Movie Recommendation based on User Interests

André Carvalho
Instituto Superior Técnico
1049-001 Lisboa
Email: andre.silva.carvalho@ist.utl.pt

Supervisor: Pável Calado
Instituto Superior Técnico
1049-001 Lisboa
Email: pavel.calado@ist.utl.pt

***Abstract*—In this document a movie recommendation system is proposed, where users specific interests are taken into account to determine recommendations. The interests are identified by a Self-organizing map (SOM), for each user, instead of one SOM clustering users into groups. The system combines movie ratings and descriptions, leading to a hybrid filtering recommendation system, which uses both collaborative and content-based filtering. Two similarity metrics were proposed to achieve the hybrid filtering algorithm. In order to evaluate the proposed system, we provide the comparative results of a set of experiments based on the MovieLens-100k dataset (ML), which include the evaluation of the Entity identifier process and accuracy evaluation of all Recommendation systems variations we have tested. Our system provided good accuracy results and is a viable solution to be used as Recommendation system.**

Keywords: Recommendation systems, Self-organizing maps, Collaborative filtering, Content-Based Filtering, Hybrid filtering.

I. Introduction

Technology evolution led to a big and diverse amount of information on the internet. This evolution caused the user to be overloaded with information [34]. Recommendation systems allow us to cope with this overload, by cataloguing a vast list of items, that later can be recommended. This recommendation is determined by a vast number of techniques, highlighted by the scientific community [5, 34]. Nowadays, recommendation systems can be found in a vast number of services, such as movies, music, news, products and services recommendation, among others [1].

Despite the research on this topic, few references make use of Self-organizing Maps [13, 22, 24, 34]. In addition, it was never applied to identify and present the specific interests of a single user, nor exploited the items descriptions, using Self-organizing maps, to enhance the recommendation system performance [13].

This work has two objectives: (1) identify users specific interests and use them to enhance the RS and (2) evaluate if the Self-organizing maps, using content based information, are a viable solution to represent the users interests. This two defined goals contribute to provide a new solution, creating models to represent users interests and also to develop new similarity metrics for hybrid systems.

The identified user specific interests were named entities and split the user in distinct organisms. Those entities can then be used by the recommendation system, since they were formed using the movies descriptions and they might provide new information to the RS.

This work is composed by eight chapters which are structured as follows: Chapter II presenting the basic concepts regarding Self-organizing maps, and the recommendation systems; Chapter III contains related work about recommendation systems and Self organizing maps; Chapter IV describes the architecture of the proposed system, including the entity identifier and the actual recommendation system; Chapter V shows how the entity identifier functions; Chapter VI clarifies how our recommendation system was developed ; Chapter VII present the evaluation of the developed system; finally, Chapter VIII contains the conclusions.

II. Basic concepts

A. Self-organizing maps

First presented by Tuevo Kohonen in 1982 [20], the Self-Organizing Map (SOM) allowed to represent signals with large dimensionality in a lower-dimensional space, so that the formed topology was observable by the human eye as a map [20]. With SOMs, patterns in the data are identified using a unsupervised neuronal network, which is responsible for representing the high dimensionality signals using a map, generally with one or two dimensions [34].

In Fig. 1, the SOM algorithm is illustrated. The *input nodes* are vectors with high dimensionality, while the *output nodes* form a map, in this case with a 5 by 5 layout.

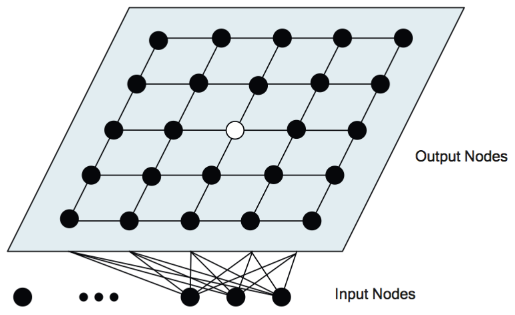


Figure 1: SOM input nodes and output nodes.

On the first stage, for each learning sample, the distance is calculated against all the output nodes. Then, the nearest node is selected as the Best Matching Unit (BMU) [31, 34]. The second stage of the algorithm is responsible for the update of all the input nodes position. When the best matching unit for each sample is identified, the SOM reprograms the nodes positions taking into consideration the BMU, using Eq. 1.

In Eq. 1, m_p represents each node, h_{cp} is a neighbourhood function that represents distances between a node and the BMU. h_{cp} can be for example a Gaussian function. $\alpha(t)$ is the learning rate of the output node, which decreases as function of t so that the algorithm can learn faster in the beginning but slower over time. The distance between the sample x_n and each m_p is also taken into consideration and the variable t refers to the number of iterations occurred.

$$m_p(t+1) = m_p(t) + \alpha(t)h_{cp}(t)[x_n - m_p(t)] \tag{1}$$

After completing T iterations, the competition training stops. The SOM training is then complete, the output nodes can be mapped on a matrix and the distances between them can be drawn, forming the Unified-Distance Matrix (U-Matrix) [36].

The U-Matrix presents the distances between contiguous nodes. This map is based on the distance between the contiguous output nodes, using their m_p vector [36]. An example of the U-Matrix visualization is presented in Fig. 2, where the values nearest to one correspond to similar nodes and zero to more distant nodes.

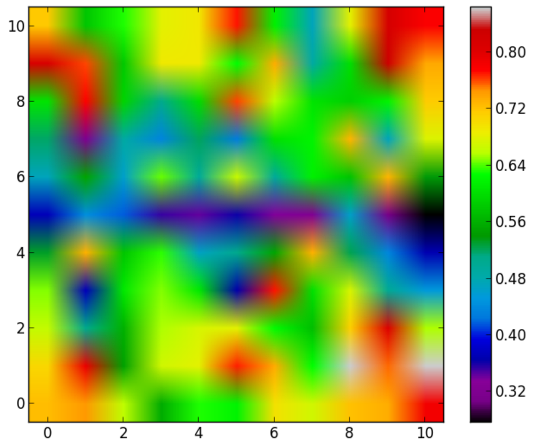


Figure 2: Example of a U-Matrix.

B. Recommendation systems

RS are a networking tool that offer dynamics and collaborative communication, interaction and knowledge [4]. On the last twenty years, the demand of RS has only increased, as it allows users to deal with big amounts of data, providing them a selection of personalized recommendations, services and contents. As a result, various techniques have been developed and studied, both from the scientific community and from companies, since it allows them to increase their profit [1].

1) Collaborative filtering: perform the comparison of users ratings, resulting in the identification of the most similar ones [3]. Most of the research has been done using this type of filtering as it is simple and provides good results. The two main focuses of research are how to define the *similarity metric* and also how to *predict a rating* to an item not rated by a user.

We present two vastly used Collaborative Filtering (CF) algorithms: K-Nearest Neighbours (KNN) and Singular Value Decomposition (SVD).

