This constrain in size is applied when developing practical algorithms but it is not a limitation of the theory.
- A corresponds the set defining the actions, defined finite for the same reasons as S is.
- R is the function that allows to evaluate the rewards when transitioning from state s to z because of the effect of an action a.
- P identifies the function that provides the probability at time t of transitionning from state s to z at time t+1 by applying the action a.

The solution for this problem goes through finding a policy function $\pi(s)$ that identifies what action a should be taken when staying in state s. That function allows to fix the actions and reduce the problem to a Markov chain [32], gathering the different probabilities of transition $P(s_{t+1}=z \mid s_t=s)$ into a Markov transition matrix. More concretely to the problem of recommendations, it can be implemented through an iterative process of refinement of the policy function $\pi(s)$. That means that starting with an initial policy function $\pi_0(s) = \arg\max_{a \in A} R(s,a)$, it can be computed the reward value function $V_i(s)$ of decisions

selected by $\pi_0(s)$. This reward value is then used for updating the policy $\pi_0(s)$ to a new policy $\pi_1(s)$. This can be done iteratively from $\pi_i(s)$ to $\pi_{i+1}(s)$ achieving a convergence in a final optimal policy [33].

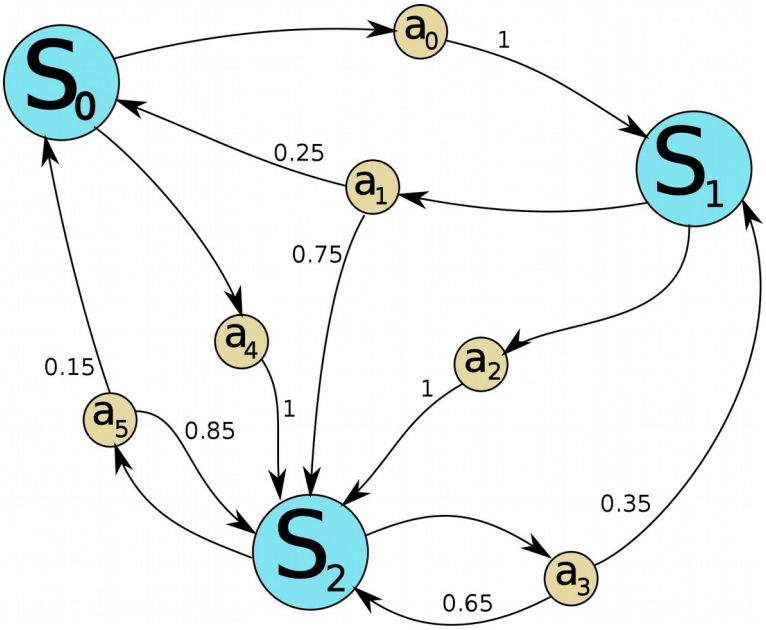

Figure 5. Illustrative sample of a Markov decision process.

A practical appliance for MDPs was proposed by Shani et al. [34] with the following concrete identification of the elements forming it:

- The states defining the set S are k-tuples of items; some of them are unknown.
- Rewards are identified as the utility of selling an item (from the e-commerce perspective) like the net-profit.
- The actions that populate A are the recommendations of items.
- The state z that follows an action (recommendation) a can be only *accepting* the recommendation, select a *non-recommended* item or *not taking* anything at all.

The comparison [34] for this specific MDP was performed on an online bookstore in an A/B testing where the baseline was not using any recommendation at all, and the metric of improvement was the business case in terms of economic profit.

2.1.2.8. Latent semantic CF models

The main purpose of this technique is to identify users' profiles that identify prototypes with specific and differentiated interests using latent class models. Those models relate the observed variables to the latent (unobserved) ones; these latent variables are discrete. In general, this technique provides better accuracy and scalability than the memory-based methods [35].

Hofmann and Puzicha [27] introduced a specific proposal named **aspect model**. This model obtains the latent class variables from the observed tuples formed by users and items,

considering those independent from each other for every of those latent variables. This has been successfully applied to *EachMovie* [37] where the results show a better performance than previous approaches like linear regression and nearest neighbour CF.

Other variations are the **multinomial model** [38] and **multinomial mixture model** [39] where the former is a simple probabilistic one that assumes just one type of user, whilst the latter considers there are several types and the rating variables are independent between them and users' identities. Finally, the **user rating profile** (URP) model [38] combines characteristics from the aspect model and multinomial mixture model with Latent Dirichlet Allocation [40] obtaining better performance that just the aspect model or multinomial mixtures model alone.

2.1.2.9. Other CF techniques based on models

Cohen et al. [36] proposed a model for determining the order of preferences rather than the rates. This **order learning CF** is a model that uses available data for training a preference function. Once this function is obtained, it is used for sorting new data affording a measure of the likelihood of this preference by pairs of elements. Some experiments performed using *EachMovie's* [37] data applying a greed-order algorithm have shown a better results than the ones obtained with nearest neighbour CF or linear regression models.

When the focus is in the top-*N* recommendations rather than the whole options set, the **association rule based** CF algorithms are specially applicable. Sarwar et al. [41] used association rule mining algorithms for finding any good enough rules (according to a established threshold) for predicting and taking the first *N* best ranked items. Another implementation of this method was applied by Fu et al. [42] for recommending web pages using the users' history of visited URLs for obtaining the association rules.

There are also models that have used **maximum entropy** [43] for predicting their results. Those first segment the data with some kind of clustering analysis and after that, on every cluster it is obtained a specific maximal entropy model.

A faster alternative to Bayesian belief networks are **dependency networks** [44], but at the cost of worst accuracy at predictions. Those models are graphs describing probabilities between conditions nested in their nodes.

Some other models are **decision trees** [45] which use these algorithms for classification, or the **horting** [46] approach based on graphs where nodes represent users and the arcs are the similarities between them. There are also other approaches like **probabilistic principal components analysis** [47], **matrix factorization** [48] or **multiple multiplicative factor models** [49].

2.2. Content-based filtering

This filtering process is based on the information from data associated to items relating them to the users' profiles [50], once again in order to be able to predict their preferences. From the item perspective, there is available information about its attributes. Those attributes can be anything like title, category, genre or any available used for classifying them. Profiling the user does not only include general information about him/her, but may also have records of previous interactions with the recommender system. As part of this characteristic information about the users, it is not unusual to explode textual records like reviews or opinions. Taking into

consideration these are based on unstructured information, there exist several techniques that allow to synthesize them into concrete values and/or categories.

One interesting approach is learning from the user's profile rather than try to force the user to provide one. This is usually done using machine learning techniques where the model is able to learn, from some training data, to classify new cases. The drawback with those models is they need to be feed with structured data, which is not necessarily how it is available. It is usual to find textual descriptions or opinions about the items, so it is needed to apply some transformation to the data in order to synthesize some specific metrics. That is not a trivial problem as there are some word characteristics that complicates it; those are the polysemy and the synonymy where there can be found words with several meanings for the former and several words for the same meaning for the latter. There are strategies like **semantic analysis** that using lexicons or ontologies is able to extract to some degree of precision the information needed for feeding the ML models.

2.2.1. Keyword-based systems

One of the most popular representation for structuring users' texts is the TF-IDF (term frequency-inverse document frequency) [51], it is used at 83% of recommender systems for digital libraries. It is based on evaluating how many times a word appears in a text, but also including some weights for specific words used very frequently because are the fundamentals of the grammar. The way this is implemented use to be applied is with a Vector Space Model (VSM) approach, which is basically a n-dimensional vector where every word from a selected vocabulary is represented by one of those dimensions. These VSMs can be identified with texts using some kind of correspondences like TF-IDF. This can defined in a more formal way like a set of documents $D=\{d_1,d_2,...,d_N\}$ usually called corpus and a selected dictionary $T=\{t_1,t_2,...,t_n\}$ that matches the VSM's dimension. It is a common practice to generate T from a selection of the relevant terms, words or tokens found in D. Narrowing down into the definition of the elements of D, those d_j can be defined as $d_j=\{w_{1j},w_{2j},...,w_{nj}\}$ and every w_{kj} is the obtained weight of the term t_k within the document d_j. With that notation TF-IDF is calculated as:

$$\text{TF-IDF}(t_k,d_j)=\text{TF}(t_k,d_j)\cdot\log\frac{|D|}{n_k} \tag{16}$$

$$\text{TF}(t_k,d_j)=\frac{f_{kj}}{max_z f_{zj}} \tag{17}$$

where in equation (16) the cardinality $|D|$ is the size of documents within the corpus, v_k is the frequency of documents where the term can be found. This frequency is different from f_{kj} the one that counts how many times the term t_k appears in the document d_j. It is a common practice to apply cosine normalization to the expression (16) in order to refine the values of the weights:

$$w_{kj}=\frac{\text{TF-IDF}(t_k,d_j)}{\sqrt{\sum_{s=1}^{|T|}\text{TF-IDF}(t_s,d_j)^2}} \tag{18}$$

Those weights are important for calculating the similarity between documents, there are some options for that but one common choice is the *cosine-similarity* that is defined as:

$$\text{simil}(d_i, d_j) = \frac{\sum_k w_{ki} \cdot w_{kj}}{\sqrt{\sum_k w_{ki}^2} \cdot \sqrt{\sum_k w_{kj}^2}} \quad (19)$$

This similarity is used for determining the interest of a user to an item, and this is possible because both profiles (user and item) are translated into vectors composed by weights. There exists several cases of web recommender systems that successfully exploit different kind of information available from the user:

- *Letizia* [52] is an add-on for web-browser able to track the user's navigation patterns and uses the key-words found for deducing the user's preferences.
- *Amalthaea* [53] uses filtering agents feed by the user using pages that represent his/her preferences.
- *iWeb* [54] introduces some innovations to the Amalthea's approach, including direct feedback and disinterests from the users. It is also able to include a temporal component for disinterests that can be evaluated as a progression in time.

For the specific domain of news filtering some samples among the most representatives systems, there are *NewT* [55], *PSUN* [56], *INFOrmer* [57], *NewsDude* [58] or *YourNews* [59]. Some of them are able to deal with every kind of news falling in different sections/categories/topics using different agents trained in this specific domain. *NewT* includes the explicit feedback from users at articles (or parts of them), authors and sources. *YourNews* uses term vectors from past consumed news for identifying the 100 terms with highest weights and generate the prototype vectors.

As a different sample of application in other domain, *LIBRA* [60] is a recommender for books that uses naïve Bayes on the description of products from Amazon. For movie recommendation, it is worth noting *INTIMATE* [61] which uses text extracted from films' synopses in IMDb. *Movies2GO* [62] has a similar approach but reducing the used synopses to just the ones from the films rated by the user. In music recommendation, *Pandora* [63] uses textual descriptions for its recommendation whilst *FOAFing the music* [64] extracts the information from RSS specialised in music.

Main conclusions about keyword-based representation are their good performance when there is enough data available. Despite of that, they do not include "semantic intelligence" which is the next step for improving their performance.

2.2.2. Semantic Analysis

This is a refinement in the understanding of the texts which allows to capture more specific details related to the language rather than just isolated terms. The aim for this approach is more ambitious than previous methods as it is intended to provide the logic and knowledge needed to understand natural language and reason about the documents' content.

There are two main categories for classify the existing systems, **ontology-based** and **encyclopedic-based** depending on the data sources and structure used for feeding them.

The additional information extracted from texts can be obtained from its meaning or context. **Table 1** shows a sample with some categories associated to several names of presidents of the USA. In order to be useful, the selected categories for enhancing the words should be relevant for the specific knowledge domain to be explored. On another hand, **Figure 6** shows some extra features generated using *Semantic Analysis*, where the new information provides a more detailed understanding about the role every specific word is having within its sentence. This kind of more sophisticated analysis allows to refine the textual information and to introduce details, but also requires more complexity for several dimensions, like conceptually and technically.

	Democrat	Republican	Former Governor	Former Vice-President	2-Full Terms in Office	Still Living
L.B. Johnson	+	-	-	+	-	-
Nixon	-	+	-	+	-	-
Ford	-	+	-	+	-	+
Carter	+	-	+	-	-	+
Reagan	-	+	+	-	+	-
Bush (sr.)	-	+	-	+	-	+
Clinton	+	-	+	-	+	+
Bush (jr.)	-	+	+	-	-	+

Table 1. Example of Semantic Features Analysis.

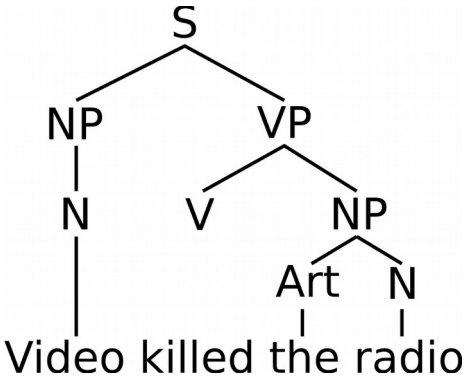

Figure 6. Sample of features obtained from Semantic Analysis.

2.2.2.1. Ontology-based SA

In information science, an ontology covers one or several domains of knowledge where it provides a representation, well defined categories, related concepts with properties and the data and entities that support it. One good example of it is WordNet [65], a lexical database for English that also has its content linked by concepts to other languages (EuroWordNet,

MultiWordNet, etc.). The following is a small sample of the most representative systems and a brief description about them.

The first system (according to actual documentation) using sense-based representation of its knowledge is *SiteIF* [66] and it uses MultiWordNet for exploring multilingual news web sites. News are associated to MultiWordNet *synsets* generating a semantic network. The news to evaluate are translated in the same way and compared using a *semantic network value technique* [68]. *ITR* (*ITem Recommender*) is a multi-domain recommender that covers interests like movies, books, music and others working in a similar way to *SiteIF* but adding some differences in its data structure. It uses synsets directly from WordNet and introduces a vector space model that generalizes the bag-of-words [69] to bag-of-synsets. One system worth to mention is *Quickstep* [70] as it is focused on the academic research papers available on-line. It uses as its basis ontology the DMOZ open directory project [71]. The way semantic is introduced by *Quickstep* is by clustering (applying K-means) the papers using the paper topic. Other systems like *Interactive Digital Television* [72] apply reasoning techniques for recommending TV programs.

The success of the previous samples and others has helped WordNet to become the most widely used lexical ontology for semantic interpretation, supporting that with their good performance. This is based on the fact the lexical ontology is the corner stone when tackling a specific domain, and for the general approach itself all studies including it have improved results when compared to more traditional/basic content-based methods.

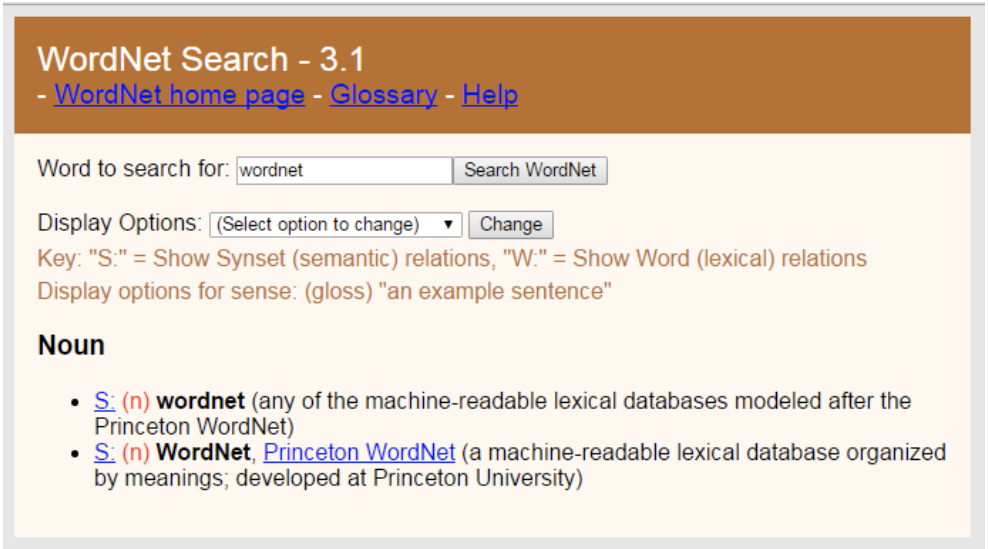

Figure 7. Fragment of a screenshot taken from the Wordnet home page showing a definition of itself.

2.2.2.2. SA based on Encyclopedic Knowledge

With the different techniques for applying *Natural Language Processing*, it can be used general knowledge databases like Wikipedia, Yahoo! Web Directory or the Open Directory Project. The first one has been used for evaluate movie similarity at the Netflix Prize [6] competition using k-nearest neighbour algorithms. Despite the results obtained are good, it does not show a clear improvement in general accuracy when compared to other methods actually implemented.

The actual status of these approaches is in early stage of study and development, those seem to be very promising areas of research but there is not available any specially remarkable result and even some of them have not been already implemented for profiling users.

2.2.3. User profiling techniques

The most popular models used for that fall within the category of machine learning. Those models can learn from training documents and their accuracy can be estimated using validation techniques like cross-validation [73]. As the final decision uses to be that a user is interested or not in one specific item, it is reduced a classificatory problem with just two classes. Nowadays, the development and popularity of machine learning is huge, this has been specially boosted by the successful cases of use of *deep learning*. The next sections are just a small summary of the most used ones rather than a complete collection.

2.2.3.1. Naïve Bayes

This is a probabilistic approach that has been explained in previous sections, but what makes the difference is the variables used for its appliance. As it is a Bayesian classifier, it can evaluate the probability of the two classes $\{c_+, c_-\}$ (interest or no interest) using information from documents. It has a technical problem related to data availability as the documents use to be evaluated only once. That does not mean that cannot be evaluated but the statistical robustness of the predictions is not good. For tackling this problem, it is introduced a known false assumption that considers that all terms from document d are conditionally independent between them. The rational behind this decision is that probabilities of terms can be estimated individually instead of all together, what is very convenient for increasing the sampling used for inference. Several results [74] show that this decision is justified in terms of obtaining good results. Despite there are other specific ML methods that perform better than naïve Bayes, in general it can be considered very efficient and easy to implement.

2.2.3.2. Relevance Feedback

This is a technique using feedback from users for improving the decisions of the recommender systems. This feedback is related to known recommendations proposed previously by the system and allows to tweak it in order to avoid those mistaken recommendations and others similar to them in the future. The way this is implemented with text categorization is a mature algorithm named Rocchio's formula [75]. This option is one of the most used by content-based recommenders for this task.

When working with linear classifier models like logistic regressions, this Rocchio's can be used for rectifying their predictions according to Rocchio's formula:

$$\vec{c}_i\,' = \alpha \cdot \vec{c}_i + \frac{\beta}{|D_+|} \cdot \sum_{\vec{c}_j \in D_+} \vec{c}_j - \frac{\gamma}{|D_-|} \cdot \sum_{\vec{c}_j \in D_-} \vec{c}_j \qquad (20)$$

where α, β and γ are the coefficients that modulate the impact (α) of the original vector $\vec{c}_i$, the set D_+ of the positive feedback documents (β) and the set D_- with negative feedback ones (γ). The vector $\vec{c}_i$ is the original classifier for the category c_i and it is formed by the scalar

values of its weights $\vec{c}_i=(w_{1i},w_{2i},...,w_{|T|i})$. where $|T|$ is the cardinality of the vocabulary T. It is a common choice to simply set α=1.

2.2.3.3. Other methods

This last section related to content-based recommender systems is a small selection of the most important ML models excluding the previous two methods (naïve Bayes and relevance feedback).

Decision trees are widely used [76] modeles (not only in this scope) that provide some benefits like the easiness to understand the underlying logic behind their "intelligence". Those receive the name from their architecture based on nodes with nested branches connected to new nodes in a non-cyclic descending way. The nodes include conditions to evaluate and the different results (usually just a boolean value as *true* or *false*) are connected to one specific branch that leads to the next node. Finally there are reached a *terminal node* that provides a prediction instead of a new condition. The **Figure 8** shows an example illustrating this procedure. Those models are created partitioning the available training data in a recursive way that selects the conditions that minimize the error obtained from the residuals, that is the error from the difference of the model generated values and the observed ones. A similar model is **decision rules** which operates in an close way but use to obtain more compact classifiers than the former ones. That is because these try to explore all the available rules applicable to the data.

Nearest neighbour is another of the widestly used models. These store samples from training data in order to have a reference when comparing new items. This comparison is performed using some kind of similarity function and the nearest neighbour or some of them are used with their classes for determining which label should be applied to a new item. They are very effective in terms of accuracy but from computational perspective these perform lots of calculations as they need to compare every new item with all the pre-existing ones.

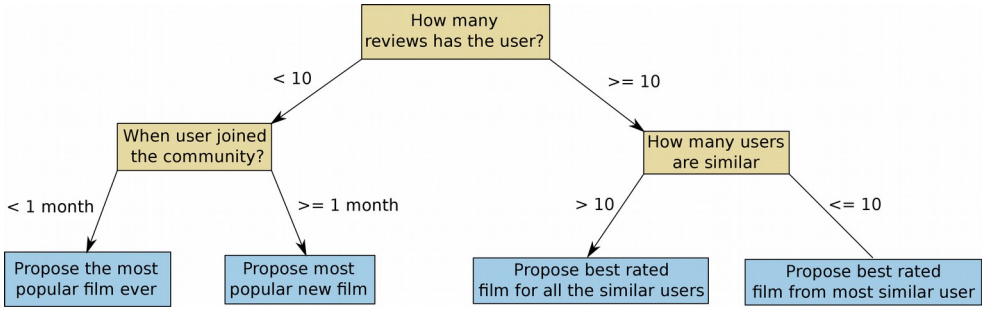

Figure 8. Example of a decision tree for selecting the best recommendation approach.

2.3. Hybrid recommender systems

Nowadays it has become popular to have recommender systems that combine different approaches for obtaining better results. These different approaches include collaborative filtering, content-based or any other that may provide an improvement. *A priori* there is no known reason for limiting any method that can be combined for forming a hybrid method. In general, evidence shows that hybrid methods perform better than single ones. One of the most popular ones combines content-based models with CF recommenders. That is performed using

the CF or the content-based one as base model and then adding the other one's characteristics. Just as sample, one of the problems of content-based recommender systems is the previously mentioned cold start; this can be solved introducing an additional method that compensates this lack.

2.3.1. Hybrid CF and content-based recommenders

As has been introduced in previous sections, the cold start problem usually faced by content-based models can be reduced combining one approach with a complementary method. This is the case of the *content-boosted CF* algorithm that successfully mixes a naïve Bayes for categorizing the content with a weighted Pearson correlation-based CF. It is generated a pseudo rating matrix combining the values from the naïve Bayes and fill the missing values with the latter model's predictions. It not only solves the cold start problem but also is a solution for dealing the sparsity problem. Empirically there are similar approaches like Greinemr et al. using *TANELR* [77] for the content prediction that have proven a performance increase from the error reduction perspective [78].

2.3.2. Models combining CF with other recommender systems

One simple way of combining models is using *switching hybrid recommenders* that are able to combine several predicting models according to certain criteria that ensure the best model is used for every evaluation. *Weighted hybrid recommenders* use several models for predicting and weights for combining their outcome. The way those weights are obtained using different methods like weighted majority voting [79] or weighted average voting [80]. Other methods worth to mention are *meta-level recommenders* [81], *cascade hybrid recommenders* [82], *mixed hybrid recommenders* [83] among others.

As main conclusion about these approaches it has empirically proved that they generate more accurate predictions than the combined methods applied alone. It is worth mentioning that there is a special benefit in the cases of new users or items, where CF methods face more difficulties.

2.3.3. Hybrid CF-only recommender systems

This section covers the hybrid systems formed exclusively by two CF different approaches, more concretely the most important two, memory-based and model based. As shown when introducing earlier hybrid models, the performance of their recommendations use to be better than their isolated components. The following is just a small selection of two of them:

- **Personality diagnosis** combines existing information with some randomness. It is selected an active user from a random pick among all the known ones; this active user represents the one his/her preferences are wanted to be predicted. The obtained preferences of this active user are slightly smothered by introducing a little randomness in their values. This active user is then used for evaluating the rates of different items. A practical application of this hybrid method was investigated by Brees et al. [24] using *EachMovie* [37] and *CiteSeer* [84], confirming the benefits of this CF method combination.
- **Probabilistic memory-based CF** [85] is another hybrid sample with better accuracy than stand-alone approaches like Pearson correlation-based CF or naïve Bayes. For

generating the predictions uses a distribution of all the ratings from known users. Some technical issues like computation effort when having big databases are solved using reduced representative subsets.

2.4. Location-based recommendation

As can be deduced from its name, this category of recommender systems gives an important role to the location information obtained from the user, mainly from devices like smartphones. This information is not only used for profiling the user but also, in the case of physical items, to consider the nearest ones to the requester. These systems are used in location-based social network (LBSN) like *Foursquare, Gowalla, Yelp* and others.

According to some authors [86] there are three main ways of calculating the similarity function: **similar users**, **similar friends** and **geographical distance**.

The similar users way of scoring is decomposing it into a mathematical expression of the probabilities of users visiting places. The function $p(u,l)$ denotes this probability of user u and the location l. Given two users u and v a set L of locations they could be interested, the similarity is calculated as:

$$\text{simil}(u,v)=\frac{\sum_{l\in L} p(u,l)\cdot p(v,l)}{\sqrt{\sum_{l\in L} p(u,l)^2}\cdot\sqrt{\sum_{l\in L} p(v,l)^2}} \tag{21}$$

Every one of the probabilities $p(u,l)$ in equation (21) are obtained from applying previously introduced approaches like user-based CF:

$$p(u,l)=\frac{\sum_{v\in U} \text{simil}(u,v)\cdot\text{simil}(u,l)}{\sum_{v\in U} \text{simil}(u,v)} \tag{22}$$

where U is the set of users. The *similar friends* variant uses the cosine similarity calculated with the common connections (colleagues, friends, etc.) according to some social network/s indicators as:

$$\text{simil}(u,v)=\eta\frac{|F_u\cap F_v|}{|F_u\cup F_v|}+(1-\eta)\frac{|L_u\cap L_v|}{|L_u\cup L_v|} \tag{23}$$

where F_u, F_v, L_u and L_v are the friend and location sets for users u and v respectively. The normalized parameter $\eta \in [0,1]$ modulates the balance of the influences from friends and places. Finally, the geographical distance approach uses it in an inverse proportion, so the closer the distance the greater the similarity.

2.5. Risk-aware

This is a very reduced category for recommender systems where it is incorporated the risk of proposing recommendations that could disturb somehow to the user. This is related to the investment in time, effort or economical that the user could apply for trying a recommendation. *DRARS* [87] is a system that incorporates this risk consideration using the approach of the multi-armed bandit [88] in order to maximize the gain of its decisions.

2.6. Empirical performance

Some authors like Seroussi et al. [108] have performed experiments where several of the previously explained approaches have been evaluated with real data. More concretely, Seroussi et al. propose a MCMU (Multi Class, Multi User) model combining other previous models in a weighted way. Some of the models just use scores from users, but others are able to include textual information from reviews or even additional external sources like e-mails or others. It is not a exhaustive check of the performance of all the available models but provides an understanding of the errors that can be expected from several of the most important approaches. Those results are introduced also for having a reference for evaluating the experiments proposed in the next sections.

	Rating-based	Text-based
All films	AIR	AIT, AIP
Films commonly reviewed	CRR	CRT, CRP

Table 2. Classification of several ways of calculating the similarity.

The **Table 2** contains a summary of the two main kinds of models evaluated by Seroussi et al., depending on if their similarity evaluations of users is based on rates or texts. The rating-based models only use information from the ratings, whilst in the other hand, the text-based models exploit uniquely documents written by the users. The second criteria for dividing these depends on if the similarity is calculated using all the films labelled by the users or it is focused just in those films in common. The following is a brief description of the different models used:

- Pan and Lee's [89] model for single users **SCSU** (Simple Class, Simple User) comparable to content-based recommendation systems.
- **MCMU** combines different classifiers using some weights. This is described by the equation (24), where $r_{f,u}$ is the rate of user u to the film f, μ_u is the average rate of this user, σ_u is the standard deviation of these rates and w_v are the weights associated to the contribution of rates from other users. There are different ways of obtaining those weights, but it is usually based on the use of the similarity function described in equation (25). There the similarity must be greater than a threshold value s_{min} in order have a positive value for the weight w_v.

$$r_{f,u} = \mu_u + \sigma_u \sum_{v \in U} \left(\frac{w_v}{\left(\sum_{v \in U} w_v \right)} \cdot \frac{r_{f,v} - \mu_v}{\sigma_v} \right) \tag{24}$$

$$w_v = \begin{cases} f(\text{simil}(u,v)) & \text{for simil}(u,v) > s_{min} \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

- **All Item Rating** (AIR) uses ratings and evaluates the similarity taking into account to what range from 1 to 10 they belong. If two users u and v have the ratings R_u and R_v, more specifically the $R_{u,r}$ and $R_{v,r}$ represent all the ratings of those users for a very specific value r. The Hellinger distance between users u and v compares them on every range $r \in \{1,2,3,...,10\}$, one by one:

$$\text{simil}(u,v) = 1 - \sqrt{\frac{1}{2} \sum_{(r=1)}^{10} \left(\sqrt{\frac{|R_{u,r}|}{|R_u|}} - \sqrt{\frac{|R_{v,r}|}{|R_v|}} \right)^2} \in [0,1] \tag{26}$$

- **Co-reviewed Ratings** (CRR) compares two users using the cosine similarity or the Pearson correlation only with their common reviews.
- **All Item Terms** (AIT) uses available documents from users for evaluating the Jackard coefficient as metric of their similarity.
- **Co-reviewed Terms** (CRT) uses a similar approach like AIT does but only with the common films that every pair of films review.
- **All Items PSP** (AIP) proposes an equivalent to AIR (equation 26) but using the Hellinger distance between PSP distributions. PSP (Positive Sentence Percentage) is a metric introduced by Pang and Lee that evaluates the percentage of positive sentences among all the available ones in a review.
- **Co-reviewed PSP** (CRP) is based on the Pearson correlation coefficients of the calculated PSP on the commonly evaluated films.

The dataset used for the evaluation is the *Prolific IMDb Users* [108] where different numbers of reviews have been considered, labelled and are used for calculating the similarities and weights, and the rest (unlabelled) are tried to be predicted. Those results can be seen at **Figure 9** where it can be concluded that initially SCSU is the worst performing approach when using little data but systematically reduces its error with a constant increment of available labelled revies. EQW is an specific case of MCMU where all the weights are set with the same normalized value. As can be observed, EQW performs better for small amount of data from users but it does not take so much benefit from increasing the total number of rates used as SCSU.

As can be directly observed at **Figure 9**, the evolution of the error through the amount of labelled reviews evinces how the size of consumed information reduces the error. That happens independently from the used model, despite some models benefits more or less of this. It is remarkable the case of SCSU that is specially sensible to the quantity of information available.

