Contents

List of Tables

List of Figures

List of Acronyms

BMU Best Matching Unit

CF Collaborative Filtering

CBF Content-Based Filtering

IDF Inverse Document Frequency

IMDB Internet Movie Database

KNN K-Nearest Neighbours

ML MovieLens-100k dataset

MAE Mean Average Error

MSD Mean Square Difference

PHIT Perfect Hit

RS Recommendation System

SOM Self-Organizing Map

SVD Singular Value Decomposition

TF Term Frequency

TF-IDF Term Frequency-Inverse Document Frequency

U-Matrix Unified-Distance Matrix

WS Weighted similarity between users and entities

Chapter 1

Introduction

1.1 Contextualization

Technology evolution led to a big and diverse amount of information on the internet. This evolution caused the user to be overloaded with information [2]. 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 [2, 3]. Nowadays, recommendation systems can be found in a vast number of services, such as movies, music, news, products and services recommendation, among others [4].

Recommendation systems allow to discover new items that might please the user, among the diversity of items available. Some websites, such as Amazon, Grouplens, Ebay and Netflix, are real examples of services that benefit from providing recommendations for their clients [5].

Despite the research on this topic, few references make use of Self-organizing Maps [2, 6, 7, 8]. 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 [6]

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 defined goals were achieved, we were able to identify users specific interests and use that information in the context of Recommendation system.

1.2 Dissertation layout

This dissertation is composed of eight chapters which are structured as follows:

- **Chapter 2** presents the basic concepts regarding Self-organizing maps, and recommendation systems;
- **Chapter 3** contains related work about recommendation systems and self organizing maps;
- **Chapter 4** describes the architecture of the proposed system, including the entity identifier and the actual recommendation system;
- **Chapter 5** shows how the entity identifier functions;
- **Chapter 6** clarifies how we developed the recommendation system;
- **Chapter 7** presents the evaluation of the developed system;
- Finally, **Chapter 8** contains the conclusions.

Chapter 2

Basic concepts

In this chapter, the concepts of Self-Organizing Map (SOM) and Recommendation System (RS) will be introduced.

2.1 Self-organizing maps

First presented by Tuevo Kohonen in 1982 [9], the SOM allows to represent signals with large dimensionality in a lower-dimensional space, so that the formed topology, is observable by the human eye as a map [9]. 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 [2].

In Fig. 2.1, 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.

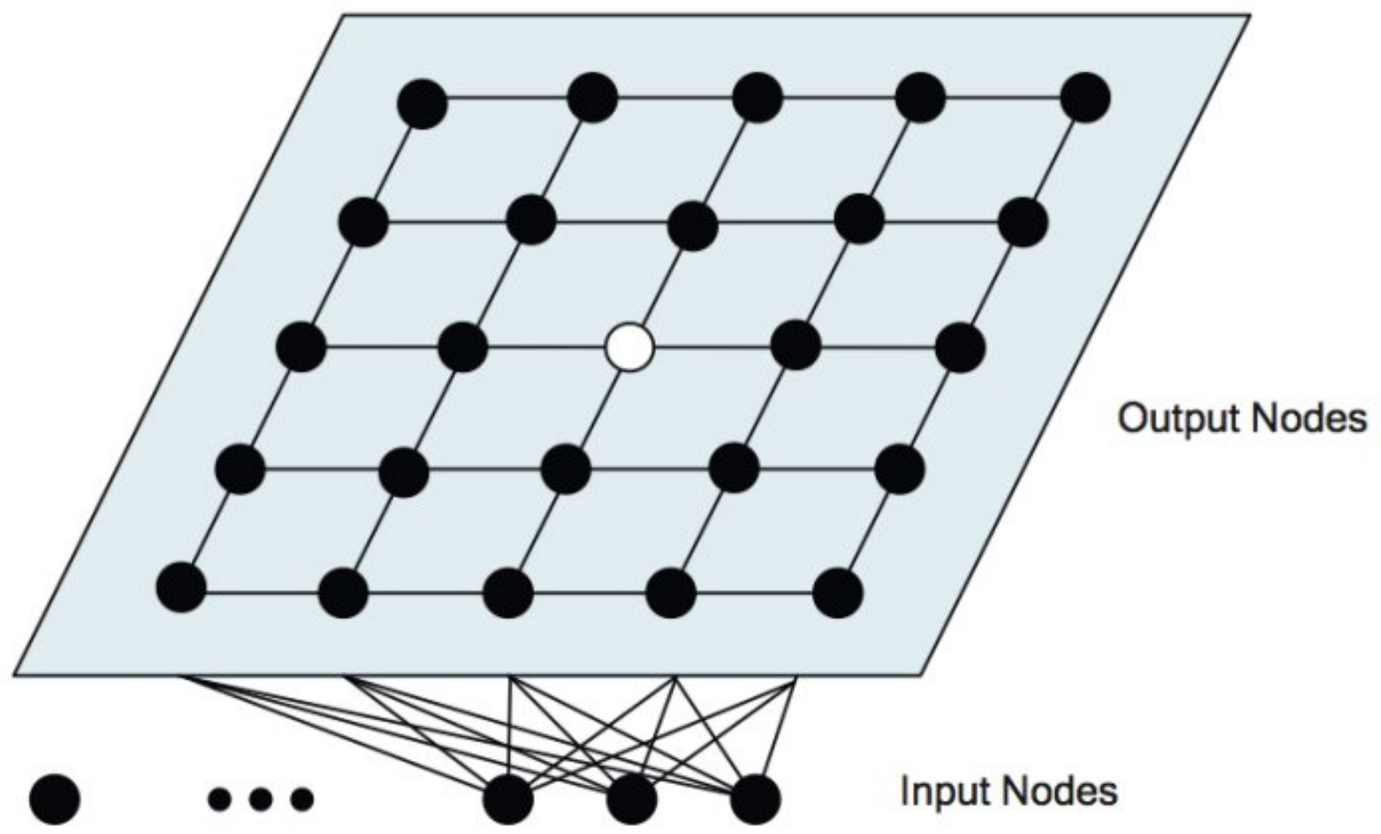


Figure 2.1: SOM input nodes and output nodes.

This clustering technique is based on a competition training, split in two separate

stages, forming a iterative algorithm that executes each of them alternately. 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) [2, 10].

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. 2.1.

$$m_p(t+1) = m_p(t) + \alpha(t)h_{cp}(t)[x_n - m_p(t)] \tag{2.1}$$

In Eq. 2.1, 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.

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) [11].

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 [11]. An example of the U-Matrix visualization is shown in Fig. 2.2, where the values nearest to one correspond to similar nodes and zero to more distant nodes.

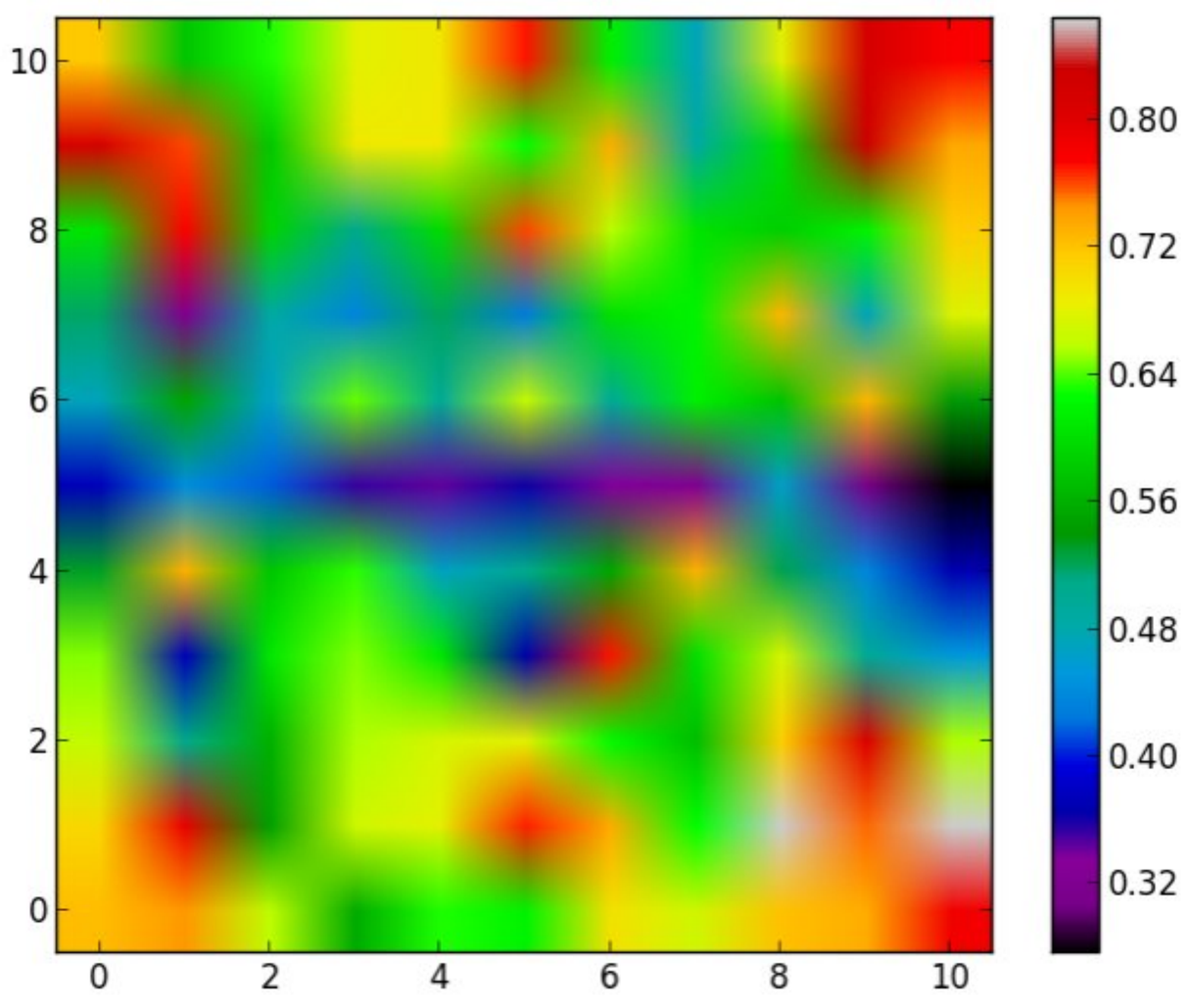


Figure 2.2: Example of a U-Matrix.

2.2 Recommendation systems

The first RS was developed by Goldberg et al. in 1992 [12], in order to solve the overload of emails that users had in their inbox. It is a filtering algorithm where users collaborate by registering their reaction to the documents after reading them [12].

RS are a networking tool that offer dynamics and collaborative communication, interaction and knowledge [13]. 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 [4].

A RS is composed by two modules: a database and a filtering technique [1]. The database is responsible for storing the information about users, items and the associated ratings. The filtering technique is composed by an algorithm, generally split in two stages. On the first, the similarities between users or items are identified providing a neighbourhood of each item or user. At the second stage, the system predicts ratings corresponding to the items unseen by the user, and only the best ones are used as recommendation [14].

As result of research, several types of filtering algorithms have been developed [1, 4, 15, 16]:

- Collaborative filtering: recommends items based on similar users and their ratings;
- Content-based filtering: the user receives recommendations based on the similarity of the profile and the items features;
- Demographic filtering: based on information regarding the user, for example, the gender, the age, the profession and the location of the user;
- Social filtering: by resorting to the user social network;
- Hybrid filtering: result of a combination of different filtering techniques.

On the following sections, we will discuss more extensively the three most common types: collaborative , content-based and hybrid filtering, which we use in our work.

2.2.1 Collaborative filtering

Collaborative Filtering (CF) algorithms perform the comparison of users ratings, resulting in the identification of the most similar ones [17]. 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.