When using KNN to develop one CF algorithm, a similarity metric is responsible for comparing users or items and attribute them a degree of similarity. A reference for the most common ones can be found in [3]. We will explain in more detail the Pearson correlation, as it is vastly used in CF and will be used in our work.

Pearson Correlation was proposed by Shardanand and Maes [30], in 1995 and it has been widely used in RS, as it provides good results [3]. It is obtained by applying Eq. 2, where I represents all items rated by both users u and j. The average of all ratings from user u is represented by $\bar{r}_u$ and the average of all ratings from user j is represented by $\bar{r}_j$.

$$sim(u,j) = \frac{\sum_{i\in I}(r_{u,i} - \bar{r}_u) \times (r_{j,i} - \bar{r}_j)}{\sqrt{\sum_{i\in I}(r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i\in I}(r_{j,i} - \bar{r}_j)^2}} \tag{2}$$

The result of Eq. 2 will be in the interval $[-1, 1]$, where a similarity equal to -1 corresponds to a inverse correlation and a similarity equal to $+1$ to a positive correlation [13]. Values near zero show that no linear correlation exists between the two users.

The prediction metric is responsible for taking advantage of the similarities between users (or items) and provide a prediction for the item. There are numerous ways to make predictions, and the most common ones are described in [3]. In our solution we used a variation of the most common Equation to make predictions.

We adopted the adjusted weighted aggregation (deviation-from-mean), represented by Eq. 3, since it is an improved equation. This equation allows us to consider the deviation from the mean of each user rating $(r_{j,i} - \bar{r}_j)$, instead of only considering the rating alone $r_{j,i}$, as the most common Equation does [3].

In Eq. 3, $p_{u,i}$ denotes the prediction value determined, $\mu_{u,i}$ is defined in Eq. 4 and corresponds to a normalizing factor [13].

$$p_{u,i} = \bar{r}_u + \mu_{u,i} \sum_{n\in G_{u,i}} sim(u,j) \times (r_{j,i} - \bar{r}_j) \tag{3}$$

$$\mu_{u,i} = \frac{1}{\sum_{n\in G_{u,i}} sim(u,j)} \iff G_{u,i} \neq \varnothing \tag{4}$$

Another CF algorithm is Singular Value Decomposition (SVD), which was firstly used on a case study [29]. It performs a decomposition of the rating matrix. As a result, it enables a reduction of dimensionality that improves the overall Recommendation System (RS) accuracy and efficiency.

The method works as follows, the ratings matrix (X) is decomposed in three matrices: U, S and V^T using SVD [16]. There is a lot of research in how to perform this decomposition since this is the most computational demanding task in the recommendation process [11, 21].

SVD allows to reduce the dimensionality of the problem by cropping U, S and V^T, on the k dimension. With Eq. 5 it is possible to predict a rating ($p_{u,i}$) for any pair user (u) and movie (i). The value $\bar{r}_u$ represents the average rating of u and k the number of features selected from the decomposed rating matrix.

$$p_{u,i} = \bar{r}_u + U_k(u) \times \sqrt{S_k} \times V_k^T(i) \quad (5)$$

2) Content based filtering: Sometimes, by using the items meta-data, recommendations can be enhanced. For that reason, Content-Based Filtering (CBF) was introduced as a filtering technique. It traditionally works by determining items related to those that users previously manifested their interest in [37].

On the cinematographic domain, for example, movie descriptions such as synopsis, genre, actors, directors, keywords and many others can be used to obtain the similarities between movies [37].

Most of the times, this descriptions have to be pre-processed and converted into something comparable between them. When dealing with text, several operations such as the *removal of stopwords*, *stemming* and the formation of the *bag of words* representation, are generally done as a text pre-processing algorithm [10].

If a RS only uses CBF, it will most likely suffer from the over-specialization problem. Thus, generally CBF is combined with other types of filtering techniques, such as Collaborative Filtering (CF) [2, 6].

3) Hybrid filtering: The previously discussed filtering techniques can be combined to produce a RS that uses hybrid filtering [35]. Combining them is expected to improve the overall quality of the RS and the provided recommendations [6, 7].

Although any filtering techniques are possible to be combined, we will only focus on combining CF an CBF, since we will use on our system. There are several ways to combine two filtering techniques [1], we adopted a case where the characteristics of the CBF were producing information that would afterwards be used by the CF. An example is that it is possible to incorporate personal information in CF to alleviate the problem of introducing a new user to the RS.

III. Related work

Research on RS has been wide and a vast number of discoveries have been done to define new techniques and enhance such systems [2, 25].

A. RS: Current research

Today, there are many studies regarding RS. Yet, there is still room for the development of new techniques [25]. Current research is mostly about: (1) hybrid filtering, (2) development of new similarity metrics and the evaluation of their results, (3) the inclusion of security and privacy methods into the RS and (4) the extraction of user consumption pattens, allowing the RS to better identify each individuals interests [2].

1) Hybrid Filtering: A large number of research focus on hybrid filtering [6, 7, 17, 35]. Most of them combine CF with another filtering technique, thus exploiting the advantages of each to enhance the RS results [2, 17].

Another example of RS that used hybrid filtering is [26]. This technique combined CBF and CF by forming a pseudo rating matrix, which would include the predictions on non rated movies by each user, computed using CBF. This pseudo rating matrix would afterwards be used as a regular rating matrix when performing CF, but the information of predictions resulting from CBF would help the CF algorithm to compute similarities.

One way to combine two different filtering algorithms is to take advantage of the combination of the similarities determined, by each metric. With the solution proposed in [18], it is possible to accomplish that goal, by weighting the relevance of each similarity with a coefficient α. Their goal was to combine two similarity metrics, one using the traditional obtained between users $sim_{user}(u,j)$ and the other with the similarities based on formed groups of users $sim_{group}(u,j)$. Those groups were created according to the descriptions of the items they rated. Based on these, a clustering algorithm was applied and the groups formed. Afterwards, a group rating matrix was created, forming a smaller rating matrix with the users within that group [18]. Equation 6 weighs the determined similarity with the user rating matrix $sim_{user}(u,j)$ and the similarity obtained by using the group rating matrix $sim_{group}(u,j)$, resorting to the term α.

$$sim(u,j) = (1-\alpha) \times sim_{user}(u,j) + \alpha \times sim_{group}(u,j) \quad (6)$$

In our work we adapted the Eq. 6, to aggregate both user and entity similarities.

2) Development of new similarity metrics: A RS searches for items to recommend. It must compare items and users to achieve this. The *Starcoll* method, in [12], helps the developed RS to alleviate the problem of cold-start. The problem of cold-start occurs when a RS either does not have enough information about the user or item. The developed method is able to compare all users, even those that do not share rated items. To make this possible, a user model was created, which contained a representation of the users features, by using all the items features as user interests and comparing them to obtain the similarities [12].