Considering that its performance is systematically improved when using more and more data, it can even cross some boundaries of MAE that the rest of models cannot achieve. It is in the range going from 100 to 200 labelled reviews when it begins to be the most accurate model, outperforming the rest. This scenario cannot even be considered small data as some other models, like EQW, provide a similar result using just 50 of those labelled reviews. The other models use to have closer behaviours, evolving in a more regular way. Independently from how well they adapt to the increase of available information, it is proven that little information has a negative impact in their performance. That it is specially remarkable with the situation of less than 5 labelled reviews, that has not even been considered for review.

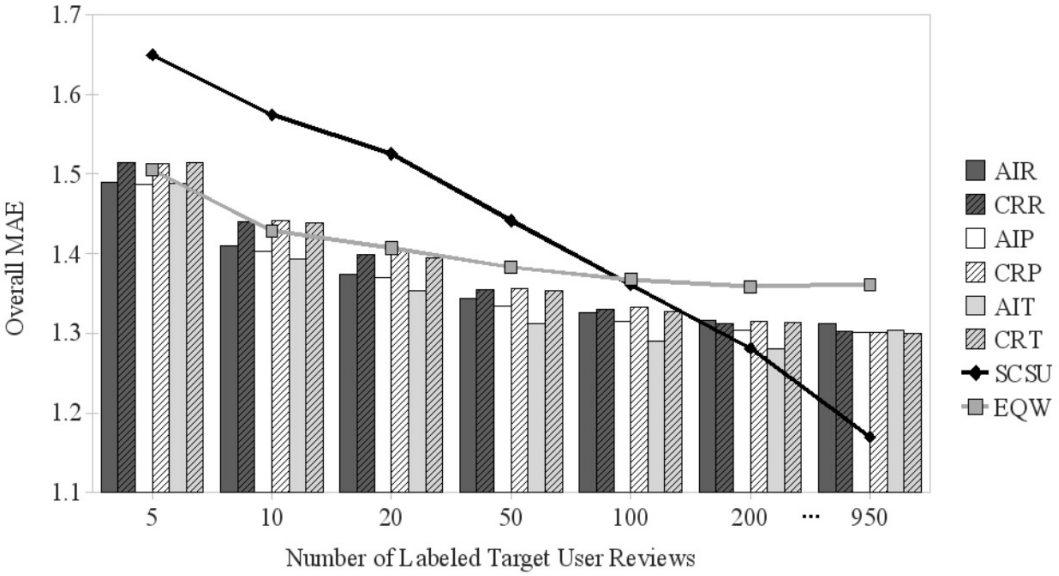

Figure 9. Similarity modelling experiment results for several approaches. This image has been directly taken from [108], showed at page 9 of this document.

2.7. Conclusions

As it has been exposed in the current section, there are several different approaches for solving the recommendation problem and even combinations of them. Some of the most advanced ones also use information obtained from reviews through some NLP that allows to extract and structure the data for generating final metrics about the processed documents. This is an interesting approach for trying to infer a user's rate for items directly from reviews. In general, all the methods introduced have some maturity degree that ensures these are able to be applicable when there is enough data from the users and/or items. It can be said that the main trend is to use as much available data as possible and try to exploit more sophisticated data sources, producing both usually an increment of the solution's complexity.

On the other side there are less mature solutions for dealing with the initial situations where not so much information is known from the users and/or items. This is a situation where every user-item interaction is really valuable because it makes the difference between nothing and and very little information, or between a very little and a bit more of data. For the general cases with a well known community of users and lots of interactions, having one more or less data per user is not going to make a difference, or in other words, this difference is not going to

be considerable. This same situation changes completely when little or no data at all is available for new users or items. It also has been explained why those initial scenarios are important for providing a good recommendation to the users, so they perceive a quality in service.

3. Hypotheses and objectives

This chapter contains the details about the proposal of the current work. For doing that, first of all some used concepts and notations are introduced, and after this, it is presented the frame hypotheses, some inspiration for selecting such a specific approach and, finally, the objectives to be achieved.

As a basic context, must be said the focus of this work is applied just on all those previously mentioned circumstances where there is almost no information about the users or even none at all. Aligned with the more general definition of collaborative filters, this thesis proposes a collection of approaches that try to extract any available information for improving the accuracies of the predictions. Due to the lack of information in those cases, some of the hypotheses proposed are based on a priori knowledge about what kind of results can be expected and also how the proposed model of evaluation works.

The basis general notation used for the proposed approach includes *users*, *entities* and *scores* understood as:

- **User** is every person that interacts with the recommendation system. Their preferences about the different options depend on the user and the domain of the options. The users are the receivers of the recommendations from these systems.
- **Entity** is the generic way of identifying every item/entity/choice that can be offered to the users as alternatives to be considered. These options are the recommended (or omitted from recommendation) objects, depending on some criteria of the expected acceptance from the user to them.
- **Score** is the numerical metric that identifies the user's acceptance/satisfaction/liking to options. It can be only one real expected score from one user to one option.

When users are mentioned, it is always referred as a set of users U of size $|U|$. The same applies to the entities, which are all covered within the set O. The reason for using O (omicron) to define the entities is because the translation of *entity* into Greek is οντότητα. For the scores, as those go from every user u from U to the available entities $|O|$, it is expected have a total size of $|S|=|U|\cdot|O|$ for all the users and entities, but this is just its upper limit. That is because there are not necessarily one score per all the combinations of user-entity; so in general it will be easier express the scores in terms of the users. Every user u_i has associated a set of scores S_i in the form of:

$$S_i=\{s_{i1,}s_{i2,}s_{i3,}\dots,s_{im}\} \tag{27}$$

In general, for the following sections, the use of indices i and j will be used exclusively for referring to the i-th user the former and the j-th entity the latter. Using that notation, every score that is an element of S_i is denoted as s_{ij}. In general, the cardinalities $|S_i|$ may be different for different users u_i and u_k, so $|S_i|\neq|S_k|$.

3.1. Hypotheses

Once the notation and concepts are specified, the fundaments, in terms of hypotheses for this work are summarized and described as:

- **No-a-priori bias.** Given an open community and a set of users participating being part of this, it is not considered any pre-existing bias that could filter the profiles of users in terms of preferring some entities rather than others. That is the likelihood of having one or other user profile and preferences depends only on the empirical data but not in a policy that restricts or bias them.
- **Archetypes.** In this community it is expected to exists similarities between users in terms of preferences and perception of the available entities. There are not expected completely random and equidistant distribution of users, all of them equally similar and distinct between comparing by random users. That is translated in the assumption of the existence of several archetypes representing some of the users interests.
- **Multiple profiling.** The way a user can be profiled is by some traits resembling him/her to the available archetypes. Those traits/similarities to the archetypes are not exclusive, so a user can be define with several or all the archetypes having some of them more or less relevance determining the user. Every user can be considered a specific combination of archetypes.

The reasons and motivation for the previous hypotheses are variated. The *no-a-priori bias* proposes a pure data driven approach where the results cannot be advanced and are completely dependant on the dataset (community) that is evaluated. The *archetypes* in combination with the *multiple profiling* are the basis and foundation of the present exercise and make the main difference with any other pre-existing approaches. As so many other works like the collaborative filtering consider the similarities with other isolated users, one by one, this approach looks for these similarities in abstract groups that simplify their behaviour. At the same time, combining the archetypes for defining unique users also provides a flexible feature model for specifying every user with the required degree of resolution in terms of preferences. On another note, this different traits/archetypes are a easier way of understanding why a user has the preferences it has and why some known entities are indicators of the motivation toward other entities.

For providing a good context and background for the additional details about the proposed approach, it is exposed before two main sources of inspiration for that. Those are presented in the next sections.

3.3.1. Eigenfaces

This is the main source of influence for the proposed method. Eigenfaces [90] was an idea developed by Sirovich and Kirby in 1987 for dealing with the problem of representing human face traits. After them, Matthew Turk and Alex Pentland applied it [91] for face recognition. When working with training images of faces, the dimension of this dataset can be reduced using a smaller basis of images that can be combined for representing the original dataset. The different features of the faces are distributed on these eigenfaces, so every face can be reproduced as a linear combination of those standard eigenfaces. Just as an easy example, let’s suppose a dataset reduced to 3 eigenfaces where the first real human face is composed from

50% of eigenface 1, 17% of eigenface 2 and -13% of eigenface 3. The contribution of the eigenfaces can be negative as some of the features they contain can be found in more than one eigenface. If eigenface 1 has two main features (A and B) but eigenface 2 has only this B feature, the only way of representing the feature A is combining eigenface 1 minus eigenface 3. Despite those are not real samples, these can help to understand why the contributions of eigenfaces can have values not necessarily bounded from 0% to 100%.

The way the eigenfaces are generated is applying *principal component analysis* (PCA) [92] to an initial dataset of human faces. This dataset should be big enough for containing all the features that would like to be captured by the eigenfaces. More formally, when having a set of images $\{\Gamma_1,\Gamma_2,...,\Gamma_M\}$ it can be defined the average face as:

$$\Psi=\frac{1}{M}\sum_{i=1}^{M}\Gamma_i \tag{28}$$

that can be used to redefined the whole set of images one by one as:

$$\Phi=\Gamma_i-\Psi \tag{29}$$

Using PCA, it can be obtained a set of orthogonal eigenvectors $\boldsymbol{u}_k$ and their associated eigenvalues λ_k allowing to form the covariance matrix C as:

$$C=\frac{1}{M}\sum_{I=1}^{M}\Phi_i\Phi_i^T=AA^T \tag{30}$$

The set of weights $\Omega^T=(w_1,w_{2,...},w_{M'})$ that define how a face is composed based on the obtained eigenvectors $\boldsymbol{u}_k$ are w_k and are calculated as:

$$w_k=\boldsymbol{u}_k^T(\Gamma-\Psi) \tag{31}$$

This approach not only reduces the dimensionality of the problem but empirically frequently happens that not many eigenfaces are needed in general for defining a human face. It also uses to happen that the eigenfaces obtained show something resembling a human face. That is because every eigenface could represent some characteristic that can be present in several images of faces but not easily interpreted as a basic human trait. On another hand, it is fairly invariant when it is used for reducing big datasets. Just as a sample, this can be seen in the **Figure 10** (extracted from [91])showing a real extraction of 4 eigenfaces obtained from real experiments.

The way eigenfaces are obtained using PCE is applying an orthogonal transformation to the training dataset in order to extract a set of images that are linearly uncorrelated. That means despite the faces images from the dataset can be reproduced as a linear combination of the found eigenfaces, those cannot be reproduced using themselves. In other words, an eigenface cannot be generated as a linear combination of the rest of them, those are orthogonal.

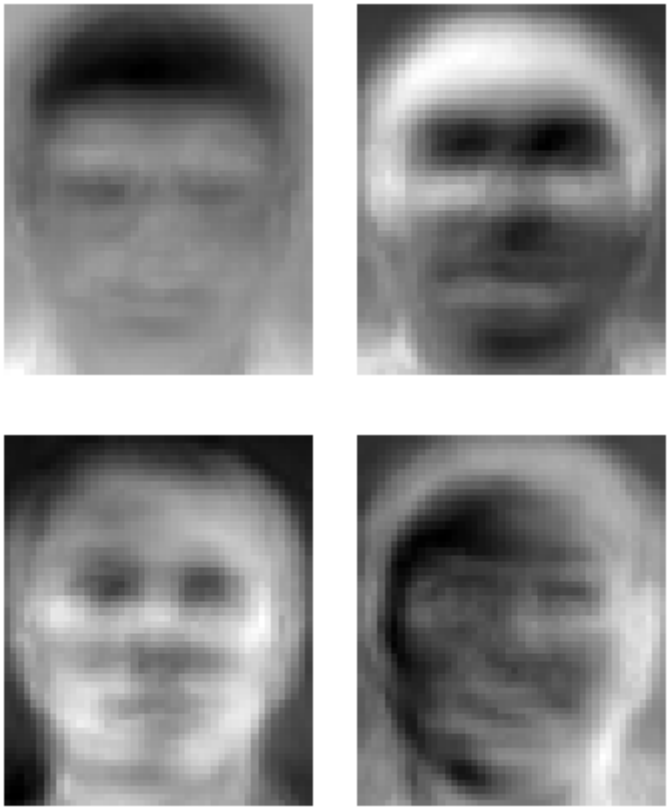

Figure 10. Example of some real eigenfaces. Image extracted from [91]

Despite this technique has been mainly developed and tested within the domain of facial recognition it has other applications like handwriting and voice recognition, medical imaging analysis and other related to communication like voice recognition, lip reading and interpretation of sing language. Because of this historical relation with facial recognition, this area has been specially developed achieving the best improvements there. Anyway, if the test dataset is considerable greater than the generating dataset it is not unlikely to have failures for identifying the new faces. The process of face recognition is performed through five main steps:

1. The training dataset is used for generating the vector base (eigenfaces).
2. Every face of the training dataset is described by the set of weights indicating the contribution of the eigenfaces.
3. New faces are also decomposed into the previous eigenfaces for obtaining the weights of the new samples.
4. Those new weights are compared with weights from training using a distance metric.
5. The closest face in terms of weights is identified as the recognized image.

As mentioned, the applicability range of this approach suggests that it may be used successfully for other domains; at least, those related to human activities where there can be found some kind of correlation between the activity. That applies directly to human interests

where despite the enormous diversity, those are not completely random and can easily fall in some kind of predefined categories.

3.1.2. Quantum superposition

This principle of quantum mechanics has a less obvious and direct influence to the eigenclusters methodology when compared to eigenfaces. In a completely different way compared to what happens in classical mechanics, the quantum state of a system can be represented as the sum of different quantum states. An equivalent situation in classical physics would be the mechanical or electromagnetic waves that also can be formed as the result of combination of several waves of their own nature.

The electron e^- spin (intrinsic angular momentum) [93] is one of the simplest cases that can be used for explaining this. This spin can take only values 1/2 and -1/2 for the electron, and those are used to be represented by the quantum states $|\uparrow\rangle$ and $|\downarrow\rangle$. Using these two states, a more general state for an electron e^- can be defined using two coefficients $c_\uparrow$ and $c_\downarrow$ as:

$$|\psi\rangle = c_\uparrow|\uparrow\rangle + c_\downarrow|\downarrow\rangle \tag{32}$$

These two coefficients in (32) are complex numbers related to the probabilities of finding the electron e^- in one of the two states; more concretely those are $|c_\uparrow|^2$ for the state $|\uparrow\rangle$ and $|c_\downarrow|^2$ for the state $|\downarrow\rangle$. As the electron must be detected in one of these to states, the total probability has to be 1, so $|c_\uparrow|^2 + |c_\downarrow|^2 = 1$ is a mandatory constrain. When generalizing this to more complete scenarios and other particles, the states can be defined as:

$$|\Psi(t)\rangle = \sum_n C_n(t)|\Phi_n\rangle \tag{33}$$

In equation (33), $|\Psi(t)\rangle$ represents the state of the particle at any time t and the different coefficients $C_n(t)$ again take complex values. This definition (33) is only valid for the cases where n takes discrete values. In an equivalent situation comparable to the electron's example, the n different coefficients evaluate the probabilities of finding the system at the state $|\Phi_n\rangle$ through the time. These probabilities are calculated as $|C_n(t)|^2$. Taking into account the constrain about covering the total probability of detecting the system with one of the different states, it must be satisfied that:

$$\sum_n |C_n(t)|^2 = 1 \tag{34}$$

The physical interpretation of this quantum superposition and the probabilities is the interesting point for the proposed approach of eigenclusters. There are several states where the system can be found; those states can be understood as a description of the system, and the right values of the coefficients $C_n(t)$ determine any prediction to be done about the system at any future time *t*. In the quantum theory, the way of obtaining the states is solving the Schrödinger equation [94]:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle=\hat{H}|\psi(t)\rangle \tag{35}$$

In this Schrödinger equation (35) the operator $\hat{H}$ is called the Hamiltonian and represents the total energy of the system; it sums the kinetic and potential energies for all the particles contained in the system. Despite that when working with eigenclusters the way those are obtained is not based in this approach at all, when there is little information about the users the different clusters representing them can be seen as the observed profiles that can describe the users behaviours. There is a parallelism between the clusters and the states as they both are the different possibilities that a user or a system can show. In the case of clusters is clear that users are intended to be profiled being influenced by more than one cluster at one; what is exactly the equivalent of the quantum superposition principle. That would became more important when tackling those cases where there is very little or not at all information about the user and using probabilities is the way to statistically try to reduce the error of the predictions. The equivalent of coefficients $C_n(t)$ in quantum mechanics would be interpreted in the eigenclusters approach like the weights w_{ik}, despite the later ones do not depend on time t. Finally, the last difference is those weights can be only sometimes obtained from a probabilistic approach, but in general do not have a probabilistic interpretation, so there is no need to apply a probability constrain like (34) to them.

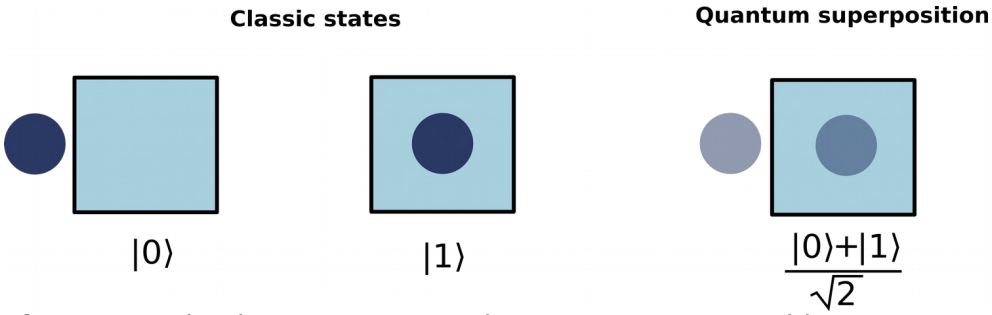

Figure 11. Classic states compared to quantum superposition.

The **Figure 11** shows a conceptual explanation of the quantum superposition for an easy understanding of the concept at high level. For the classical mechanics the state of a system is a perfectly determined situation that lacks of any ambiguity. Oppositely to that, in the world of quantum mechanics the final state can be a combination of more basic states. It can be even found a different set of convenient base states for representing the combined states. The **Figure 12** shows how this can be interpreted when working with two wave functions associated to a pair of states. The final wave function is itself a linear combination of the two basis wave functions in the proportion of the coefficients associated with the states. Regardless this interpretation is fully accepted, it has been historically discussed by some important physicists like Albert Einstein or Erwin Schrödinger. This later scientists created the paradox of the Schrödinger's cat as a way of criticizing this concept of quantum superposition. Basically, this critics presents a hypothetical scenario where a cat staying into a closed box is under the superposition of two clear states. In a very simplified version, into the box there is also a poison able to kill the cat and this can only be activated by an atom in one specific state. For instance a decaying atom emitting a concrete radiation measured by a Geiger counter. As this (hypothetical and simplified) atom can be in both states at once, the poison can be liberated or

not, producing a simultaneous reality where the cat is also in a superposed state of being dead or alive.

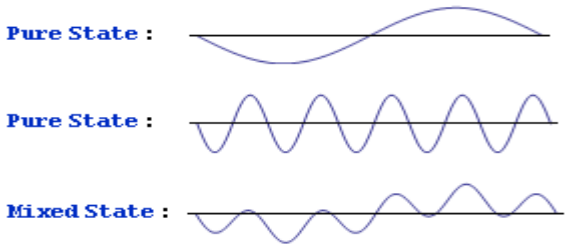

Figure 12. Quantum superposition of wave functions from basic states.

3.1.3. Eigenclusters

This is the proposed method in this work. As has previously explained, it is a variant of collaborative filtering that introduces some new ideas about how to profile users, even when there is very reduced information about them. Some CF approaches use direct information about the closest neighbours in terms of similarity and some of the more sophisticated CFs apply some kind of segmentation that identifies every user with one specific group or cluster depending on the user's preferences. This assumption, proven to be statistically right, can be generalized starting from the point there exists at least a cluster that represents the user. Despite this basic assumption seems sensible, it constrains the number of clusters to profiling the user to just only one. *A priori* there is no further justification for selecting just one cluster; this is an assumption that simplifies the algorithm but not necessarily reflects the reality. This proposal explores the fact that a user can be profiled as a linear combination of different categories. In other words, a user can be identified with one group of users to some degree, but it also can be identified to other groups sharing other preferences with them. Similarly to a multidimensional space where a point is determined by several coordinates that perfectly define it, this different grade of identification of a user with every one of the clusters can be seen as a vector that identifies every user with the right precision. The selected name for this method, **eigenclusters** is inspired in the *eigenfaces* approach explained earlier. For both cases a dimension reduction is applied for representing the training dataset.

3.2. Objectives

Given an open community of users where there are several entities that can be offered to them our methodology eigenclusters has the aim of predicting the expected scores these users would give to that entities. Those predictions are expected to have an acceptable accuracy and be purely based on the hypotheses introduced previously. It will be also added some assumptions and constrains, fully compatible with the mentioned hypotheses, in order to improve the overall performance of the predictions.

The main goal when introducing this method is to provide a new perspective and a different way of solving the problem of recommendation. Because eigenclusters propose an intuitive methodology that also minimizes the requirements of information needed for applying

it, this is intended to be a good basis, a corner stone upon it can be built more sophisticated systems that use it and (maybe) combine its results with other complementary methods.

It is not intended to challenge the state of the art of the most outperforming models, but rather than that just achieve a reasonable good results. Those general results evaluated in some standard datasets do not have to provide the least error but one that is acceptable in real environments.

One specific objective for this methodology is being able to generate results with reduced errors in those specific cases where the information is really reduced. These results should be small enough for a system in production being able to provide active recommendations with clearly more successful cases than mistakes.

The last of the objectives is proposed to be achieved through the fulfilment of the previous ones, using them as a proof of the viability of the general proposal, the existence of archetypes being able to be decomposed into eigenclusters.

4. Our proposal: A recommender system based on eigenclusters

As a remainder of what has been exposed previously, a simple way of understanding eigenclusters is to consider that there are some user archetypes that represent the main different behaviours of users. Each one of those archetypes may have more or less users identified by it, so some profiles could be more popular than others. These would be applicable to lots of users whilst other archetypes only represent some minor groups. There is no initial restriction about the size of this clusters in terms of users, so they can have similar number of users or not at all. Here lays the main difference to other statistical approaches, what has been shown in the state of the art is a set of methods that predict the preferences/rates by averaging the closer neighbours, one or several of them. This method is more focused on the qualities or traits that define them as a group instead of their individual behaviour. Those qualities are captured by the clusters and the user's preferences/rates are then evaluated by this cluster influence rather than individual contributions. It moves the contribution from closest users to contribution of closest clusters. These clusters can be identified as an ideal profile, not a real one; in other words, something that defines one user with a very extreme tastes. It seems very rare to happen that there could be any user only defined by one cluster and to some degree maybe there can be more inclination to belong to one cluster than to other, but this purity in behaviour seems very unlikely to happen in a absolute way as people are not 100% equal ones to the others, even in the cases where people share lots of treats in common; there is always some kind of difference.

From the mathematical point of view, this approach can also be interpreted as a compensation to the possible dataset bias. There is no reason for thinking that all the different trends, interests and opinions are equally represented. That means that some groups with some specific preferences can be more or less eager to share their opinion. That can happen by social discrimination; when someone has unpopular preferences from the point of view of social acceptance, exposing those interests to others can have some negative reaction from antagonist groups. It is not clear how those interactions work but it is reasonable thinking that it can be expected some effect in the distribution or presence of different opinions, tastes and points of views. Translating this to statistical terms means that there could be an over-representation of interests that cannot be applied to the majority of users, even if those interests are the majority. On another note, it also may happen that important groups in terms of presence would have very few recorded interactions representing they interests. If that unbalance between recorded interests and real users' interests is present, working with archetypes identified with some groups of interests can help to balance back the predictions. That works when predicting using clusters instead of total number of recorded interactions.

According to this introduced logic, another point for taking into account is the total number $|C|$ of defined clusters. Let's suppose there are really a set A of archetypes with cardinality $|A|$, but the chosen cluster set C is smaller, so $|C| < |A|$. In that case, some of the archetypes are going to be distributed among other clusters, it is not going to be a cluster per archetype. When that happens, the cluster representing several archetypes is going to be an weighted average of the profiles of the different users described by these archetypes.

Obviously, the only way for those archetypes to be represented properly is choosing a value of $|C|$ that at least is as great as $|A|$.

For the previously explained cases of a user belonging to only one of the clusters, a very common representation of him/her is using a vector $\vec{w}=(w_{1,..}, w_k)$ where every $w_i \in \{0,1\}$ and all of them satisfying the following constrain:

$$\sum_{i=1}^{K} w_i = 1 \tag{36}$$

With the eigenclusters approach this representation is still valid, but rather than restricting $w_i \in \{0,1\}$, the new paradigm is defined mainly as $w_i \in (0,1)$ and, despite the extreme values 0 and 1 are (in theory) allowed, technically those are seldom seen. This statement obviously excludes the case where there exists only one cluster. This approach is inspired in two main precedents used with successful results, **eigenfaces** and **quantum superposition**. Finally, it also will be shown some approaches where it is applied even less restricting constrains like $w_i \in (-1,1)$ just as a sample.

As an advanced of the next sections, eigenclusters is more similar to eigenfaces than quantum superposition. Like in the former, clusters are used and combined as representative traits of the user that specially define him/her. In quantum superposition, when a measure is performed, only one of the states is finally observed and it becomes the new state describing the system. Anyway, in our approach several questions are raised about those clusters:

1. How can those clusters be found?
2. How much a user should belong to each cluster?
3. How these weights of users belonging to different clusters can be articulated for generating predictions?

From the last three questions, the last one is the only one that can be defined in a very specific way that will be always the same for all the cases. Knowing that helps to answer the first two questions. Intuitively, every cluster can make a prediction for one or several entities; there is no limit for how many entities a cluster can evaluate. Let's define a set of clusters named as C and with cardinality $|C|$. For every cluster c_k belonging to C where $\forall k \in [1,|C|]$ there is an **associated score set** named Z_k:

$$Z_k = \{z_{k1}, z_{k2}, z_{k3,}..., z_{km}\} \tag{37}$$

In (37) the value m is the cardinality of the set of entities $|O|$, so $|Z_k| = |O|$ and is a constant depending on the size of the used dataset. In previous studies where every user may belong only to one cluster, the different values $\{z_{k1}, z_{k2}, z_{k3}, ..., z_{km}\}$ can be interpreted as the scores that every user belonging to this k-th cluster would apply for every entity. In the actual case, where every user may belong to several (or even all) cluster to some degree, this definition gets smoother. It can be still interpreted as the scores coming from the cluster c_k, but more concretely it is the contribution of cluster c_k to the expected scores for every of its

users. For gathering all this contributions together in one final predicted score it is needed to define how much every user belongs to every cluster.

The way a user is categorized by the clusters in C is given by $\vec{w}=(w_1,...,w_{|C|})$ and with those vectors it is defined the weight matrix **W** with dimension $|U|\times|C|$. This matrix defines every user per row and every k-th column is the weight (equal to the user's w_k) that identifies the user with the cluster c_k. According to this notation, every user u_i correlated to cluster c_k now defines a double-indexed weight w_{ik}, so every user's profile can be defined as:

$$u_i=(w_{i1,}w_{i2,}w_{i3,}...,w_{i|C|}) \tag{38}$$

Now, (37) and (38) can be related to predict the final scores of user u_i for any j-th entity. Let's be $\hat{y}_{ij}$ this final evaluated score, it is calculated according to:

$$\hat{y}_{ij}=\sum_{k=1}^{K} w_{ik}\cdot z_{kj}+\alpha_i \tag{39}$$

According to (39), depending on the values of w_{ik},z_{kj} could be interpreted as a score or a contribution to the final score (a partial score from the clusters). This interpretation helps to understand the different values that w_{ik} and z_{kj} may have. Intuitively it can be expected that w_{ik} are normalized but despite it would bring some elegance in the approach, it is not mandatory. It can be easily understood that some values could be 0, but nothing prohibits having negative values. Further explanation on this cases will be introduced in the next parts. On another note, despite that z_{kj} would seem to be reduced to the range of the different values of the scores, once again, *a priori* there is no restriction what values those may have. What it is clear, is that according to (39) the election of w_{ik} and z_{kj} needs to provide the most accurate values for $\hat{y}_{ij}$. Despite of the different interpretation for the values z_{kj}, for simplicity, the different vectors Z_k will be always named **cluster scores vector** and it will be always a vector Z_k associated to every cluster c_k.

4.1. How to generate clusters

There are several techniques that can be applied for clustering. Some of them can seem more logical a priori than others, but for this specific exercise the **optimal clustering** method would be the one that maximizes the accuracy of the predicted preferences using all the clusters. As the clusters are defined for all the users, and are going to be immutable, its definition is going to affect them all. It is also remarkable that for defining the optimal clustering, first it must to be known how users are correlated to the clusters, and how the user's information about the cluster is going to be used for prediction. This is the **fitness function** that will be defined latter. A perfect optimal clustering algorithm would begin with the whole population of users U, distributing them in different clusters and selecting the one that

maximizes this fitness function. If $|C|$ is the total number of clusters created, and $|U|$ the total number of users, it should be always considered that $|C|<|U|$. It does not make any sense to have more clusters than users. If that would be the case, in theory it may be generated a unique cluster per user that only defines this user's preferences. That situation would break the principle of collaboration because if every user only needs one cluster to define him/her, because this cluster is adapted to his/her needs there will be no other users belonging to it. Other users also would have their own totally customised cluster and lead to use clusters with only the information of the user they define. Knowing that, $|C|<|U|$ will force the users to share their clusters, it can be explored different values of $|C|$ that satisfy this constrain.

Even considering every user can belong to different clusters, it would be nice to define clusters that identify one specific profile as much differentiated from the others as possible. The rational behind that is equivalent to find an orthonormal vector basis; despite there are several equivalent elections, selecting a right one can simplify the representation of any element of the subspace. In this case, the elements are the different users' profiles, more concretely their preferences.

Because clustering based on maximizing the accuracy of the fitness function comes with a heavy computational cost, it is specially interesting look for some kind of heuristic method for segmenting the users able to provide good enough results. In the next sections it is going to be exposed several intuitive ideas for clustering the users.