The two most used CF approaches are [1, 4, 18, 19]:

- user-based: identifying the user neighbours (i.e. the most similar users) and then predicting a rating, based on the ratings of the neighbours;
- item-based: identifying the items neighbours (i.e. the most similar items) and then predicting a rating, based on the ratings of the neighbours;

In Fig. 2.3, we show an example of a RS using CF with a user-based approach. The similarity between two users (u and j), is obtained by computing the complement of the euclidean distance (Eq. 2.2). The variable $r_{u,i}$ corresponds to a rating from user u to the item i. The equation uses only the set I, as it contains only the ratings shared by both u and j (rows in the matrix).

$$sim(u, j) = 1 - \sqrt{\sum_{i \in I} (r_{u,i} - r_{j,i})^2} \tag{2.2}$$

As an example, we present one way to compute the top 3 recommendations for user 4. First, the similarity $sim(4, j)$ between 4 and all the other users would be computed. Using the K-Nearest Neighbours (KNN) algorithm the most similar are selected, in this example they are only three users (2, 5 and 7). Then, using only those neighbours, the RS predicts the rating on all the items i that the user has not rated. In this example, the prediction equation is the average of the rating of the item, considering only the neighbour users. Finally, the top 3 items are selected (1, 7 and 9) and recommended [1].

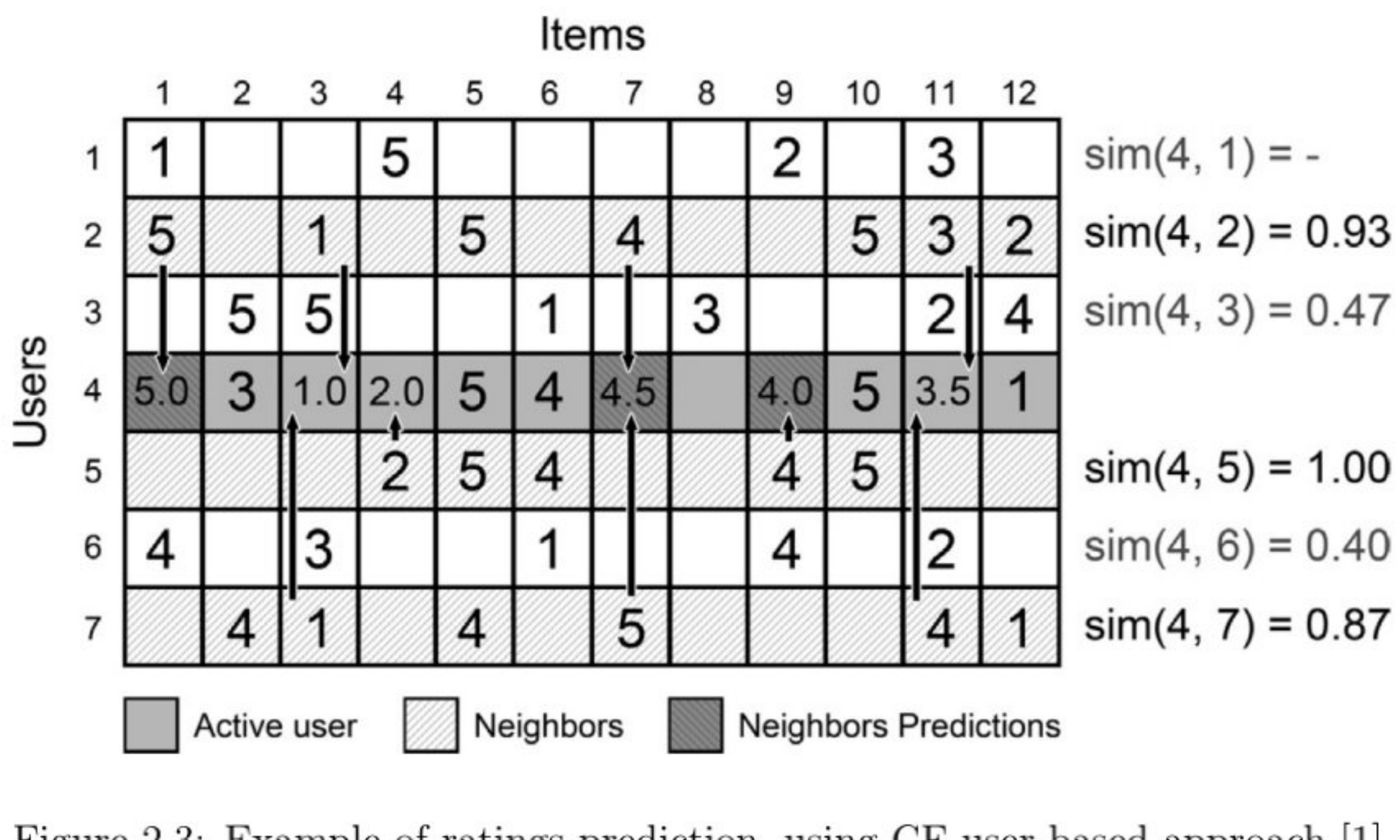


Figure 2.3: Example of ratings prediction, using CF user-based approach [1].

In practice, many other similarity metrics and prediction equations can be, and are used. We now explain the most commonly adopted.

Similarity metric

When using CF algorithms, 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 [17]. 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 [20], in 1995 and it has been widely used in RS, as it provides good results [17]. It is obtained by applying Eq. 2.3, 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 $\bar{r}_j$.

$$sim(u,j) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u) \times (r_{j,i} - \bar{r}_j)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I}(r_{j,i} - \bar{r}_j)^2}} \tag{2.3}$$

The result of Eq. 2.3 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 [6]. Values near zero show that no linear correlation exists between the two users. In Figure 2.4 we illustrate an example of three users who have four items in common. The users 1 and 2 have a positive correlation according to their ratings, even though they did not rate the items exactly the same, there is a correlation between the two.

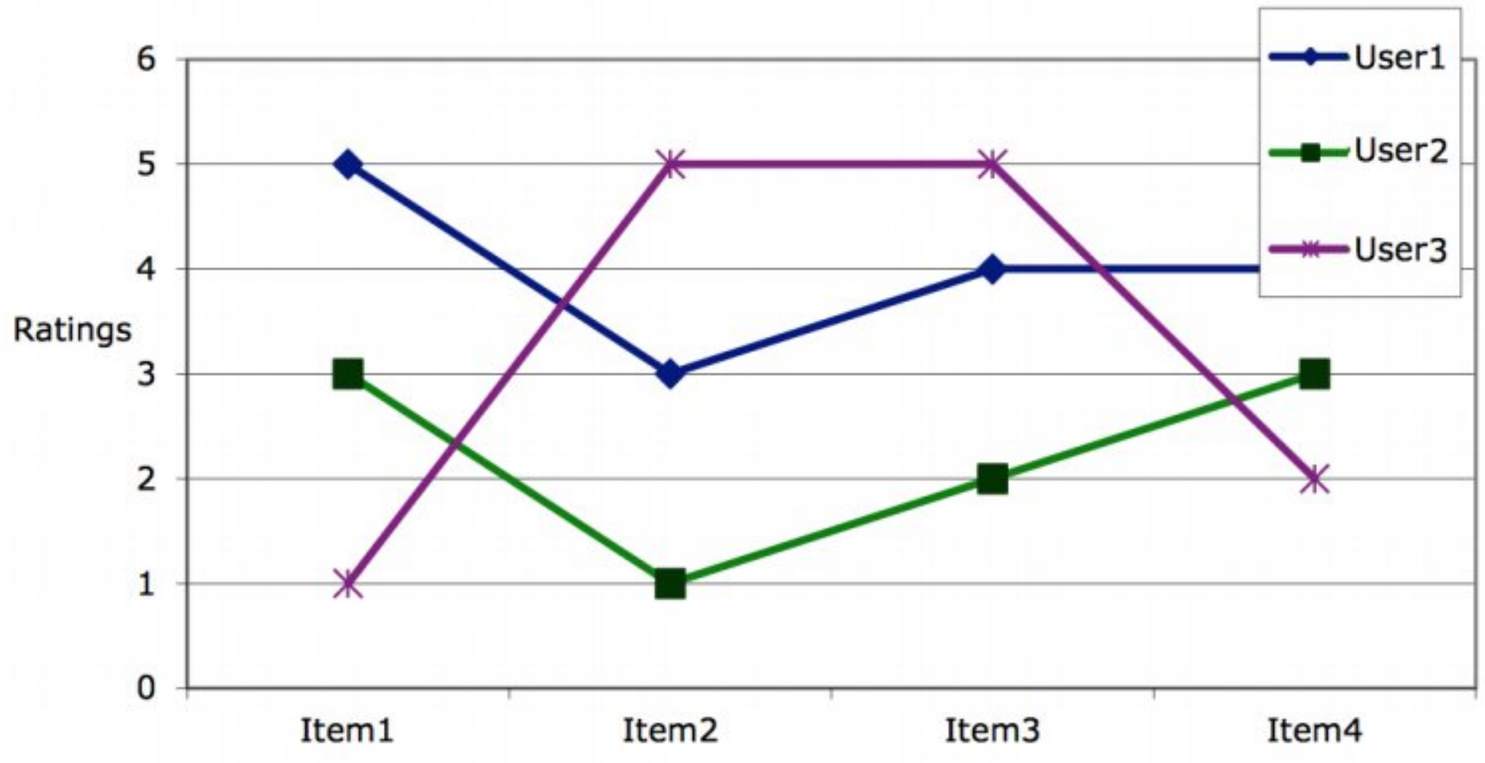


Figure 2.4: Example of the users ratings and their correlation.

Prediction metric

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 [17]. 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. 2.4, 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 [17].

In Eq. 2.4, $p_{u,i}$ denotes the prediction value determined, $\mu_{u,i}$ is defined in Eq. 2.5 and corresponds to a normalizing factor [6].

$$p_{u,i} = \bar{r}_u + \mu_{u,i} \sum_{n \in G_{u,i}} sim(u,j) \times (r_{j,i} - \bar{r}_j) \tag{2.4}$$

$$\mu_{u,i} = \frac{1}{\sum_{n \in G_{u,i}} sim(u,j)} \iff G_{u,i} \neq \varnothing \tag{2.5}$$

Collaborative filtering using SVD

Singular Value Decomposition (SVD) is used as a recommendation method, and it was firstly used on a case study [21]. It performs a decomposition of the rating matrix. As a result, it enables a reduction of dimensionality that improves the overall RS accuracy and efficiency.

The method works as follows. The ratings matrix (X) is decomposed in three matrices: U, S and V^T (Fig. 2.5) using SVD [22]. There is a lot of research in how to perform this decomposition since this is the most computational demanding task in the recommendation process [23, 24]. SVD allows to reduce the dimensionality of the problem by cropping U, S and V^T, on the k dimension. In Fig. 2.5, n is the number of items, m the number of users and k the number of kept features, being a lower or equal than n and m.

Regarding the decomposed matrices, U contains features describing each user, S provides information about the weight of each element in a row or column, on the matrix U and V^T respectively, and V^T which contains vectors with features describing the items.

Note that the original matrix X can not be rebuilt after cropping the decomposed matrices, therefore resulting in $\widetilde{X}$, which is an approximate rating matrix.