Another technique developed a similarity metric that outperforms the Pearson correlation [3]. This new metric was able to enhance the CF techniques, using the complement of the Mean Square Difference (MSD), shown in Eq. 7, and combining it with the Jaccard similarity, shown in Eq. 8, resulting in the similarity Eq. 9. Note that r_u and r_j represent the users rating set, containing only the items shared by both. The cardinality of ratings shared by both users is represented by $\#r_{u,j}$.

$$MSD(u,j) = \frac{|r_u - r_j|}{\#r_{u,j}} \quad (7)$$

$$jaccard(u,j) = \frac{|r_u \cap r_j|}{|r_u \cup r_j|} \quad (8)$$

$$new\ similarity(u,j) = (1 - MSD(r_u, r_j)) \times jaccard(r_u, r_j) \quad (9)$$

One of the main advantages of this metric is that, with a lower number of neighbours, it still obtains better results than the Pearson correlation, which is one of the most used similarity metrics [3].

B. RS using Self-organizing Maps

SOM have been used in several ways to help RS. The first example is of an hybrid system, able to combine demographic information about the user [22]. Demographic information allows to better compute similarities between the users. The SOM clusters users by taking into account only their demographic information. Then, the identified clusters can be used to determine the neighbourhood of the user for CF. Is worth noticing that their work improved the system both in computational cost and scalability [22].

The algorithm of Probabilistic SOM [24] was a result of the combination of Principal component analysis, which is commonly used when dealing with high dimensionality problems, followed by a SOM. The probabilistic stands for the fact that this solution predicts the uncertainty of the principal components that will be used on the SOM. Although the results presented were not good when Probabilistic SOM was used in isolation, this solution was seen in works developed for Netflix Prize [27], because it can be ensemble with other models, and combined have an improved global performance.

Finally, S. Gabrielsson compared the performance neighbourhoods using either KNN or SOM [13]. However, they did not gave must emphasis to the use of CBF to try to enhance the system. Their CBF approach used keywords as item descriptions. One difference that must be highlighted is that this RS did not produce a SOM specific for each user but a generic SOM shared by all users, using all the movies information. Ours will only use the movies rated by the respective user, therefore we will have a SOM model per user in our solution.

C. Movie recommendation systems

A large number of RS were tested with movie-based datasets [2, 14, 15, 33]. Yet, not all of them are specifically for movie recommendation, being easily applicable in other domains. In [9] an improved movie RS is presented, which exploits the movies genres. Their solution represented users and movies based on their genres. Afterwards, they tried to correlate users and movies in two different ways: (1) according to their number of equal genres; (2) according to the decade when the movies were released. Their findings shown that more precise recommendations were obtained using (2) a decade-based genre correlation.

Also, the *More* system [23] is a RS developed with the movie domain in mind, by exploiting the movies descriptions and the rating matrix. Their goal was to develop an hybrid system that combines the CF and CBF. Yet, they did not include the movies synopsis as description.

IV. System architecture

In most RS, it is not clear why are the users getting their recommendations [33]. Our primary goal is to identify the users most specific interests, and take advantage of this knowledge when making rating predictions. To accomplish this goal, we propose to split each user into several entities, each representing different tastes. This difference in tastes is based on movie descriptions and not the ratings that users gave to a movie. Once identified, the entities will play the role of users with a specific interest on a given topic (e.g. action movies with superheroes or movies about detectives or unsolved investigations).

Two separate components are presented: (1) Entity identifier - which is able to split a user into entities, based on the description of watched movies; (2) Recommendation system - which performs predictions for movies not yet rated by the users and recommends only those with better predictions. Both are independent but can be combined to produce an hybrid filtering solution.

The Entity model will store information regarding the formed entities. This information is produced by our Entity identifier, based on the information about movies, such as: title and release date and will be latter used by the Recommendation system to improve predictions.

A. Overview

Each user will have their specific entities discovered using a SOM. The SOM was used to discover the entities since it allows clustering the movies, according to their description and also a map visualization of the formed entities.

The proposed system, first identifies the user entities. As a result, two rating matrices are available. The first, is the original rating matrix, containing each user and its ratings on movies. The second is an entity rating matrix, containing each entity and its ratings on movies.

To exemplify how the entity matrix creation is done, Fig. 3 illustrates both user rating matrix and the resulting entity rating matrix. User 1 is split into two entities and User 2 into three entities according to their respective Entity model.

The entities are determined by a SOM, which clusters the movies rated by the user, according only to their descriptions.

V. Identifying the user and his entities

As the name implies, the Entity Identifier component, determines the entities of each user. Basically, the system takes the movies and their descriptions into consideration and, resorting to a SOM, discovers the groups of movies that form the entities. Formally, it can be seen as a CBF component of an hybrid Recommendation System (RS).

There are three information sources that can be exploited to describe movies: the synopsis, the keywords or the rating matrix. Since, for now, we only are interested in CBF, only the first two will be considered. To better understand our approach Fig. 4 illustrates how the entities are obtained.

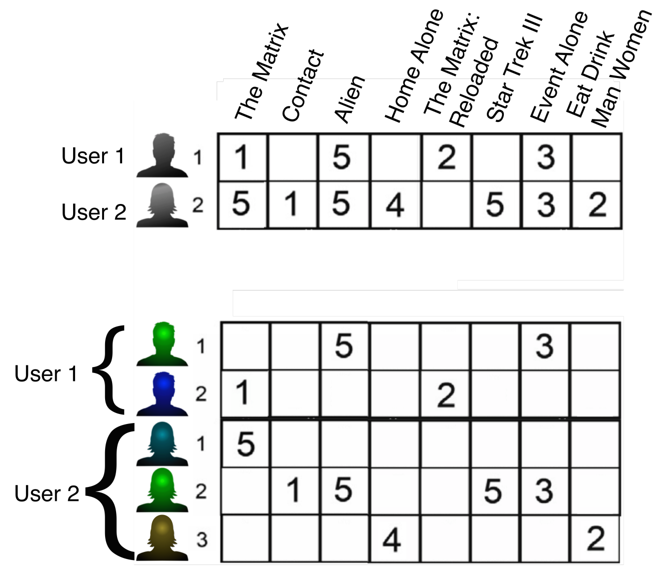


Figure 3: Example of entity rating matrix formation.

The two components developed were: (1) Movie description Getter, which downloads the synopsis and keywords of each movie; and (2) Entity identifier that splits the user into a set of entities, using only the watched movies and their descriptions.

The Movie Description Getter component interacts with the SOM Entity Identifier by providing a matrix with movies descriptions. As a result, the Entity Identifier produces an *Entity Model* as output. This model contains information about entities and their respective movies, and also information regarding the User SOM. The information can be used to produce an entity visual map with the identified user entities, a table presenting the most relevant terms in the movie descriptions and the associated movies to each of the entities. Finally, it can be used in the Recommendation System (RS) context to enhance its performance.