4.1.1. Partition sorted by average scores

This is a basic distribution of the entities for defining the c_k clusters instead of using the users themselves. Given all the average scores for every entity j, those are sorted in descendant order and distributed within the K different buckets. The value K is equivalent to $|C|$. This distribution is a partition, so every entity is associated to only one cluster. This can be intuitively understood as a categorization of entities in groups of same average score, from top rated to the lowest rated ones. Every cluster c_k has then an associated constant number of $|c_k|$ entities except the last one $|c_K|$ according to:

$$|c_k| = \left\lfloor \frac{n}{K} \right\rfloor \tag{40}$$

$$|c_K| = n - K\left\lfloor \frac{n}{K} \right\rfloor \tag{41}$$

Looking for the same number of entities for every cluster, $|c_k|$ is the entire part of dividing n per K. As n may not be a multiple of K, this difference is gathered in the last cluster $|c_K|$ that captures the remainder of the division *n/K*, modulo operation [95]. The set of entities associated to every cluster c_k defines them as $c_k = \{o_{k1}, o_{k2}, o_{k3}, \ldots, o_{k|c_k|}\}$ with the previously determined cardinality. The different elements o_{kl} represent the *l-th* entity belonging to the cluster c_k. The way these entities o_{kl} are identified comes from the partially ordered set

[96] $S=\{o_1,o_2,\ldots,o_{|S|}\}$. Those are sorted using as ordering criterion their average score, so o_1 is always the entity with the highest average score, whilst o_m is always the entity with the lowest average score. The formal definition for the binary relation ≤ for comparing two entities o_a and o_b is evaluated as their average score from all the users. That is $o_a \leq o_b$ is true only if the average score of o_a is equal or lesser than the average score of o_b. From this partially ordered set S the first $|c_1|$ entities belong only to c_1, while the next $|c_2|$ entities belong only to c_2. This sequence ensures that o_1 always belongs to the first cluster c_1 and o_m always belongs to the last cluster c_K. The associated cluster scores vectors Z_k have their elements $Z_k=\{z_{k1},z_{k2},z_{k3},\ldots,z_{k|S|}\}$ defining every z_{kj} according the following evaluation:

$$z_{kj}=\begin{cases} avg(f_j) & \text{if } f_j \in c_k \\ 0 & \text{if } f_j \notin c_k \end{cases} \tag{42}$$

As this method uses a partition for the entities (42), every entity is evaluated only by one cluster. Different users will predict the same entities using their associated clusters, just weighting by their personal values according to (39).

4.1.2. Partition sorted by standard deviation of scores

This distribution is very similar to the previously introduced *partition sorted by average scores* but with the difference of using the standard deviation of the entities scores instead of the average of the scores for sorting the entities. The number of clusters K and their associated cardinalities $|c_k|$ defined in (40) and (41) and everything else is exactly the same as described in the *partition sorted by average scores* except the sorting criterion, defined by the binary relation $\leq$ with a different evaluation. Given two entities o_a and o_b, let's define their respective standard deviation for their scores as σ_a and σ_b. The binary relation applied as $o_a \leq o_b$ is only true when $\sigma_a \leq \sigma_b$ is also true, and false otherwise.

The way the different components z_{kj} for the associated vectors Z_k are defined is also the same as it is described in (42).

4.1.3. Binary partition sorted by average scores

This way of obtaining the clusters is, once again, equivalent to *partition sorted by average scores*, but with one big difference on how the vectors $Z_k=\{z_{k1},z_{k2},z_{k3},\ldots,z_{k|S|}\}$ are evaluated. Instead of using (42), the next criterion is applied:

$$z_{kj}=\begin{cases} 1 & \text{if } f_j \in c_k \\ 0 & \text{if } f_j \notin c_k \end{cases} \tag{43}$$

The rational sustaining this segmentation approach (43) is that every cluster c_k will contribute with the same value 1 for all the entities included within it; or 0 otherwise. That means several entities with a very close average scores fall within the same cluster c_k and the

prediction for a given user is be the same for those entities. As it is also a partition, all the entities within c_k share the same prediction for a user despite it will be different predictions for different users because of the weights and (39). This is because the contributions of any other cluster that is not c_k is zero because (43), so every entity is evaluated only using one cluster.

4.1.4. Binary partition sorted by standard deviation of scores

This is also based on a previously introduced clustering method, concretely on *partition sorted by standard deviation of scores*. The relation is equivalent to the one between *binary partition sorted by average scores* and *partition sorted by average scores*. In this case all the way the clusters are calculated is quite equivalent to *partition sorted by standard deviation of scores* but, exactly as happened in the previously described *binary partition* the way the vectors $Z_k=\{z_{k1},z_{k2},z_{k3},...,z_{k|S|}\}$ are evaluated is based on the criterion (43) not the (42). The rational behind this approach is also equivalent to the previous *binary partition*: it is used a criterion for sorting (standard deviation of scores) but all the entities associated to a cluster c_k are scored with the same value for a same user.

4.1.5. Clustering by users

This a conceptual idea rather than a concrete method. It is based on clustering the users in different groups according a similarity criterion. These clusters c_k would then define groups of similar users where this similarity can be extrapolated to their preferences. Knowing how much a user is related to the different clusters is used for predicting this user's preferences using the function (39). In this approach, the clusters are defined by the users they represent, this is the main difference with the previous four clustering methods.

There is a concept that is worth to be highlighted: how a user belongs to a cluster, and that is different depending to the situation. It has been explained previously that every user belongs to several clusters but in this specific methodology the goal is to define the clusters based on users. This creates a recursive situation where the user and the clusters may fall in a endless spiral of definition based on each other. The proposed way for solving this is starting from something known, information about the users, those users including their information are named as **clustering users** or *c-users*. This c-users are the basis and foundation for *clustering by users*. Once this clusters are generated, the **profiled users** or p-users are the ones that may belong to several clusters at once as described in (38) by the vector $u_i=(w_{i1,}w_{i2,}w_{i3,}...,w_{i|C|})$. This process is divided in two steps: the first step is using the c-users for generating the clusters, the second one is using the clusters for determine the p-users. This different naming for the users it is useful to determine at what state the user is, either defining the cluster or being defined by the cluster.

Independently of how to distribute the c-users within the different clusters, it must be established a method for generating the different score vectors Z_k associated to every cluster c_k. For doing that let's define U_k the set of *c*-users $U_k=\{u_{k1},u_{k2},u_{k3},...,u_{k|U_k|}\}$ defining (by belonging) a cluster c_k. As all this users U_k define c_k, the score vector Z_k can be obtained from the scores of these users. Considering every user with the same importance/weight into the cluster c_k, then all them must be considered equally; that means the different scores for the entities must be averaged for having the cluster consensus for them. This cluster consensus

defines then the vector $Z_k=\{z_{k1},z_{k2},z_{k3},...,z_{k|S|}\}$ where every c-user u_{ki} contributes with the scores of the entities reviewed. It is important to mentioned that not all the c-users u_{ki} have evaluated and scored the same entities. It could even happen that there are only a reduced set of entities seen and evaluated by these users; if these users are similar it is not unlikely they have pretty close interests. This can be mathematically defined as:

$$z_{kj}=\frac{1}{n}\sum_{i=1}^{n} s_j(u_{ki}) \tag{44}$$

$$z_{kj}=\frac{1}{|c_k|}\sum_{i=1}^{n} s_{jki} \tag{45}$$

Both equations (44) and (45) are equivalent ways of expressing the evaluation of the cluster's score z_{kj} using different notations. In the expression (44) $s_j(u_{ki})$ is the score of c-user u_{ki} (user *i*-th from cluster *k*-th*)* to the entity identified with index *j*; this $s_j(u_{ki})$ is a function of u_{ki}. For the expression (45), s_{jki} is the result of this evaluation $s_j(u_{ki})$, s_{jki} is exactly the score of user u_{ki} to the entity associated to the index *j*. Independently of what notation is used, both (44) and (45) define every score z_{kj} to the entity *j* from a cluster c_k as the average of its values only from the c-users of the cluster; the rest of c-users do not contribute to this score.

4.1.6. Clustering using K-means

This is a classical but also very popular machine learning algorithm for segmenting populations of any kind of items; those only have to be defined by their attributes. The benefit of this algorithm is that is generic, in the sense that can be applied independently of the nature of the problem to be tackled. That does not mean it can be used systematically for any kind of problem, but it has been proved to be useful in a very wide range of problems with different natures [97].

K-means uses a distance function for evaluating the similarity/difference/distance between the elements to be compared. This function can be specified for special purposes but it is a common choice using the euclidean distance between elements. The way this is performed is characterizing every element as a multidimensional vector of n dimensions, where every one of those dimensions correspond to one of its attributes. These attributes can be continuous values or categorical ones. In the last case, the way an item is defined as belonging or not to a category is using values like 1 or 0.

The specific case to be faced is the clustering of a set of c-users depending on the scores of the entities they have reviewed. Fore every user u_i the c-user is represented by his/her known scores for all the entities:

$$u_i=(s_{i1},s_{i2},s_{i3},...,s_{im}) \tag{46}$$

It is important to differentiate the alternative ways of representing a user. On one hand the vector (46) identifies a specific user with his/her scores for the entities, but on the other hand the vector (38) does the same profiling u_i with the clusters instead of the scores.

In (46), m is the cardinality of the entities set $|O|$, so there is a score for every entity. That represents a problem because there may be entities not evaluated by a given c-user u_i but there must be a value for those in order to apply the similarity function with the k-means. As the similarity function is the euclidean distance, this function applied to two users u_i and u_j is checked as:

$$|u_i - u_j| = \sqrt{\sum_{l=1}^{|O|} (s_{il} - s_{jl})^2} \tag{47}$$

When evaluating (47) the partially unrated entities constitute a problem, that is the entities rated by one user but not by the other. If an entity is rated equally from u_i and u_j the difference between those $s_{il} - s_{jl} = 0$, so there is no increment of the similarity (47) because of that. This same rule can be applied to a completely unrated entity that neither u_i nor u_j have rated it. This can be implemented by excluding this rate from the sum or using one specific value for the unrated entities. This value is not going to make any difference because $s_{il} - s_{jl} = 0$. A completely different scenario happens when there is a partially unrated entity, this happens when one of the c-users (let's say u_i) has scored the *l-th* entity with s_{il} but the other c-user u_j has not. In this case, the default value used for representing the unrated scores from user u_j is going to have an impact to $s_{il} - s_{jl}$ and consequently to $|u_i - u_j|$ according to (47). There has been explored two entities for establishing the default value for an unrated entities and those are exposed in the next sections.

4.1.7. Null default value

This is the first approach proposed for dealing with the default values for unrated entities when clustering by users using a k-means algorithm. The rational behind this is considering that having or not scored an entity is a preference in itself. The assumption is that users see (and rate) the entities they expect that they are going to like. Obviously, these expectations may be wrong and finally they could see an entity with a difference experience, disappointing and then scoring poorly or just the opposite, better than expected applying a high score.

The numerical implementation of the null default value is based on using the zero value for unrated entities. This value uses to be so different to the averages scores that guarantees a considerable difference for the cases $s_{il} - s_{jl}$. If it is assumed that u_i is the c-user who has scored the entity and u_j has not, then the previous expression is reduced to s_{il}. Comparing this scenario to the cases where there is a score for both c-users u_i and u_j, and considering that always $s_{il} > s_{il} - s_{jl}$ it is easy to see that this rule penalizes the comparison of partially unrated entities. The null default value forces the cases where there are partially scored entities having a similarity (fitness function) greater than the hypothetical case where these same users would have scored the same entities. This behaviour is a way of forcing the k-means algorithm to consider the users that have scored the same entities more similar than the ones that do not. This, intuitively, can be understood as facilitating the k-means to generate clusters with c-users that have seen the same entities, which may be the preference pattern to validate.

4.1.8. Average default value

This alternative to set the default value is based on a different assumption. This considers that for every c-user their average score is the most likely expected value for an unrated entity. Despite that scoring an entity depends on several factors, it assumes there is a personal trend for every c-user to rate within his/her specific personal range. This specific range is reduced just to its average value, that is every user u_i has an average score $\bar{s}_i$. When two c-users u_i and u_j are evaluated by the k-means (using the euclidean distance), the contribution of the entities not rated at all will be evaluated according to (47) as $\bar{s}_i - \bar{s}_j$. If the are n entities not rated at all by any of both u_i and u_j, from (47) it can be conclude that their similarity will be not less than:

$$|u_i - u_j| \geq n \cdot |\bar{s}_i - \bar{s}_j| \tag{48}$$

For the partially rated entities, the contribution of the l-th (if the rating c-user is u_i) is $s_{il} - \bar{s}_j$.

4.1.9. Cluster densification

One of the problems associated to the previous methods for obtaining the scores matrices is their sparsity. This sparsity may appear when the common number of entities reviewed by several users is low. As it is shown later, one of the datasets used has this specific characteristic. Mathematically speaking, this sparsity is noticed in clusters where there are entities not rated at all by any of their c-users. In those cases, the cluster's proposed score would be zero. Taking a look at (39) and considering some cases with small number of clusters $|C|$, when there is no score at all for the j-th entity in the cluster c_k, there is no contribution (score is 0) from this cluster. That impacts the final evaluation of $\hat{y}_{ij}$ reducing its value. When there are several of those clusters without an evaluation of the j-th entity, the impact on the value $\hat{y}_{ij}$ is stronger. The lack of score for an entity from a cluster c_k can be understood as rating it 0 from this cluster. The rational behind this cluster densification is that clusters without a score for an entity should have a neutral contribution to the evaluation of $\hat{y}_{ij}$ as much as possible. The way this is articulated is replacing the null value in the cases where there is no score from a cluster with a value. That is, for every cluster c_k having an associated scores vector Z_k the replacement of the unrated values from 0 to some **default value**. Not all the zeros, just the ones coming from a lack of scores.

One of the options for this default value is looking for a neutral one, but this desirable neutrality has to be checked as being feasible or not. According to (39), every cluster c_k contributes to $\hat{y}_{ij}$ with a proportion weighted as w_{ik} which is different for every p-user. That makes impossible to select a value that does not affect $\hat{y}_{ij}$ so absolute neutrality cannot be guaranteed. On another hand, the clusters and their scores vectors are introduced for contributing somehow to the evaluation of the p-user's scores instead of avoiding them. Taking those last facts into consideration, the final default value will not be neutral but a committed one, like any other within Z_k. According to that, the proposed solution is to use a default score

from the rest of values z_{kj} belonging to $Z_k=\{z_{k1},z_{k2},z_{k3},\dots,z_{k|S|}\}$. Applying a partition to Z_k, its elements can be split into two different subsets, V_k for the evaluated/labelled scores and X_k for the unevaluated/unlabelled ones. The two subsets V_k and X_k must satisfy the following relations:

$$\begin{gathered} V_k \subseteq Z_k \\ X_k \subseteq Z_k \\ V_k \cap X_k = \emptyset \end{gathered} \tag{49}$$

All the v_{kj} values from V_k are used for determining the elements of X_k. In fact, all the elements of X_k are evaluated with the same value $\bar{z}_k$. This value is obtained averaging all the scores from V_k, as:

$$\bar{z}_k = \frac{1}{|V_k|}\sum_{l=1}^{|V_k|} v_{kl} \tag{50}$$

The rationale to use (50) for obtaining the default values $\bar{z}_k$ for every Z_k is that an average value from the known scores is going to be the most accurate value that can be obtained considering no addition information. At least, it is easy to assume it may be a better value than just zero, which is a specific score very biased from the habitual values.

4.2. Relating users to clusters

This is the process of **profiling** the p-users by their proximity to every of the clusters C. In other words, generating the weight matrix **W** that contains all the relations between users and clusters . This matrix defines every p-user per row i whilst each column k contains the weight w_{ik} that identifies the p-user u_i with the cluster c_k. According to this definition the dimensions of the weights matrix **W** are $|C|\times|U|$ and there are lots of ways of obtaining them. Because the weights w_{ik} are continuous values, they *a priori* could have any value, so their combinations are infinite. According to this, there is no right way of obtaining the weights but the most convenient according to one specific purpose. In the present problem, the ultimate goal is to maximize the accuracy of the predictive model for $\hat{y}_{ij}$ (39). The prediction $\hat{y}_{ij}$ is the estimated score that user u_i would apply for the j-th entity, and the closer it gets to the user's real preference $\hat{y}_{ij}\approx s_{ij}$ the better the model is. This goal determines how measure an optimization of the matrix **W**, but it does not completely determines the way the weights w_{ik} may be obtained. On the next sections it is explained some of the methods proposed for finding those weights and the rational behind them.

4.2.1. Linear regression

This is the simplest and most direct method for evaluating the weights of the matrix **W**. Taking into account that (39) is a linear model [98], for every p-user u_i depending on the

regressor variables s_{jk} and the Y-intercept α_i, the problem of finding the weights w_{ik} can be understood as a linear regression to be fitted in order to maximize the $\hat{y}_{ij} \approx s_{ij}$ accuracy. The way these weights are fitted is using the method of *least squares* [99], a standard approach for obtaining the most approximated solution to these problems. There is calculated a regression for every p-user u_i.

It has been explored two variants to this approach depending on the treatment applied to the Y-intercepts α_i. This intercept can be evaluated directly from the linear regression or can be forced to be zero for some specific purposes; both cases have been studied because they can be interpreted in different ways. For illustrating the meaning of the Y-intercepts α_i let's explore the extreme case where there is no one single cluster, that is $|C|=0$. This no-clusters case reduces the model (39) to just $\hat{y}_{ij}=\alpha_i$ its intercept; this can be interpreted as every p-user u_i will be predicted with an average score $\alpha_i=\bar{s}_i$. As it was mentioned earlier, every p-user has his/her own score range where falls the majority of his/her scores. This score range is like a fingerprint that helps to identify the p-user and also improves the predictions. The minimum representation of this range is the average score $\bar{s}_i$ obtained from all the entities the user has reviewed. When there are some clusters $(|C|>0)$, the Y-intercepts α_i from (39) can then be understood as the personal bias towards some specific score.

4.2.2. Saturated linear model

One of the problems when working with linear regressions is there are no upper or lower limits to the values they predict. It could predict scores above the maximum logic value 1 or even negative values. Obviously, those values never are going to be found when evaluating the prediction and can be fixed with a very simple correction on the prediction boundaries. This fix is the saturated linear model depicted at **Figure 27**.

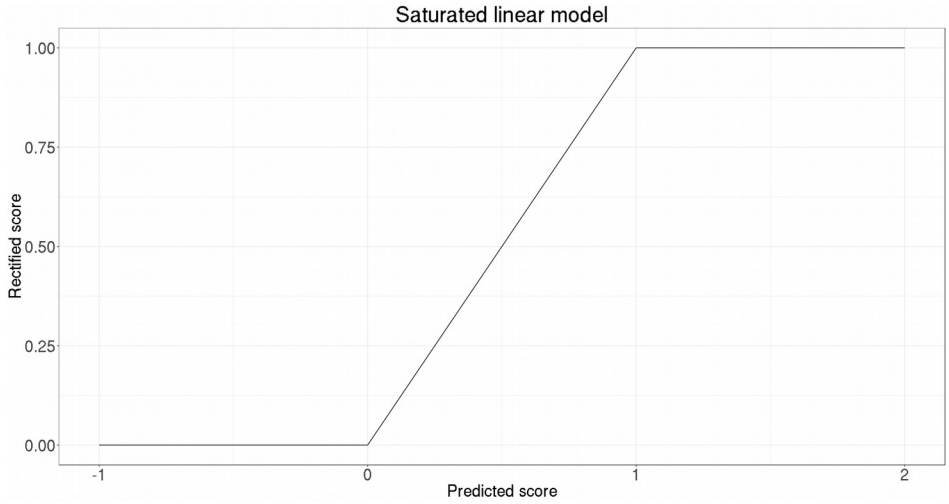

Figure 13. Saturated linear model as rectifier for predictions.

The model (39) is used as the basis prediction to be improved where three different cases can be found depending on the range of the obtained value of $\hat{y}_{ij}$:

a) The predicted score $\hat{y}_{ij}$ is greater than 1 (maximum possible value)
b) The predicted score $\hat{y}_{ij}$ is within the range [0,1] (allowed values)
c) The predicted score $\hat{y}_{ij}$ is negative

The case *b* (prediction falling within the allowed values) is the one not requiring fix at all. It may be a more or less accurate but *a priori* it cannot be evaluated. The case *a* can be fixed using as output the minimum value between $\hat{y}_{ij}$ and 1. A similar logic can be applied to case *c* when the output value is lesser than 0, selecting always the maximum between $\hat{y}_{ij}$ and 0 guarantees a realistic score. Putting all these scenarios together, the saturated linear model $\hat{y}'_{ij}$ would be a correction to $\hat{y}_{ij}$ described as:

$$\hat{y}'_{ij} = \begin{cases} 1 & \text{for} \quad \hat{y}_{ij} > 1 \\ \hat{y}_{ij} & \text{for} \quad 0 \leq \hat{y}_{ij} \leq 1 \\ 0 & \text{for} \quad \hat{y}_{ij} < 0 \end{cases} \tag{51}$$

Without having to compare their accuracies or entering in further mathematical demonstrations, it is a fully safe conjecture that $\hat{y}'_{ij}$ will be always equally or more accurate than $\hat{y}_{ij}$ when predicting p-user's scores.

4.2.3. Heuristic methods

Using linear regressions for obtaining the matrix weights **W** has some drawbacks. Maybe the main con is the need of using at least $|C|$ known scores for every p-user. This is because a linear function with n regressors and one Y-intercept, requires no less than n+1 records for adjusting the function coefficients. This is not a big problem when the total number of samples (reviewed entities) per p-user is really greater than the final number of clusters $|C|$. On another note, it is just the opposite when there are just a few of samples for evaluating the weights in comparison to the number of clusters $|C|$. That does not mean there are not ways of finding **W**, but certainly no using a linear regression.

The heuristic methods cover a collection of several ways of calculating the weights introducing some assumptions that constrain the final values of **W**. The aim of these methods is not obtaining the most efficient way of reducing the error of the predictive model (39), but at least a solution that is accurate enough. As mentioned, these are specially important in the cases where there are almost no previous information about the p-users; maybe a few rates or even none at all.

4.2.4. Equal weights

The simplest approach when trying to figure out what are the right weighs that define how much a p-user is describe by every cluster c_k, is accepting that without more information or additional assumptions, a priori all the clusters share the same likelihood. In other words, all the clusters have the same chance of describing an unknown p-user. Setting as K the total number of clusters, all them should contribute with the same weights to the final score

evaluation $\hat{y}_{ij}$ using (39). As the α_i Y-intercept can be understood as the contribution of a virtual cluster c_{K+1} with systematic ratings of 1, α_i should have the same value as the rest of weights. According to all this, depending on using or not the Y-intercept, the weights can be calculated as:

$$w_{ik}=\begin{cases}\frac{1}{K+1} & \text{when using Y-intercept} \quad \alpha_i=w_{ik}\\ \frac{1}{K} & \text{when not using Y-intercept} \quad \alpha_i=0\end{cases} \tag{52}$$

$$\alpha_i=\begin{cases}\frac{1}{K+1} & \text{when using Y-intercept}\\ 0 & \text{when not using Y-intercept}\end{cases} \tag{53}$$

The decision of using or not the Y-intercept can be understood as relying just on the clusters for evaluating $\hat{y}_{ij}$ or adding some personal bias to the final score. This uses to be the case when α_i can be properly evaluated for every p-user u_i. Because this method is applied when there is very little or not information at all about the p-user, if α_i is considered, having the same value as the rest of weights, then the interpretation has to be understood just as the contribution of an extra virtual cluster c_{K+1}.

Finally, it is worth to mention that the advantages of this method are its simplicity but also, because of the way of evaluating the weights, the final prediction does not need any correction; it always falls within the range [0,1].

4.2.5. BFGS

The *Broyden-Fletcher-Goldfarb-Shanno* [100] is a specialised algorithm that performs numerical optimizations through several iterations. It belongs to the group of quasi-Newton methods, applying a hill-climbing optimization in order to achieve a stationary point. This stationary point is reached when the gradient of the cost function is zero. In the specific case of the weights optimization, the cost function is the error of the all known evaluations $\hat{y}_{ij}$ vs the real scores s_{ij} for a p-user. As the weights have a direct impact (39) in $\hat{y}_{ij}$ then the function cost depends on **W** and its gradient can be used for minimizing this error.

Despite the BFGS algorithm has a well defined logic, the quality of the final results has a strong dependency on the initial values $\mathbf{W_0}$ used as starting point. Those initial weights $\mathbf{W_0}$ can be closer or further of the optimal **W** that minimizes the error, so its election is also a decision to be taken into account. Exactly the same as there are several ways of obtaining **W** there are also several ways of selecting the initial weights $\mathbf{W_0}$; the one applied is based on the previous method of fixed-weights. The rational behind this decision compared to a random within a range selection of weights is because the random selection would lead to different values of errors for every time the BFGS is applied, so its performance would be random despite constraint into a domain. On the other hand, the fixed weights election for initial weights $\mathbf{W_0}$ combined with the deterministic BFGS guarantees the same results and accuracy for every execution.

Finally, as this method relies on an optimization it cannot be assured that the final weights will not predict invalid results for $\hat{y}_{ij}$ so there is room for applying a saturation rectifier in order to limit those to [0,1].

4.2.6. Successive approximations

This method is directly inspired by the several mathematical ones covered by this name with the objective of finding a solution. Like those methods, the proposed one is a recursive process where on every step it is found the best simple solution and the resultant error of that is reduced at the next steps. More concretely, for every c-user u_i searches the cluster c_k which Z_k combined with an optimized w_{ik} minimizes the error ε_{ik} for this user's predictions using just the cluster c_k. That is, minimizes:

$$\varepsilon_{ik}=\sum_{j=1}^{|O_i|}|s_{ij}-w_{ik}\cdot z_{kj}| \tag{54}$$

In the expression (54), $|O_i|$ is the cardinality of O_i the set of all the entities reviewed by the c-user u_i. For every of these c-users u_i the evaluation is performed through all the clusters selecting just the c_k with the least error ε_{ik} and its associated weight w_{ik}. This error ε_{ik} for the selected cluster c_k is obtained from a collection of different individual errors ξ_{ikj}, one per c-user u_i and the j-th entity; that can be expressed as:

$$\varepsilon_{ik}=\sum_{j=1}^{|O_i|}|\xi_{ikj}| \tag{55}$$

$$\xi_{ikj}=s_{ij}-w_{ik}\cdot z_{kj} \tag{56}$$

These individual errors per c-user and entity ξ_{ikj} are the ones used in the next iterations. This method is applied for evaluating all the errors for every cluster c_k but it can be also generalized for obtaining the Y-intercept α_i for (39). As it has been explained earlier, the Y-intercept can be understood as the prediction of a virtual cluster that systematically scores with value 1 every entity. Mathematically this is translated directly from (54) as:

$$\varepsilon_{i\alpha}=\sum_{j=1}^{|O_i|}|s_{ij}-\alpha_i| \tag{57}$$

The equivalent for (56) when defining the individual errors from the Y-intercept are:

$$\varepsilon_{i\alpha}=\sum_{j=1}^{|O_i|}|\xi_{i\alpha j}| \tag{58}$$

$$\xi_{i\alpha j}=s_{ij}-\alpha_i \tag{59}$$

Using (57), (58) and (59) as the way of evaluating the errors of the Y-intercept, this can be evaluated like the clusters. In fact, this is part of the selection of the cluster c_k, the Y-intercept is like an extra cluster. For simplifying the explanation of the method, it will considered as a virtual cluster c_α that will be treated exactly the same way as the others, except for the evaluation of its errors.

For describing the next iterations let's set k the index of cluster c_k that minimized the error for the user u_i. That may include the Y-intercept as the selected entity. During the next iteration it will be available all the clusters but not c_k (or maybe c_α if it is the case), so all the remaining clusters will be evaluated looking for the one $c_{q\neq k}$ (or $c_{q\neq\alpha}$) that will reduce the new error ε_{iq}. Those iterations are performed with almost the same exact logic explained but using ξ_{ikj} as the target value to match instead of s_{ij}. The equation (54) adapted from the first iteration to any of the next ones takes the form:

$$\varepsilon_{iq}=\sum_{j=1}^{|O_i|}|\xi_{ijk}-w_{iq}\cdot z_{qj}| \tag{60}$$

The evaluation of individual errors ξ_{iqj} and the sum of these absolute values ε_{iq} is now performed as it is described in (60). Taking that into account, instead of using (56), the next iteration will define ξ_{iqj} as:

$$\xi_{iqj}=\xi_{ikj}-w_{iq}\cdot z_{qj} \tag{61}$$

This process can be generalized from one iteration where the selected cluster is c_k (or c_α) to the next iteration where the selected cluster will be c_q. The only consideration to (60) and (61) is the first iteration, where instead of using the values ξ_{ikj} it is used s_{ij} instead, as it is shown in (57) and (59). For the virtual Y-intercept cluster c_α, its specific evaluation equations are:

$$\varepsilon_{i\alpha}=\sum_{j=1}^{|O_i|}|\xi_{ijk}-\alpha_i| \tag{62}$$

$$\xi_{i\alpha j}=\xi_{ikj}-\alpha_i \tag{63}$$

What it is expected from this approach is not to evaluate all the weights at once but one after the other considering the Y-intercept like any other weight. This process begins with the most influencing one in the evaluation of $\hat{y}_{ij}$ until the least influencing one. The term influencing must be understood as the weight or Y-intercept that has minimized most the error at every step. Once introduced the general approach, it is needed to explain how to obtain the weights w_{iq} or α_i that minimize the error for c_q or c_α at every step. The procedure is not based in the first iteration but it can be easily generalized for that just switching from ξ_{ikj} to s_{ij} in

every equation. Obtaining the optimal w_{iq} for (60) is equivalent to find this weight for the equation:

$$\varepsilon_{iq}=\sum_{j=1}^{|O_i|}\left(\xi_{ijk}-w_{iq}\cdot z_{qj}\right)^2 \tag{64}$$

Despite that (60) and (64) do not have the same behaviour, they share the same minimum value w_{iq} so it can be obtained from (64) and just apply it at (60). In order to simplify the notation, (64) can be rewritten as a function $\varepsilon=\varepsilon(w)$ of w that is specific for every c-user u_i, cluster c_q and its previous errors ξ_j from the preceding iteration:

$$\varepsilon(w)=\sum_{j=1}^{|O_i|}\left(\xi_j-w\cdot z_j\right)^2 \tag{65}$$

At the equation (65) it has been removed the index i that defines the user u_i for the sake of simplicity. For obtaining the extreme value w that minimizes $\varepsilon(w)$ the standard approach is the least squares method. As $\varepsilon(w)$ is a minimum, it cannot be decreased changing the value of w, that is:

$$\frac{\partial\varepsilon(w)}{\partial w}=0 \tag{66}$$

$$\frac{\partial\varepsilon(w)}{\partial w}=\frac{\partial\left(\sum_{j=1}^{|O_i|}\left(\xi_j-w\cdot z_j\right)^2\right)}{\partial w}=2\cdot\sum_{j=1}^{|O_i|}z_j\cdot\left(\xi_j-w\cdot z_j\right)=0 \tag{67}$$

Finally, isolating w from (67), its evaluation is performed as:

$$w=\frac{\sum_{j=1}^{|O_i|}z_j\cdot\xi_j}{\sum_{j=1}^{|O_i|}z_j^2} \tag{68}$$

This calculation for w is only for the clusters c_k but not the Y-intercept or the first iteration. The expression (68) rewritten for the first iteration only needs to replace the previous errors ξ_j by the c-user's score s_j for the j-th entity:

$$w=\frac{\sum_{j=1}^{|O_i|}z_j\cdot s_j}{\sum_{j=1}^{|O_i|}z_j^2} \tag{69}$$

For the calculation of the Y-intercept α_i, the expression takes the form:

$$\alpha = \frac{1}{n} \cdot \sum_{j=1}^{|O_i|} \xi_j \tag{70}$$

For a better understanding of the whole method, the **Table 3** shows a pseudo-code describing how *successive approximations* are applied:

```
# W is the users' weight matrix
# Z is the clusters' score matrix
# S is the users' scores matrix

function weights_estimator(Zk,R)
  # Weights for cluster k are obtained applying (68)
  let Wk = sum(Zk·R)/sum(Zk·Zk)
  # E (errors matrix) is obtained applying (54) or (60)
  let E = abs(Wk·Zk) - S
  # the function returns a tuple of two values
  return (E,Wk)

function successive_approximations(Z,S)
  let W = empty collection
  let first_iteration = true
  for Zk in Z
    if first_iteration
      # initially S is used as the error to be reduced
      (new_R,Wk) = weights_estimator(Zk,S)
      # it is no longer the first iteration
      let first_iteration = false
    else
      (new_R,Wk) = weights_estimator(Zk,old_R)
    # reassigned the W weights matrix
    let W = append(W,Wk)
    # reassigned the value old_R
    let old_R = new_R
  # after the loop let's calculate different α values with (70)
  let α = mean(old_R)
  # the function returns weights and Y-intercepts as tuples
  return (W,α)
```

Table 3. Pseudocode for algorithm that obtains weights by the *successive approximations* method.