With Eq. 2.6 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.

$$p_{u,i} = \bar{r}_u + U_k(u) \times \sqrt{S_k} \times V_k^T(i) \tag{2.6}$$

2.2.2 Content based filtering

Sometimes, by using the items meta-data, recommendations can be enhanced since it provides detailed information about either the item or the user. For that reason, Content-Based Filtering (CBF) was introduced as a filtering technique. It traditionally

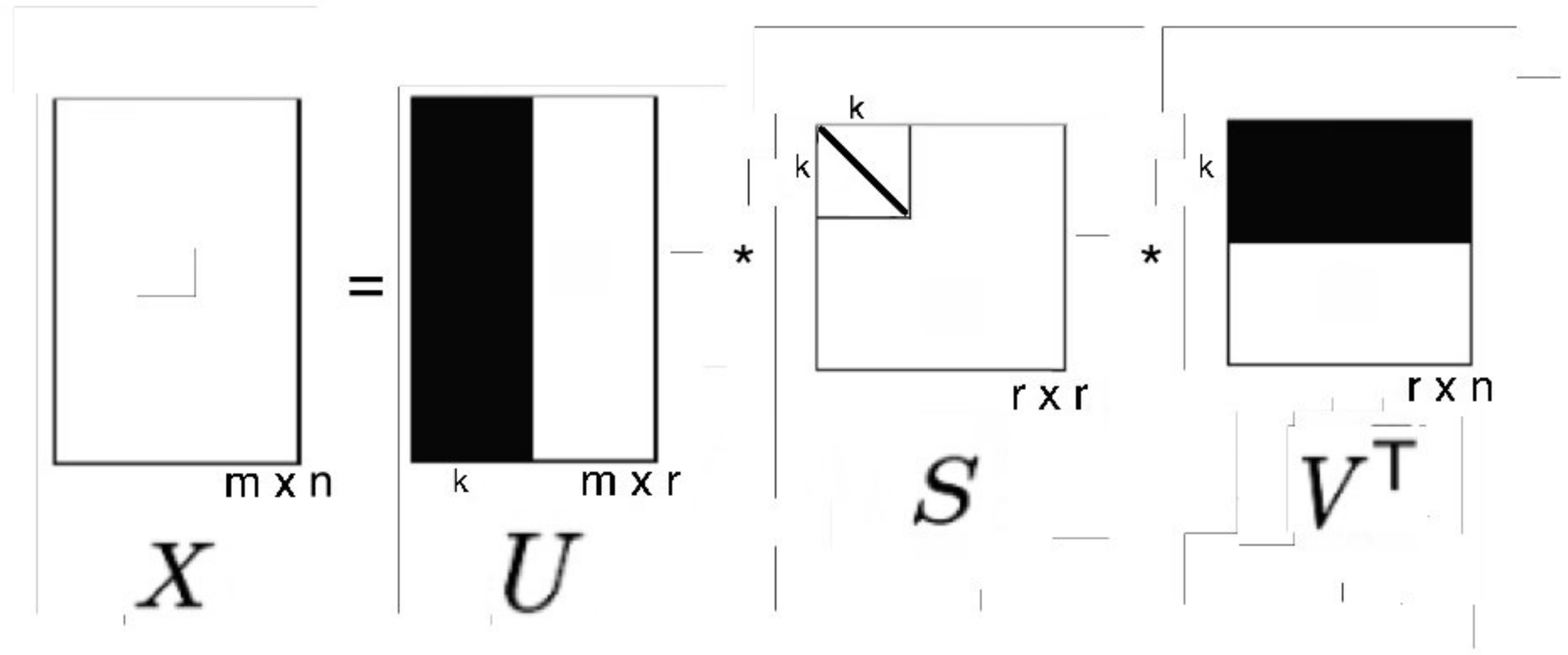


Figure 2.5: SVD matrix decomposition into U, S and V^T.

works by determining items related to those that users previously manifested their interest in [14].

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 [14].

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 [25].

The removal of stopwords is a process that deletes from the text, the most common and usually not relevant terms, such as: "the", "in", "on", "at", "from", "I", "he" and many others.

To further simplify the terms, stemming can be performed, e.g. aggregating terms with a similar root [26]. Considering, for example, the terms: "win", "winner" and "wining". When stemmed, they would be transformed to the form "win". Therefore, the complexity of the system can be reduced since the diversity and the number of different terms will become smaller.

After removing stopwords and performing stemming on all textual descriptions on a collection of items, a bag of words representation can be created. This consists in representing each item as a vector, where each dimension corresponds to a word.

To describe each item, a vector with the weight of each term on the text can be computed. The most common weighting scheme used is the so-called Term Frequency-Inverse Document Frequency (TF-IDF) [27].

Term Frequency (TF), shown in Eq. 2.7, determines the relevance of the term in the item description, where $f_{i,j}$ is the number of times that a term j is present on each item description i.

$$TF_{i,j} = \frac{f_{i,j}}{max\{f_{z,j}\}} \tag{2.7}$$

The Inverse Document Frequency (IDF) is computed by Eq. 2.8. The values represent how rare terms are in the collection of items. The variable N is the number of item descriptions available and n_j the total of documents containing the term j. Thus, rarer terms have higher IDF weight.

$$IDF_j = \log \frac{N}{n_j} \tag{2.8}$$

Finally, the TF-IDF value takes into consideration the frequency of the term on the document (TF) and weights it with the frequency of the term on the collection of items (IDF). To compute each element of the it Eq. 2.9 is used.

$$w_{i,j} = TF_{i,j} \times IDF_i \tag{2.9}$$

Then each text corresponding to the item description is finally represented by the vector:

$$\overrightarrow{d_i} = (w_{i,1}; w_{i,2}; \ldots; w_{i,k}) \tag{2.10}$$

A user can then be represented by average of items descriptions $\overrightarrow{d_i}$. For example, by computing the average of all item description vectors rated by the user. Then it can be used to represent each of the users, making users and items comparable. In order to identify the relevance of each item to a user, a similarity metric must be defined between the user and the items. An example is Eq. 2.2 [4].

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) [1, 28].

2.2.3 Hybrid filtering

The previously discussed filtering techniques can be combined to produce a RS that uses hybrid filtering [29]. Combining them is expected to improve the overall quality of the RS and the provided recommendations [15, 28]. Although any filtering techniques are possible to be combined, we will only focus on combining CF an CBF, since we will use them on our system. There are several ways to combine two filtering techniques [4], as shown in Figure 2.6:

- Case A: implementing a CF and a CBF separately and combining their recommendations afterwards. For example, if we use the average between the predictions from both CF and CBF;

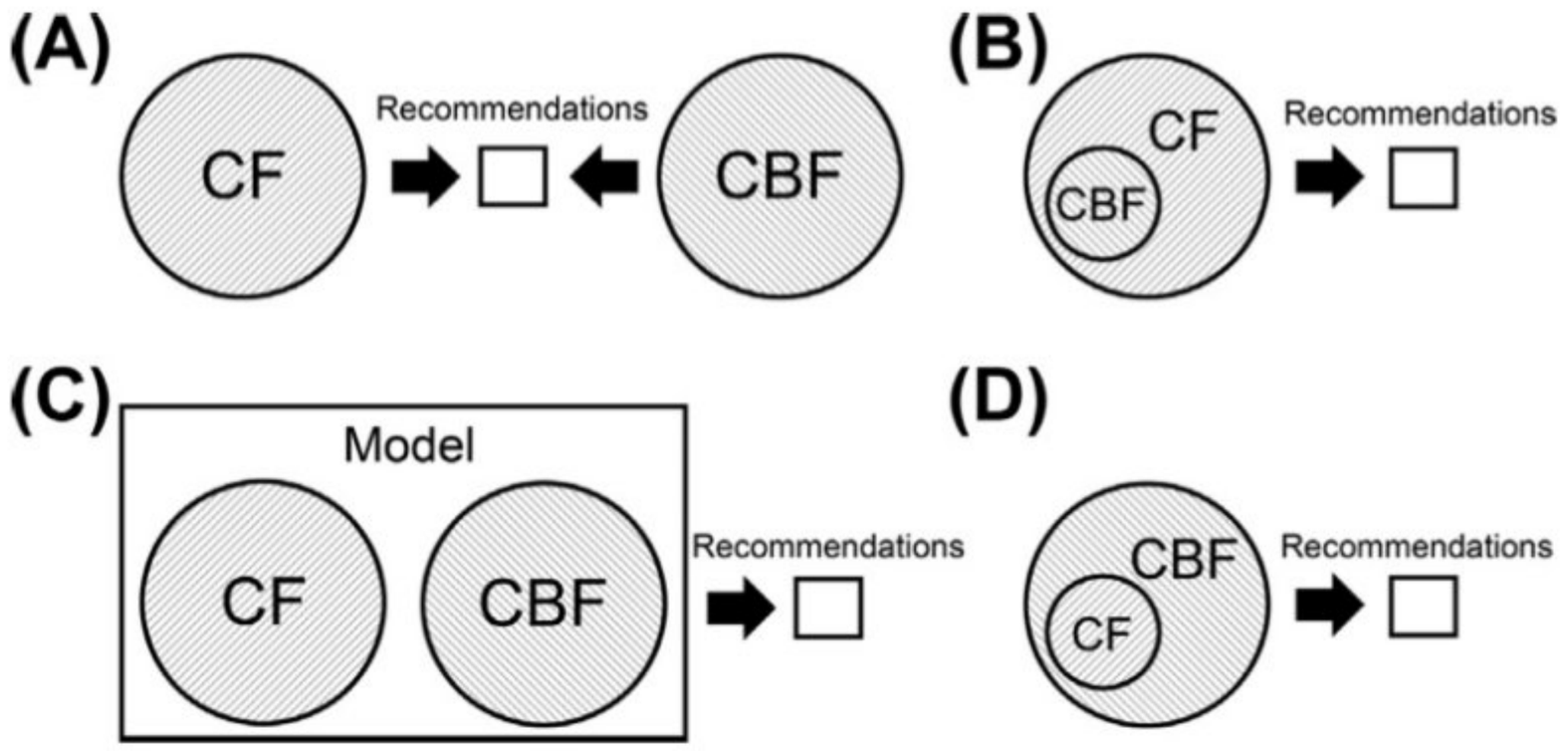


Figure 2.6: Several models of hybrid filtering, using both CF and CBF [1].

- Case B: incorporating some of the characteristics of the CBF into the CF, e.g. the incorporation of personal information in CF, to alleviate the problem of introducing a new user to the RS;
- Case C: building a general model that takes into account characteristics from both CF and CBF, therefore combining their results with a machine-learing algorithm (e.g. using Bayes-Networks);
- Case D: incorporating some of the characteristics of the CF into the CBF. For example, using the determined recommendations from the CF used as an input to the CBF algorithm;

Chapter 3

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 [1, 30]. In this Chapter, a summary analysis will be done over the main research topics of the last decade, between 2002 and 2014. This analysis will be based on surveys [1, 4, 30, 31, 32]. Then, we will present some works that we consider relevant to the development of our work and present solutions for the current problems in RS. To conclude, some of the works that use SOM to enhance RS will be discussed followed by RS that were developed specifically for producing movie recommendation.

3.1 RS: Historical Contextualization

Although we can find reference to early RS since 1992 [12], until 2002 the main research topic on RS was how to combine various filtering techniques to enhance the recommendations. In other words, how to develop a RS using hybrid filtering. At the same time, the topic of knowledge-based filtering was emerging [31].

Around 2004 and 2005, due to the emerging of larger-scale internet services, the current research topics were no longer the same. Most of the research was about the computational cost of the RS, either regarding the used memory to run the RS or about the time that the recommendations would take to be computed [4]. The general use of RS also led to the conclusion that they need to be tuned to work properly in different domains. For example a RS designed for recommending in the movie domain will probably have a different performance in the music domain. The robustness and the transparency of the RS were, thus, at that time a recent research field [4, 32].

3.2 Current research

Today, there are many studies regarding RS. Yet, there is still room for the development of new techniques [30]. 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 [1]. The topics (1) and (2) will be discussed on the following subsections, since they are the most relevant for the purpose of our work.

3.2.1 Hybrid Filtering

A large number of research focus on hybrid filtering [15, 28, 29, 33]. Most of them combine CF with another filtering technique, thus exploiting the advantages of each to enhance the RS results [1, 33].

In [29], an hybrid approach is used to build a RS for the book domain, using the following matrices: $user \times book$ with the ratings of the user per book and the $user \times author$ with the average rating of all books from the same author. Then, we use two item-based CF approaches: (1) a matrix of book similarity $book \times book$ and (2) an author similarity matrix $author \times author$. With the two similarity matrices computed, they were combined using the *weighted arithmetic mean* Equation [29], responsible for weighting both and produce a hybrid filtering. The authors suggested that even though the proposed solution improved the overall predictions, further investigation was needed regarding the weights use in the weighted arithmetic mean equation.

Another example of RS that used hybrid filtering is [34]. 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 [35], 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 [35]. Equation 3.1 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.

$$sim(u, j) = (1 - \alpha) \times sim_{user}(u, j) + \alpha \times sim_{group}(u, j) \tag{3.1}$$

In our work we adapted the Eq. 3.1, to aggregate both user and entity similarities. More details are shown in Chapter 6.

3.2.2 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 [36], 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 [36].

Another technique developed a similarity metric that outperforms the Pearson correlation [17]. This new metric was able to enhance the CF techniques, using the complement of the Mean Square Difference (MSD), shown in Eq. 3.2, and combining it with the Jaccard similarity, shown in Eq. 3.3, resulting in the similarity Eq. 3.4. 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 $\#r_{u,j}$.

$$MSD(u,j) = \frac{|r_u - r_j|}{\#r_{u,j}} \tag{3.2}$$

$$jaccard(u,j) = \frac{|r_u \cap r_j|}{|r_u \cup r_j|} \tag{3.3}$$

$$new\ similarity(u,j) = (1 - MSD(r_u, r_j)) \times jaccard(r_u, r_j) \tag{3.4}$$

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 [17].