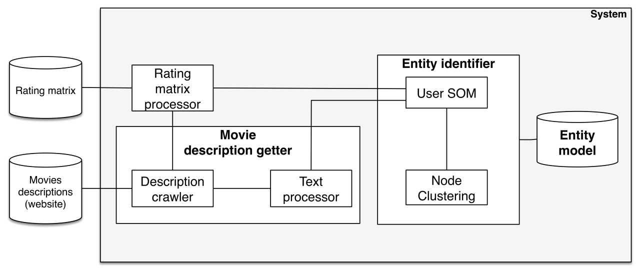


Figure 4: Architecture of our Entity Identifier.

A. Movie description getter

Since we want to identify the entities of a user, the description of the movies will be used as indicative of the user tastes. For example, for two movies of a sequel or two action movies about a hostage situation, we expect them to share similar synopsis and keywords. This is the type of information that we want to extract and take advantage of, to identify user entities.

For all the movies, the synopsis and keywords were obtained. To do that we used one of the most widely known movie database IMDB[1]. The Movie description getter can then process the text, from each synopsis, and create TF-IDF vector of the movie, allowing movies to be comparable between them.

1) Description crawler: Since the Internet Movie Database (IMDB) is one of the most widely used movie database it was selected as our movie description source. The *Description crawler* was developed to store on our system the movies synopsis and keywords.

Our developed Description crawler component has two tasks: (1) find the IMDB movie id; (2) obtain the movie description.

The text is then stored on a file, so it can be pre-processed to achieve a representation that allow us to compare movies between them.

In order to reach the movies descriptions on IMDB, an IMDB id must be found. If using the rating matrix from MovieLens-100k dataset (ML), it also provides for all movies their respective IMDB URL. Unfortunately, some of the provided URL's no longer work, therefore a more complex procedure must be adopted to obtain the IMDB id.

After obtaining the movies ids for the IMDB database, we used the library *IMDBpy*[2] to perform the requests of synopsis and keywords.

For movies without synopsis available, all the plots summaries written by users were obtained instead. Since this cannot be done using the IMDBpy library, we crawled the IMDB movie URL. In addition to the keywords, we also collected the producer names, actor names and genres from each movie, using the IMDBpy library. After the task was completed, all available movies descriptions and IMDB ids were stored on the system.

2) Text processor: Once all the movies descriptions are obtained, they have to be converted into something comparable. For this purpose we used the well-known Vector Space Model [8] with a Term Frequency-Inverse Document Frequency (TF-IDF) term weighting scheme [10]. We now explain how this was performed.

Once known the synopsis, text can be turned into a vector. The steps of this process that we refer as text pre-processing are presented in Figure 5.

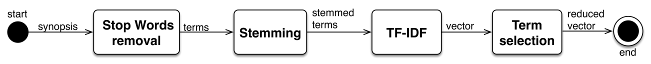


Figure 5: Pre-processing the synopsis to build a term vector.

First, as a standard procedure in text pre-processing, stopwords are removed [10]. Also standard is the conversion of terms to their stems. In our system, we used the stemming algorithm provided in the NLTK library[3]. With the remaining

[1]IMDB website: http://www.imdb.com
[2]*IMDBpy* website: http://imdbpy.sourceforge.net
[3]NLTK website: http://www.nltk.org

set of stems, we can now compute the TF-IDF weights, as explained in Chapter II, and build the term vectors that represent the movies.

3) Selecting the most relevant terms: Since many of the terms describing the movies are still uninformative, some of the less relevant must be excluded.

First, terms that exist only in one movie were dropped out, as they can not be compared to those of other movies. Following terms were filtered by setting an upper and lower limit on their TF-IDF and IDF values.

B. SOM entity identifier

To find user entities, the movie description vectors are provided to the SOM, which finds groupings of similar movies.

1) User SOM: Each user has a SOM that represents his tastes. Figure 6 exemplifies the process of forming the user entities using the SOM.

The process starts by training the SOM, with all of the users movies TF-IDF vectors. The SOM will compute the nodes for each movie, by grouping similar movie descriptions. Finally, the U-matrix is computed, which contains the similarity between contiguous entities.

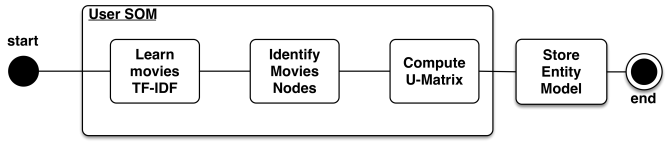


Figure 6: How to form a User SOM .

The formed entities and User SOM will then, be stored in the Entity model and used to identify the entities.

2) Node clustering: Since the nodes that the SOM produces are often too specific and some have very few movies, we further cluster them into groups, which we called entities.

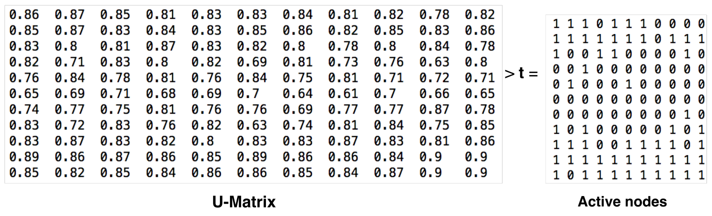
To create the entities, we applied a threshold (t) on the U-Matrix, using Eq. 10, which resulted on a binary matrix $U' = \{u'_{x,y}\}$. This is illustrated in Fig. 7.

$$u'_{x,y} = \begin{cases} 1, & \text{if } u_{x,y} > \text{ t} \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

Afterwards, a labelling algorithm, *connected-component analysis* [32], is applied, using the neighbourhood and the resulting Entity map, shown in Fig. 8. Each different label corresponds to a different identified entity.

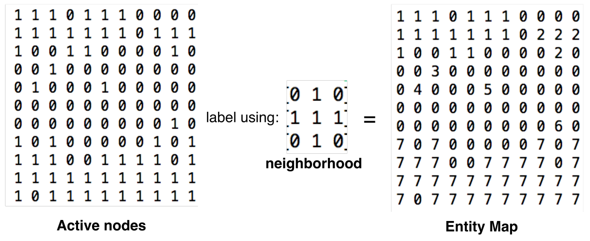
3) Entity Model: Once the nodes have been discovered and grouped into entities, they can provide useful information on the users preferences. This information is recorded by the *Entity Model*.