4.2.7. Cluster-weight optimization

This method combines the process of weight estimation with the optimization of the clusters' scores vectors Z_k. It does not introduces any new method for obtaining the weights, in fact, it uses any of the available ones previously introduced in this document. What is new is that it performs a double-step iteration for refining the clusters when the weights are obtained. With this improved clusters scores vectors, a new evaluation of weights is performed and the process is repeated several times until there is not achieved any further improvement.

Let's call WGM to a selected *weight generation method* (any of the available ones) that gives a good collection of weights **W** given a collection of K clusters and their scores vectors Z_k. Let's now define as CO the *cluster optimization* algorithm proposed in this section that given a weight matrix **W** adjust the scores vectors from Z_k to Z'_k. Those improvements when moving from Z_k to Z'_k are small and also is small the error reduction between the new predicted $\hat{y}'_{ij}$ and the real c-user's scores s_{ij}. Because of the new scores vectors Z'_k the values **W** do not guarantee any longer being the best ones (in terms of reducing the error). Using the same WGM to the new scores vectors Z'_k a new weights matrix **W'** may be obtained. If **W** and the new **W'** are the same, then an optimal 3-tuple weights-score-vectors has been achieved. If there is a significant difference between **W** and **W'**, then the CO is applied again to Z'_k for obtaining a new Z''_k, iterating until having no further improvement. This process can be more easily understood viewing the next figure that shows the double-step iteration for reducing the overall evaluation error.

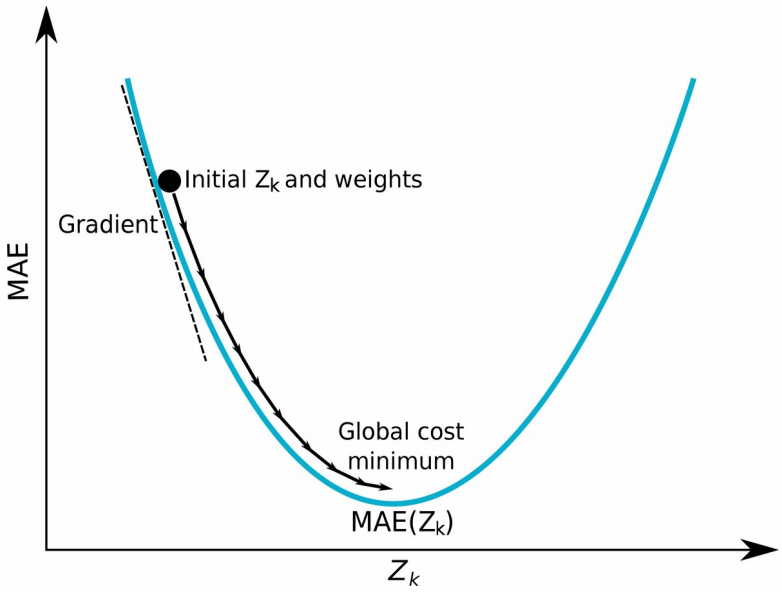

Figure 14. Graphical explanation of the gradient descent optimization.

The first part of this optimization is the WGM which is known and only has to be chosen, whilst the second part of every step is the CO and it is calculated using the gradient descent method. Using the matrix notation $\boldsymbol{Z}$ for all the clusters' score vectors Z_k and considering (39) the error of evaluation can be written as:

$$E=\sum_{i=1,j=1}^{|U|,|O|}|\varepsilon_{ij}|=\sum_{i=1,j=1}^{|U|,|O|}|s_{ij}-\hat{y}_{ij}|=\sum_{i=1,j=1}^{|U|,|O|}\left|s_{ij}-\sum_{k=1}^{K}w_{ik}\cdot z_{kj}+\alpha_i\right| \tag{71}$$

The gradient is obtained considering the evaluation of the errors ε_{ij} in (71) and applying to it the nabla [101] operator ∇ as ∇E. In the specific case of the scores vector, ∇ is defined as:

$$\nabla = \sum_{k=1,\, j=1}^{|C|,|F|} \vec{u_{kj}} \frac{\partial}{\partial z_{kj}} \tag{72}$$

The vectors introduced in (72) are unitary and define a hyperspace of dimension $|Z|$; every of those dimensions z_{kj} is a variable that can be improved. With a similar rational to previously minimized errors, optimizing (71) is equivalent to the minimization of the quadratic substitution of the absolute values per their squared version:

$$E = \sum_{i=1,\, j=1}^{|U|,|O|} \left(s_{ij} - \sum_{k=1}^{K} w_{ik} \cdot z_{kj} + \alpha_i \right)^2 \tag{73}$$

When applying ∇ to (73) the total error's E first partial derivative with respect every of the dimensions z_{kj} becomes:

$$\frac{\partial E}{\partial z_{kj}} = -2 \cdot \sum_{i=1,\, j=1}^{|U|,|O|} w_{ik} \cdot \left(s_{ij} - \sum_{k=1}^{K} w_{ik} \cdot z_{kj} + \alpha_i \right) \tag{74}$$

When evaluating those partial derivatives (74) there are obtained small increments that reduce the error as:

$$\boldsymbol{Z}_{n+1} = \boldsymbol{Z}_n - \gamma \nabla E(\boldsymbol{Z}_n) \tag{75}$$

In fact, (75) should be considered as an iterative improvement. Every time this is applied, it is considered the provided CO contribution to the double-step. The improved scores vector $\boldsymbol{Z}_{n+1}$ comes from the previous scores vector $\boldsymbol{Z}_n$ and a small contribution of the gradient of the error $E(\boldsymbol{Z}_n)$ evaluated in $\boldsymbol{Z}_n$. It is not difficult to assume that $\forall \gamma \in |\mathbb{R}|$ small enough, the error evaluated at $E(\boldsymbol{Z}_{n+1})$ is always equal or lesser than its previous $E(\boldsymbol{Z}_n)$.

4.2.8. Inverse error

That is the last of the weight evaluation methods proposed. Like some of the previous ones it is heuristic because is based in an assumption and also can be applied when having very little information about the p-users; just one single known score is enough for it. The assumption introduced is that the clusters with lesser errors, the ones that their scores vectors Z_k are closer to the p-users scores $\boldsymbol{S}$ should have more impact (greater weights) compared to the ones with more error in the final evaluation of $\hat{y}_{ij}$ using (39). Naming the error of a cluster c_k for a specific p-user u_i as ε_{ik} and considering it only depends on its scores' vector Z_k, this is evaluated as:

$$\varepsilon_{ik} = \sum_{j=1}^{|F_i|} |s_{ij} - z_{kj}| \tag{76}$$

In the present method, ε_{ik} is evaluated differently than (54) in the method of *successive approximations*. In this case, there is not defined any weight that optimizes (54), the weight is obtained directly from the inverse of ε_{ik}. Initially, all the p-user's weights w_{ik} are defined as:

$$w_{ik}=\frac{1}{\varepsilon_{ik}+\gamma} \tag{77}$$

The γ value shown in (77) satisfies $\gamma \geq 0$ and is a parameter empirically obtained. Once having the weights w_{ik}, those are normalized for obtaining the final ones w'_{ik}. This is performed using:

$$w'_{ik}=\frac{w_{ik}}{\sum_{l=1}^{K} w_{il}} \tag{78}$$

As can be directly deduced from (77) and (78), there is no evaluation for α_i. This is because it is considered $\alpha_i=0$ by default. In this case, the Y-intercept is not considered as a virtual clusters c_α.

4.3. Alternatives for predicting preferences

Despite the general model for predicting the scores is using the model (39) there are some alternative ways for the same purpose. Those are based just on averages, a very simple approach that can be used as a baseline for comparing the performance of other approaches.

The first of these alternative predicting methods is the **user's scores average**. This basically predicts $\hat{y}_{ij}$ using the average of the p-user's known scores like:

$$\hat{y}_i=\sum_{j=1}^{|O_i|} s_{ij} \tag{79}$$

In (79) all the different $\hat{y}_{ij}$ for a p-user u_i have the same predicted value $\hat{y}_i$ that is systematically applied to any prediction. The different values covered by the index j are the known scores for the user's evaluated entities O_i.

A similar approach is the **entity's scores average**, where the average is applied to all the known scores for an entity. It is equivalent to the user's scores average but focusing the average in the entity rather than in the user. In this case all the different $\hat{y}_{ij}$ for an entity o_j are the same $\hat{y}_j$ for independently of the p-user u_i. That is:

$$\hat{y}_j=\frac{1}{|U_j|}\cdot\sum_{i=1}^{|U_j|} s_{ij} \tag{80}$$

In (80) the index i covers all the users U_j that have scored the j-th entity o_j, averaging these users' scores for the entity.

Considering the earlier methods of scores averages as exceptions, the rest of the evaluations are based on (39) combining different methods for generating the weights and, maybe, a saturation rectifier.

5. Experimental evaluation

One of the previously explained ideas about the eigenclusters methodology is that it is designed to be generic. For the evaluation it is proposed a more specific domain of well known entities. That is the case of films, where there are available datasets with the required data quality in terms of quantity and variety so the performance of eigenclusters can be measured.

The idea is apply all the proposed hypotheses, objectives and the whole methodology to this specific domain, so from this point until the conclusions, the entities become just films, so it can be used both words for the same purpose.

5.1. Datasets

In order to have some standard datasets, the experiments have been performed using three concrete ones. From these three datasets, two of them are popular enough for being a *de facto* standard and the third one is a just a *dense subset* of the first one.

5.1.1. Prolific IMDb Users dataset

The first dataset is the IMDb62, a subset of the *Prolific IMDb Users* dataset [108] created in 2009. This is a big collection of reviews for 62000 reviews distributed into 27222 different films. The number of users is 62 with a total of 1000 reviews per user. One of the main characteristics of this dataset is its sparsity; 18340 of the films only have one score, that is the 67.31% of the cases. Despite this dataset includes scores and reviews written from these 62 users, all the used methodology completely ignores the text information available. This dataset has a great sparsity that is an important factor for explaining several of the results obtained from several experiments performed with it. Observing the **Figure 15**, as a rule of thumb it can be said that two random users have less than 30% of their reviews (same films) in common. The average number of films in common for two users is around 60, just a 6% of their total reviewed films.

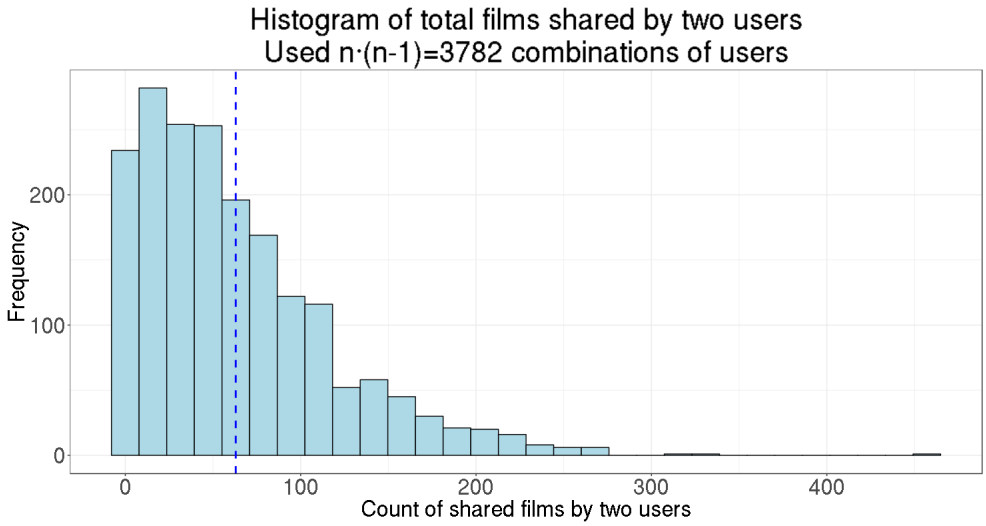

Figure 15. Histogram of common films by pairs of users for the *Prolific IMDb Users* dataset.

5.1.2. Dense subset of Prolific IMDb Users dataset

The second dataset is a *dense subset* of the first dataset, in the sense that only considers the films that at least have 4 reviews. That filter reduces the total number of films to 4111, only a 15.1% of the films from the original dataset. As can be seen at **Figure 16**, despite the number of films is smaller and the sparsity has been reduced, the distribution of films in common for two users is still very similar, lesser than 60.

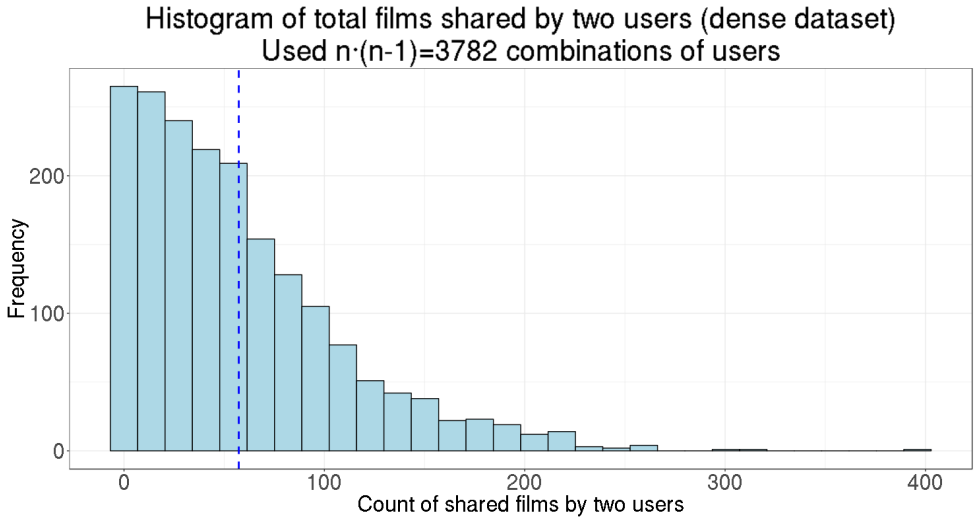

Figure 16. Histogram of common films by pairs of users for the dense subset of the *Prolific IMDb Users* dataset.

5.1.3. MovieLens

The third of the used datasets is the *MovieLens* one [109], a synthetic set created for such kind of specific purposes. This is a more dense selection compared to the plain IMDb62.

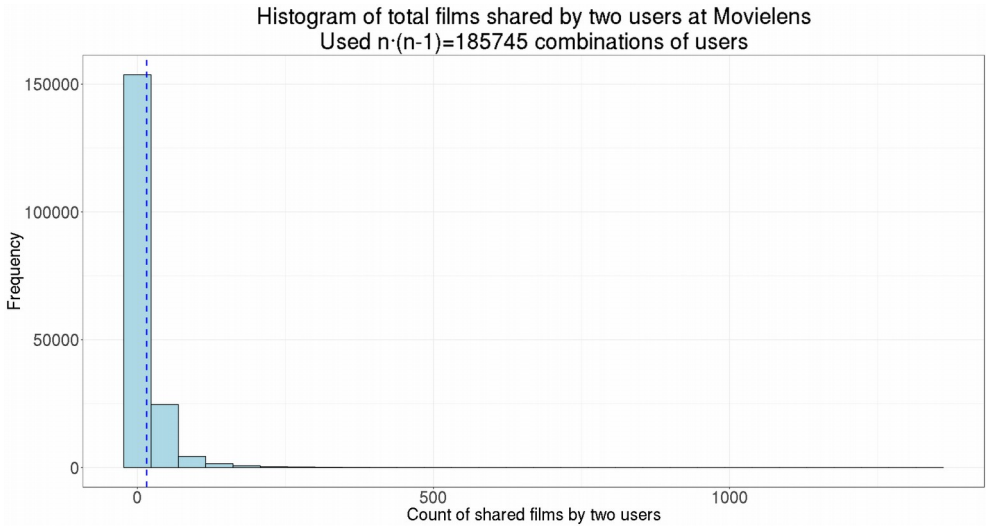

Figure 17. Histogram of common films by pairs of users for the Movielens dataset.

The main problem when working with different datasets is their specific but incompatible format. That requires a special treatment for transforming them in a common data model that is used internally by the algorithms and the models. Once the format problem is addressed, the scale for the scores is the second one. The scores for the IMDb62 dataset cover the whole range between 0 and 10, allowing the mentioned extremes. In the other hand, the MovieLens' range is reduced to 0 an 5. Once again, for having a standard format, both datasets have been re-scaled to values going from 0 to 1 and obviously including decimals. The main reason behind this decision is the easiness when comparing the different weights for the model (39) and the Y-intercept. Using this normalized scoring system, the Y-intercept can be understood in two different ways: first as the systematic user's bias but also as the hypothetical weight of a virtual cluster where all the scores were always 1 (the maximum one). As explained in previous sections, this double interpretation provides criteria when deciding to include or ignore the Y-intercept in different weight evaluation methods.

5.2. Full stack

All the implementations introduced in the present document are based in a compact code developed using the *Scala* [102] programming language. It has a good balance between efficiency and performance whilst combining the best practices mixing the functional programming paradigm with the object-oriented one; both with a strong static type system. Another of the reasons for using Scala is its easiness parallelizing processes, which is a good strategy for reducing the computation time for heavy load processes.

The JVM [103] used is concretely the OpenJDK 1.8.0_212 [104] for 64 bits. Using the JVM as support for the Java bytecode [105] generated with the compilation of the Scala code, also contributes for high efficiency among other features, like Garbage Collection [106], native class implementation and JIT compilation [107].

The operating system supporting the execution of the JVM with the Scala code is a Linux Debian distribution based on the Linux Kernel 4.17 for 64 bits. The operating system election guarantees the best performance for generic execution of native implementations compared to any other common operating systems. This performance is obtained out of the box with the standard installation.

The hardware used is a Intel i7-3770 CPU at 3.40 GHz, with 8 virtual cores for parallel execution of different threads. The RAM memory is 32 Gb and a hard disk of 1 Tb, where the used space was around 19 Gb.

5.3. Evaluation

This section explains the way all the previous methods are articulated for evaluating them. Independently of the method to be treated, the process for the evaluation is a **cross-validation** [73] where some data is used for generating a predictive model, and the rest of the data is used for measuring the accuracy of the model. In this process is specially important to completely split the data in two datasets, the **training dataset** and the **test dataset**. These datasets must be a partition of the whole available data sharing no content at all. If that were not the case, then some of the data to be predicted would be used for generating the model. It would have part of the information to be predicted which is not only invalid, but it is a circumstance that never would happen in reality; there is no need at all to predict information

already known. The notation for identifying those datasets is U^T and U^P for the c-users and the p-users. The same naming convention is applied for the scored films used for training and prediction F^T and F^P.

The training dataset, independently of its size, is used for fitting the model in a way or another. Once the model is ready, it is applied to the specific cases of the test dataset. The model's predictions could or not match the expected value. The general case when predicting continuous values is a numerical difference between the **actual value** and the **predicted value**. From these differences it is obtained the **prediction error**.

The cross-validation is an extra layer introduced to this way of evaluating the accuracy. Its rational is based in that there is a non-null probability that a specific combination of the training and test datasets would maximize the accuracy of the predicted results. That probability is present no matter which model is used despite its final value related to the nature of this model. The cross-validation handles this repeating the process several times with different distributions of the training and test datasets applied to the same model family and averaging the final accuracy among all the performed individual evaluations. Despite the cross-validation can be exhaustive, exploring all the available ways of dividing the original dataset in different training and test subsets, in this exercise it has been applied a non-exhaustive approach, where the obtention of the subsets and evaluation of results has been performed 10 times per experiment. Despite 10 times is not a statistically high number, it is enough for avoiding in most of the cases the distribution of the subsets that lead to extremes values for the accuracy. It does not avoid that some of those cases could happen, put these results will be averaged with others so the final impact is compensated. Obviously, the benefits of this approach would be yield using greater numbers than 10, but the reason for not going beyond 10 times is that some of the algorithms for training the models are really time consuming. This value provides a good balance between effectiveness and computation effort.

5.3.1. Evaluation metric

The accuracy measure used for evaluating the predictions' errors is the MAE [110] Mean Absolute Error, defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|\varepsilon_i| \tag{81}$$

In (81) every i-th case is considered an evaluation where $\hat{y}_i$ is the predicted p-user's score for a film and y_i is the known actual score for it. That difference $\hat{y}_i - y_i$ defines the error ε_i. Is important not to get confused with the i-th case and the i-th p-user u_i. In order to facilitate the notation understanding, it is introduced the specific notation where the index i refers to the i-th p-user u_i and the index j refers to the j-th film. Considering this in combination with (81) it can be rewritten as:

$$MAE = \frac{1}{|F^P|}\cdot\sum_{i=1,\,j=1}^{|U^P|,|F_i^P|}|\hat{y}_{ij} - s_{ij}| \tag{82}$$

In (82) the cardinality $|U^P|$ refers to the test subset of the p-users used for evaluating, whilst $|F_i^P|$ is the cardinality of the films subset rated by the p-user u_i. The cardinality $|\boldsymbol{F}^p|$ considers the whole test subset of the rated films for all the p-users.

As the cross-validation is perform 10 times, every iteration/time t the error is evaluated as MAE_t, this is included for the obtaining the final MAE in the way:

$$MAE = \frac{1}{10} \cdot \sum_{t=1}^{10} MAE_t = \frac{10^{-1}}{|F_t^P|} \cdot \sum_{i=1,\, j=1,t=1}^{|U_t^P|,|F_{it}^P|,10} |\hat{y}_{ijt} - s_{ij}| \tag{83}$$

This equation (83) is the final general way of evaluating the resulting MAE for model using cross-validation.

5.4. Experiments

All the methods previously introduced can be combined in different ways; among those just a few ones have been considering as the proposed experiments. In this section it is explained with some degree of detail these experiments.

5.4.1. Overview of clustering methods

The main purpose of this exercise is provide a first approach for identifying what clustering methods are the most promising for minimizing the prediction error. It has not been conceived for pointing out the most optimized clustering method but to have a global idea about their expected performance. This evaluation is not based on cross-validation and predictions but an analysis of the **residuals** [111] of the models. The residuals are fitting deviations from the model evaluation of scores and the real scores. The models and the residuals share the same definition $\varepsilon_{ij} = \hat{y}_{ij} - s_{ij}$ as it was introduced earlier but its main difference is the domain of their evaluation. The residuals are obtained from evaluating the model through the training subset of U^T and F^T (c-users and films). As there is no need for predicting with those models, the training subsets are just the whole subsets $U^T = U$, $F^T = F$ (and the c-users become also p-users). Even considering the possible effect of overfitting in the models, it is difficult to have a prediction error lesser than the residuals. That means that the evaluation of the residuals of a model is a good indicator of the best error that may be obtained from using this model for predicting. In other words, the expected prediction errors are very likely to be equal or higher but not lesser than the residuals.

The way those residuals are used is considering their final MAE, where the smaller ones are selected for focusing on their development and refine the way of reducing their prediction error. As the selected methods are the ones to improve, lately it is evaluated their prediction power using the errors at the test set, not just the residuals. This approach is a way of introducing an initial approximation for selecting the best methods as a rational alternative to simply try to refine every proposed method to its limit. That aims to reduce the scope and effort while achieving the best results.

The clustering approaches (see previous sections) selected for this initial comparison include the following that will named by their acronyms:

- **PSAS** - Partition sorted by averages scores
- **PSSDS** - Partition sorted by standard deviation of scores
- **B-PSAS** - Binary partition sorted by average scores
- **B-PSSDS** - Binary partition sorted by standard deviation of scores
- **K-M** - Clustering using K-means

For having a fair comparison when compared these methods for clustering, the way for relating the p-users to clusters (obtaining the weights for them) must be the same for all the five. The selected method for this purpose is the linear regression introduced at section 4.2.1. This method requires more information than others but on another note it also guarantees obtaining the optimum weights in order to reduce the residuals. It is worth to mention that the first four methods PSAS, PSSDS, B-PSAS and B-PSSDS can be evaluated using as many clusters as wanted, there is no limit on that as far as there are less clusters than films evaluated, that is $|C|<|F|$. On another hand, the K-M approach introduces a limitation coming from the K-means method; that is the constrain for the total number of clusters, it cannot be greater that the total number of users. In other words, K-means requires at minimum one user per clusters, or in mathematical expression $|C|<|U|$.

5.4.2. Results from clustering methods

The evaluation of the several methods for clustering (5.4.1.) are summarized at the **Figure 18** where on the abscisses are represented the number of clusters from 1 to 100 and the error (MAE) at ordinates. It can be easily observed the big difference in terms of error for the different methods proposed. The worst ones are the binary partition methods with a MAE that begins being closer to 10 and is reduced when clusters are increased; B-PSSDS has an almost linear decrease whilst B-PSAS sinks quickly when increasing the clusters from 1 to 10. From this point of 10 clusters until arriving to the 100, B-PSAS is under a MAE of 5, closer to the errors of models like PSSDS and PSAS. This allows concluding the binary version of these partition methods has a noticeable worse performance than their non-binary equivalents.

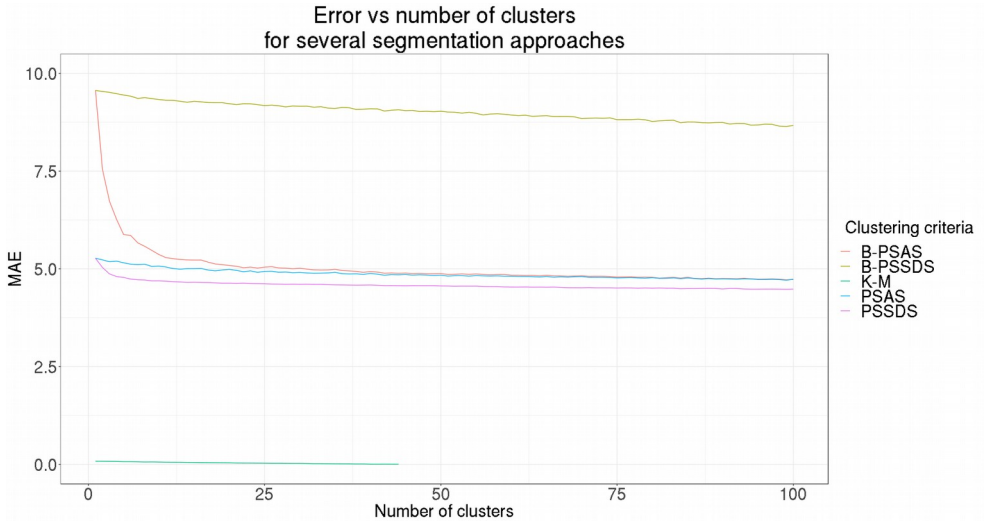

Figure 18. Evolution of prediction error using several partition methods with different total number of clusters.
Used the *Prolific IMDb Users* dataset with 44 users for clustering and weighting the clusters and 9 for evaluating with a total number of folds

for the cross-validation of 10. The binary partition methods B-PSAS and B-PSSDS are the worst while K-M reveals itself as the most promising one.

The case of the K-M is clearly different, with a enormous smaller error despite with some drawbacks, it is not defined for any number of defined clusters. It covers from 1 to 44 clusters because K-means requires at least as many users as clusters, that is $|C| \leq |U|$. For the specific case of the *Prolific IMDb Users* dataset, the total number of c-users for creating the clusters is using 44 for clustering and 9 for evaluating the results. It is also worth mentioning the case of $|C| = 24$, where the error 10^9 is so high that it is completely out of scale. This is something observed from time to time, despite K-M looks very promising compared to the other methods there are some specific cases where it performs really badly. Despite of that, it is clear that K-means is the clustering method with most promising results to be developed and in the next sections it is shown the improvements introduced around it but also the results obtained from that.

5.4.3. K-means and linear regression for well profiled users

Just as an advance of the results to be shown in the next sections, there is a special interest in the clustering using K-means and the linear regression as it is a combination with better performance than the other ones. This second experiment focus and zooms these methods combined in order to understand which variations provide the best results. In this case, the evaluation method is the proposed cross-validation instead of just the residuals. There are two main variants for this exercise:

1. **Full population**. The training and test dataset are just the same for all the users but not the films scored. That is $U^T = U^P = U$ but , where the score sets $S^T \cap S^P = \emptyset$, $S^T \subset S$ and $S^P \subset S$. The training set is 80% of total scores, $|S^T| = 0.8 \cdot |S|$ and the test set is the other 20%, $|S^P| = 0.2 \cdot |S|$.
2. **Subpopulation**. It uses 80% of users (c-users) with 100% of their scores for generate the clusters and the 20% of the users (p-users) for evaluating the model. That is $U^T \cap U^P = \emptyset$, $U^T \subset U$, $U^P \subset U$, $|U^T| = 0.8 \cdot |U|$ and $|U^P| = 0.2 \cdot |U|$. Viewed from the scores perspective $S^T \cap S^P = \emptyset$, $S^T \subset S$, $S^P \subset S$, $|S^T| = 0.8 \cdot |S|$ and $|S^P| = 0.2 \cdot |S|$. These 20% of remaining users (p-users) and the remaining scores S^P have their weights evaluated using the 80% of their scores S^{PW}, and the rest 20% of films' scores are then predicted S^{PP}. That is $S^P = S^{PW} \cup S^{PP}$ and $S^{PW} \cap S^{PP} = \emptyset$, $|S^{PW}| = 0.8 \cdot |S^P|$ and $|S^{PP}| = 0.2 \cdot |S^P|$. This last constrain can be rewritten as $|S^{PP}| = 0.04 \cdot |S|$.

As a general conclusion for both variants it can be observed that it is used a lot of information for clustering and profiling the users. The *subpopulation* method, which is the one using lesser information for profiling, uses 800 different scores for profiling every p-user. This situation contrasts with the next experiments using heuristic methods where the situation is dealing with users with a relatively unknown profile.

There are two additional variants that are combined with the previous ones. Those come from the circumstance where the score to be predicted is not present in any of the clusters. That is not an unusual situation, in the case of working with the IMDb62 dataset, it was

announced that 67.31% of the films only had been scored once. This situation is tackled using one of the two following alternatives for evaluating the prediction $\hat{y}_{ij}$ (p-user u_i and j-th film):

a) User's average. The inexistent score $\hat{y}_{ij}$ is evaluated averaging all the known scores (no matter to what films) of the p-user u_i.

b) Film's average. The lacking score $\hat{y}_{ij}$ is obtained averaging all the known scores (no matter which p-users) from all the films.