The authors, in [37], developed an improved similarity metric, that instead of using software, the similarity is computed using cheap electronic components. Their solution brought improvements that proved to be around two times faster to compute than Pearson correlation, without compromising the recommendations [37]. Their solution contributes to a more scalable RS since the time computational cost is reduced [37].

3.3 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 [8]. 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 [8].

The algorithm of Probabilistic SOM [7] 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 [38], because it can ensemble with other models, and combined have an improved global performance.

Finally, S. Gabrielsson compared the performance neighbourhoods using either KNN or SOM [6]. 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.

3.4 Movie Recommendation systems

A large number of RS were tested with movie-based datasets [1, 19, 39, 40]. Yet, not all of them are specifically for movie recommendation, being easily applicable in other domains. In [41] 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 [42] 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 descriptions.

They used a traditional CF with the KNN algorithm and the cosine similarity metric [42]. To develop the CBF, the Naive-bayes algorithm [1] was used, in which the class labels were the possible ratings. They presented a hybrid filtering with two possible modes: (1) in which the CF was only used if there were at least five neighbours, otherwise it would be applied CBF; (2) which was based on the number of ratings that the user had made: if they were fewer than 40, only CBF would be used otherwise only CF was applied. Their findings showed a slight improvement over traditional RS [42].

In [43] the authors proposed an interactive RS, which was able to produce personalized recommendations, resorting to online social networks, using a group-aware social network model, a promising technique for identifying the dynamics of social information. Therefore, they were able to deal with constant and rapid changes in users preferences and their social influence. Their solution was split in two main modules: (1) group-aware social trust management and (2) the recommendation module [43]. A similar approach was used in our work, since we developed two modules: (1) An entity

identifier and (2) a recommendation system. Yet, their approach is more focused on social filtering and ours on CBF.

Chapter 4

System architecture

In most RS, it is not clear why are the users getting their recommendations [19]. 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 an 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).

Figure 4.1 shows that our system has two separate components:

- Entity identifier - which is able to split a user into entities, based on the description of rated movies;
- 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.

4.1 Overview

Each user will have their specific entities discovered using a SOM. The SOM is 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

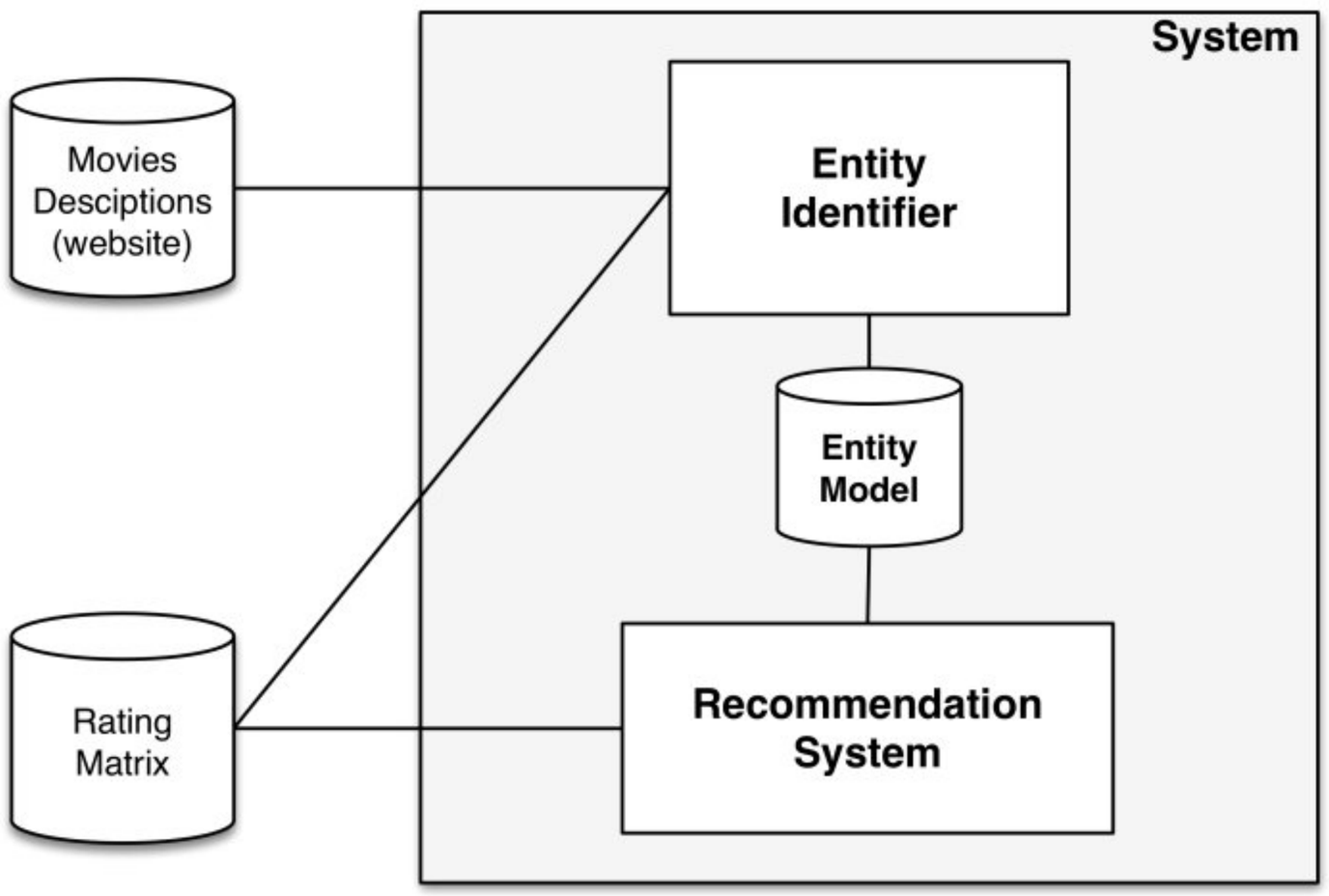


Figure 4.1: Main components of the RS.

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. 4.2 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.

To determine the entities we use a SOM, which clusters the movies rated by the user, according only to their descriptions. This problem is described detailedly in Chapter 5.

4.2 Entity identifier architecture

Our main component, the Entity Identifier has the goal of identifying users specific tastes, using movie descriptions. Our solution employs a SOM to automatically identify each user's entities. Each user will thus, be represented by a SOM, using the rated movies and their respective descriptions. In Figure 4.3 we can observe the various elements that allow the Entity Identifier to accomplish this goal.

In the Entity identifier, there are two components that deal with external information: (1) *Movie description getter*, which downloads the movie information and performs the text pre-processing so that movie descriptions can be used for clustering; and (2) *Rating matrix processor* responsible for loading the rating matrix.

In our solution, we use the movie synopsis as a description, but, we use other available features, such as keywords or the producers and actors.

The component named *user SOM* uses the SOM to identify clusters of movies for

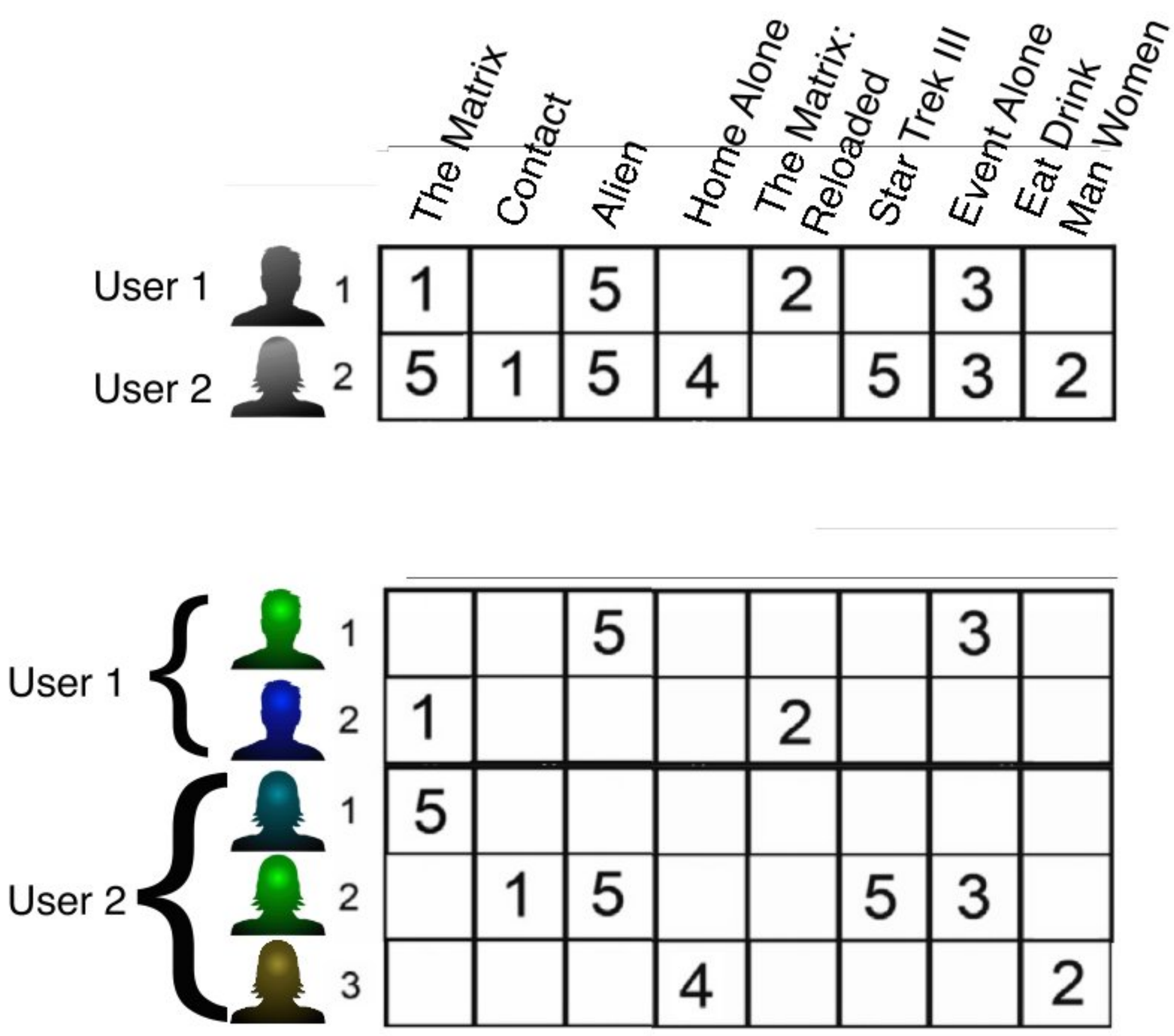


Figure 4.2: Example of entity rating matrix formation.

a given user. As mentioned before, each node, represents one of the user's tastes. The *Node clustering* component is responsible for grouping the most similar nodes to form the entities.

As output, the Entity identifier produces an Entity model, which stores the user SOM and a table with the entities formed and their respective nodes and movies. Using this approach is possible to create a visual map representation for each user, showing the set of entities discovered and their associated nodes. This is illustrated in Figure 4.4. The three keywords on each node are the most relevant words taken from the movie descriptions that represent the node. The colours differentiate the entities, based on the similarity between the node. The detailed functioning of these components will be explained in Chapter 5.

4.3 Recommendation system architecture

A Recommendation System (RS) is able to suggest new items to users that may be interesting to them. The RS proposed can use the information extracted by the Entity Identifier, by exploiting the Entity model to improve its recommendation performance.