The entity model is composed by: (1) all users SOM; (2) a table with all users entities and their respective movies.



$$\begin{matrix} 0.86 & 0.87 & 0.85 & 0.81 & 0.83 & 0.83 & 0.84 & 0.81 & 0.82 & 0.78 & 0.82 \\ 0.85 & 0.87 & 0.83 & 0.84 & 0.83 & 0.85 & 0.86 & 0.82 & 0.85 & 0.83 & 0.86 \\ 0.83 & 0.8 & 0.81 & 0.87 & 0.83 & 0.82 & 0.8 & 0.78 & 0.8 & 0.84 & 0.78 \\ 0.82 & 0.71 & 0.83 & 0.8 & 0.82 & 0.69 & 0.81 & 0.73 & 0.76 & 0.63 & 0.8 \\ 0.76 & 0.84 & 0.78 & 0.81 & 0.76 & 0.84 & 0.75 & 0.81 & 0.71 & 0.72 & 0.71 \\ 0.65 & 0.69 & 0.71 & 0.68 & 0.69 & 0.7 & 0.64 & 0.61 & 0.7 & 0.66 & 0.65 \\ 0.74 & 0.77 & 0.75 & 0.81 & 0.76 & 0.76 & 0.69 & 0.77 & 0.77 & 0.87 & 0.78 \\ 0.83 & 0.72 & 0.83 & 0.76 & 0.82 & 0.63 & 0.74 & 0.81 & 0.84 & 0.75 & 0.85 \\ 0.83 & 0.87 & 0.83 & 0.82 & 0.8 & 0.83 & 0.83 & 0.87 & 0.83 & 0.81 & 0.86 \\ 0.89 & 0.86 & 0.87 & 0.86 & 0.85 & 0.89 & 0.86 & 0.86 & 0.84 & 0.9 & 0.9 \\ 0.85 & 0.82 & 0.85 & 0.84 & 0.86 & 0.86 & 0.85 & 0.84 & 0.87 & 0.9 & 0.9 \end{matrix} > \mathbf{t} = \begin{matrix} 1&1&1&0&1&1&1&0&0&0&0 \\ 1&1&1&1&1&1&1&0&1&1&1 \\ 1&0&0&1&1&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \\ 1&0&1&0&0&0&0&0&1&0&1 \\ 1&1&1&0&0&1&1&1&1&0&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \\ 1&0&1&1&1&1&1&1&1&1&1 \end{matrix}$$

Figure 7: Identifying nodes that must be clustered.



$$\begin{matrix} 1&1&1&0&1&1&1&0&0&0&0 \\ 1&1&1&1&1&1&1&0&1&1&1 \\ 1&0&0&1&1&0&0&0&0&1&0 \\ 0&0&1&0&0&0&0&0&0&0&0 \\ 0&1&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&1&0 \\ 1&0&1&0&0&0&0&0&1&0&1 \\ 1&1&1&0&0&1&1&1&1&0&1 \\ 1&1&1&1&1&1&1&1&1&1&1 \\ 1&0&1&1&1&1&1&1&1&1&1 \end{matrix} \quad \text{label using: } \begin{matrix} 0&1&0 \\ 1&1&1 \\ 0&1&0 \end{matrix} = \begin{matrix} 1&1&1&0&1&1&1&0&0&0&0 \\ 1&1&1&1&1&1&1&0&2&2&2 \\ 1&0&0&1&1&0&0&0&0&2&0 \\ 0&0&3&0&0&0&0&0&0&0&0 \\ 0&4&0&0&0&5&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0&0&6&0 \\ 7&0&7&0&0&0&0&0&7&0&7 \\ 7&7&7&0&0&7&7&7&7&0&7 \\ 7&7&7&7&7&7&7&7&7&7&7 \\ 7&0&7&7&7&7&7&7&7&7&7 \end{matrix}$$

Figure 8: How entities were formed, using connected component analysis.

VI. Movie Recommender system

A RS must be able to predict a rating for any user and movie pair. Our system combines both CF and CBF into a Hybrid Filtering RS. On this Chapter we show how to exploit the new information extracted by the Entity identifier to improve rating predictions. For this, we will evaluate two unsupervised machine learning techniques: Singular Value Decomposition (SVD) and KNN, and compare how they perform on RS.

A. SVD as a Recommender system

In order to use entities, the original user rating matrix must be modified producing the entity rating matrix, as illustrated in Fig. 3.

After creating the entity rating matrix, it is decomposed by the SVD technique. Afterwards, the number of features to use is selected, this will allow to perform a dimensionality reduction of the problem.

Finally, the predictions are obtained, based on the number features, resorting to the Eq. 5, which is responsible for the predictions. We can, afterwards, use this predictions to evaluate the produced predictions of the RS or to make recommendations to a user, based on each entity.

B. KNN as a Recommender system

K-Nearest Neighbours (KNN) is a classic implementation of RS using CF. To use KNN one must define a similarity metric, to perform the comparison between two users and define a prediction equation to estimate a rating.

To compute the similarity, both user and entity rating matrix must be used. There will be a similarity metric for users and a similarity metric for entities.

We can, afterwards, compute predictions for movies not rated by users, using the previously computed similarities in the Eq. 3.

1) Prediction: When predicting ratings in the context of CF, one possible approach to this problem is Eq. 3, previously presented. That equation takes into account only the user ratings and, for this reason can not use the entity model that we developed to enhance the prediction.

In order to modify it, the average user ratings must be replaced with the average entity ratings. Equation 11 presents this modification, where $\bar{r}_e$ represents the entity average rating. $G_{u,i}$ represents the set of neighbours who rated the movie i, excluding the user u, to whom we are producing a recommendation.

$$p_{u,i} = \bar{r}_e + \mu_{u,i} \sum_{n \in G_{u,i}} sim(u,j)(r_{j,i} - \bar{r}_j) \tag{11}$$

The variable μ has a function of normalizing factor:

$$\mu_{u,i} = \frac{1}{\sum_{n \in G_{u,i}} sim(u,j)} \Longleftrightarrow G_{u,i} \neq \varnothing \tag{12}$$

The elements contained in $G_{u,i}$ are constrained by the number of k neighbours selected and by the users that rated the item i. $G_{u,i}$ will only include the users that belong to the nearest neighbours and rated the item.

2) Modified Pearson correlation as similarity metric: Pearson correlation is one of the most common similarity metrics used in the context of CF. We tried to improve it, by taking the entity similarity $sim_{entity}(u,j,i)$ into account.

Our modified Pearson correlation, compares both user u and j ratings and multiplies that correlation by the $sim_{entity}(u,j,i)$ on each movie, using Eq. 13.

$$sim(u,j) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u) \times (r_{j,i} - \bar{r}_j) \times sim_{entities}(u,j,i)}{\sqrt{\sum_{i \in I}(U_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I}(r_{j,i} - \bar{r}_j)^2}} \tag{13}$$

The function $sim(u,j,i)$ computes the similarity between the entity of user u that contains the movie i and the entity of user j that contains also the movie i, using the Eq. 13.

3) Weighted similarity between users and entities: This similarity has the purpose of combining user and entity similarities, Eq. 14 denotes this weighted similarity. Notice that a zero α value means that only the users similarity is applied, while α equal to one applies only the entities similarity.

$$sim(u,j,i) = (1-\alpha) \times sim_{users}(u,j) + \alpha \times sim_{entities}(u,j,i) \tag{14}$$

$sim_{entities}(u,j,i)$ needs to know the users and also the movie, since this is the only way to identify the correspondent entities.

$$p_{u,i} = \bar{r}_u + \mu_{u,i} \sum_{n \in G_{u,i}} sim(u,j,i)(r_{j,i} - \bar{r}_j) \tag{15}$$

VII. Evaluation and results

A. Protocol

To build the entity identifier the rating matrix is obtained from the MovieLens-100k dataset (ML) and the movies descriptions extracted from the IMDB website.