The combination of the former variants and the later alternatives becomes into the 4 next flavours: 1-a (full population, user's average), 2-a (subpopulation, user's average), 1-b (full population, film's average) and 2-b (subpopulation, film's average). All these 4 flavours have been evaluated.

There is one more case of study included in this experiment for the simple purpose of having a reference about the impact of using weights for relating p-users with several clusters at once. This reference is the classical approach (1) of using only one cluster per user, that is that every p-user has only one weight $w_{ik} = 1$ and the rest are 0. The way of obtaining those weights is just applying directly the results from the K-means clustering. With this classical approach, the experiment also evaluates the previous 4 flavours having only one cluster per user.

5.4.4. Results from K-means variants

At **Figure 19** can be observed the results related to the previous experiment 5.4.2 and its variants and flavours: 1-a (full population, user's average), 2-a (subpopulation, user's average), 1-b (full population, film's average) and 2-b (subpopulation, film's average). The dataset used corresponds to *Prolific IMDb Users* and the evaluation metric is MAE for predictions done with different number of clusters obtained with the K-means method. Every point from each series is obtained applying cross-validation with 10 folders.

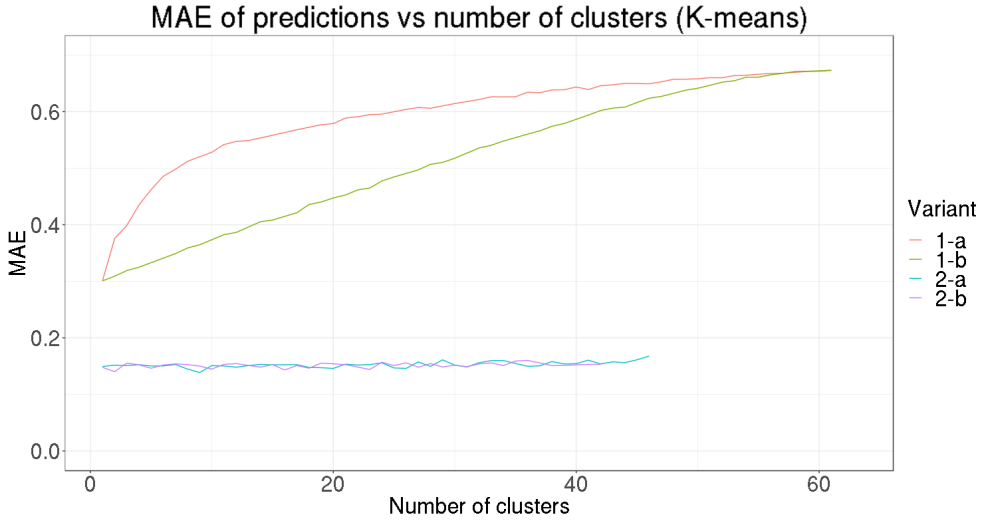

Figure 19. Evolution of prediction error using K-means with different total number of clusters and ways of using the users and their missing scores.
Used the *Prolific IMDb Users* dataset for predicting scores with a total number of folds for the cross-validation of 10. The variants 1-a and 1-b show the worst error compared to 2-a and 2-b.

It is easy to identify at the results from **Figure 19** that methods 1-a and 1-b, both related to the full population perform noticeably worse to the cases where a subpopulation is selected. Despite that all the series show a general trend to augment the error when increasing the total number of clusters, it is more evident for the full population variants (1-a, 1-b). The increasing trend for the subpopulation variants (2-a and 2-b) is so smooth that the variability of the error is greater than the trend (slope) itself. There is also some variability in the full population variants (1-a, 1-b) but in general it is smaller than the trend itself. This variability is due to the random initialization of the K-means algorithm; it does not provide the most optimal clusters but a good result that is always different in proportion to the total number of elements for distributing among the clusters. Another of the previously mentioned behaviours is the limit of clusters that can be obtained using K-means, always $|C| \leq |U|$ with 44 c-users used from the *Prolific IMDb Users* dataset.

The **Figure 20** shows how often films are repeated per cluster for the used dataset, pointing out a small density of scores per film that quickly decreases. The average unique film count per cluster is close to 2 scores per film when having only 1 cluster. This is reduced to less than 1.25 scores per film before reaching 4 clusters; from this point it slowly decreases until the maximum number of clusters. This behaviour is important to explain why the results from **Figure 19** get worse when increasing $|C|$. It may seem counterintuitive that adding more clusters that allows to profile users with higher precision results have their error increased, but that can be explained in relation to this frequency of unique films. As the eigenclusters method is based on an statistical approach, it is very important that every cluster has enough cases per film in order to provide not only a good segmentation but also for having a robust prediction. When only a few samples per firm are available on every cluster, this lack of information is translated in worse evaluations for the clusters' associated scores vectors Z_k and worse accuracy on predictions. This problem is more associated to the sparseness of the *Prolific IMDb Users* dataset than the method itself.

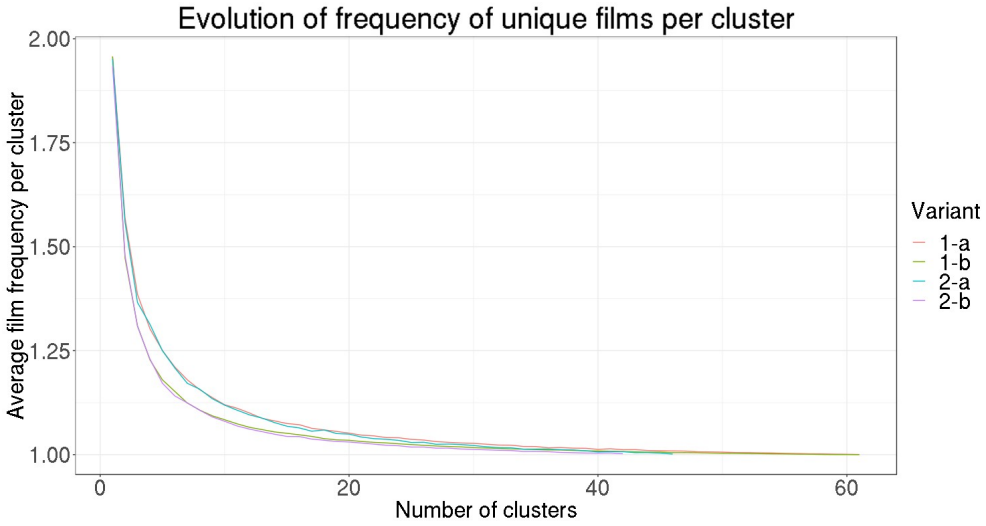

Figure 20. Average count of the unique films per cluster generated using the *Prolific IMDb Users* dataset.

The high sparsity of the dataset leads to quickly fall in a situation where the films only have one or two samples for evaluating the average score.

This proportion tends quickly to just one score per film when increasing the total number of clusters generated. This evolution of the cluster density of scores is pretty similar independently of the variant used.

Independently of the convenience of using the evaluated dataset and being consistent with the results obtained for the different variants, it is easy to conclude the *subpopulation* method intrinsically provides a clear better performance. As the scores are normalized, it is easy to identify the minimal MAE for the *full population* variant 0.3 as a 30% of error from the score scale. This error for the *full population* series is able to increase to values greater than 0.6, that is a 60% when using the maximum number of clusters.

5.4.5. K-means and linear regression for different profiles

The next experiment after the previous overview about K-means and linear regression is zooming into the most promising variants; more concretely exploring different ranges of profiled p-users. For doing that it has been created different groups of profiling defined by the total number of known/labelled scores from their users. Every group is identified by a **profiling percentage** that determines how many of the known scores per user are used for profiling. These scores falling into the profiling percentage are used for calculating the p-users weights, and the rest of scores are used for evaluating the accuracy/error of the model. The following are the selected profiling percentages explored by this experiment and their translation when using the *Prolific IMDb Users* dataset.

Profiling percentage	Scores per user (profiling)	Scores per user (predicting)
0.1%	1	999
0.2%	2	998
0.3%	3	997
0.4%	4	996
0.5%	5	995
1%	10	990
10%	100	900
20%	200	800
40%	400	600
80%	800	200

Table 4. Profiling percentages for the *Prolific IMDb Users* dataset.

For all the different profiling percentages it is selected 80% of the users population for training the models; that is to feed the K-means and generate the clusters, so $|U^T|=0.8\cdot|U|$. The remaining 20% of the users are fully used for the predictions, $|U^P|=0.2\cdot|U|$. Finally, for the evaluation of the distances using K-means, the cases of non-scored films are fulfilled applying the previously explained *null default value*.

As can be seen from the profiling percentages, the model is going to be evaluated in a very wide range of knowledge about the users. The reason for this and the next experiments is the validation or refusal of the hypothesis that the effectiveness of the method varies noticeable with the total information about the user.

Another dimension introduced in this exercise is the use or absence of the Y-intercept into the linear regression. As has been already discussed, this Y-intercept can act like the average score for a user, so it may have a strong influence in the model.

5.4.6. Results for different profiling

The following figures show the results of the previous experiment applied to the *Prolific IMDb Users.* **Figure 21** is a selection of small percentages of scores from p-users used for profiling them that covers from 0.5% to 10%, using the Y-intercept as part of the predicting model. In this figure and the rest of them exposed at this section it has been filtered out all the results with extremely bad MAE, those go from 10^9 to 10^{13}. The results exposed at this figure do not cover the same number of total clusters $|U|$ because the linear regression needs at least as much points (scores in this case) as total number of clusters $|U|$. That implies that the smaller the percentage of scores used for training the smaller the number of clusters can be. This reduced number of training points is directly interpreted as knowing not so much about the user. What is important from **Figure 21** is the slope of the series. The error uses to be almost the same for 1 cluster and tends to increase when adding more clusters. This rule has an exception when using 0.5% of the scores (5 c-users) for clustering; in this case the initial error (1 cluster) is greater than the rest despite it also follows the trend of increasing with the number of clusters. It can also be concluded that the slope is flatter the greater the scores percentage is used for clustering. That supports the rationale that more samples (scores) provides more information and more statistically robust evaluations with better predictions. That is translated as the flatter slopes point out the more accurate models.

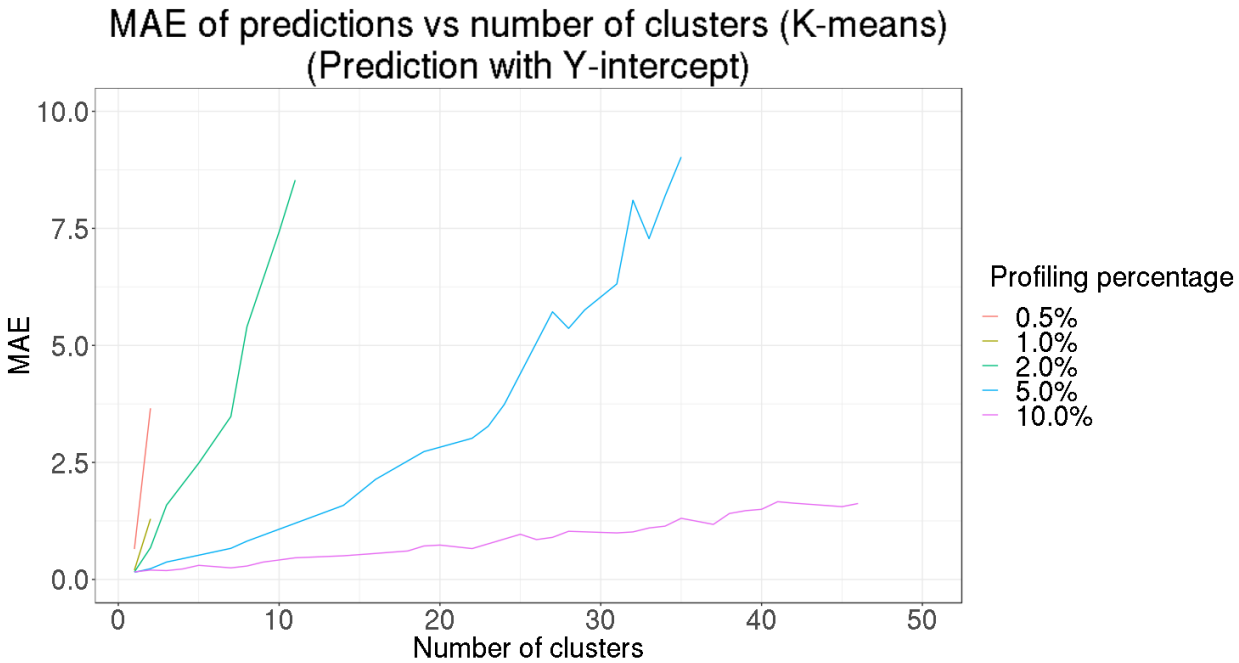

Figure 21. Group of the smaller profiling percentages and evolution of their errors (MAE) when increasing the number of used clusters.

All the outliers with an error greater than 10^{11} have been removed. The predicting model is the regular linear regression including its Y-intercept. The series with less known scores cannot be explored with more clusters and provide an error proportionally greater

compared to the ones with more scores percentage. Except the 0.5% series, the rest almost begin from a pretty similar error for the case of just one cluster.

The comparison of the results from **Figure 21** vs **Figure 22** shows an evident better performance in predictions when using the Y-intercept compared to when ignoring it. Just with one glance, it can be seen only the cases of 0.5% and 2% of profiling percentage (using Y-intercept) reach an error greater than 7.5 for the prediction. For the case of non-using Y-intercept, almost all the series reach errors greater than 7.5. **Figure 22** reveals a scenario with similar behaviours for the series of different percentages of scores but with different slopes and initial errors for just 1 cluster. The included-Y-intercept version of the predicting linear model manages to almost have the same MAE for the different series whilst the excluded-Y-intercept option shows worse initial error inversely proportional to the population percentage. The slopes are also flatter in the included-Y-intercept.

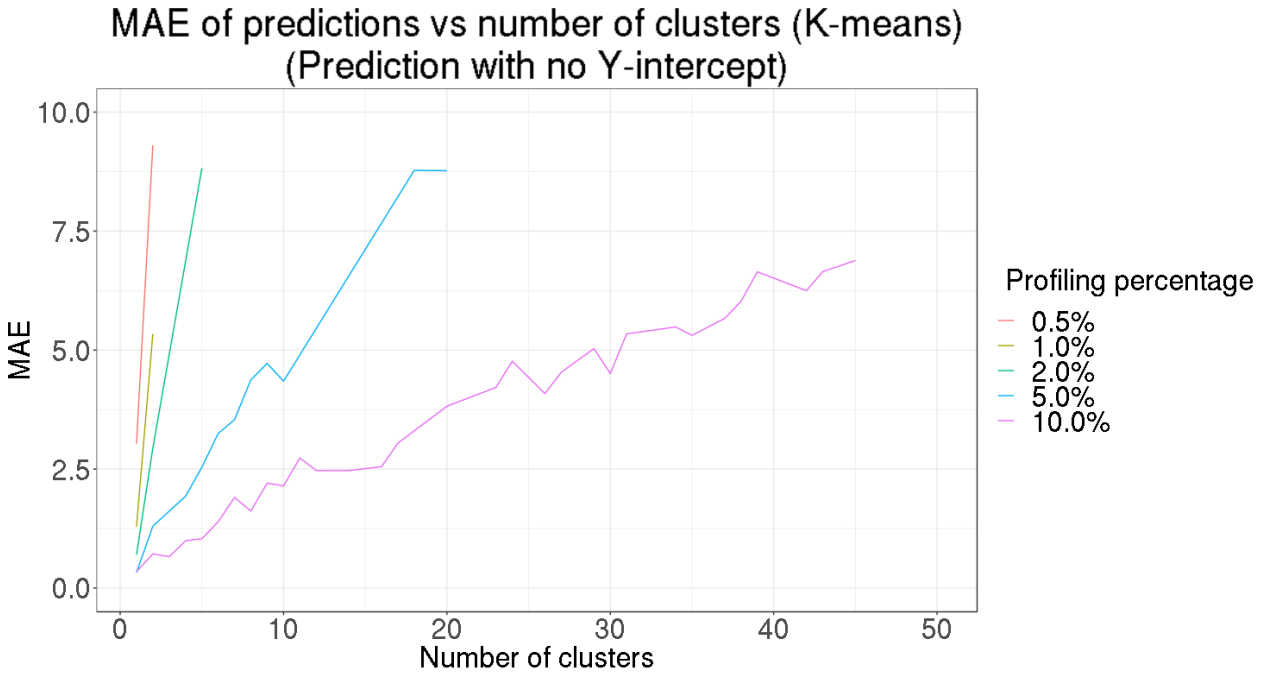

Figure 22. Group of the smaller profiling percentages and evolution of their errors (MAE) when increasing the number of used clusters (no Y-intercept).

All the outliers with an error greater than 10^{11} have been removed. The predicting model is the regular linear regression excluding the Y-intercept at all. The series with less scores cannot be explored with more clusters and provide an error proportionally greater compared to the ones with more known scores. Almost any series increases its error until overreaching the value 7.5.

Figure 23 is the natural continuation of **Figure 21** but comparing profiling percentages that go from 10% to 80% on the same dataset *Prolific IMDb Users*. The results are equivalent to the smaller profiling percentages in terms of having an initial error that is almost independent from the series and it tends to increase when using more clusters; it also shows an equivalence having a flatter slope when increasing the known scores percentage. Despite this slope reduction is proportional to the scores count, it is not directly proportional because the

reduction of the slope when moving from the 10% series to the 20% is greater than when moving from 20% to 40%.

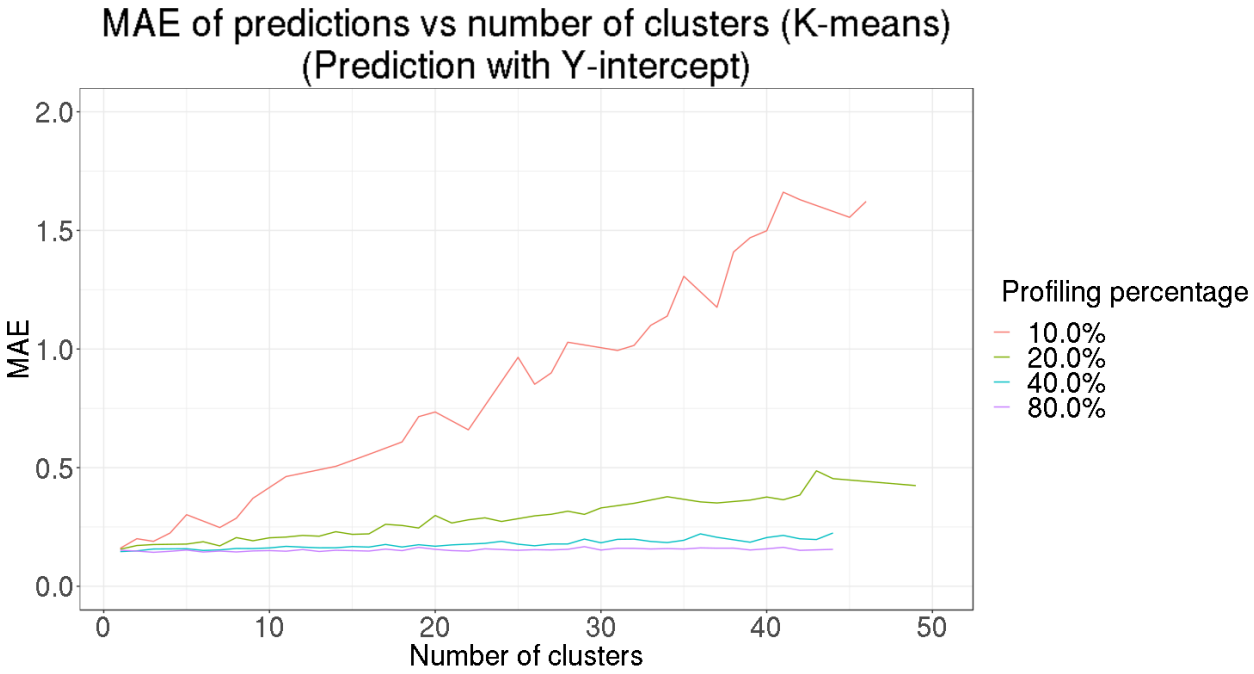

Figure 23. Second group of profiling percentages (greatest ones) and evolution of their errors (MAE) when increasing the used clusters.

All the outliers with an error greater than 10^9 have been removed. The predicting model is the regular linear regression including its Y-intercept. All the series begin from a similar error for the case of just one cluster and tend to increase it with the addition of more clusters.

Similarly to the parallelism between **Figure 23** and **21**, the **Figure 24** is the equivalent of **Figure 22** also evaluating cases without Y-intercept and profiling percentages that go from 10% to 80% using the dataset *Prolific IMDb Users*. In these new cases there is a closer initial error for all the series when using just one cluster meaning that greater known scores percentages are directly translated into more statistically significant results, more stability of the models and better accuracy. When comparing the higher profiling percentages (from 10% until 80%) in the variants of using and excluding the Y-intercepts from the predicting models, **Figure 23** and **24** expose how every series has a worse behaviour with the non-Y-intercept version. In **Figure 23** all series stay below an error smaller than 0.5 except for the series of 10% of scores that crosses the 1.5 MAE limit, but their equivalent series seen in **Figure 24** overpass the values 0.5, 1, 2 and 6 of error. All this leads to a general conclusion of the importance of the Y-intercept for improving the accuracy of the model.

This comparison of the same approach using and ignoring the Y-intercept evidences how this term gathers information that improves the prediction making more stable the model for different profiling percentages. As the model is a personalized linear regression per every p-user it is easy to conclude that the Y-intercept gathers the user's bias to predict one specific average value. The greater is the sample of scores for fitting the linear regression with the clusters more cases are included in this Y-intercept so its evaluation is less probably an isolated atypical score but a clear trend of the user.

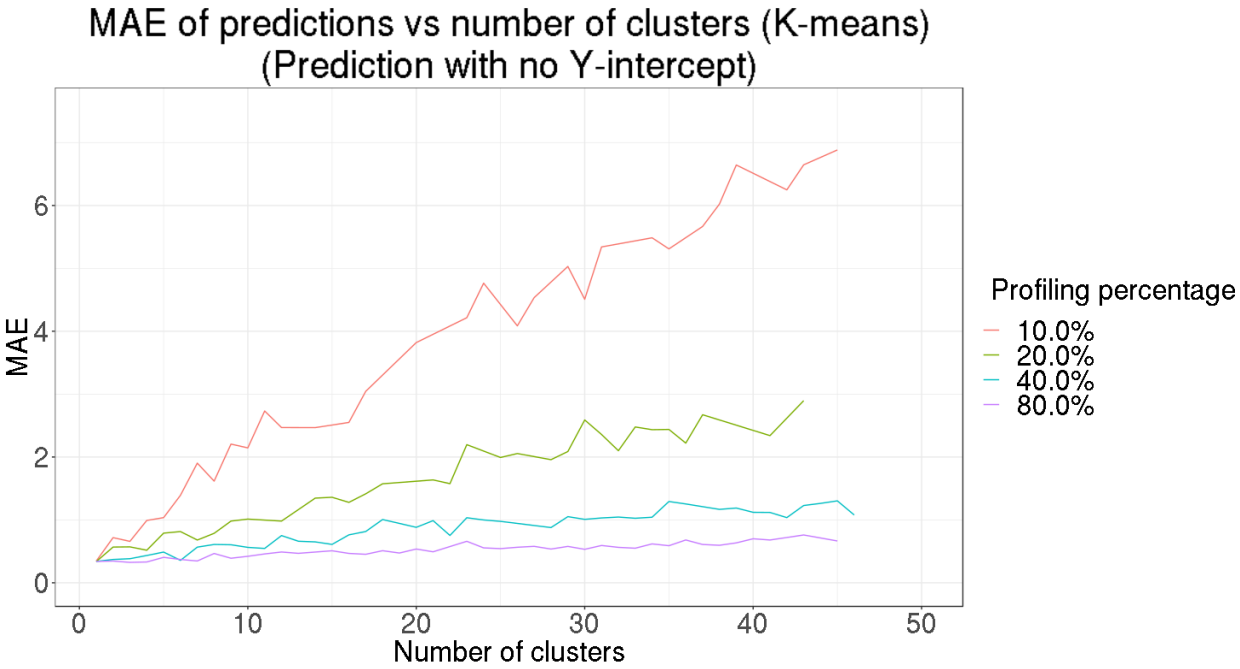

Figure 24. Second group of profiling percentages (greatest ones) and evolution of their errors (MAE) when increasing the number of clusters.
All the outliers with an error greater than 10^9 have been removed. The predicting model is the regular linear regression avoiding to use the Y-intercept. All the series begin from a similar error for the case of just one cluster and tend to increase it with the use of more clusters.

For supporting this explanation about the impact of the Y-intercept, the **Figure 25** shows the results of using just this Y-intercept as a predictor, in other words the case of using 0 clusters. When no clusters at all are used, the linear regression is performed just over a constant becoming a simple average of all the scores to predict. The most remarkable evidence exposed at **Figure 25** is how reduced the error is compared to previous approaches. The maximum error when using just one score per user as predictor is lesser than 0.2112 of MAE, falling very quickly when increasing the number of scores until reducing MAE to the value 0.1545. This range delimiting the error is clearly smaller than any other error range shown in previous results. The explanation for this behaviour lies in the intrinsic reduced variability of the whole set of scores when comparing to the predictions. The maximum possible variability for scores is 1 (from 0 to 1) in the worst scenario.

Despite the errors filtered out from the previous **Figure 21**, **22**, **23** and **24** are humongous, those are easy to detect; whatever prediction out of the range [0,1] can be automatically excluded. The reason why those errors can be so greater is based on the linear model used for prediction (39) and the values of their weights. A priori, a linear model has no bounds for its predictions so the same happens for its errors; those are not limited and only depend on the precision of the regression itself. This is one of the problems explained in section 4.2.2 for justifying the introduction of *saturated linear models* that enclose any prediction coming from a linear model into the ranges [0,1].

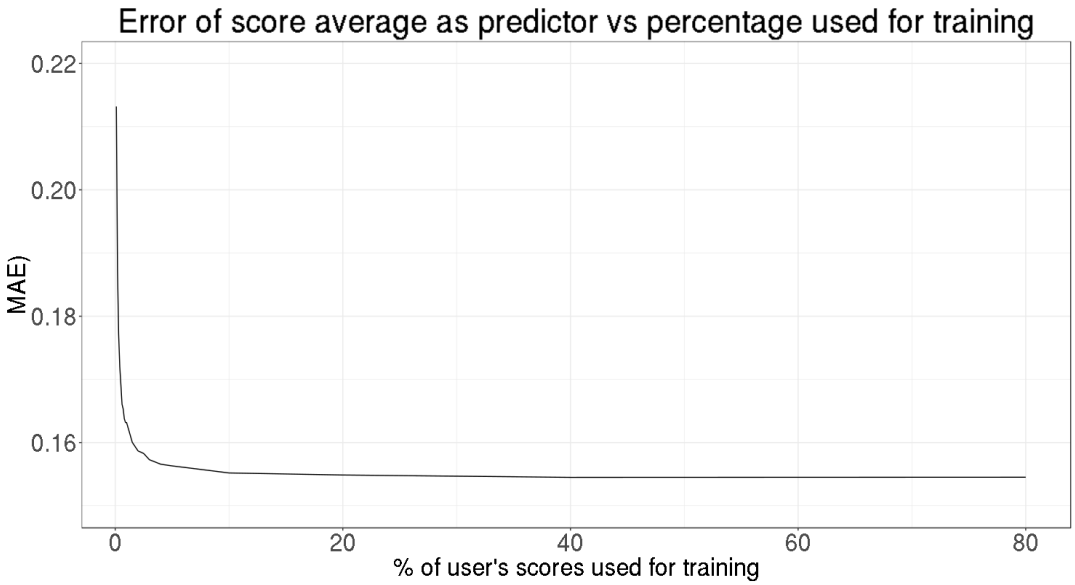

Figure 25. Error of predictions using as model just the user's average of score. The predictions has been performed using different percentages of the user's scores for training. The maximum and minimum errors are between the range (0.15, 0.22), falling this error very quickly when increasing slightly the number of scores used for evaluating the model.

5.4.7. Improvements to K-means and linear regression

Based on the conclusions of previous experiment 5.4.6, it seems obvious that applying some kind of method that can guarantee more stable predictions, preferably into the range [0,1], will lead to an obvious reduction of the prediction errors. In order to do that, this exercise covers two small improvements that can be applied to the K-means and linear regression on several profile levels for the users. Those improvements are the following:

1. Using the **saturated rectifier** for improving the output of the linear model. That should help to contain the extreme predictions when those get out of range. That may happen in the case of small profiling percentages, where the number of clusters *K* is similar to the total number of scores used. In those cases, the found weights may lead to unexpected values instead of taking advantage of a robust regression with lots of samples.
2. Introducing the **dense clusters** approach for having more stable weights. Using sensible values for the non-evaluated elements of the scores vector avoids using extreme values like 0 as part of the evaluations (section 4.1.9). That helps to have a more viable weight for a cluster c_k that reduces the range of the predictions within the limits [0,1].

These two actions are only changes introduced, the rest of the experiment is under the constrains and conditions described for the experiment *K-means and linear regression for different profiles*.

5.4.8. Results of the improvements

After introducing the previously mentioned improvements (saturated rectifier and dense clusters), their results are shown at **Figure 26** and **Figure 27** for the different order of percentages of scores used for profiling the p-users. The experiment has been performed with the *Prolific IMDb Users* dataset. These figures can be compared directly with **Figure 21** and **Figure 23** and the exact mapping between them corresponds as {*Figure 26* → *Figure 21*} and {*Figure 27* → *Figure 23*}. When comparing the results from **Figure 26** with **Figure 21** it is shown a clear improvement in the error for the new results that include both improvements. The MAE obtained applying *dense clusters* and the *saturated rectifier* is clearly under 0.4, whilst the previous experiments arrive to values above 1 very easily. The general reduction happens in the order of magnitude. That is even more important when considering that the final results obtained and shown at **Figure 26** have not been filtered at all. The methods applied introduce improved accuracy and stability. The same happens when comparing the group of great percentages of scores shown at **Figure 27** and **Figure 23** where the prediction error is also noticeably reduced. It is easy to conclude that those improvements combined have empirical evidence supporting the rationale that justify their use.