Our system can use the classic algorithms used on RS literature, such as SVD [44,

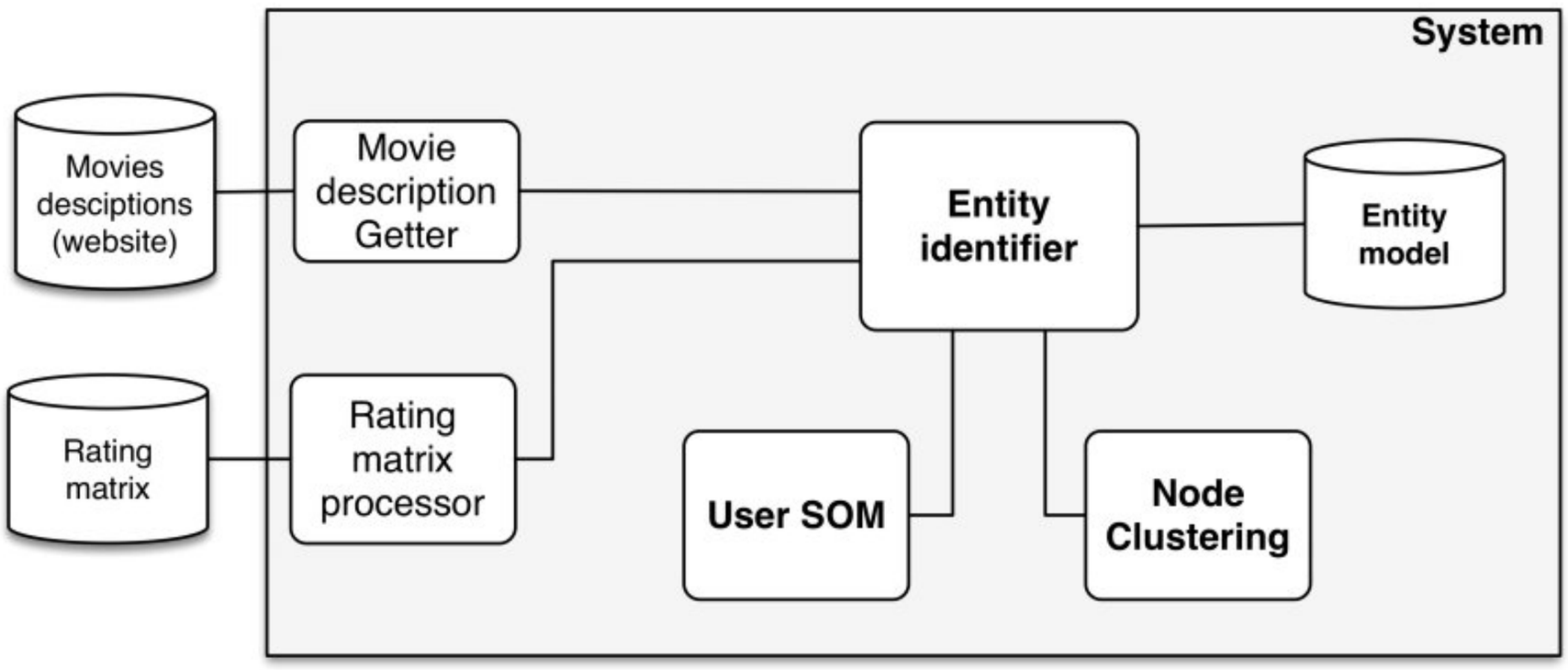


Figure 4.3: Entity Identifier Architecture.

45] or KNN [17, 18]. Nevertheless, the main focus is how to exploit entities (representing users tastes) to improve the recommendation yielded by those classic techniques. This is accomplished by the architecture shown in Fig. 4.5.

Our proposed Recommendation System uses the user entities generated by the Entity Identifier component. The *Rating matrix processor* component provides the rating matrix, listing the movies watched by the user. The RS can then use the rating matrix and the entity model to form an entity rating matrix, where each user is split in the corresponding identified entities.

Once this is done, we can apply CF techniques, as presented in Fig. 4.5 these can be either SVD or KNN. For KNN, we can select a *similarity metric* and a *prediction equation* for the RS, as described in Chapter 2.

As referred in Section 3.2, the similarity metric allows the RS to compare two users (or items) by quantifying the similarity of each pair with a value. In Chapter 6 we present a more detailed view of how we enhance the similarity method to include both user and his respective entity information to improve the RS.

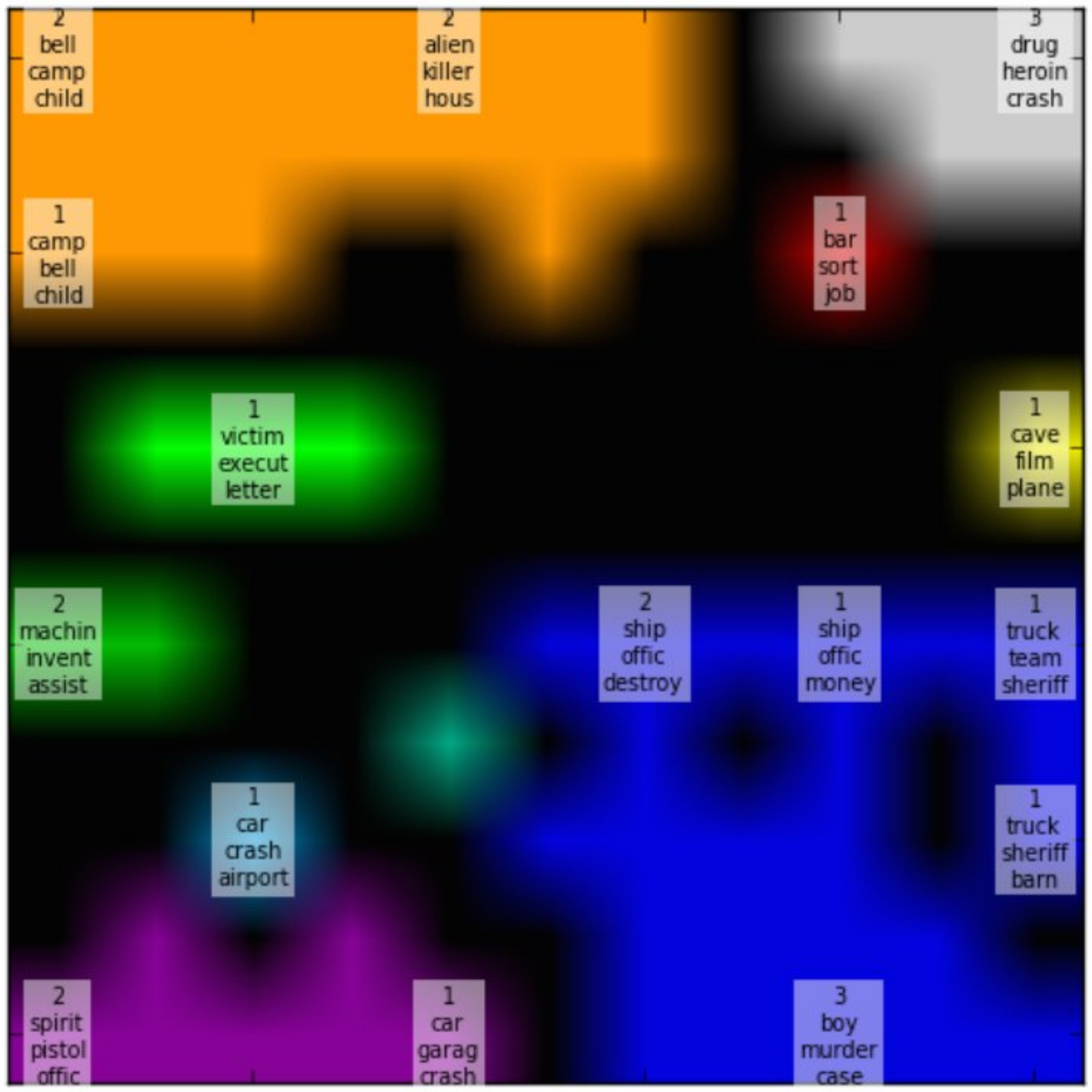


Figure 4.4: An example of an entity visual map for a user.

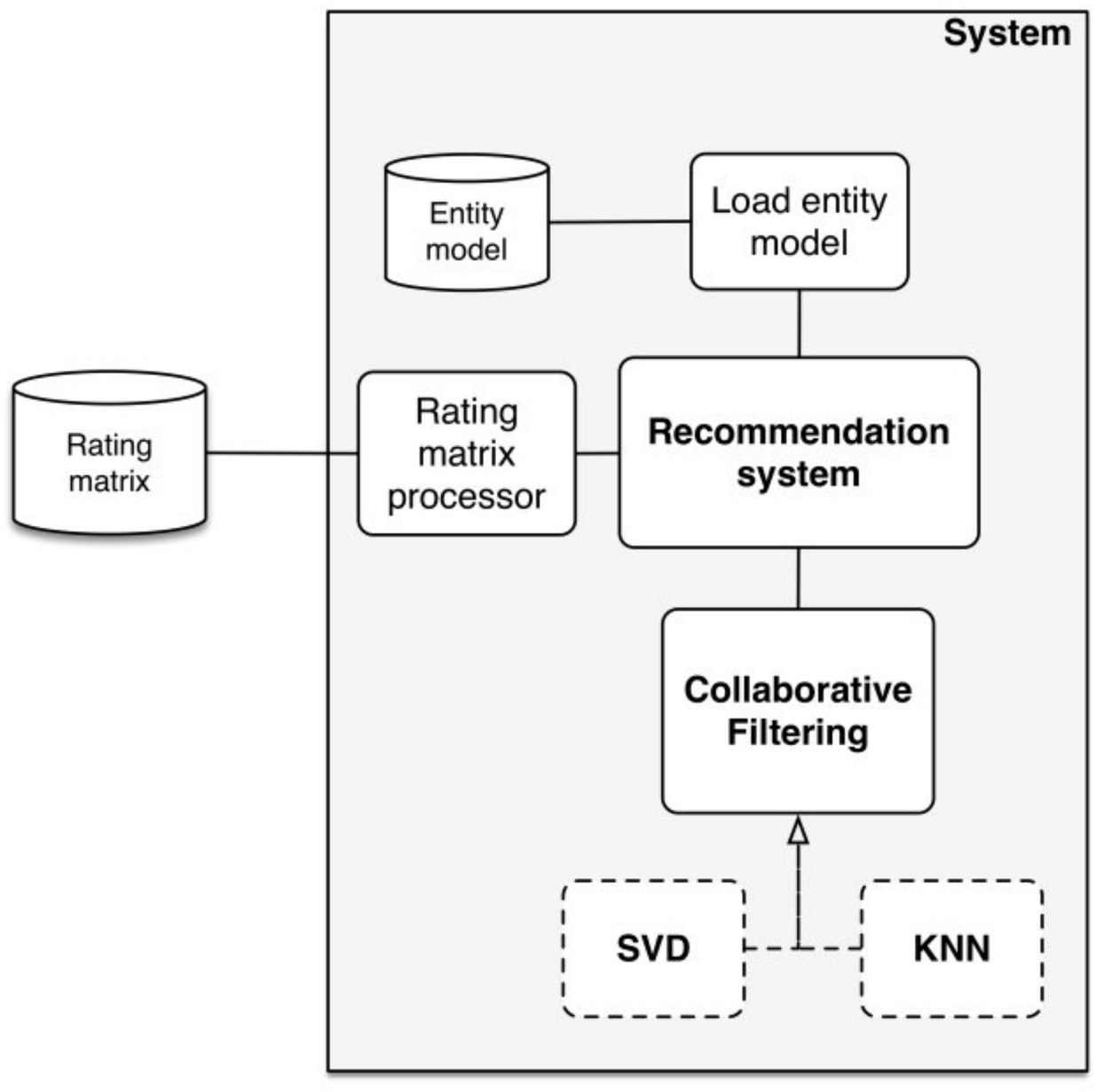


Figure 4.5: Recommendation system architecture.

Chapter 5

Identifying the user and his entities

As the name implies, the Entity Identifier component, mentioned in Chapter 4, 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 a 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. 5.1 illustrates how the entities are obtained. The three components developed were:

- Movie description Getter: downloads the synopsis and keywords of each movie;
- Entity identifier: splits the user into a set of entities, using only the watched movies and their descriptions.
- Rating matrix processor: parses the rating matrix in a database to be able to inform both the Movie Description Getter and the SOM Entity Identifier, which movies exist on the database and which users watched each movie.

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 contained in the entity model can be used to produce an entity visual map, as shown in Chapter 4, with the identified user entities, a table presenting the most relevant terms in the movie descriptions and the associated movies to each entity. Finally, it can be used in the Recommendation System (RS) context to enhance its performance.

On the following sections the Movie description getter and the Entity identifier will be explained in detail.