The ML dataset has a rating matrix composed by 943 users and 1682 movies. From those, all movies except one have descriptive information, such as title, release date and a URL to IMDB.

The Entity identifier was evaluated by comparing the results obtained by the RS, using different movie features to describe entities. To evaluate the quality and accuracy of rating predictions we use [4, 28]: Mean Average Error (MAE) (Eq. 16) and Perfect Hit (PHIT) (Eq. 17). MAE stands for how much is, on average, the error when making predictions. As for the PHIT, it estimates the percentage of rating predictions that the RS correctly produced. The goal is to minimize the MAE and maximize the PHIT.

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{u,i} - r| \tag{16}$$

$$PHIT = \frac{\sum_{(u,i,r) \in T}(round(p_{u,i}) - r) == 0}{|T|} \tag{17}$$

In the Eq. 16 and Eq. 17, T represents the test set, which contains the users (u), movies (i) and the ratings (r). The rating prediction determined by the RS is represented by $p_{u,i}$.

We used the 5-fold cross validation [19], that is pre-computed on the ML dataset, thus the train set and test set are randomly split into subsets, where $4/5$ are used for training and $1/5$ is used for evaluating the RS.

B. Results

1) Entity identifier: Even-though we obtained the IMDB id for all movies, not all had descriptions available. In order to evaluate the Text Processor, we needed to extract the descriptions, either the synopsis or keywords. The number of movies that we were able to obtain the synopsis and keywords is shown in Table I. We also present how many movies ended up without words, caused by text pre-processing performed.

Description number of movies	with content	TF-IDF vector
Containing synopsis	1652	1618
Containing keywords	1679	1575

Table I: Number of movies with description.

In Table II we present the size of the bag of words for both synopsis and keyword models, before and after the pre-processing. The drastic reduction of the number of words helped, not only in the computation cost, but also improved the performance of the entity identifier. One downside, although, was that a small amount of movies ended up being with a empty $TF-IDF$ vector, which is crucial to identify the entities.

Description Bag of words (size)	before	after
Containing synopsis	10532	3191
Containing keywords	79312	2516

Table II: Number of words in a bag of words.

The result of the entity identifier was not only the entity model, which was used on the RS, but also the visualization of the formed entities and their respective descriptive words. The User SOM was configured to use a $2d$ matrix with 6 nodes on each dimension. The learning rate for all the users SOM was equal to 0.05. To compute the SOM we used the Orange python library[4]

Figure 9 illustrates an entity visual map. The first thing to notice is the different entities, which are represented by colours. Those entities resulted from clustering similar nodes, which were too specific. The nodes that were too specific are represented by each box with three stemmed words. Each word resulted from the highest TF-IDF values in the SOM node vector.

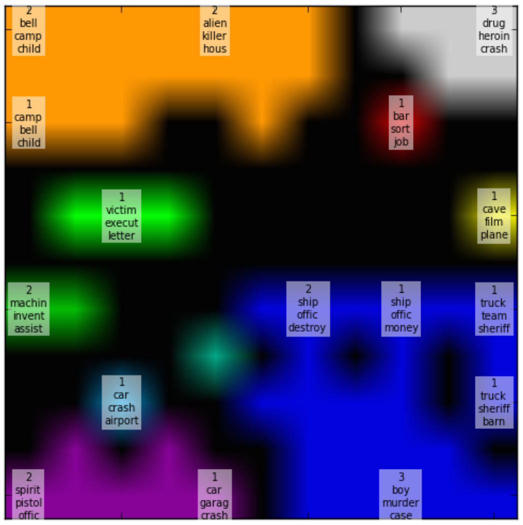
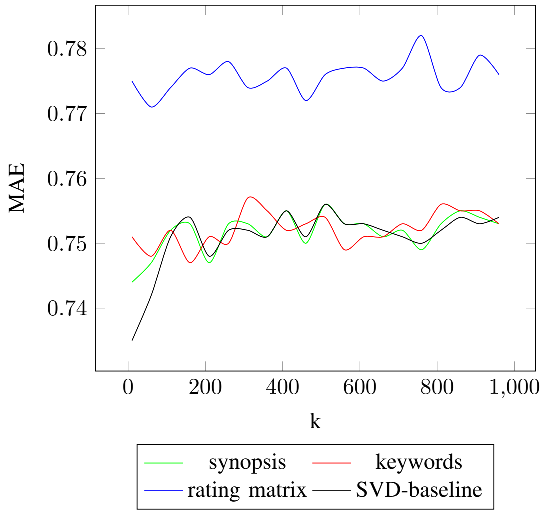
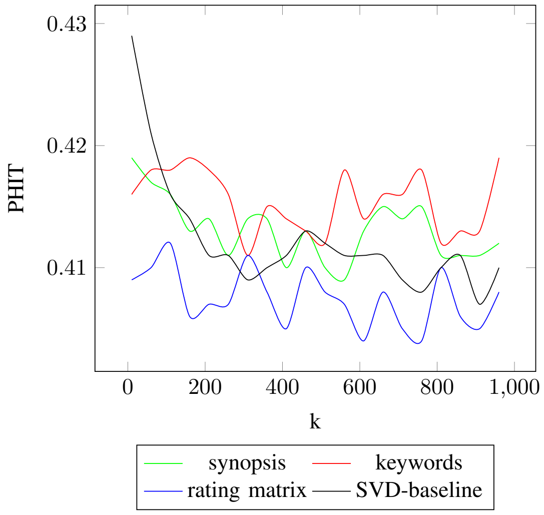


Figure 9: Entity visual map.



(a) MAE



(b) PHIT

Figure 10: SVD performance, when using the entities.

2) Recommendation system using SVD: In Fig. 10, the RS uses the entities, which have been clustered using the Node clustering module. In this case, the performance in MAE and PHIT increases.

As SVD baseline, the MAE and PHIT is computed using the user rating matrix, neglecting the information provided by the entities formed using our Entity identifier.

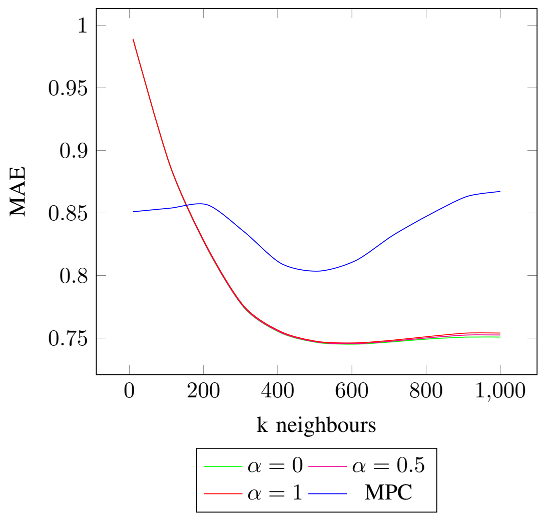
In conclusion, baseline, synopsis entity model and keyword entity model perform very similar to each other. Apart from that, when using around 150 features, we can actually see that both synopsis and keyword entity models outperform our baseline, in both MAE (Fig.10a) and PHIT (Fig.10).