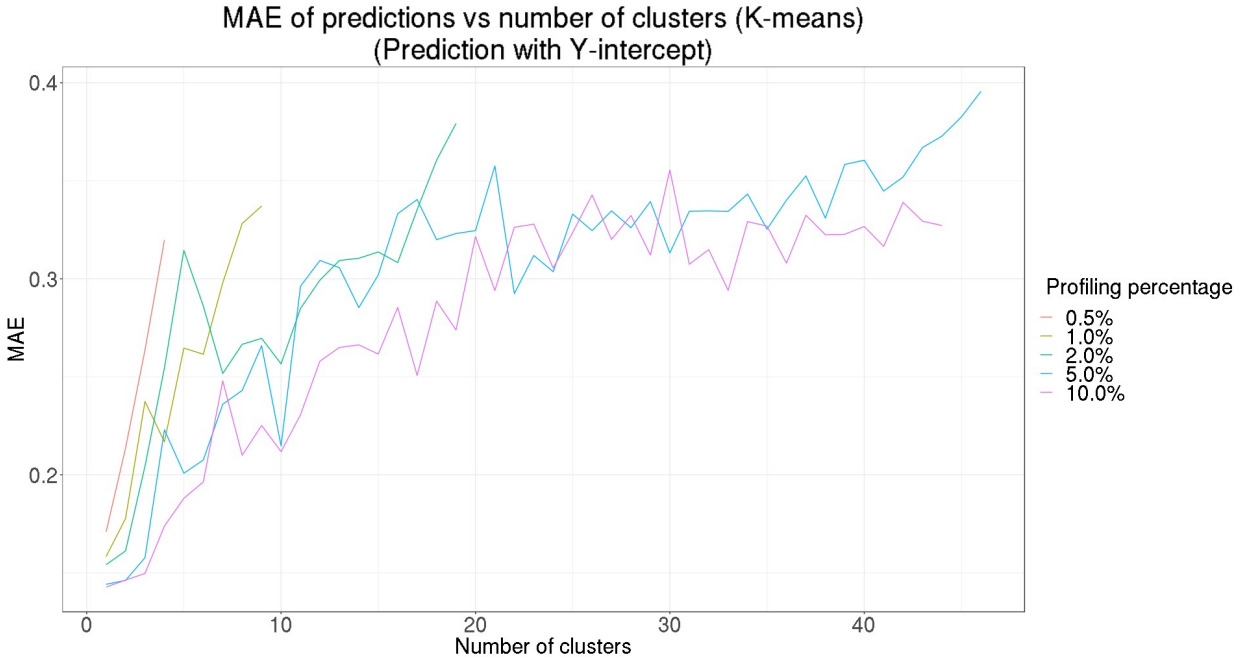

Figure 26. First group of profiling percentages (smallest ones) and evolution of their errors (MAE) when increasing the number of clusters.
The clusters have been treated to become dense and a saturated rectifier has been applied to the predictions.

Despite it is not the reason why the saturated rectifier has been introduced, it also can be used for evaluating the difference of errors between a specific model that uses it for compacting its output and the same model without this restriction. This difference of errors can help to understand the impact of the saturated rectifier or, viewing it from a different angle, how much the error is increased because of predictions that fall out of the range [0,1]. That is useful for having a specific metric about if the model is using enough information for guarantee

the stability of its predictions into the valid range [0,1]. As can be seen in the comparison of the group of great percentages of scores at **Figure 27** and **23**, the error is reduced using linear saturation in an inverse proportion to the percentage of scores consumed for profiling. That can be understood as the model has more information for understanding that predictions must fall in the valid range. In other words, the model not only has more understanding for performing better predictions but it also learns what is the expected range of values where the predictions can fall in.

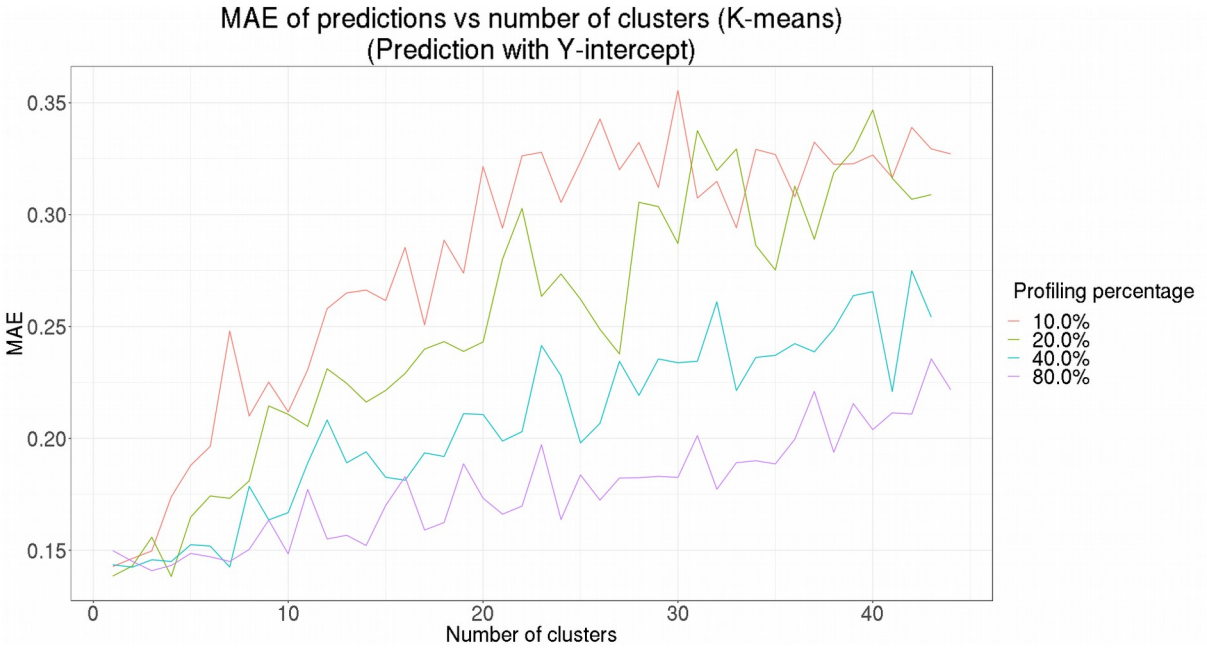

Figure 27. Second group of profiling percentages (greatest ones) and evolution of their errors (MAE) when increasing the number of clusters.
The clusters have been treated to become dense and a saturated rectifier has been applied to the predictions.

5.4.9. Model competition

This is the last one of the experiments performed and it has been evaluated using the *Prolific IMDb Users* dataset, its dense version and also *Movielens*. It is a global comparison of different ways of obtaining the clusters and the weights; this also include small variations like using rectifiers, different comparison of unknown scores for K-means, including the use or not of Y-intercept and others. The codes identifying the elements that can be combined are gathered at the **Table 5**. Using the acronyms for the elements from this table, there are generated the different pipelines shown at **Table 6**.

Code	Used for	Element
UAvg	Evaluating of missing values for the K-means distance	Average of know user's scores
null	Evaluating of missing values for the K-means distance	Null as default value
dense	Clusters' scores vector improvement	Dense clusters
LR	Profiling	Linear regression
Eqw	Profiling	Equal weights
BFGS	Profiling	BFGS
SApr	Profiling	Successive approximations
CWO	Clustering method + profiling	Cluster-weight optimization
inv(0.05)	Profiling	Inverse error with size factor 0.05
inv(0.1)	Profiling	Inverse error with size factor 0.1
inv(0.2)	Profiling	Inverse error with size factor 0.2
inv(0.5)	Profiling	Inverse error with size factor 0.5
inv(1.0)	Profiling	Inverse error with size factor 1.0
LS	Prediction rectification	Lineal saturation
w[-1,1]	Constrain applied to profiling	Weights between ±1
w[0,1]	Constrain applied to profiling	Weights normalized
w[0,x]	Constrain applied to profiling	Positive weights
YI	Constrain applied to profiling	Y-intercept is part of the model
NYI	Constrain applied to profiling	Y-intercept is null

Table 5. List of all the different elements to combine in pipelines. Elements represent models, rectifiers, constrains and others that can be combined in order to generate pipelines of algorithms that predict the expected scores of a p-user to certain films.

The result expected from this experiment is the final prediction error (MSE) for every of the methods in the **Table 6**. The content of this table has been created combining different models and approaches from the previous sections, but it also includes two concrete cases:

- *baseline-user-avg* is the code for a baseline model that is very easy to implement when there is little information about the user. The model just provides a prediction based on his/her mean scores from the previous reviews, even if there are just a fistful or one.

- *baseline-film-avg* is the other code for the second baseline model. It is very similar to *baseline-user-avg* but with the difference that the aggregation level is performed around the film, so all the previously known scores about this film are averaged for generating the prediction.

Short name	Clustering	FFNS	Complement to clustering	Profiling	Complement to profiling	Rectifier
null-dense-LR	K-means	null	dense	LR	-	-
null-dense-Eqw	K-means	null	dense	Eqw	-	-
null-dense-Eqw-NYI	K-means	null	dense	Eqw	NYI	-
null-dense-SApr	K-means	null	dense	SApr	-	-
null-dense-SApr-LS	K-means	null	dense	SApr	-	LS
null-dense-BFGS-w[-1,1]	K-means	null	dense	BFGS	w[-1,1]	-
null-dense-BFGS-w[-1,1]-LS	K-means	null	dense	BFGS	w[-1,1]	LS
null-dense-BFGS-w[0,1]	K-means	null	dense	BFGS	w[0,1]	-
null-dense-BFGS-w[0,1]-LS	K-means	null	dense	BFGS	w[0,1]	LS
null-dense-BFGS-w[0,x]	K-means	null	dense	BFGS	w[0,x]	-
null-dense-CWO	K-means	null	dense	CWO	-	-
UAvg-dense-LR	K-means	UAvg	dense	LR	-	-
UAvg-dense-Eqw	K-means	UAvg	dense	Eqw	-	-
UAvg-dense-Eqw-NYI	K-means	UAvg	dense	Eqw	NYI	-
UAvg-dense-Inv(0.05)	K-means	UAvg	dense	Inv(0.05)	-	-
UAvg-dense-Inv(0.1)	K-means	UAvg	dense	Inv(0.1)	-	-
UAvg-dense-Inv(0.2)	K-means	UAvg	dense	Inv(0.2)	-	-
UAvg-dense-Inv(0.5)	K-means	UAvg	dense	Inv(0.5)	-	-
UAvg-dense-Inv(1.0)	K-means	UAvg	dense	Inv(1.0)	-	-
UAvg-dense-BFGS-w[-1,1]	K-means	UAvg	dense	BFGS	w[-1,1]	-
UAvg-dense-CWO	K-means	UAvg	dense	CWO	-	-
Short name	**Prediction method**					**Rectifier**
baseline-user-avg	Every user's prediction is the average of their known scores					-
baseline-film-avg	Every film's prediction is the average of their known scores					-

Table 6. Different pipelines formed from combinations of several elements. Pipelines of algorithms predict the expected scores of a p-user to certain films. At the table's bottom there are two additional entries for the baseline models that are used for providing a reference of the error values.

5.4.10. Results of the model competition

The model competition results for the Prolific IMDb Users dataset are shown in the **Table 7** where every row identifies one of the possible combinations described in the **Table 6**.

The short name of these combinations are shown in the first column (with the same name) of the **Table 7** and every additional column is the results associated to a specific percentage of scores per p-user used for profiling it. Every cell (combination of row and column) of this table shows information about the optimal number of clusters that minimize the error. That is obtained repeating the combination of methods given by the row with the scores percentages identified by the column several times using from 1 to 63 clusters, and from those results selecting the result with lesser error. This information is shown in the cell, which is divided in two lines; the first line is the error measured as the MAE and the second the value of $|C|$ that minimized this error. For every column with different percentages of scores, it is selected a cell highlighted with boldface that indicates the combination with lesser error among all in the same column.

Combination	0.1%	0.2%	0.3%	0.4%	0.5%	1%	10%	20%	40%	80%
null-dense-LR	-	98.539 C1	3.154 C1	3.455 C1	0.6498 C1	0.1982 C1	0.1604 C1	0.1563 C1	0.1462 C1	0.1432 C3
null-dense-Eqw	0.1695 C30	0.1681 C31	0.1676 C38	0.1694 C35	0.1662 C39	0.1641 C26	0.1666 C40	0.1681 C37	0.1672 C42	0.1683 C34
null-dense-Eqw-NYI	0.1644 C32	0.1676 C43	0.1660 C23	0.1672 C21	0.1672 C24	0.1668 C6	0.1681 C21	0.1671 C40	0.1673 C18	0.1679 C12
null-dense-SApr	0.1935 C19	0.1846 C1	0.1711 C1	0.1708 C1	0.1646 C1	0.1624 C1	0.1477 C1	0.1555 C1	0.1502 C1	0.1562 C1
null-dense-SApr-LS	0.1946 C19	0.1940 C1	0.1739 C1	0.1810 C1	0.1673 C1	0.1653 C1	0.1570 C1	0.1465 C1	0.1529 C1	0.1487 C1
null-dense-BFGS-w[-1,1]	0.1916 C3	0.1792 C37	0.1698 C24	0.1723 C36	0.1729 C21	0.1676 C9	0.1621 C4	0.1551 c2	0.1573 C1	0.1644 C39
null-dense-BFGS-w[-1,1]-LS	0.1833 C22	0.1763 C27	0.1707 C29	0.1724 C34	0.1743 C25	0.1694 C38	0.1570 C2	0.1623 C2	0.1570 C1	0.1649 C33
null-dense-BFGS-w[0,1]	0.1925 C34	0.1792 C44	0.1716 C32	0.1733 C30	0.1724 C35	0.1670 C3	0.1647 C3	0.1569 C2	0.1593 C2	C.1646 C14
null-dense-BFGS-w[0,1]-LS	0.1866 C13	0.1755 C37	0.1728 C41	0.1705 C37	0.1707 C2	0.1713 C6	0.1638 C1	0.1581 C1	0.1606 C1	0.1659 C41
null-dense-BFGS-w[0,x]	0.1919 C39	0.1774 C41	0.1701 C37	0.1725 C1	0.1728 C39	0.1678 C41	0.1663 C2	0.1628 C19	0.1597 C2	0.1694 C6
null-dense-CWO	0.1854 C23	0.1739 C34	0.1737 C32	0.1697 C38	0.1703 C36	0.1672 C30	0.1654 C10	0.1669 C19	0.1655 C29	0.1671 C37
UAvg-dense-LR	-	∞	∞	∞	∞	∞	**0.1446 C2**	**0.1371 C1**	**0.1412 C1**	**0.1374 C5**
UAvg-dense-Eqw	0.1698 C11	0.1670 C38	0.1684 C25	0.1677 C11	0.1660 C20	0.1654 C38	0.1655 C11	0.1648 C19	0.1625 C17	0.1643 C11
UAvg-dense-Eqw-NYI	0.1639 C7	0.1620 C6	0.1643 C5	0.1658 C6	**0.1587 C9**	0.1662 C5	0.1648 C12	0.1643 C7	0.1612 C8	0.1640 C8
UAvg-dense-Inv(0.05)	0.1604 C11	0.1613 C14	**0.1580 C14**	**0.1578 C7**	0.1590 C5	**0.1564 C5**	0.1546 C7	0.1560 C4	0.1564 C22	0.1560 C5
UAvg-dense-Inv(0.1)	0.1604 C8	0.1620 C7	0.1599 C10	0.1605 C6	0.1591 C14	0.1581 C7	0.1562 C8	0.1581 C5	0.1581 C5	0.1581 C6
UAvg-dense-Inv(0.2)	**0.1600 C5**	**0.1605 C7**	0.1616 C10	0.1588 C4	0.1628 C6	0.1612 C8	0.1631 C11	0.1589 C6	0.1584 C14	0.1624 C4
UAvg-dense-Inv(0.5)	0.1633 C2	0.1620 C5	0.1614 C12	0.1588 C5	0.1602 C11	0.1609 C9	0.1597 C10	0.1626 C18	0.1645 C14	0.1629 C8
UAvg-dense-Inv(1.0)	0.1611 C7	0.1619 C3	0.1605 C16	0.1650 C6	0.1625 C5	0.1638 C10	0.1627 C6	0.1640 C12	0.1631 C8	0.1621 C9
UAvg-dense-BFGS-w[-1,1]	0.1822 C8	0.1756 C15	0.1667 C34	0.1682 C4	0.1670 C18	0.1650 C2	0.1605 C5	0.1583 C1	0.1567 C1	0.1635 C30
UAvg-dense-CWO	0.1764 C13	0.1756 C25	0.1711 C27	0.1702 C16	0.1676 C20	0.1661 C11	0.1662 C10	0.1651 C5	0.1662 C12	0.1656 C28
baseline-user-avg	0.2112	0.1875	0.1778	0.1725	0.1697	0.1618	0.1548	0.1546	0.1546	0.1545
baseline-film-avg	0.1804	0.1791	0.1829	0.1805	0.1802	0.1821	0.1963	0.2088	0.2088	0.2291

Table 7. Results for different combinations of pipelines. Different combinations of algorithms predict the expected scores of a p-user and films of the *Prolific IMDb Users* dataset. Every row one combination of several methods and every column identifies a specific percentage of scores used for profiling p-users. Cells contain the smallest error for a whole range of clustering options going from 1 to 63 clusters, showing the MAE and the cluster with lesser error.

It is easy to see that the best combinations are based on heuristic models for small percentages of scores (for profiling) until 1%, from 10% until 80% the linear regression is systematically the most accurate model. The heuristic weights models are the *inverse error* in

combination with the *average default value* and values 0.2 and 0.05 for the parameter *γ* and very closely to this model the *equal weights* (without Y-intercept) combined with the same processing variants.

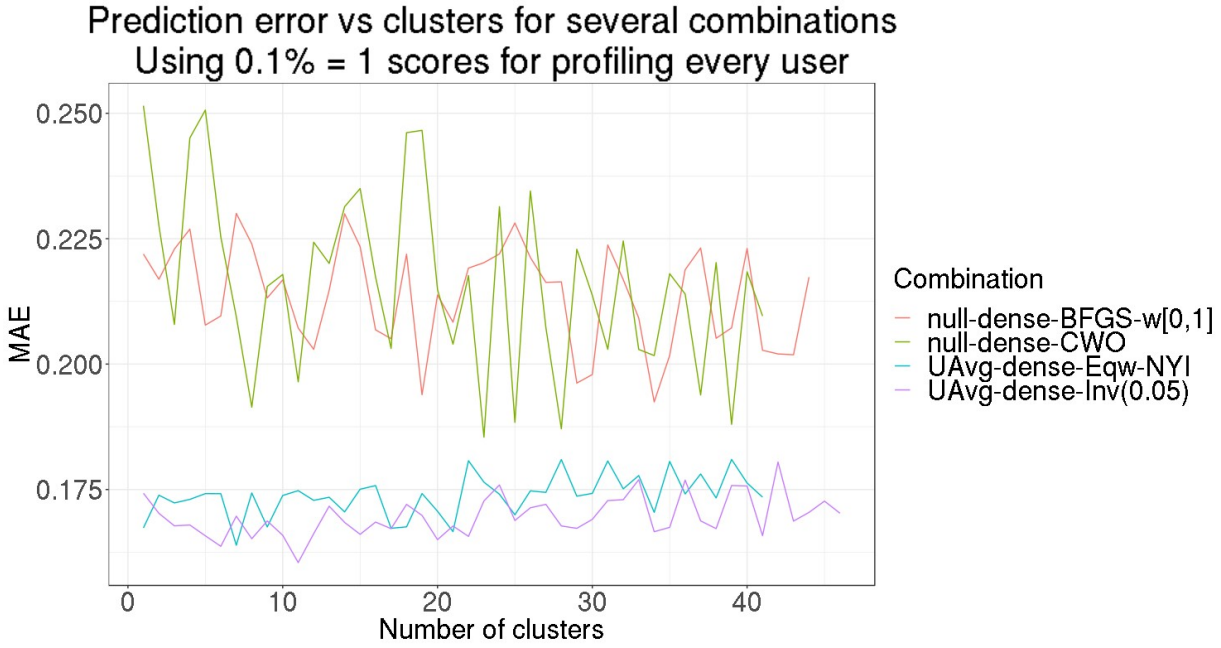

Figure 28. Prediction error vs clusters for several combinations using 0.1% of scores for profiling.

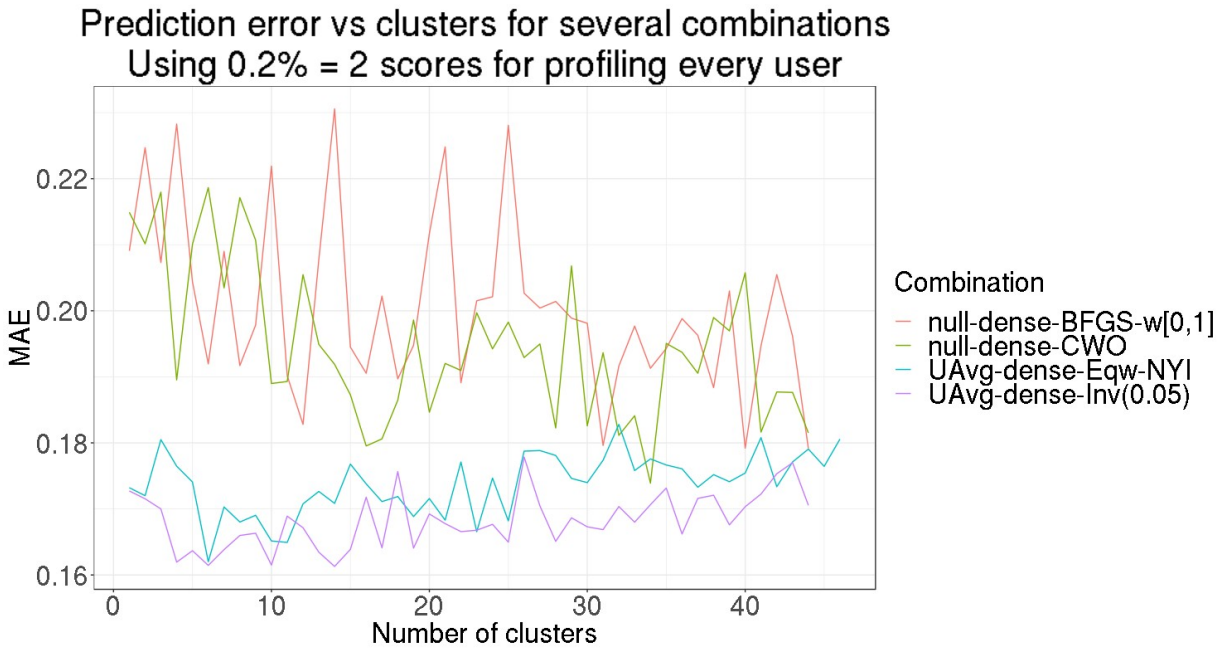

Figure 29. Prediction error vs clusters for several combinations using 0.2% of scores for profiling.

Additionally to those general results at **Table 7**, the figures from **Figure 28** to **Figure 32** show a specific zoom on the results of model BFGS (and others for reference). **Figure 28** exposes the case of using just 1 known score per user for profiling where the most

simple methods *equal weights* and *inverse error* have a clear better performance to more sophisticated ones like BFGS and CWO. When increasing the total known scores per user to 2 (0.2%), **Figure 29** shows that the two couples of models are closer in terms of error, but still with a clear difference that begins to smooth when using 32 clusters or more; at this range the error series begin to slightly mix. This progressive approach of the prediction errors is stimulated by the increase in the known scores as can be seen at **Figure 30**, **Figure 31** and **Figure 32**, that show the equivalent series of errors for the (scores) percentages 0.5%, 10% and 80% respectively.

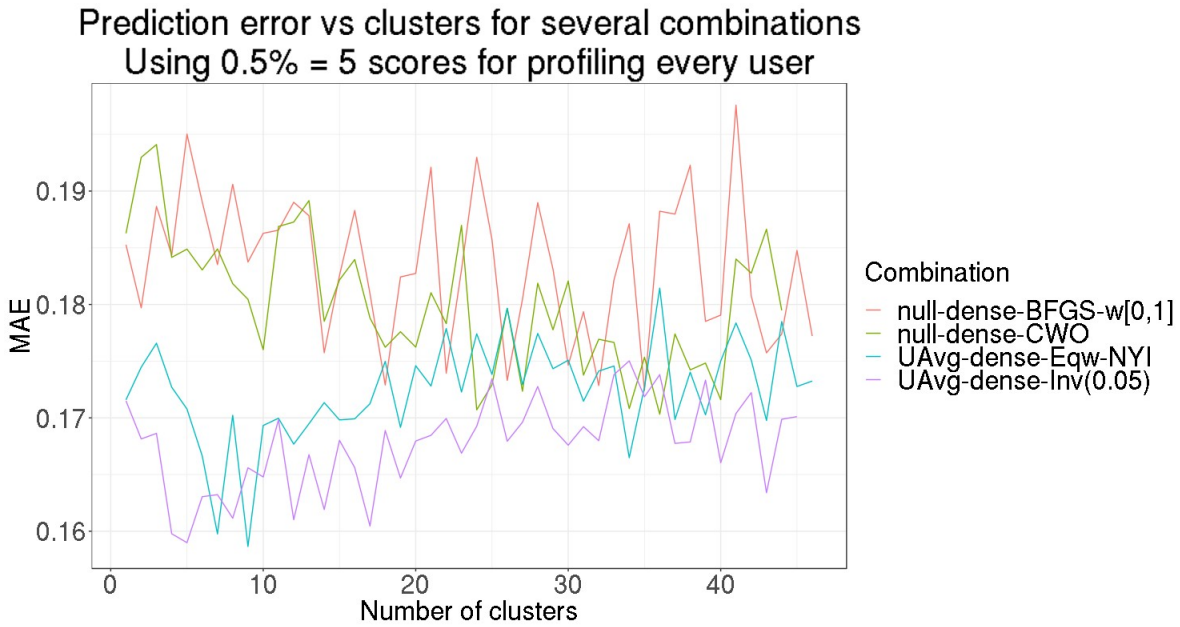

Figure 30. Prediction error vs clusters for several combinations using 0.5% of scores for profiling.

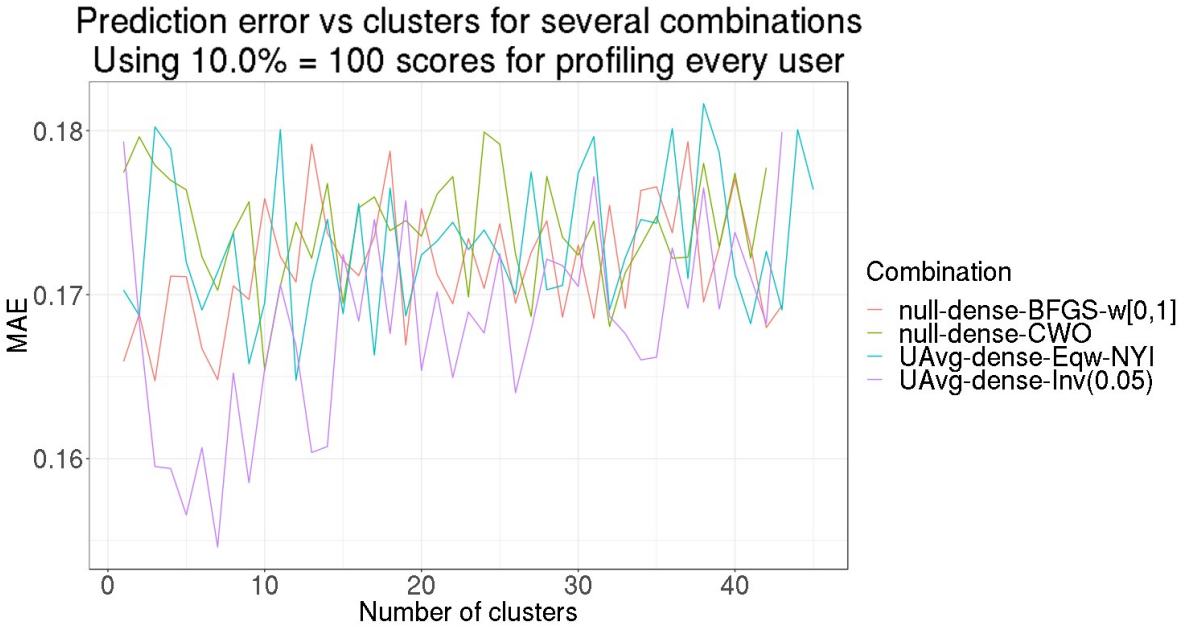

Figure 31. Prediction error vs clusters for several combinations using 10% of scores for profiling.

It is also worth mentioning the gap from **Figure 31** to **Figure 32** (from 10% of scores to 80%) in terms of error. For lesser percentages of known scores it is a clear reduction of the error for every of the different ways of obtaining the weights of the users (equal weights, inverse error, BFGS and CWO), but starting from this 10% this trend is stuck and in general the error is not reduced by increasing the total number of known scores.

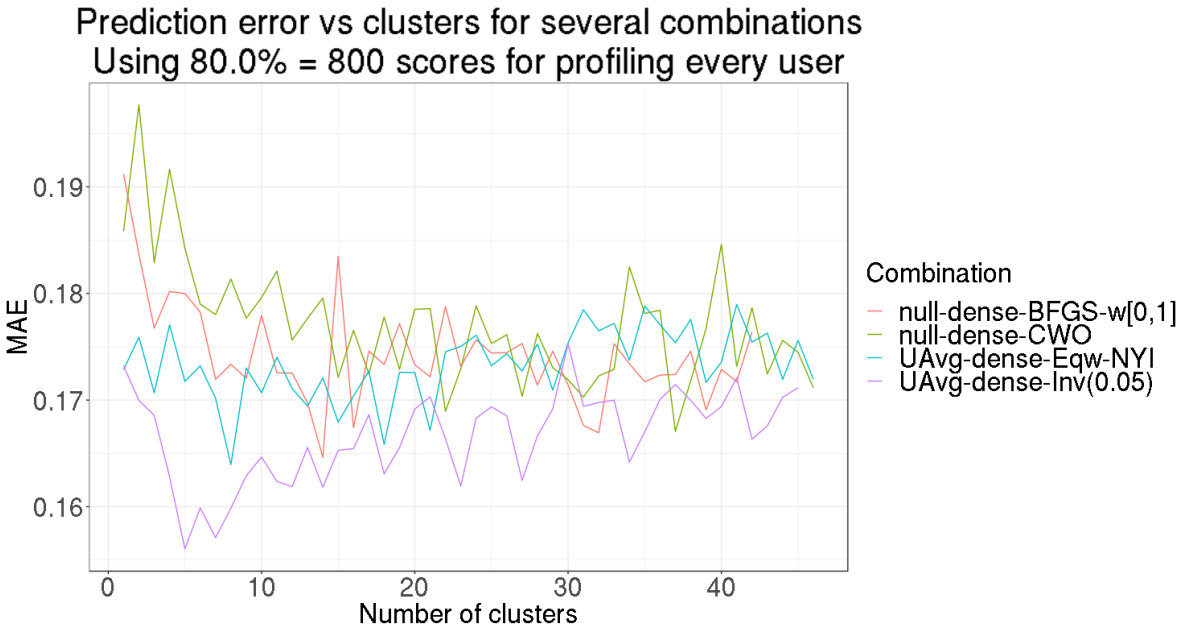

Figure 32. Prediction error vs clusters for several combinations using 80% of scores for profiling.

For the case of the *dense Prolific IMDb Users* subset the **Table 7** shows the model competition for some of the available combinations of pipelines. As its equivalent results **Table 7**, the **Table 8** has highlighted in boldface the combinations with lesser error (MSE) for every column. As general conclusion can be said that the error is reduced around 1% for the best combinations when comparing the best results with their equivalent in the **Table 7**.