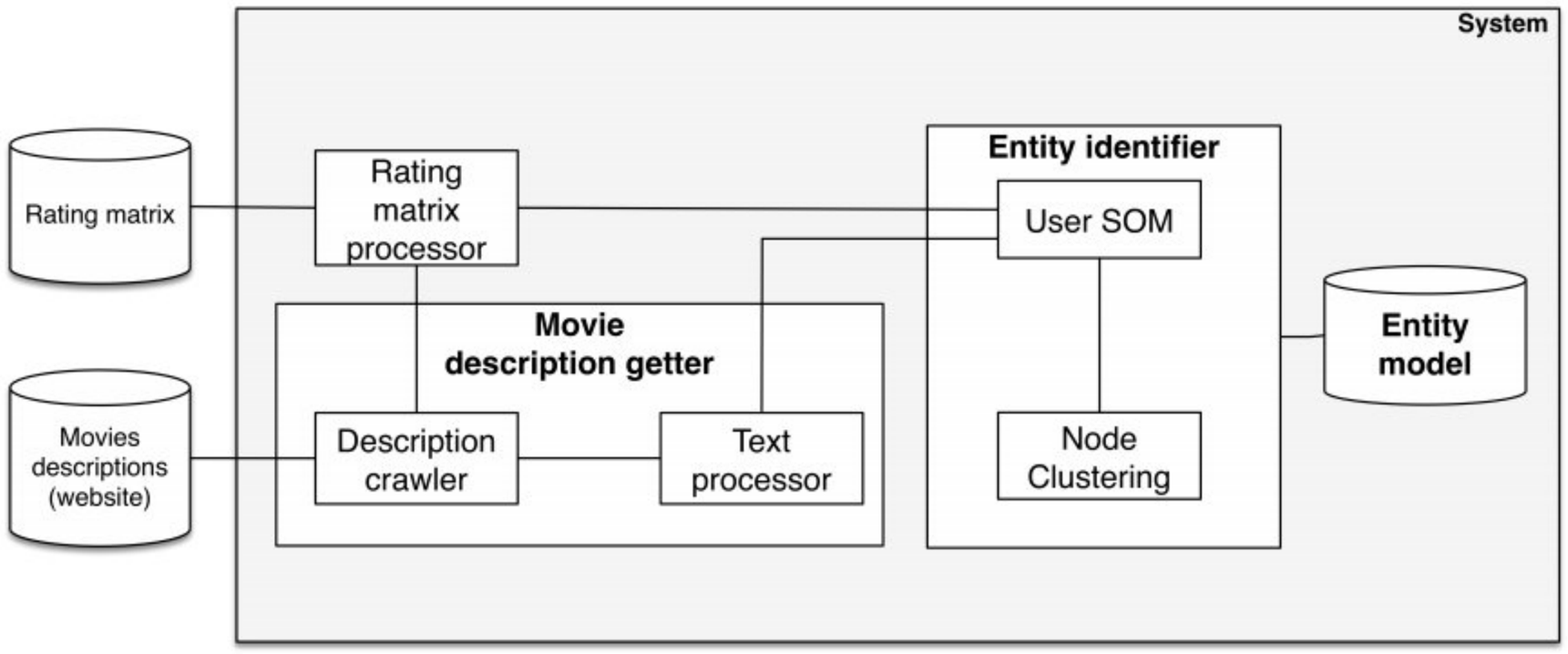


Figure 5.1: Architecture of our Entity Identifier.

5.1 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]. A crawl of the movie database was done and, afterwards, the movies descriptions obtained. The Movie description getter can then process the text, from each synopsis, and create TF-IDF (explained in Chapter 2) vector of the movie, allowing movies to be comparable between them.

5.1.1 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:

- Find the IMDB movie id;
- 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.

[1]IMDB website: `http://www.imdb.com`

Find the IMDB movie id

In order to reach the movie descriptions on IMDB, an IMDB id must be found. In our case, we used the list of movies from the MovieLens-100k dataset (ML)[2]. This dataset 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.

First, the movie title and the release date were used to perform a direct query to the IMDB website. The provided titles in the dataset had generally two formats, such as: "Toy Story (1995)" and "Flower of My Secret, The (Flor de mi secreto, La) (1995)". The first movie, has the original title in English, while the second movie has two titles, one in English and the other is the original title. If the movie contains both the english title and the original title, both are used to perform the query. As exemplified below, for the movie "Flower of My Secret, The (Flor de mi secreto, La)", released in 1995:

```
Query URL =  http://www.imdb.com/search/title?
                    title_type=feature&
                    release_date=1995,1995&
                    title=Flower+of+My+Secret%2C+The+
                    %28Flor+de+mi+secreto%2C+La%29
```

The parameter "title_type=feature" allows to filter for only movies, thus neglecting TV-Series among others.

If the first method fails to find the movie IMDB id, a second method is used. In this case, a similar query is made to the domain `akas.imdb.com`, using only the original movie title and the year of release. Also it requests that only movies are presented, using the parameters "s=tt&ttype=ft", and that the title is exactly equal to the one provided, using the parameter "title?exact=true". The query is exemplified below:

```
Query URL =   http://akas.imdb.com/search/
                    title?exact=true&s=tt\&ttype=ft&
                    title=La%20Flor%20de%20mi%20secreto&
                    release_date=1995,1995
```

Finally, if all previous attempts fail, the last method consists of performing a query to the google search engine, using a string composed of the URL for IMDB and term "/title" to enforce that where searching for a movie title. An example query is:

```
Query String = "imdb.com\" "title\"
                    "Flower of My Secret, The (Flor de mi secreto, La) 1995"
```

After submitting the query, the first result was obtained, hopefully containing the URL to the movie in the IMDB.

[2]Movielens datasets website: `http://grouplens.org/datasets/movielens/`

Obtaining the movie description from a movie

After obtaining the movies ids for the IMDB database, we used the library *IMDBpy*[3] 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.

5.1.2 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 [46] with a TF-IDF term weighting scheme [25]. We now explain how this was performed.

Turning the synopsis text into a term vector

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 5.2.

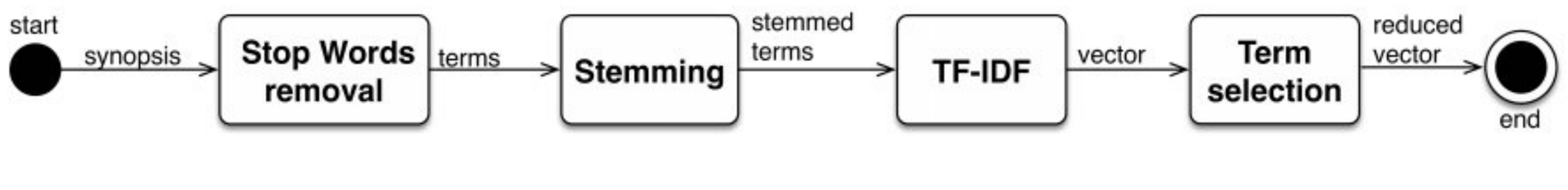


Figure 5.2: Pre-processing the synopsis to build a term vector.

First, as a standard procedure in text pre-processing, stopwords are removed [25]. Also standard is the conversion of terms to their stems. In our system, we used the stemming algorithm provided in the NLTK library[4]. With the remaining set of stems, we can now compute the TF-IDF weights, as explained in Chapter 2, and build the term vectors that represent the movies.

Since the number of terms is still considerably large, some of them will be dropped out, as shown in the following Section.

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 cannot 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.

[3] *IMDBpy* website: `http://imdbpy.sourceforge.net`

[4] NLTK website: `http://www.nltk.org`

We note that SVD could have been used instead, to reduce the number of features. However, we opted not to use it, since features would no longer be readable by humans, making not clear to users which terms caused the formation of the entities, when presenting the entity visual map.

5.2 SOM entity identifier

To find user entities, the movie description vectors are provided to the SOM, which finds groupings of similar movies. The User SOM component will be explained in Section 5.2.1, followed by the presentation of the Node clustering component in Section 5.2.2, responsible for grouping similar nodes into a entity.

5.2.1 User SOM

Each user has a SOM that represents his tastes. Figure 5.3 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.

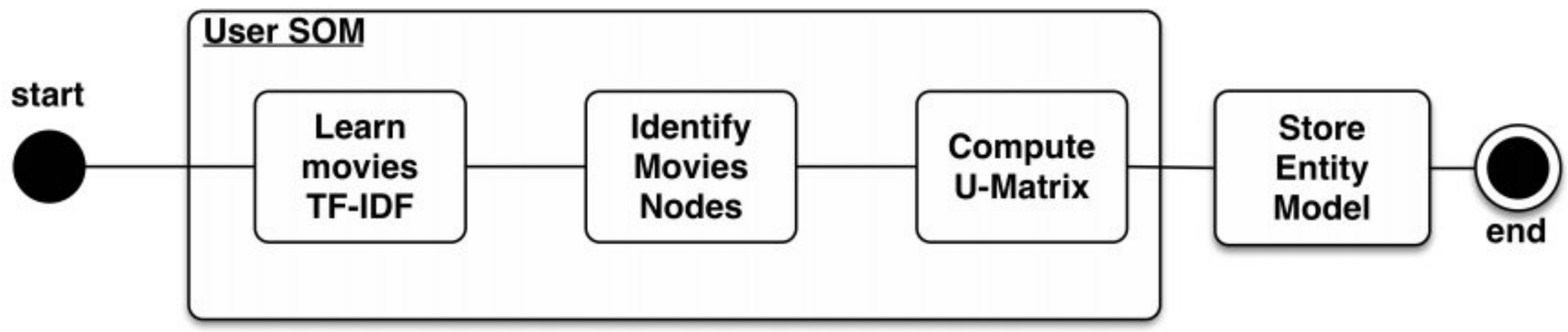


Figure 5.3: How to form a User SOM .

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

5.2.2 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. 5.1, which resulted on a binary matrix $U' = \{u'_{x,y}\}$. This is illustrated in Fig. 5.4.

$$u'_{x,y} = \begin{cases} 1, & \text{if } u_{x,y} > \text{ t} \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

Afterwards, a labelling algorithm, *connected-component analysis* [47], is applied, using the neighbourhood and resulting in the entity map,shown in Fig. 5.5. Each different label corresponds to a different identified entity.

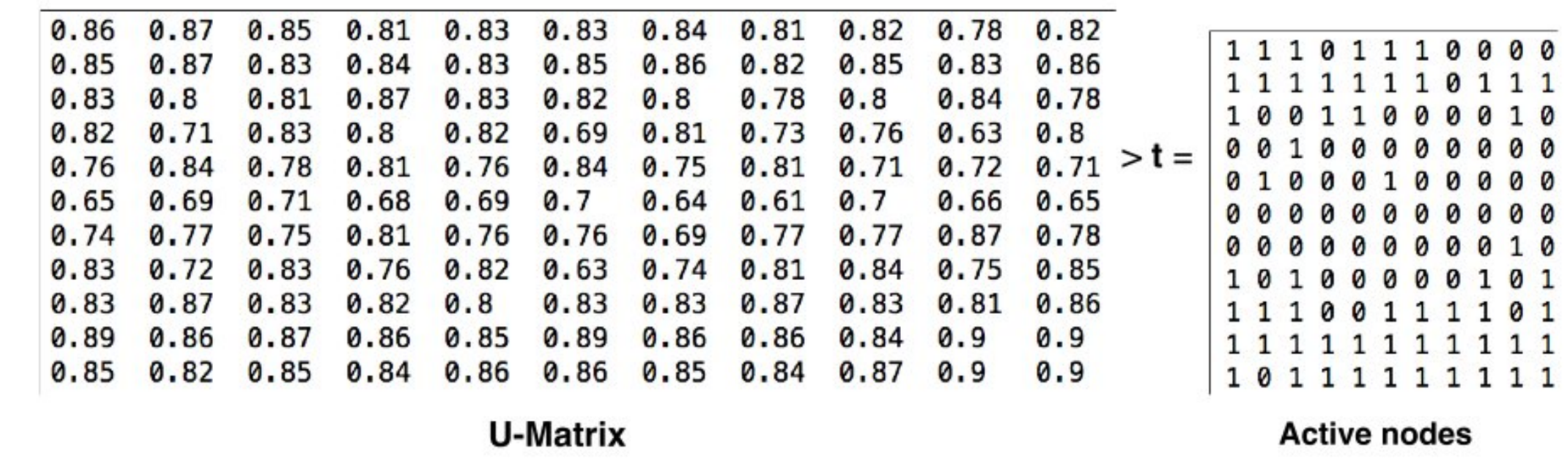


Figure 5.4: Identifying nodes that must be clustered.