3) Recommendation system using KNN: To evaluate the RS using KNN, we began by comparing the performance of the developed metrics: modified Pearson correlation (Eq. 13) and Weighted similarity between users and entities (Eq. 14). By computing the modified Pearson correlation (MPC) and the Weighted similarity between users and entities (WS) using: $\alpha = 0$, $\alpha = 0.5$ and $\alpha = 1$, we can determine the influence of the neighbours by comparing MAE (Fig. 11a) and PHIT (Fig. 11b). Note that, for $\alpha = 0$ only the users are going to be computed, as for $\alpha = 1$, only the entities and finally for $\alpha = 0.5$ both user and entity similarities will be used and have the same weight in the determined similarity.

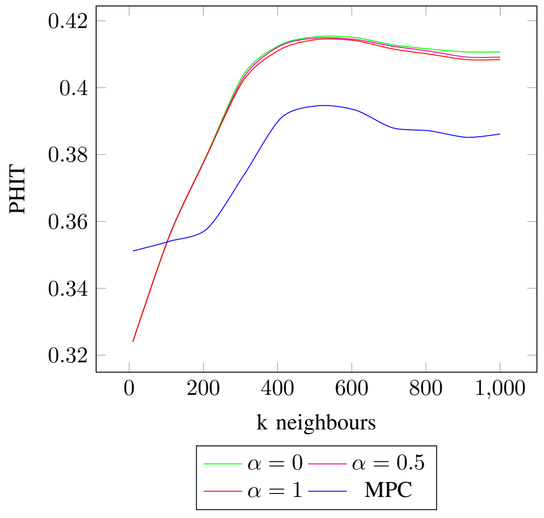
The similarity metric used for both users and entities was Pearson Correlation. The modified Pearson correlation (MPC) performance was worst than the weighted similarity with $\alpha = 1$, which did not overcome the performance of the weighted similarity, with $\alpha = 0.5$ or $\alpha = 0$. Both $\alpha = 0.5$ or $\alpha = 0$, performed almost exactly the same, as can be seen in Fig. 11.

Figure 12 shows the impact of α, with the number of neighbours fixed at 500, since it was previously determined as the best value. We can state that the impact of the entities

[4]Orange python library : https://pypi.python.org/pypi/Orange.



(a) MAE



(b) PHIT

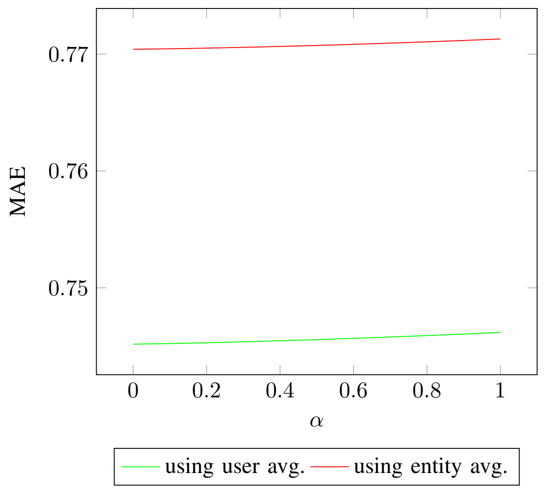
Figure 11: KNN performance, when tuning the number of neighbours.

on the RS using KNN is negative and does not improve the results. When, inspecting the actual values, we observed that the best performance occurred when using $alpha = 0.1$ were the MAE and PHIT were equal to 0.7452 and 41.52%, respectively. Although MAE got worst, the PHIT increased slightly, thus contradicting the pattern, in which generally the MAE increases and the PHIT reduces.

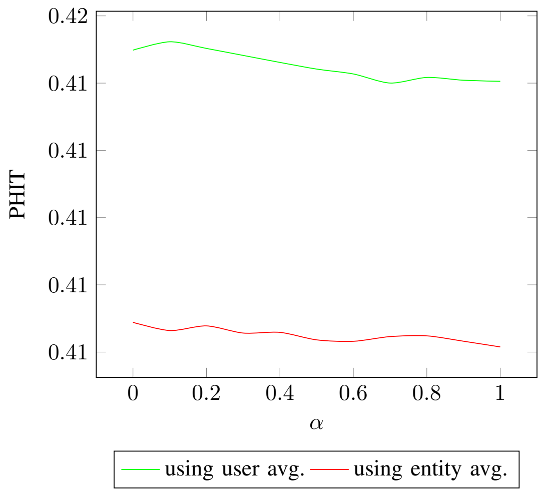
We also compared the impact when using the average of entities instead of the user, but it did not have a positive impact on the system accuracy.

4) Overall RS results: The RSs proposed in the work, combined several techniques. We present two Tables that summarize all the obtained RSs and their best registered results regarding MAE and PHIT.

Table III presents the performance of the entity models when combined with the SVD to create a RS. The results were compared with a baseline, which is the RS using SVD applied to the original rating matrix, provided by the MovieLens-100k dataset (ML).



(a) MAE



(b) PHIT

Figure 12: KNN performance, when tuning the coefficient α.

Entity Model	Using	K-features	MAE	PHIT
Synopsis	nodes	200	0.755	0.411
	entities	200	0.747	0.414
Keywords	nodes	160	0.756	0.416
	entities	160	0.747	0.419
Rating matrix	nodes	200	0.799	0.397
	entities	300	0.774	0.411
baseline	—	200	0.748	0.413

Table III: Evaluation results for the RS using SVD, with different Entity models, using the nodes or entities.

The presented results shown an improvement, when using the entity model based on: synopsis or keywords. The best performance was achieved with the keywords entity model, yet almost equal to the one using synopsis or the baseline.

The summary evaluation of the RS, using the KNN algorithm is presented in the Table IV. In this evaluation, the RS parameters were configured based on the entity model using synopsis which was the only one evaluated in subsection VII-B3, for entity models based on synopsis and keywords. We did not include the rating matrix because of the obtained results in Table III.