Combination	0.1%	0.2%	0.3%	0.4%	0.5%	1%	10%	20%	40%	80%
null-dense-BFGS-w[0,1]	0.1763 C14	0.1714 C37	0.1681 C28	0.1679 C44	0.1668 C36	0.1603 C16	0.1548 C35	0.1570 C5	0.1527 C5	0.1521 C16
null-dense-BFGS-w[0,1]-LS	0.1852 C19	0.1803 C23	0.1717 C36	0.1682 C3	0.1626 C36	0.1605 C6	0.1585 C6	0.1567 C1	0.1533 C1	0.1499 C1
null-dense-CWO	0.1678 C40	0.1715 C29	0.1625 C4	0.1631 C3	0.1644 C10	0.1602 C37	0.1548 C39	0.1547 C4	0.1559 C18	0.1572 C16
UAvg-dense-LR	-	∞	∞	∞	∞	∞	**0.1332 C1**	**0.1346 C1**	**0.1338 C1**	**0.1300 C3**
UAvg-dense-Eqw-NYI	0.1515 C1	0.1540 C6	0.1546 C7	0.1518 C1	0.1549 C12	0.1518 C6	0.1530 C5	0.1518 C10	0.1536 C6	0.1518 C15
UAvg-dense-Inv(0.05)	**0.1500 C7**	**0.1502 C6**	**0.1496 C8**	0.1509 C4	0.1507 C8	0.1511 C3	0.1485 C7	0.1474 C3	0.1471 C8	0.1428 C13
UAvg-dense-Inv(0.2)	0.1525 C6	0.1509 C6	0.1516 C6	**0.1482 C7**	**0.1484 C9**	**0.1491 C7**	0.1483 C9	0.1491 C5	0.1441 C3	0.1484 C6
baseline-user-avg	0.2109	0.2142	0.1944	0.1889	0.1796	0.1711	0.1567	0.1559	0.1558	0.1545
baseline-film-avg	0.1730	0.1747	0.1760	0.1737	0.1757	0.1753	0.2028	0.2211	0.2216	0.2132

Table 8. Results for different combinations of pipelines for the *dense Prolific IMDb Users* subset.

For a better understanding of the MSE difference when using the *Prolific IMDb User* dataset and its dense subset, the **Figure 33** shows a summary of the best results coming from these subset and dense selection. The two series appearing at **Figure 33** are created selecting for every percentage of known scores the best error from **Table 7** and **Table 8** respectively. Despite the scale for the abscisses is neither linear nor logarithmic it clearly exposes how this error is reduced for the best combination of every series whilst increasing the percentage of used scores. Despite the two series show a similar behaviour keeping a difference in benefit of the *dense subset*, in the case of percentage 0.1% of scores the error difference is 1% globally and when arriving at the percentage 80% their difference is reduced to a global 0.75%. There are fluctuations where sometimes using more scores seems to lead to worse errors but those seem caused by how scores per used are distributed; in other words, those are specific of the dataset and its data.

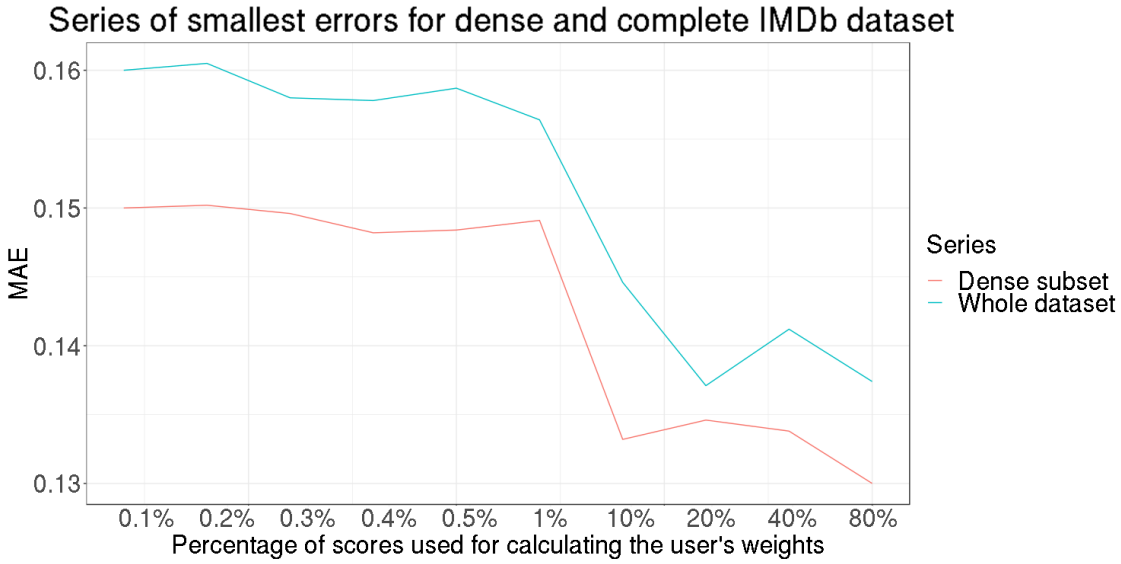

Figure 33. Series of smallest errors for the dense and complete *Prolific IMDb User* dataset.

Combination	0.1%	0.2%	0.3%	0.4%	0.5%	1%	10%	20%	40%	80%
UAvg-dense-LR	-	∞	∞	∞	∞	∞	0.1481 C1	**0.1358 C1**	**0.1340 C1**	**0.1314 C3**
UAvg-dense-Inv(0.05)	0.1475 C16	0.1479 C10	C.1473 C20	**0.1461 C7**	0.1461 C19	0.1469 C3	**0.1453 C4**	0.1454 C19	0.1446 C9	0.1441 C4
UAvg-dense-Inv(0.1)	**0.1474 C2**	0.1483 C13	**0.1472 C9**	0.1472 C3	0.1472 C17	**0.1459 C9**	0.1465 C4	0.1462 C17	0.1455 C9	0.1449 C9
UAvg-dense-Inv(0.2)	0.1478 C2	**0.1465 C4**	0.1477 C2	0.1483 C14	**0.1458 C3**	0.1479 C6	0.1475 C11	0.1462 C9	0.1485 C14	0.1473 C19
baseline-user-avg	0.1948	0.1898	0.1863	0.1815	0.1796	0.1686	0.1491	0.1475	0.1463	0.1460
baseline-film-avg	0.1712	0.1711	0.1705	0.1722	0.1732	0.1754	0.1972	0.2006	0.1999	0.1953

Table 9. Results for different combinations of pipelines for the *Movielens* database.

The last dataset evaluated is the *Movielens* database which has a different distribution of users and scores per film. This different content processed with the same combinations of pipelines obtains different results but similar behaviours. Those are shown at **Table 9**, where

not all the possible ones are exposed but only the ones with the best results from the **Table 7** and **Table 8** in order to being able to make comparisons. These comparisons evidence the importance of the dataset for having better or worse results, at least for the smaller percentages of scores used for profiling. Comparing the results from *Prolific IMDb Users* and *Movielens* starting from 1% to 80% (of scores) are improved with the latter but not as much as for the percentages from 0.5% (of scores) and below, where the reduction of the MAE is greater than 1%. On another note, the same comparison between the *dense Prolific IMDb Users* and the *Movielens* database shows a smoother difference. For the small percentages (of scores) from 0.1% to 0.5% the *Movielens* database provides better results with improvements of less than 0.5% of MAE, but for greater percentages (of scores) from 1% to 80% the *dense Prolific IMDb Users* subset is the one improving the MAE reducing it until more than 1% in some specific cases.

5.4.11. Understanding the clusters

Despite in the previous experiments the methodology has been detailed as far as the results obtained, the clustering process can be studied at results level. In section 4.1.6 it has been explained how the K-means algorithm works and what can be obtained from it, but the results shown so far do not provide light about how specifically those clusters look like. In order to focus in some practical cases, this experiment explores applying K-means to the *Prolific IMDb User* dataset in conjunction with the two methods for evaluating the c-users' missing scores. That is when the c-user has not rated a film but some significant value is needed for being able to evaluate the distance between this c-user and others or the centroids. At section 4.1.7 it was proposed the *null default value* and at section 4.1.8 the *average default value*, both used in almost all the previous experiments.

As the results have exposed that *average default value* is the approach that helps to provide the smallest errors (see **Table 7**), it is not clear how this is affecting the way the clusters are generated. For giving visibility to all questions, this exercise is divided in a collection of different appliances of K-means with the following specific details gathered at **Table 10**.

Combination code	Number of clusters	Default value method
null-2	2	null default value
avg-2	2	average default value
null-5	5	null default value
avg-2	5	average default value
null-10	10	null default value
avg-10	10	average default value
null-40	40	null default value
avg-40	40	average default value

Table 10. List of segmentations performed with K-means.

5.4.12. Results for different segmentations

The results for the experiments with K-means are shown in the following figures of this section. For the case of *null default value* and 2 clusters, the **Figure 34** evidences that K-means produces a cluster for containing all the users except one (outlier) that has his/her own cluster.

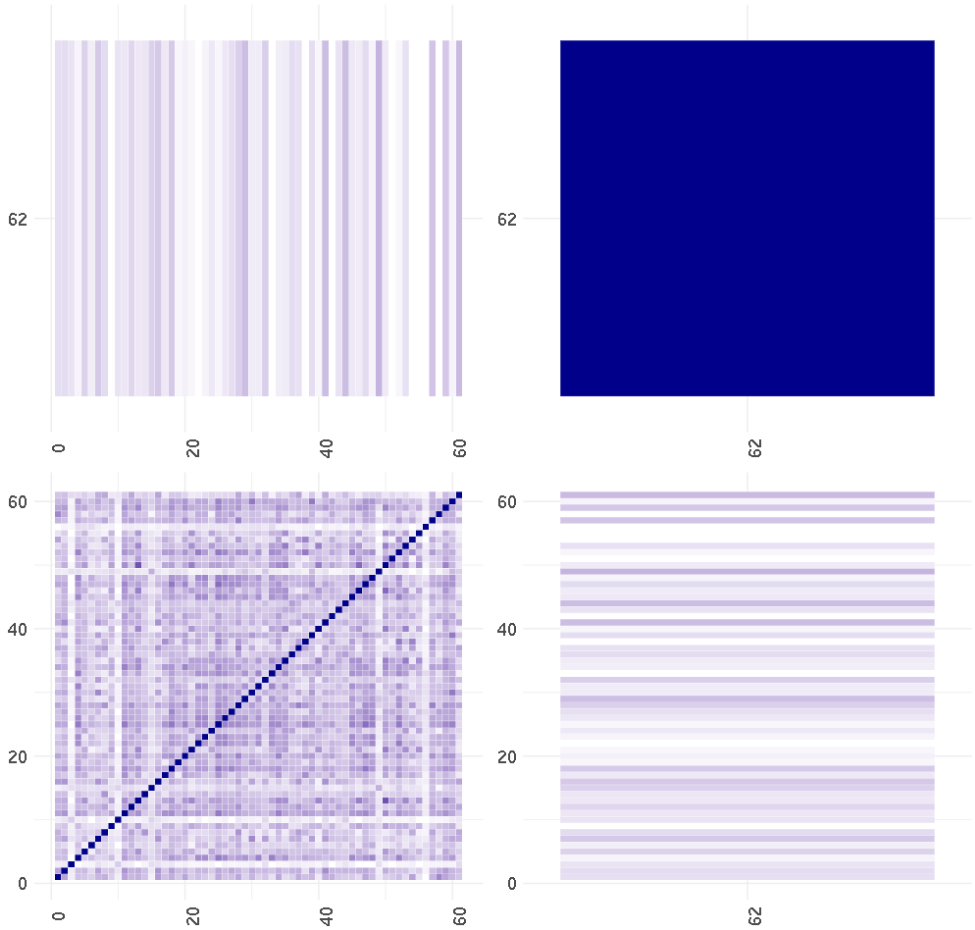

Figure 34. Results of K-means with 2 clusters and *null default value*.

The **Figure 34** is a matrix of range 2×2 where every row i and column j represents a cluster, so given a cell (i,j) this is the comparison of cluster c_i with cluster c_j. The cell (i,j) is in itself nesting a user matrix comparing the c-users from cluster c_i and the c-users from cluster c_j. These nested user matrices have $|c_i|$ rows and $|c_j|$ columns. The way those cardinalities are evaluated is considering $|c_i|$ the total number of c-users belonging to c_i and $|c_j|$ the total number of c-users belonging to c_j. Basically, every user matrix (i,j) is a *heat map* [112] where its cells (k,l) show a colour intensity from white to dark blue proportional to the number of common reviewed films of the k-th c-user and the l-th c-user. The cases where i=j and k=l show the maximum intensity in dark blue because it is a c-user compared to itself obtaining a value of 1000 scored films for this *Prolific IMDb User* dataset. The cases where i=j compare a

cluster with itself, showing the nested matrix of its users while the rest $i \neq j$ compare different clusters. The **Figure 35** is an equivalent clustering for 2 centroids using the *user default value* approach; this figure has to be interpreted with the same format and colour code as the **Figure 34** despite its content is completely different.

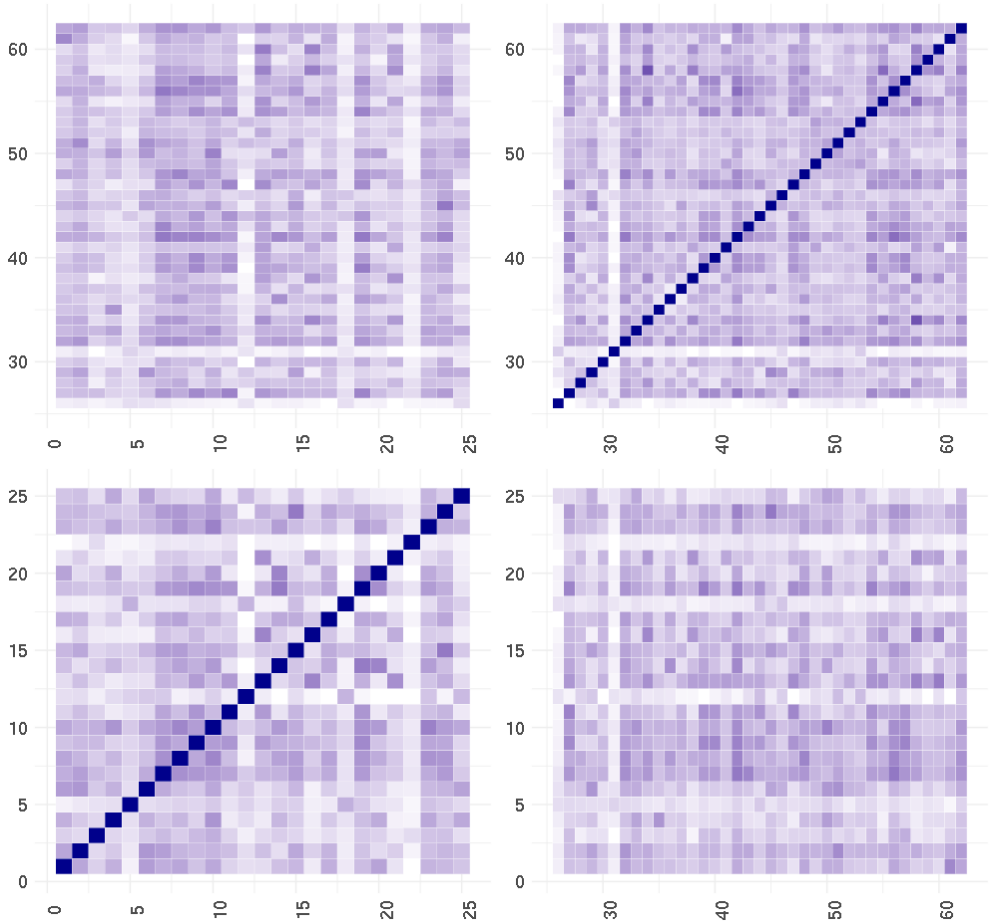

Figure 35. Results of K-means with 2 clusters and *average default value*.

Despite both clusterings have been performed on the same dataset *Prolific IMDb User* dataset it is clear that **Figure 35** shows a more equilibrated clustering, distributing the population in 26 c-users for the first cluster and 36 c-users for the second one. The second clustering does not considers users as outliers.

The results of clustering again with *null default value* and *average default value* using 5 clusters are exposed in the **Figure 36** and **Figure 37** respectively. These show a parallelism to what has been observed comparing both methods with 2 clusters in the sense that **Figure 36** shows again 5 clusters where almost all the population is in the 2nd cluster and the other four clusters contain only 1 or 2 c-users. When clustering with *null default value* the population use to be considered pretty homogeneous except for a few users considered outliers and placed in additional clusters as K-means considers them not similar enough (even between them).

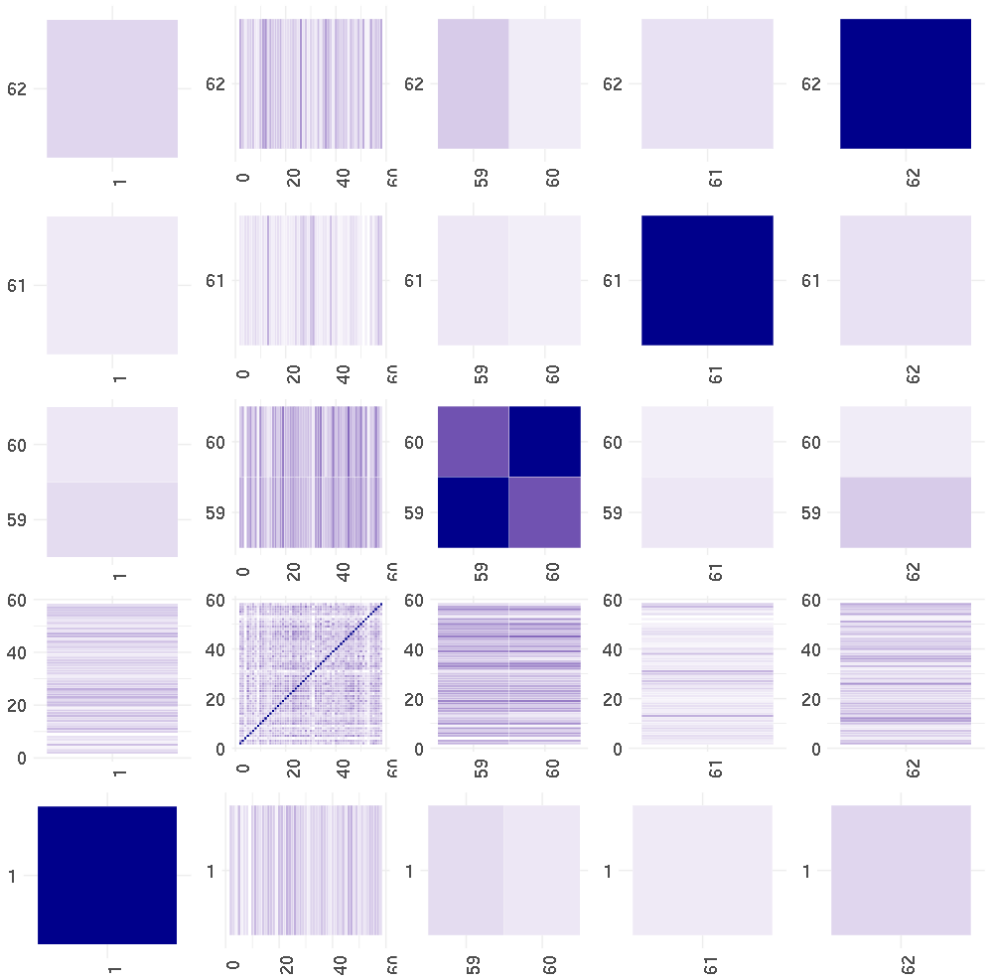

Figure 36. Results of K-means with 5 clusters and *null default value*.

The **Figure 37** shows how *average default value* helps K-means to find a more equilibrated distribution in terms of the clusters' sizes. The clusters have relative close sizes except the last one with just a couple of users that the algorithm interprets as outliers that deserve their own cluster as not being similar enough to the rest of them.

For the next cases where it has been used 10 clusters, the previous heat map requires so much resolution that makes it unfeasible for being used to illustrate the results. These are presented using a simpler approach counting every cluster's users in bars. The **Figure 38** shows that for the *null default value* that follows its usual trend of concentrating the majority of users in one cluster and isolating the rest as exceptions in their own clusters of one or two users. On another note, the **Figure 39** shows again how *average default value* finds more proportional clusters. In this case it is clear that users are distributed with closer similarities in a population-descendant way, as the first cluster is the greatest one with 18 members, the second has 16 and so on, being the latest clusters the ones keeping two pairs of outliers each one.

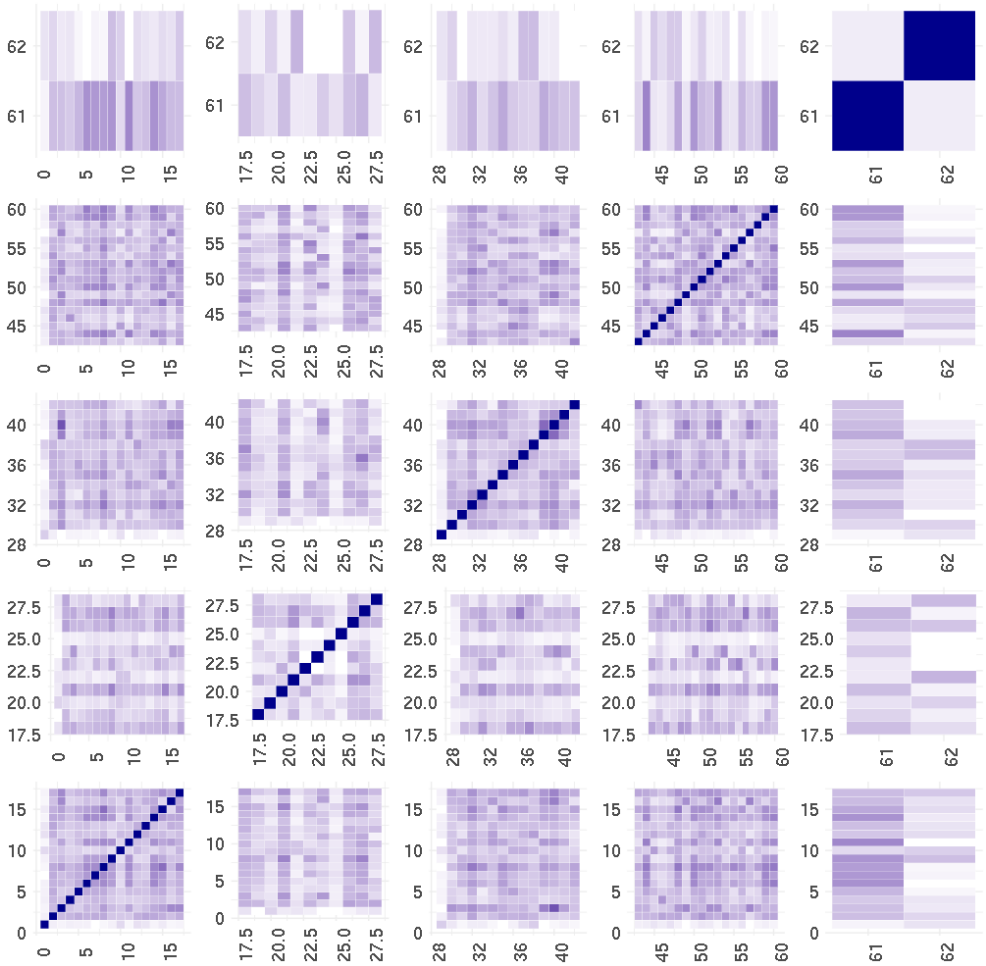

Figure 37. Results of K-means with 5 clusters and *average default value.*

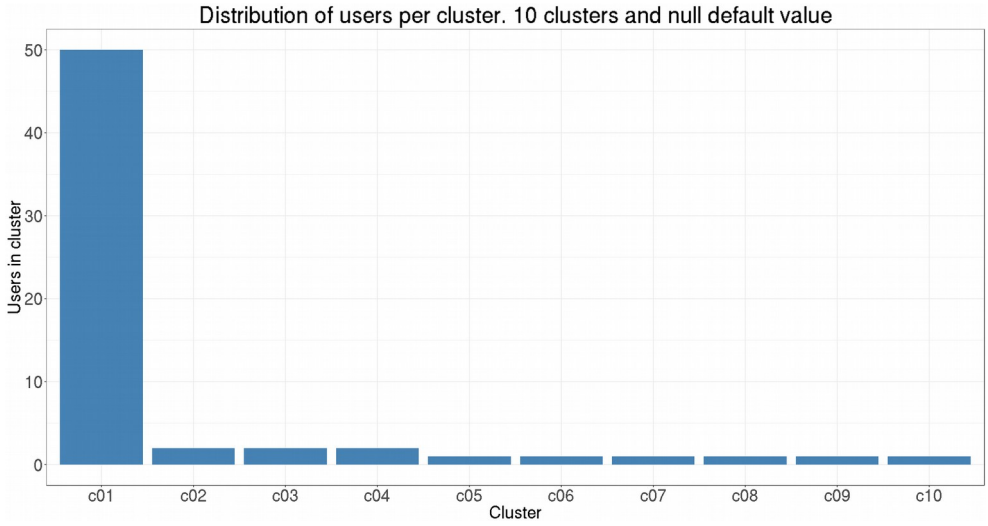

Figure 38. Distribution of users per cluster for 10 clusters and *null default value*.

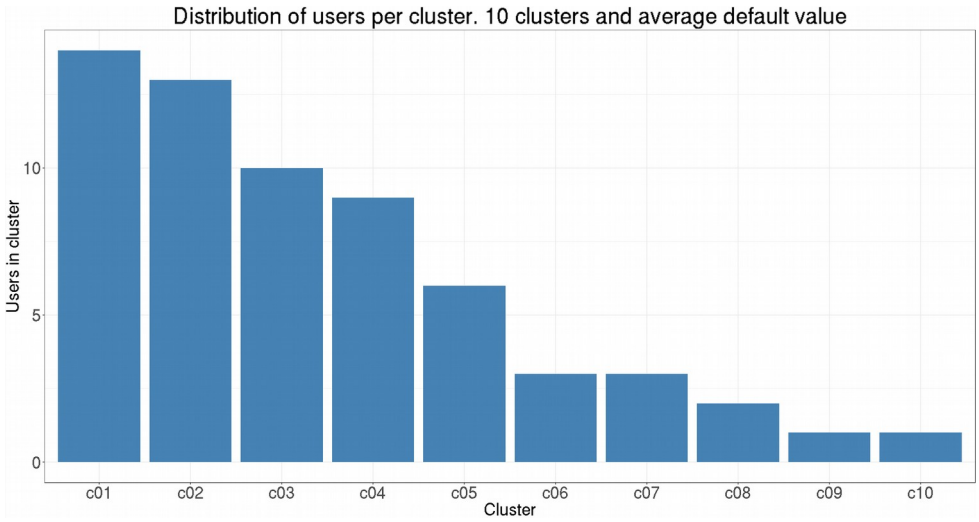

Figure 39. Distribution of users per cluster for 10 clusters and *average default value.*

The final results involve a total of 40 clusters on the *Prolific IMDb User* dataset and can be observed at the **Figure 40** and **41**. Similarly to previous results obtained applying *null default value*, the **Figure 40** shows a very biased distribution of the users among the clusters. The first clusters gather the greatest number of users, more than 3 times the content of the second clusters. From the 3rd to the 40th cluster there is a collection of small clusters form around just 1 user. When evaluating again with 40 clusters and the same dataset but using *average default value* the result varies from the previous one. This is shown in **Figure 41**, where the same count of clusters contains a different distribution of users. Despite that previous segmentations of *average default value* show more heterogeneous distributions of users, this case is more homogeneous despite not as much as **Figure 40**. The **Figure 41** shows a big first clusters with 6 users, 3 clusters of 4 users, the next 3 clusters with 3 users and the rest containing just 1 user. According to those results, can be said that in this case the difference between both methods only appears at the 8 first clusters, and the 32 resting have the same size. This is just for the content's distribution but it is not specified if the distribution of the users per cluster coincides or not.

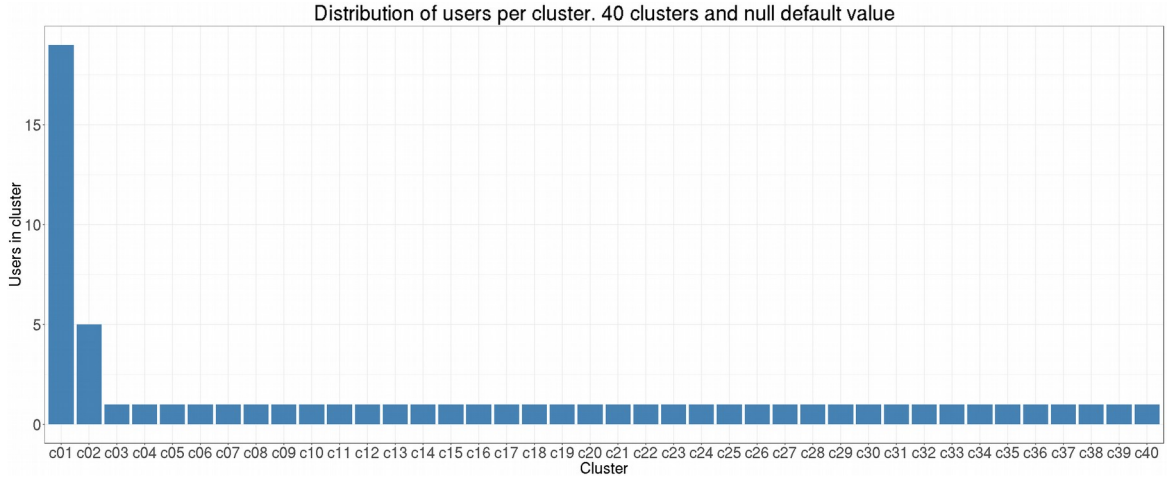

Figure 40. Distribution of users per cluster for 40 clusters and *null default value*.