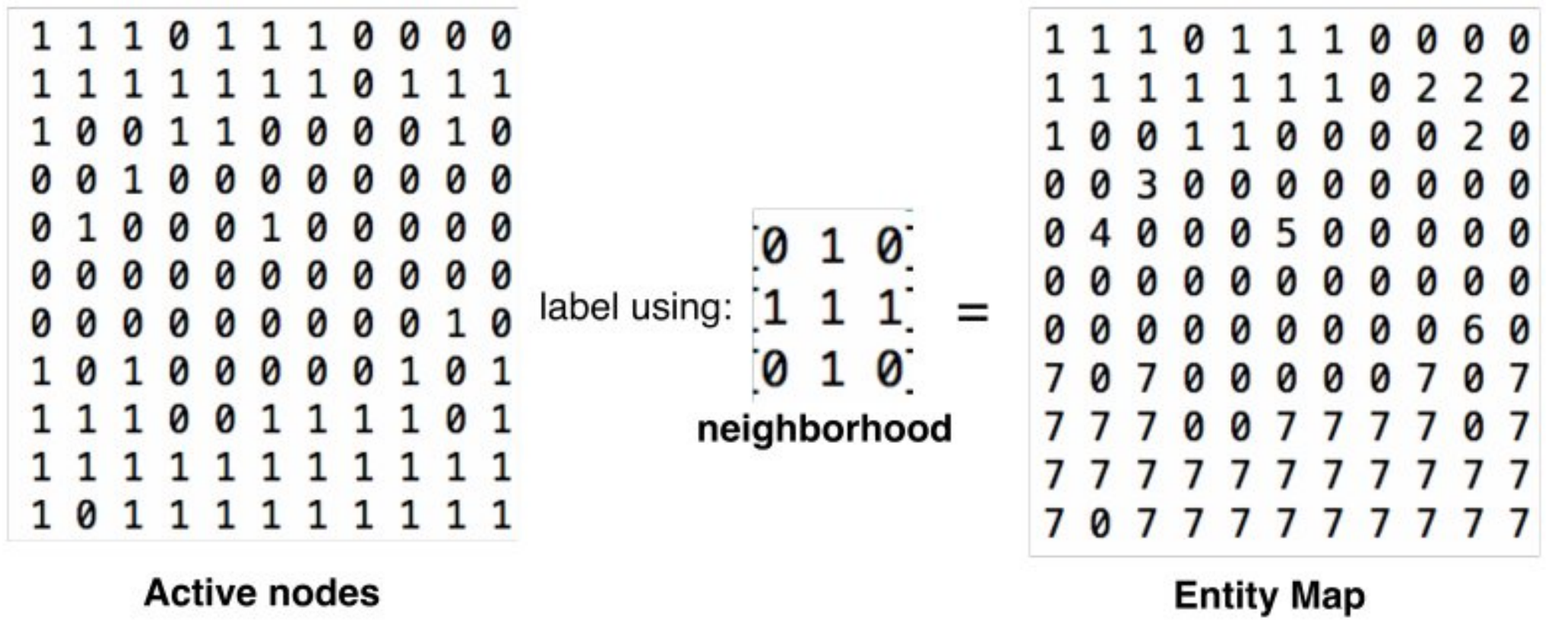


Figure 5.5: How entities are formed, using connected-component analysis.

5.2.3 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. With content, we are able to provide new information to the RS and present more details to the user regarding the formed entities. The discovered entities and their details can be presented in two different forms.

In the first, a list of entities is presented. For each entity the most relevant terms, plus the movies associated are shown. Figure 5.6 exemplifies one of the entities determined by our system.

The second form presents a more visual approach, using the U-Matrix extracted from the user SOM, an entity visual map, where all the user entities are represented and is possible to get an general overview of:

```
Relevant terms:
   Entity TF-IDF vector:     ship, alien, crew, planet, ...
   Sum movies TF-IDF vector: ship, crew, planet, signal, ...

   Movies List:
         1 - Star Trek: First Contact (1996)
         2 - Event Horizon (1997)
         3 - Star Trek III: The Search for Spock (1984)
         4 - Mars Attacks! (1996)
         5 - Under Siege (1992)
         6 - Contact (1997)
         7 - Alien (1979)
         8 - Star Trek IV: The Voyage Home (1986)
```

Figure 5.6: Example of the detailed information regarding one entity.

- how many entities exist;
- the entity tastes that each node represents;
- how many movies are associated to each node.

Figure 5.7 shows an example of an entity visual map. Each box corresponds to a node, containing an identifier, the 3 most relevant terms from the TF-IDF vector, and how many movies are associated to the entity. The different colours correspond to each of the entities.

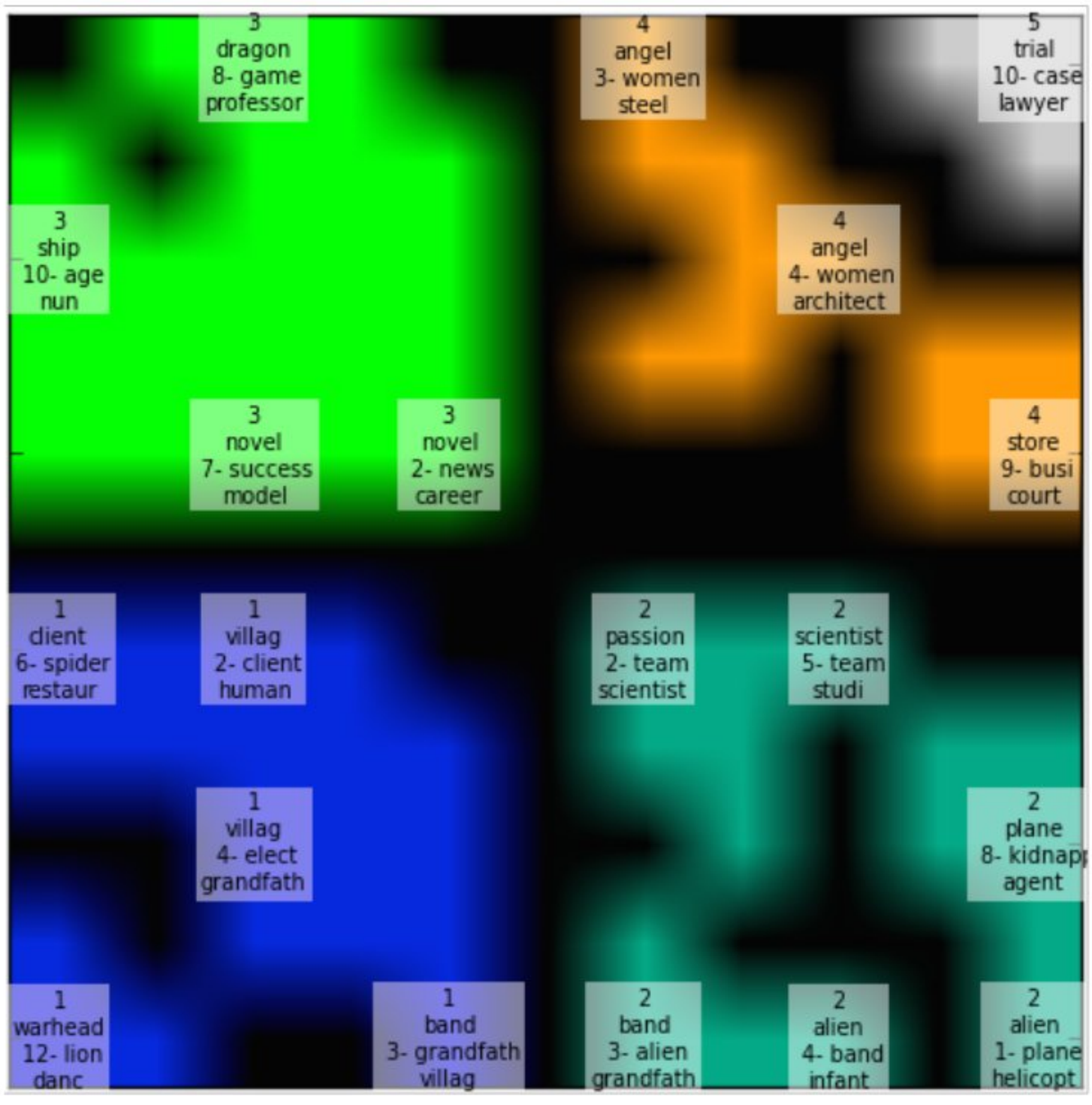


Figure 5.7: Example of an entity visual map.

Chapter 6

Movie recommendation system

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

SVD has been largely used in RS [21, 45, 48]. It presents several advantages, dimensionality reduction being the most significant one, which usually improves the computational cost, without loosing the quality of recommendations [21]. KNN [5, 37] is a more flexibly and modifiable algorithm, where we tried out several similarity metrics between users and entities [5].

In the context of KNN we have used two different similarity metrics between users and entities: Pearson correlation (Eq. 2.3) and a novel metric (Eq. 3.4) which combines the Mean Squared Difference and the Jaccard similarity [17]. After identifying the similarities between users, the Eq. 2.4 was used to predict a rating for a given pair user and movie.

In the following sections, each of these approaches, will be explained in more detail.

6.1 SVD as a Recommendation system

This recommendation method allows the decomposition of the rating matrix, enabling a reduction of dimensionality, which boosts the overall RS accuracy and improves the computational cost of traditional CF [21].

Figure 6.1 shows the architecture of our RS based on SVD. In order to use entities, the original user rating matrix must be modified.

The *Rating matrix processor* loads the dataset and the users can be replaced by their respective entities, which is done by the component *Entity rating matrix creation*. This generated the entity rating matrix, as illustrated in Fig. 4.2. After creating the

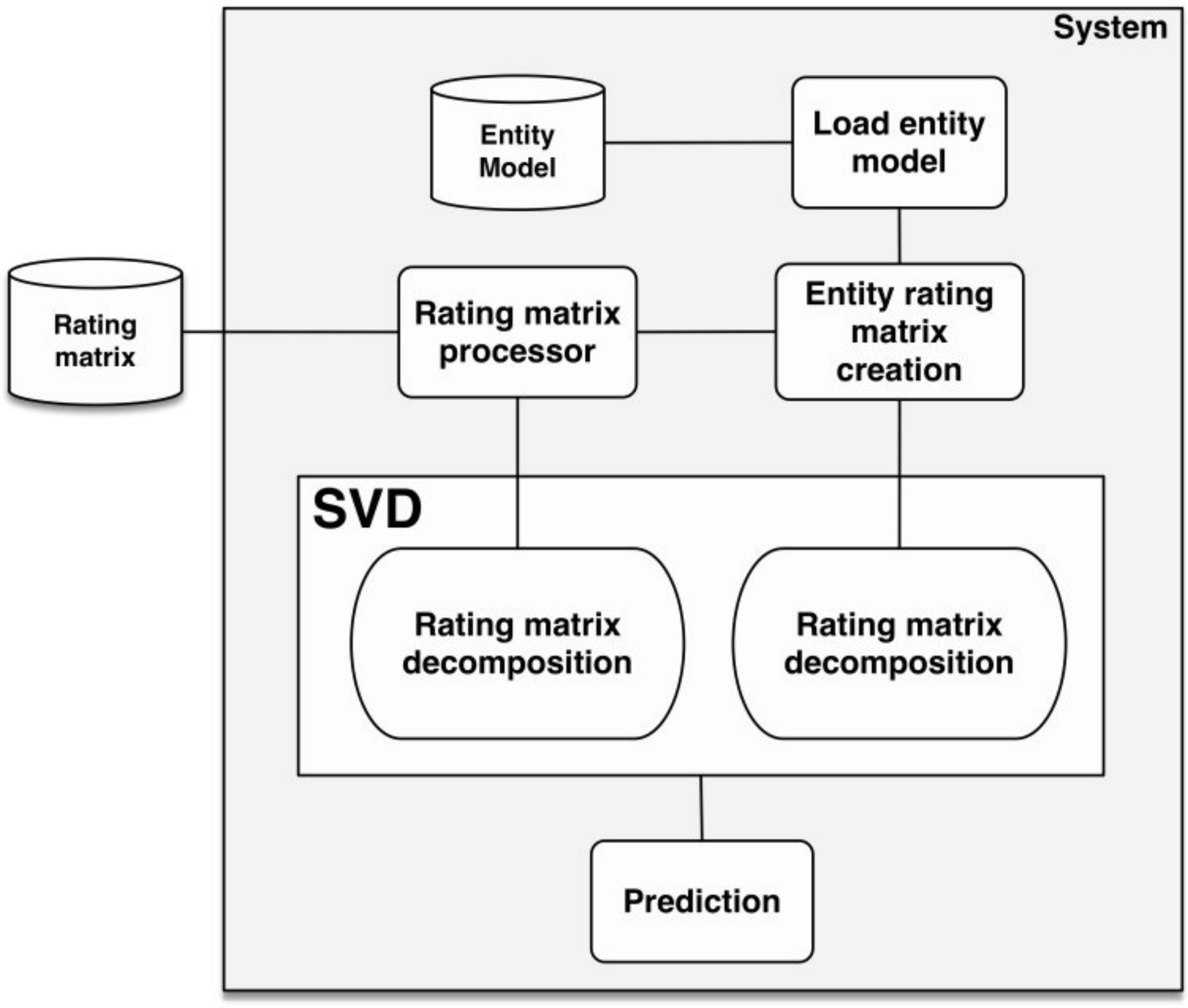


Figure 6.1: Architecture of a Recommendation System using SVD.

entity rating matrix, it is decomposed by the SVD technique by the component *Rating matrix decomposition*. By selecting a number of features to use, this will allow to perform a reduction of dimensionality.