Entity Model	similarity metric			predition $\bar{r}$	MAE	PHIT
	Equation	user	entity			
synopsis	WS ($\alpha = 0.1$)	pearson	pearson	user	0.7463	0.4152
	WS ($\alpha = 0.1$)	pearson	novel	user	0.7462	0.4154
	WS ($\alpha = 0.1$)	novel	pearson	user	0.7480	0.4118
	WS ($\alpha = 0.1$)	novel	novel	user	0.7464	0.4130
	WS ($\alpha = 0.1$)	pearson	pearson	entity	0.7704	0.4066
	MPC	—	pearson	user	0.8034	0.3946
keywords	WS ($\alpha = 0.1$)	pearson	pearson	user	0.7460	0.4155
	WS ($\alpha = 0.1$)	pearson	novel	user	0.7459	0.4157
	WS ($\alpha = 0.1$)	novel	pearson	user	0.7482	0.4118
	WS ($\alpha = 0.1$)	novel	novel	user	0.7458	0.4134
	WS ($\alpha = 0.1$)	pearson	pearson	entity	0.7922	0.4022
	MPC	—	pearson	user	0.7937	0.3983
—	baseline WS ($\alpha = 0.0$)	pearson	pearson	user	0.7463	0.4153

Table IV: Evaluation results for the RS using KNN, using $k = 500$, with different Entity models.

VIII. Conclusions

In this work was presented a RS for the cinematographic domain, which was able to identify the users specific interests that we named entities. To identify them we used a SOM that allowed a map representation of the user preferences. The identified entities were afterwards applied to a CF algorithm, thus providing a more detailed and specific user information in order to help in the predictions of ratings.

Our solution differs from others since the entities of each user are mapped according to groups of similar movies evaluated by the user, information that is used to make predictions.

The defined objectives were accomplished: the developed work shows that our solution is comparable to traditional RS regarding prediction accuracy and that SOM allow the identification of users specific interests. Through our evaluation process, we concluded that a marginal prediction accuracy was obtained in some of the presented systems.

As future work, we propose a more extensive study regarding the configuration with a fine tune of the SOM parameters and the exploit of the identified entities to enhance the process of recommendation, as our main focus was to enhance the accuracy of ratings predictions.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-Based Systems*, 46(0):109 – 132, 2013.

[3] J. Bobadilla, F. Serradilla, and J. Bernal. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, 23(6):520 – 528, 2010.

[4] J. Bobadilla, F. Serradilla, and A. Hernando. Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22(4):261 – 265, 2009. Artificial Intelligence (AI) in Blended Learning (AI) in Blended Learning.

[5] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Information Processing and Management*, 49(1):13 – 33, 2013.

[6] Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García-Sánchez. Social knowledge-based recommender system. application to the movies domain. *Expert Systems with Applications*, 39(12):10990 – 11000, 2012.

[7] Walter Carrer-Neto, María Luisa Hernández-Alcaraz, Rafael Valencia-García, and Francisco García-Sánchez. Social knowledge-based recommender system. application to the movies domain. *Expert Systems with Applications*, 39(12):10990 – 11000, 2012.

[8] Yen-Liang Chen and Yu-Ting Chiu. An ipc-based vector space model for patent retrieval. *Information Processing and Management*, 47(3):309 – 322, 2011.

[9] Sang-Min Choi, Sang-Ki Ko, and Yo-Sub Han. A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, 39(9):8079 – 8085, 2012.

[10] R.F. Correa and B.F. Pinheiro. Self-organizing maps applied to information retrieval of dissertations and theses from bdtd-ufpe. In *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pages 31–36, 2010.

[11] J. A. Diaz-Garcia and F. J. Caro-Lopera. Shape theory via SVD decomposition I. *ArXiv e-prints*, March 2010.

[12] Alan Eckhardt. Similarity of users' (content-based) preference models for collaborative filtering in few ratings scenario. *Expert Systems with Applications*, 39(14):11511 – 11516, 2012.

[13] Sam Gabrielsson and Stefan Gabrielsson. The use of Self-Organizing Maps in Recommender Systems. Uppsala University, Department of Information Technology, 2006.

[14] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.

[15] Jason J. Jung. Attribute selection-based recommendation framework for short-head user group: An empirical study by movielens and {IMDB}. *Expert Systems with Applications*, 39(4):4049 – 4054, 2012.

[16] Dan Kalman. A singularly valuable decomposition: The svd of a matrix. *College Math Journal*, 27:2–23, 1996.

[17] Ahmad A. Kardan and Mahnaz Ebrahimi. A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups. *Information Sciences*, 219(0):93 – 110, 2013.

[18] Byeong Man Kim, Qing Li, Chang Seok Park, Si Gwan Kim, and Ju Yeon Kim. A new approach for combining content-based and collaborative filters. *J. Intell. Inf. Syst.*, 27(1):79–91, July 2006.

[19] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[20] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[21] S. Lahabar and P.J. Narayanan. Singular value decomposition on gpu using cuda. In *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–10, May 2009.

[22] Meehee Lee, Pyungseok Choi, and Yongtae Woo. A hybrid recommender system combining collaborative filtering with neural network. In Paul Bra, Peter Brusilovsky, and Ricardo Conejo, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 2347 of *Lecture Notes in Computer Science*, pages 531–534. Springer Berlin Heidelberg, 2002.

[23] George Lekakos and Petros Caravelas. A hybrid approach for movie recommendation. *Multimedia Tools and Applications*, 36(1-2):55–70, 2008.

[24] Monika Mandl, Alexander Felfernig, Erich Teppan, and Monika Schubert. Consumer decision making in knowledge-based recommendation. *Journal of Intelligent Information Systems*, 37(1):1–22, 2011.

[25] Nikos Manouselis, Hendrik Drachsler, Katrien Verbert, and Erik Duval. Recommender systems for learning. 2012.

[26] Prem Melville, Raymod J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth National Conference on Artificial Intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[27] Netflix. Netflix prize, January 2009.

[28] Royi Ronen, Noam Koenigstein, Elad Ziklik, and Nir Nice. Selecting content-based features for collaborative filtering recommenders. In *Proceedings of the 7th ACM conference on Recommender systems*, RecSys '13, pages 407–410, New York, NY, USA, 2013. ACM.

[29] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[30] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating ;word of mouth;. In *Proceedings of the SIGCHI Conference on Human*

Factors in Computing Systems, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[31] D. Sovilj, T. Raiko, and E. Oja. Extending self-organizing maps with uncertainty information of probabilistic pca. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–7, 2010.

[32] George Stockman and Linda G. Shapiro. *Computer Vision*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[33] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Moviexplain: a recommender system with explanations. In *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, pages 317–320, New York, NY, USA, 2009. ACM.

[34] Chih-Fong Tsai and Chihli Hung. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing*, 12(4):1417 – 1425, 2012.

[35] Paula Cristina Vaz, David Martins de Matos, Bruno Martins, and Pável Calado. Improving an hybrid literary book recommendation system through author ranking. *CoRR*, abs/1203.5324, 2012.

[36] T. Yamaguchi and T. Ichimura. Visualization using multi-layered u-matrix in growing tree-structured self-organizing feature map. In *Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on*, pages 3580–3585, Oct 2011.

[37] Xujuan Zhou, Yue Xu, Yuefeng Li, Audun Josang, and Clive Cox. The state-of-the-art in personalized recommender systems for social networking. *Artificial Intelligence Review*, 37(2):119–132, 2012.