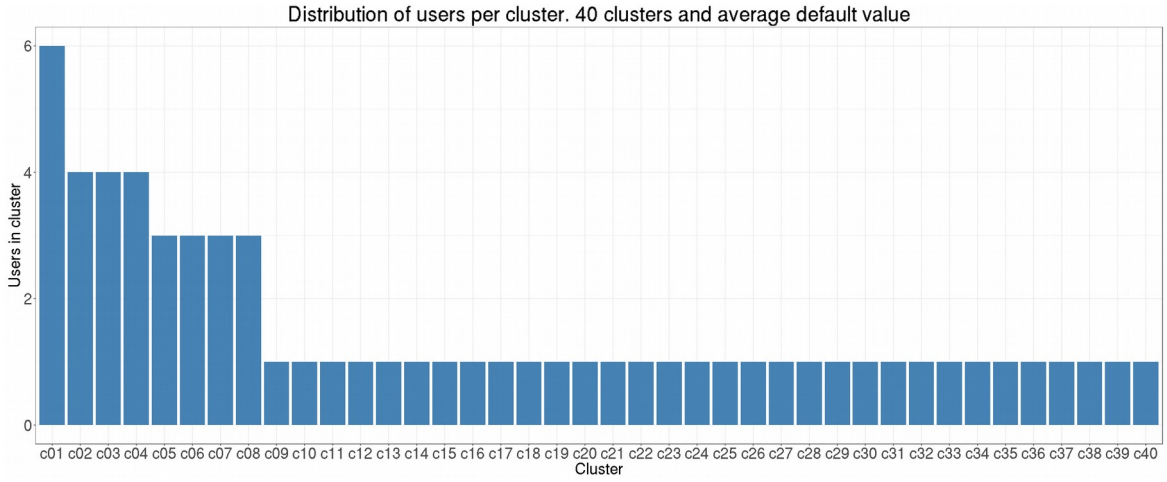

Figure 41. Distribution of users per cluster for 40 clusters and *average default value*.

5.4.13. Summary

Compiling the several conclusions extracted from all the previous results the most important insights can be extracted as a timeline. From section 5.4.1 and 5.4.2 the alternative ways of clustering show a clear winner in terms of efficiency. K-means produces the best results with a difference several times more reduced than any other method explored.

The sparsity of the *Prolific IMDb User* dataset has a strong impact augmenting the error whilst increasing the total number of clusters used for modelling. That is directly extracted from sections 5.4.3 and 5.4.4, where it is also evinced that using information from the same users is an important key for improving the accuracy of predictions.

As it was expected, the use of more data is the way that error can be reduced. Sections 5.4.5 and 5.4.6 expose this in an opposite way. As the number of clusters increments the error because the sparsity of the dataset, this is compensated more efficiently when more data about the user's scores is used for profiling this user.

Sections 5.4.7 introduces some rational about how the predictions can be improved, introducing the *dense clusters* approach and the *saturated rectifier* as simple fixes. The results from section 5.4.8 prove these conjectures are right and yield clearly better results.

The model competition is the final comparison between different combinations of approaches. Its results from section 5.4.10 expose that heuristic methods, based on probability more than pure data-driven modelling provide good results. Those are especially relevant when considering the reduced amount of data used for that.

Finally, the analysis of the clusters allows to understand that different approaches for dealing with the fitness function of the k-means produce different behaviours and distributions of the users among the clusters. Being *average default value* a more balanced way of grouping users by archetypes, knowing that this is reflected in better results at the model competition.

6. Conclusions

Considering all the different approaches, experiments and results reviewed, the conclusions obtained from them are also diverse and this section tries to analyse these. It has been introduced how recommendation systems are important nowadays and it is expected to increase their influence and popularity in a near future. It has also been going through the state of the art of different approaches for obtaining the most accurate recommendations, showing different categories like collaborative filters or content-based and how those can be combined for improving the results. Despite the actual trends go in the direction of using more data and making more sophisticated systems, the present approach wanted to explore the opposite side, proposing simpler methods when there is little information or even nothing available at all.

6.1. General conclusions

According to the objectives introduced for the present work, it has been designed, implemented and proved a mechanism able to segment users by archetypes and use them for performing reasonable good predictions. It has been specially achieved the goal of having at least one feasible alternative for the cases with no information, while in another hand, several variants of the heuristic methods are able to generate good results when there is at least very reduced information.

The way of obtaining clusters and predicting the scores for the users provides so many combinations that not all of them can be explored, but the diversity of results allows to conclude constant behaviours. The main general conclusions related to the proposed approach can be summarized as the following ones, not sorted necessarily in order of importance:

- **There is not a single approach that is better than the others**. The different results obtained and exposed at **Table 7** (*Prolific IMDb Users*) and **Table 9** (*Movielens*) evidence that when having different amounts of information available for profiling the users it is best to use the method that provides the best results from the data. The different *heuristic methods* combined with other approaches produce the best results when little information. On another hand, *linear regressions* are the best for obtaining the weights that profile users when there is lot of information statistically robust about them. Even the same models for obtaining the weights produce different results for different hyperparameter tuning. That can even be applied to the baseline models built using the scores average aggregated by user or film. Which one is the best of them in terms of reducing the prediction error depends on the data availability so, when very little information about a user is found using the film information is the best option, but when there begins to be some samples about the user's preferences this generates better predictions.

- **The dataset impacts the performance of predictions**. When comparing the three different datasets (and subsets) used for the experiments it is clear that the expected prediction's error is going to be very depending on the dataset. This is a two sides coin where on one hand the data is going to limit the results, but on another hand if gathering more data is possible, it shows this is a path that can be explored when the

accuracy of predictions needs to be improved. Independently of that, the datasets used do not allow to determine what are the limits of improvement in the results when increasing the amount and kind of data used for prediction. This is a very interesting analysis that can initiate future lines of investigation.

- **Simple methods may provide better results than sophisticated approaches**. When observing the results from different pipelines of methods used obtained for the three different explored datasets it is clear that some simple heuristic methods provide good results when using from 1 to 10 scores for profiling users. This is the case of the *inverse error* way of generating the weights relating the users with the clusters. This method is the best performing one almost systematically in this range [1,10] of scores when it is combined with the values {0.05, 0.1, 0.2} for the parameter γ. Another of those methods (even simpler than this) is the *equal weights*, specially its variation with no Y-intercept. This case is not the best performing one but it is very close in error to *inverse error* within this small range of scores. For the range of 100 scores or more per user, the linear regression is the outperforming model for obtaining the weights. This simple model provides results comparable to the obtained from other methods developed by previous researchers [108].

- **The methodology proposed may be applied to other recommendation domains.** Despite all the experiments proposed have been performed using datasets related to preferences about films, more concretely scores, the approach and the rationale supporting its conception is general enough an susceptible of being exported to other domains like books, music, TV series and others without the need of having to tweak so much the approach, it can be almost directly applied.

6.2. Clustering methods

From the experiments performed at section 5.4.1 it is easy to conclude how much convenient is the K-means algorithm for creating the clusters that most reduce the modelling error. Other methods cannot even provide a segmentation that is comparable in terms of error. Despite of that, nothing indicates that the clustering process cannot be improved. An approach similar to the proposed *cluster-weight optimization* at section 4.2.7 is very likely to improve the final results. That does not mean that the improvement is going to be enormous or that the way of obtaining the segmentation is going to be as fast and computationally light as K-means, but that would be a future line to explore. Applying other optimization methods like gradient descent to randomly initialized clusters will obtain minimums of the error that could fall close enough of the absolute minimum. Another way of obtaining good results for the clusters goes trough the use of genetical algorithms. Anyway, both methods, despite they are powerful and performant optimizing, require a lot of computation consumption that is proportional to the number of users and films reviewed. K-means seems a very reasonable way of clustering with a good balance between efficiency and computation requirements. That does not mean that other clustering methods, heuristic or more generic cannot be applied for this purpose.

6.3. Default values for unrated films

Understanding the importance of these variants is key for explaining why *average default value* provides more accurate results that *null default value*. In this case, results of the evaluation of the final error (**Table 7**, **Table 8** and **Table 9**) and how users are distributed among the clusters (**Figure 34** vs **Figure 35**, **Figure 36** vs **Figure 37**) provides a reasonable explanation: *average default value* tends to lesser concentrated distributions of users through the clusters. The more the clusters contain as many users as possible, the more combinations of users can be covered without having to increase the number of clusters. Obviously this is a rule of thumb that is not the only criteria for improving the clustering process. The way *average default value* achieves that is considering the unrated films as being more probably close to the user's own average value. That focus the similarity of the users when clustering mainly in a comparison of average values, making the difference the ones in common that can be closer or not. On another hand, *null default value* has a strong emphasis on basing the similarity process in a proportion of how many films users have seen. That allows to infer that when clustering, similarity in average scoring seems to be a more important factor for relating users than their common interests. The average user's score impacts more on the results' accuracy than what films he/she has scored.

6.4. Profiling the user

One of the main hypotheses of eigenclusters is that identifying the user with one or more archetypes represented by clusters may provide a better understanding of their behaviour and would make easier to predict the future preferences. The results evince that the presented approach provides benefits in the form of a reasonable MAE for several ways of obtaining the weights for predicting. A complete different debate is how many of those clusters are relevant and how this number is related to the amount of information available. Once again, the two main ranges of known scores for profiling (1% or less of the scores vs 10% and more) show different needs in terms of the total number of clusters. For the small range, it is easy to find the best performing approaches with 7 or more clusters in general. There are some outliers with 3 or 2 clusters, but do not represent the trend. For the big range of scores, where the linear regression is the constant winner as best method for generating the weights, it is usual to need only 1 cluster for that. It also has some exceptions with 2 or 4 clusters, but even those are small numbers. For the special case of using 80% of the scores, this situation is reverted again, having cases of needing from 3 to 5 clusters. All these different amounts of clusters and how many scores are used for profiling the user allow to infer several behaviours:

- **Profiling with less than 10% of user's scores** can easily use more clusters than scores are known from the user. In those cases there is not enough information for deterministically obtain the weights, so the heuristic models provide their best; more concretely the most simple ones. As they are not constrained under a very rigid logic and are based on probabilistic assumptions this is how the results can be interpreted. The weights in such situations can be interpreted as a **probability** of the user being significantly defined by one specific cluster-archetype. This is because the heuristic model cannot infer the real profiling in terms of the available data. Only some of the heuristic models are able to refine these probabilities increasing the ones of the clusters

that explain a bit better the known user's scores. The concrete case of the *equal weights* (equation 52), that do not use any of the available information of the user at all, shows (**Table 7**) a clear improvement vs the baseline models and it is purely probabilistic.

- **Using 10% or more of user's scores** can be interpreted less in terms of probability and more in terms of pure profiling; those weights represent how much the clusters define the **traits of the user**. In those cases the linear regression model is the best performing and in general, the more information it uses the more accurate the predictions are because the improvement of the profiling.

6.5. Comparison with previous experiments

One of the reasons why the section 2.6 (Empirical performance) has been included as part of the state of the art is for providing a reference point; helping to understand the scale of the results obtained from the eigenclusters' method. Taking into account the evaluation that Seroussi et al. [108] performed is not completely the same developed here, both are based in the same database of *Prolific IMDb Users* (IMDb) and allow to extract some conclusions. Looking at **Figure 9**, it can be seen the *Overall MAE* exposed is in terms of the scores ranges from 1 to 10. For a fare comparison those must be normalizing simply dividing them by 10, so these MAE values move from 0 to 1. Using this scale and comparing these with the results gathered at the **Table 7**, it can be seen how models like *SCSU* perform worse than *UAvg-dense-Inv*(0.2) or *UAvg-dense-Inv*(0.2) for the cases where the labelled scores used for profiling are less than 10. For the case of 200 labelled scores, *SCSU* is the best performing model also with *AIT*. At this point *UAvg-dense-LR* (**Table 7**) performs slightly better than *EQW*.

7. Future work

When comparing the obtained results from **Table 7** to the references at **Figure 9** it can be seen that the MSE obtained is in the order of what can be considered good enough for a ready-to-production model. This conclusion is even more reasonable when it is also considered how reduced is the information used for profiling the users and that external information like written documents from the users is not necessary at all for this approach.

One of the future lines of development may be use additional datasets for defining the clusters. As those represent the archetypes, real user profiles that exist beyond one specific website, it makes sense to expect that different datasets from other places could be combined to have a better determination of clusters, and also to correlate the information available per user with the ideal number of clusters.

Observing the approach from Seroussi et al. [108] it seems obvious that adding external information like documents from users would be natural way of continuing improving the predictions. The following sections summarize several ways of expanding this approach using external information.

7.1. Model ensemble

The results from the eigenclusters approach can be the foundation of a prediction to refine using the text-based analysis about sentiment or any other kind. These base results can be just refined or combined with other models for providing the final values. Merging with other available ways of predicting, it can be applied some kind of classificatory model based on the user attributes, like decision trees, that may apply a predictive way (among available ones) personalized to the user.

7.2. Feature generator

This is an alternative way of using the insights that eigenclusters provides. Every cluster can be identified with a unique feature, despite these are not necessarily uncorrelated. These features are directly applicable to ML algorithms in different ways. One use for these features is directly predicting the final score of a new film. This approach can be used also in combination with a clustering of the films in an equivalent way to how the users have been segmented. This segmentation of films would be a second set of features to be used in combination with the ones coming from the user. A second way of using them (in addition or not to the features obtained from clustering films) is for feeding as additional variables some of the actual ML classificatory algorithms.

7.3. Clustering improvement

Using different algorithms or additional information from texts is another of the ways of improving the predictions. As was shown at **Figure 18** the results from diverse segmentation techniques lead to significantly different values of the error; that keeps the door open to assume that the used *K*-means is not necessarily the absolute optimal approach but just some

kind of good approach for that. On another note, using flexible mixture model (FMM) is a reasonable candidate for improving the performance as it is able to cluster both users and films at once.

8. References

[1] Olenski, S. (2015, December 29). The Evolution Of eCommerce. *Forbes*. Retrieved from https://www.forbes.com/sites/steveolenski/2015/12/29/the-evolution-of-ecommerce/

[2] https://www.omnicoreagency.com/youtube-statistics/

[3] Amazon Annual Report 2017.

[4] Number of movies released in the United States and Canada from 2000 to 2018. *Statista*. Source: Box Office Mojo.

[5] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[6] "The Netflix Prize". Archived from the original on 2009-09-24. Retrieved 2012-07-09.

[7] Karlgren, Jussi. 1990. "An Algebra for Recommendations." Syslab Working Paper 179 (1990).

[8] Bobadilla, Jesús; Ortega, Fernando; Hernando, Antonio; Bernal, Jesús (February 2012). "A collaborative filtering approach to mitigate the new user cold start problem". *Knowledge-Based Systems*.

[9] Hou, Lei; Pan, Xue; Liu, Kecheng (7 March 2018). "Balancing the popularity bias of object similarities for personalised recommendation". *The European Physical Journal* B. 91 (3).

[10] http://license.umn.edu/technologies/z05173_movielens-database

[11] John S. Breese, David Heckerman, and Carl Kadie, Empirical Analysis of Predictive Algorithms for Collaborative Filtering, 1998 Archived 19 October 2013 at the Wayback Machine.

[12] Terveen, Loren; Hill, Will (2001). "Beyond Recommender Systems: Helping People Help Each Other". Addison-Wesley. p. 6. Retrieved 16 January 2012.

[13] Koza, John R.; Bennett, Forrest H.; Andre, David; Keane, Martin A. (1996). *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*. Artificial Intelligence in Design '96. Springer, Dordrecht. pp. 151–170.

[14] "Collaborative recommendations using item-to-item similarity mappings"

[15] Xiaoyuan Su, Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence archive, 2009.

[16] Pearl, Judea (2000). Causality: *Models, Reasoning, and Inference*. Cambridge University Press. ISBN 978-0-521-77362-1.

[17] Dawid, A. P. (1979). "Conditional Independence in Statistical Theory". *Journal of the Royal Statistical Society, Series B*. **41** (1): 1–31.

[18] Gut, Allan (2013). *Probability: A Graduate Course* (Second ed.). New York, NY: Springer. ISBN 978-1-4614-4707-8.

[19] C.D. Manning, P. Raghavan and M. Schütze (2008). *Introduction to Information Retrieval*. Cambridge University Press, p. 260.

[20] K. Miyahara and M. J. Pazzani, “Collaborative filtering with the simple Bayesian classifier,” in *Proceedings of the 6th Pacific Rim International Conference on Artificial Intelligence*, pp. 679–689, 2000.

[21] X. Su and T. M. Khoshgoftaar, “Collaborative filtering for multi-class data using belief nets algorithms,” in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI ’06)*, pp. 497–504, 2006.

[22] R. Greinemr, X. Su, B. Shen, and W. Zhou, “Structural extension to logistic regression: discriminative parameter learning of belief net classifiers,” *Machine Learning*, vol. 59, no. 3, pp. 297–322, 2005.

[23] B. Shen, X. Su, R. Greiner, P. Musilek, and C. Cheng, “Discriminative parameter learning of general Bayesian network classifiers,” in *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pp. 296–305, Sacramento, Calif, USA, November 2003.

[24] J. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI ’98)*, 1998.

[25] MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. **1**. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.

[26] M. O’Connor and J. Herlocker, “Clustering items for collaborative filtering,” in *Proceedings of the ACM SIGIR Workshop on Recommender Systems (SIGIR ’99)*, 1999.

[27] T. Hofmann and J. Puzicha, “Latent class models for collaborative filtering,” in *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI ’99)*, pp. 688–693, 1999.

[28] J. Canny, “Collaborative filtering with privacy via factor analysis,” in *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 238–245, Tampere, Finland, August 2002.

[29] S. Vucetic and Z. Obradovic, “Collaborative filtering using a regression-based approach,” *Knowledge and Information Systems*, vol. 7, no. 1, pp. 1–22, 2005.

[30] R. E. Bellman, *Dynamic Programming*, Princeton University Press, Princeton, NJ, USA, 1962.

[31] "Definition of *system*". *Merriam-Webster*. Springfield, MA, USA.Retrieved 2019-01-13.

[32] Gagniuc, Paul A. (2017). *Markov Chains: From Theory to Implementation and Experimentation*. USA, NJ: John Wiley & Sons. pp. 1–235. ISBN 978-1-119-38755-8.

[33] R. A. Howard, *Dynamic Programming and Markov Processes*, MIT Press, Cambridge, Mass, USA, 1960.

[34] G. Shani, D. Heckerman, and R. I. Brafman, “An MDP-based recommender system,” *Journal of Machine Learning Research*, vol. 6, pp. 1265–1295, 2005.

[35] T. Hofmann, “Unsupervised learning by probabilistic latent semantic analysis,” *Machine Learning*, vol. 42, no. 1-2, pp. 177–196, 2001.

[36] W. W. Cohen, R. E. Schapire, and Y. Singer, “Learning to order things,” *Journal of Artificial Intelligence Research*, vol. 10, pp. 243–270, 1999.

[37] EachMovie dataset, http://www.grouplens.org/node/76.

[38] B. Marlin, “Modeling user rating profiles for collaborative filtering,” in *Neural Information Processing Systems*, 2003.

[39] B. Marlin, *Collaborative filtering, a machine learning perspective*, M.S. thesis, Department of Computer Science, University of Toronto, 2004.

[40] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 993–1022, 2003.

[41] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, “Analysis of recommendation algorithms for E-commerce,” in *Proceedings of the ACM E-Commerce*, pp. 158–167, Minneapolis, Minn, USA, 2000.

[42] X. Fu, J. Budzik, and K. J. Hammond, “Mining navigation history for recommendation,” in *Proceedings of the International Conference on Intelligent User Interfaces (IUI ’00)*, pp. 106–112, 2000.

[43] D. Y. Pavlov and D. M. Pennock, “A maximum entropy approach to collaborative filtering in dynamic, sparse, high dimensional domains,” in *Advances in Neural Information Processing Systems*, pp. 1441–1448, MIT Press, Cambridge, Mass, USA, 2002.

[44] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, “Dependency networks for inference, collaborative filtering, and data visualization,” *Journal of Machine Learning Research*, vol. 1, no. 1, pp. 49–75, 2001.

[45] D. Nikovski and V. Kulev, “Induction of compact decision trees for personalized recommendation,” in *Proceedings of the ACM Symposium on Applied Computing*, vol. 1, pp. 575–581, Dijon, France, April 2006.

[46] C. C. Aggarwal, J. L. Wolf, K. Wu, and P. S. Yu, “Horting hatches an egg: a new graph-theoretic approach to collaborative filtering,” in *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’99)*, pp. 201–212, 1999.

[47] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society. Series B*, vol. 21, no. 3, pp. 611–622, 1999.

[48] G. Takács, I. Pilászy, B. Németh, and D. Tikk, “Investigation of various matrix factorization methods for large recommender systems,” in *Proceedings of the IEEE International Conference on Data Mining Workshops (ICDM ’08)*, pp. 553–562, Pisa, Italy, December 2008.

[49] B. Marlin and R. S. Zemel, “The multiple multiplicative factor model for collaborative filtering,” in *Proceedings of the 21st International Conference on Machine Learning (ICML ’04)*, pp. 576–583, Banff, Canada, July 2004.

[50] Peter Brusilovsky (2007). *The Adaptive Web*. p. 325. ISBN 978-3-540-72078-2.

[51] Robertson, S. (2004). "Understanding inverse document frequency: On theoretical arguments for IDF". *Journal of Documentation*. 60 (5): 503–520. doi:10.1108/00220410410560582.

[52] Manouselis, N., Costopoulou, C. Experimental Analysis of Design Choices in Multi-Attribute Utility Collaborative Filtering. *International Journal of Pattern Recognition and Artificial Intelligence*, 21(2):311–332, 2007.

[53] Perny, P., Zucker, J.D. Collaborative Filtering Methods based on Fuzzy Preference Relations. In *Proc. of EUROFUSE-SIC 99*, pages 279–285, 1999.

[54] Adomavicius, G., Tuzhilin, A. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[55] Siskos, Y., Grigoroudis, E., Matsatsinis, N.F. *UTA Methods*. Springer, 2005.

[56] Stolze, M., Rjaibi, W. Towards Scalable Scoring for Preference-based Item Recommendation. *IEEE Data Engineering Bulletin*, 24(3):42–49, 2001.

[57] Stolze, M., Stroebel, M. Dealing with Learning in eCommerce Product Navigation and Decision Support: The Teaching Salesman Problem. In *Proc. of the 2nd Interdisciplinary World Congress on Mass Customization and Personalization*. Munich, Germany, 2003.

[58] Cantador, I., Fernandez, M., Castells, P. A Collaborative Recommendation Framework for Ontology Evaluation and Reuse. In *Proc. of the International ECAI Workshop on Recommender Systems*. Riva del Garda, Italy, 2006.

[59] Adomavicius, G., Kwon, Y. New Recommendation Techniques for Multi-Criteria Rating Systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.

[60] Pazzani, M., Billsus, D. Learning and Revising User Profiles: The Identification of InterestingWeb Sites. *Machine Learning*, 27(3):313–331, 1997.

[61] Masthoff, J. Modeling the Multiple People That Are Me. In *Proc. of the International Conference on User Modelling (UM2003)*, pages 258–262. Johnstown, USA, 2003.

[62] Perny, P., Zucker, J.D. Preference-based Search and Machine Learning for Collaborative Filtering: the Film-Conseil Movie Recommender System. *Information, Interaction, Intelligence*, 1(1):1–40, 2001.

[63] Adomavicius, G., Tuzhilin, A. Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[64] Falle, W., Stoefler, D., Russ, C., Zanker, M., Felfernig, A. Using knowledge-based advisor technology for improved customer satisfaction in the shoe industry. In *Proc. of International Conference on Economic, Technical and Organisational aspects of Product Configuration Systems*, Technical University of Denmark, Kopenhagen, 2004.

[65] WordNet - A Lexical Database for English

[66] Recommendation of e-Learning Quality Approaches. In *Proc. of the 16th World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA)*. Lugano, Switzerland, 2004.

[67] Manouselis, N., Sampson, D. A Multi-criteria Model to Support Automatic Recommendation of e-Learning Quality Approaches. In *Proc. of the 16th World Conference on Educational Multimedia, Hypermedia and Telecommunications (EDMEDIA)*. Lugano, Switzerland, 2004.

[68] Tang T.Y., McCalla, G. The Pedagogical Value of Papers: a Collaborative-Filtering based Paper Recommender. *Journal of Digital Information*, 10(2), 2009.

[69] Schmitt, C., Dengler, D., Bauer, M. The MAUT-Machine: An Adaptive Recommender System. In *Proc. of the Workshop on Adaptivitat und Benutzermodellierung in Interaktiven Softwaresystemen(ABIS)*. Hannover, Germany, 2002.

[70] Mukherjee, R., Dutta, P.S., Jonsdottir, G., Sen, S MOVIES2GO: An Online Voting Based Movie Recommender System. In *Proc. of the 5th International Conference on Autonomous Agents*, pages 114–115. Montreal, Canada, 2001.

[71] Ariely, D., Lynch, J.G.Jr., Aparicio, M. Learning by Collaborative and Individual-based recommendation Agents. *Journal of Consumer Psychology*, 14(1&2):81–95, 2004.

[72] Cheetham, W. Global Grade Selector: A Recommender System for Supporting the Sale of Plastic Resin. *Technical Information Series*, GE Global Research, TR 2003GRC261, 2003.

[73] Kohavi, Ron (1995). "A study of cross-validation and bootstrap for accuracy estimation and model selection". *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann. **2** (12): 1137–1143.

[74] Manouselis, N., Costopoulou, C. Analysis and Classification of Multi-Criteria Recommender Systems. *World Wide Web: Internet and Web Information Systems*, 10(4):415–441, 2007.

[75] Ricci, F., Venturini, A., Cavada, D., Mirzadeh, N., Blaas, D., Nones, M. Product Recommendation with Interactive Query Management and Twofold Similarity. *Proc. of the 5th Internatinal Conference on Case-Based Reasoning*, Trondheim, Norway, 2003.

[76] Reilly, J., McCarthy, K., McGinty, L., Smyth, B. Incremental Critiquing. *Knowledge-Based Systems*, 18(4-5):143–151, 2005.

[77] R. Greinemr, X. Su, B. Shen, and W. Zhou, “Structural extension to logistic regression: discriminative parameter learning of belief net classifiers,” *Machine Learning*, vol. 59, no. 3, pp. 297–322, 2005.

[78] X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu, “Hybrid collaborative filtering algorithms using a mixture of experts,” in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI ’07)*, pp. 645–649, Silicon Valley, Calif, USA, November 2007.

[79] M. J. Pazzani, “A framework for collaborative, content-based and demographic filtering,” *Artificial Intelligence Review*, vol. 13, no. 5-6, pp. 393–408, 1999.

[80] X. Su, R. Greiner, T. M. Khoshgoftaar, and X. Zhu, “Hybrid collaborative filtering algorithms using a mixture of experts,” in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI ’07)*, pp. 645–649, Silicon Valley, Calif, USA, November 2007.

[81] M. Balabanović, “Exploring versus exploiting when learning user models for text recommendation,” *User Modelling and User-Adapted Interaction*, vol. 8, no. 1-2, pp. 71–102, 1998.

[82] R. Burke, “Hybrid recommender systems: survey and experiments,” *User Modelling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[83] B. Smyth and P. Cotter, “A personalized TV listings service for the digital TV age,” in *Proceedings of the 19th International Conference on Knowledge-Based Systems and Applied Artificial Intelligence (ES ’00)*, vol. 13, pp. 53–59, Cambridge, UK, December 2000.

[84] CiteSeer ResearchIndex, digital library of computer science research papers, http://citeseer.ist.psu.edu.

[85] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, “Probabilistic memory-based collaborative filtering,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004.

[86] Ye, Mao; Yin, Peifeng; Lee, Wang-Chien; Lee, Dik-Lun (2011-01-01). *Exploiting Geographical Influence for Collaborative Point-of-interest Recommendation. Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '11. New York, NY, USA: ACM. pp. 325–334. ISBN 9781450307574.

[87] Bouneffouf, Djallel (2013), DRARS, *A Dynamic Risk-Aware Recommender System* (Ph.D.), Institut National des Télécommunications.

[88] Gittins, J. C. (1989), *Multi-armed bandit allocation indices*, Wiley-Interscience Series in Systems and Optimization., Chichester: John Wiley & Sons, Ltd., ISBN 978-0-471-92059-5.

[89] Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, Michigan (2005) 115–124.

[90] L. Sirovich; M. Kirby (1987). "Low-dimensional procedure for the characterization of human faces". *Journal of the Optical Society of America A*. **4** (3): 519–524.

[91] M. Turk; A. Pentland (1991). "Face recognition using eigenfaces". *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. pp. 586–591.

[92] Pearson, K. (1901). "On Lines and Planes of Closest Fit to Systems of Points in Space". *Philosophical Magazine*. 2 (11): 559–572.

[93] Griffiths, David (2005). *Introduction to Quantum Mechanics* (2nd ed.). pp. 183–4.

[94] Schrödinger, E. (1926). "An Undulatory Theory of the Mechanics of Atoms and Molecules". *Physical Review*. **28** (6): 1049–1070.

[95] Leijen, Daan (December 3, 2001). "Division and Modulus for Computer Scientists". Retrieved 2014-12-25.

[96] Simovici, Dan A. & Djeraba, Chabane (2008). "Partially Ordered Sets". *Mathematical Tools for Data Mining: Set Theory, Partial Orders, Combinatorics*. Springer. ISBN 9781848002012.

[97] MacQueen, J. B. (1967). *Some Methods for classification and Analysis of Multivariate Observations*. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability 1. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201.

[98] Rencher, Alvin C.; Christensen, William F. (2012), "Chapter 10, Multivariate regression – Section 10.1, Introduction", *Methods of Multivariate Analysis*.

[99] *Stigler, Stephen M. (1981). "Gauss and the Invention of Least Squares". Ann. Stat. **9** (3): 465–474.*

[100] *Broyden, C. G. (1970), "The convergence of a class of double-rank minimization algorithms", Journal of the Institute of Mathematics and Its Applications, **6**: 76–90.*

[101] Schey, H. M. (1997). *Div, Grad, Curl, and All That: An Informal Text on Vector Calculus*. New York: Norton. ISBN 0-393-96997-5.

[102] Odersky, M.; Rompf, T. (2014). "Unifying functional and object-oriented programming with Scala". *Communications of the ACM*. **57** (4): 76.

[103] "The Java Virtual Machine Specification : Java SE 7 Edition" (PDF). Docs.oracle.com. Retrieved 2015-06-26.

[104] Darcy, Joe (June 8, 2009). "OpenJDK and the new plugin". Retrieved September 5, 2009.

[105] Lindholm, Tim; Yellin, Frank; Bracha, Gilad; Buckley, Alex (2015-02-13). *The Java Virtual Machine Specification* (Java SE 8 ed.).

[106] Matthew Hertz; Emery D. Berger (2005). "Quantifying the Performance of Garbage Collection vs. Explicit Memory Management". OOPSLA 2005. Retrieved 2015-03-15.

[107] Aycock, J. (June 2003). "A brief history of just-in-time". *ACM Computing Surveys*. **35** (2): 97–113.

[108] Seroussi Y., Zukerman I., Bohnert F. (2010) Collaborative Inference of Sentiments from Texts. In: *De Bra P., Kobsa A., Chin D. (eds) User Modeling, Adaptation, and Personalization.* UMAP 2010. Lecture Notes in Computer Science, vol 6075. Springer, Berlin, Heidelberg.

[109] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872.

[110] Willmott, Cort J.; Matsuura, Kenji (December 19, 2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". *Climate Research*. **30**: 79–82.

[111] Cook, R. Dennis; Weisberg, Sanford (1982). *Residuals and Influence in Regression* (Repr. ed.). New York: Chapman and Hall. ISBN 041224280X. Retrieved 23 February 2013.

[112] Wilkinson, Leland; Friendly, Michael (May 2009). "The History of the Cluster Heat Map". *The American Statistician*. **63** (2): 179-184.