Finally, the predictions are obtained, based on the number features, resorting to the Eq. 2.6, 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.

6.2 KNN as a Recommendation 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.

In Figure 6.2, a RS architecture using KNN is presented. First, the *Rating matrix processor* loads the dataset and builds the rating matrix, then *Entity rating matrix creation* knowing the user rating matrix and the entity based model, builds an entity rating matrix.

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. The two

similarity metrics are computed using KNN (Fig. 6.2), by the *Compute user similarities* and the *Compute entity similarities* components, respectively, allowing to identify their nearest neighbours.

We can, afterwards, compute predictions for movies not rated by users, using the previously computed similarities in the Eq. 2.4.

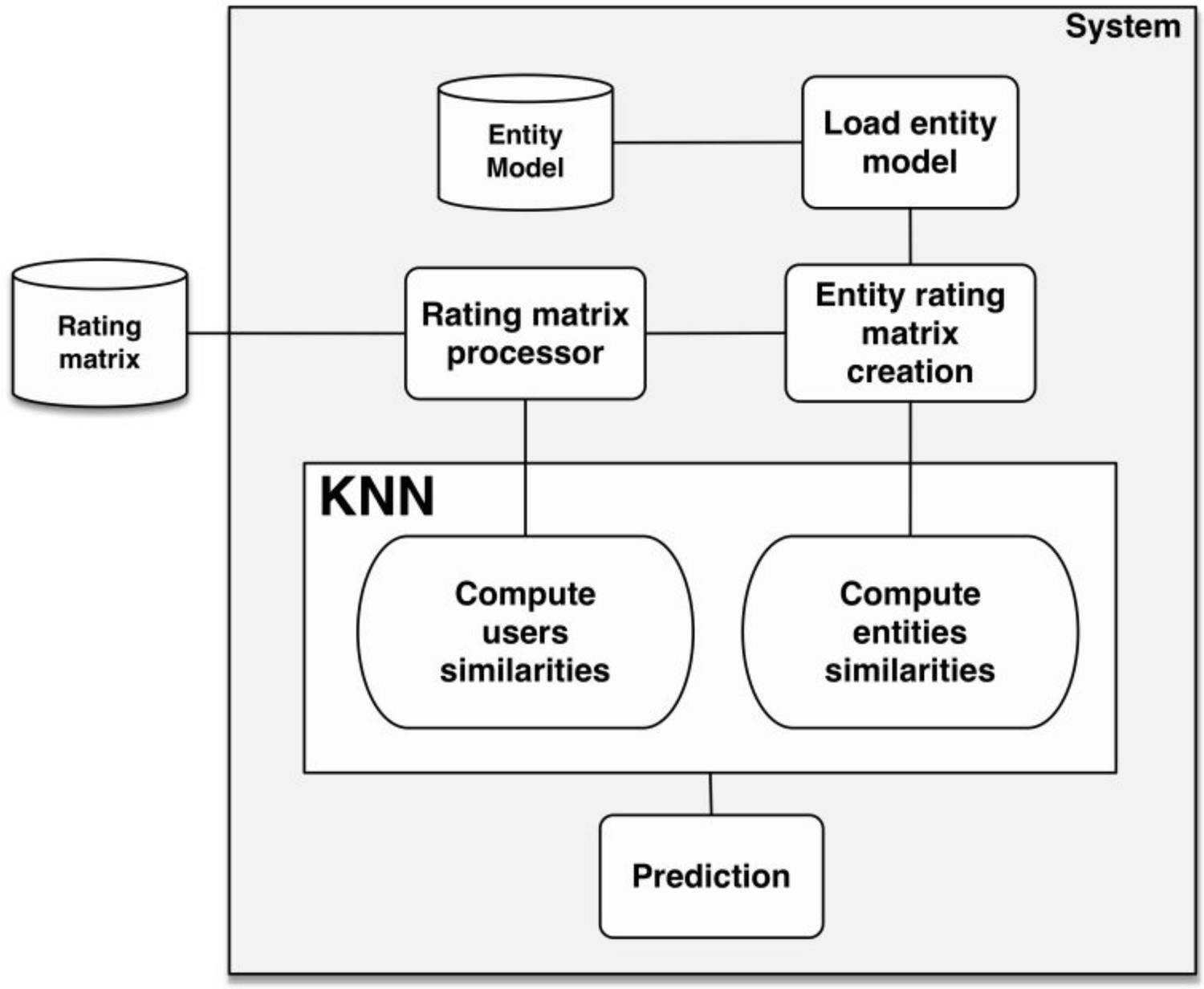


Figure 6.2: Architecture of a Recommendation System using KNN.

6.2.1 Prediction

When predicting ratings in the context of CF, one possible approach to this problem is Eq. 2.4, 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 6.1 presents this modification, where $\bar{r}_e$ represents the 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.

$$p_{u,i} = \bar{r}_e + \mu_{u,i} \sum_{n \in G_{u,i}} sim(u,j)(r_{j,i} - \bar{r}_j) \tag{6.1}$$

As before, the value μ is a normalizing factor:

$$\mu_{u,i} = \frac{1}{\sum_{n \in G_{u,i}} sim(u,j)} \iff G_{u,i} \neq \varnothing \tag{6.2}$$

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.

6.2.2 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. 6.3.

$$sim(u,j) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u) \times (r_{j,i} - \bar{r}_j) \times sim_{entities}(u,j,i)}{\sqrt{\sum_{i \in I}(U_{u,i} - \bar{r}_u)^2} \times \sqrt{\sum_{i \in I}(r_{j,i} - \bar{r}_j)^2}} \tag{6.3}$$

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. 6.3.

6.2.3 Weighted similarity between users and entities

This similarity has the purpose of combining user and entity similarities, Eq. 6.4 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.

$$sim(u,j,i) = (1-\alpha) \times sim_{users}(u,j) + \alpha \times sim_{entities}(u,j,i) \tag{6.4}$$

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

Note that this similarity also depends on the movie i being rated. Thus, the prediction equation must also include this difference as presented in Eq. 6.5.

$$p_{u,i} = \bar{r}_u + \mu_{u,i} \sum_{n \in G_{u,i}} sim(u,j,i)(r_{j,i} - \bar{r}_j) \tag{6.5}$$

Chapter 7

Evaluation and results

The developed RS is a complex system. Several variables must be tuned to produce good predictions. Most of our effort during the development of this work was experimenting new solutions to enhance the system performance in rating prediction, which is the step before the recommendation.

In this Chapter, we will present our procedure, the used settings and their impact over the RS overall performance.

7.1 Protocol

To build the entity identifier the rating matrix is obtained from the MovieLens-100k dataset (ML) dataset 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 determine the quality and accuracy of rating predictions we use [13, 49]: Mean Average Error (MAE) (Eq. 7.1) and Perfect Hit (PHIT) (Eq. 7.2). 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.

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{u,i} - r| \tag{7.1}$$

$$PHIT = \frac{\sum_{(u,i,r) \in T} (round(p_{u,i}) - r) == 0}{|T|} \tag{7.2}$$

In the Eq. 7.1 and Eq. 7.2, 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 [50], 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.

The evaluation of the RS using KNN is a computational requiring task. For that reason we decided to pre-compute both user and entity similarities for each of the train sets and test sets. This way when tuning, for example the number of used neighbours, the similarities were only computed once for all the tests.

7.2 Evaluated parameters

7.2.1 Entity identifier

The movie description getter module was evaluated by observing the amount of movies ids that each of the methods developed in the Description Crawler were found and also present the amount of movies that were able to extract the synopsis and keywords. To evaluate the pre-processing of text to obtain a TF-IDF vector, will be presented the number of terms present in the bag of words, before and after the pre-processing also a histogram with the absolute frequency of terms in TF-IDF vectors.

To evaluate the entity identifier SOM, an unsupervised machine-learning algorithm, we configured it initially using empirical testing by evaluating the formed entities. The variables tuned included the output nodes map dimension, or the threshold defined to limit the amount of words present in the bag of words, as example.

Regarding the entity clustering module, since it was implemented before the RS, we decided to try tuning the entity identifier in order to try maximize the RS performance only when configuring RS. Since after concluding the implementation of the RS, we tuned more accurately the system. Nevertheless, it is a time consuming task and there is still room for improvements in the produced entity models that might enhance the RS performance.

7.2.2 Recommendation system

The RS created has two different CF algorithm, one using SVD and the other KNN. When using the RS with the SVD, as a CF algorithm, only the entity rating matrix was used. On the other hand, when using the KNN both user and entity rating matrices were used to compute the user and entity similarity, respectively.

Regarding the SVD solution, we combined various entity models (see Fig. 6.1) and compared them. The models are generated by the entity identifier module, to compare between six different models tested. One of them includes the rating matrix as movie description, instead of using either the synopsis or the keywords.

Our first goal is to optimize the number of used features in the SVD and discuss the impact on the RS performance, for each entity model. A baseline will be computed, which is the original user rating matrix with ratings, without using any of the information provided by the entity model representation of users.

To evaluate the RS using KNN, we only tested the entity model using synopsis. We decided to try several similarity metrics and compare their impact in terms of RS prediction quality. Also compared the influence of the average user rating on the prediction equation , by replacing it with the entity average rating. Afterwards, we compared the impact of user and entity similarity, by only using different similarity metrics and leaving all the other parameters the same.

Finally, two tables present a summary of the results obtained in the several RSs developed. The first regarding the RS using SVD and the second regarding the RS using KNN. Using only the Entity model based on synopsis or keywords.

7.3 Results

7.3.1 Entity Identifier

To evaluate the Description crawler, in Table 7.1, we present how many movies have a valid URL to obtain the IMDB id, and also the amount of movies crawled using the methods that we developed (see Section 5.1). In addition, we present the number of movies that was not possible to obtain the IMDB id. We can see that if we did not use the methods, only 61% of the movies would have descriptions available. Only one movie was not identified, because the dataset did not provided any content information regarding it.

Method	number of ID obtained	Increased %	Accumulated %
valid URL	1025	61%	61%
1	412	24%	85%
2	131	8%	93%
3	113	7%	99.95%
not available	1	$< 0.05\%$	100%

Table 7.1: Evaluation of methods to find the movies IMDB id.

In order to evaluate the Text Processor, we needed to extract the synopsis or the keywords. The number of movies that we obtained both synopsis and keywords is shown in Table 7.2. We also present how many movies ended up without words, caused by the text pre-processing performed.

With descriptions	Nº of movies	Nº of movies (after preprocessing)
Containing synopsis	1652	1618
Containing keywords	1679	1575

Table 7.2: Number of movies with description.

Regarding the TF-IDF vector that describes each movie, we also present an histogram with the number of words that describes each movie using synopsis (Fig. 7.1). It is clear that most of the movies did not have a considerable amount of words, a consequence of the stopwords removal, stemming and the selection of stemmed words we opted to keep only words with a TF-IDF value between 3 and 7.5.

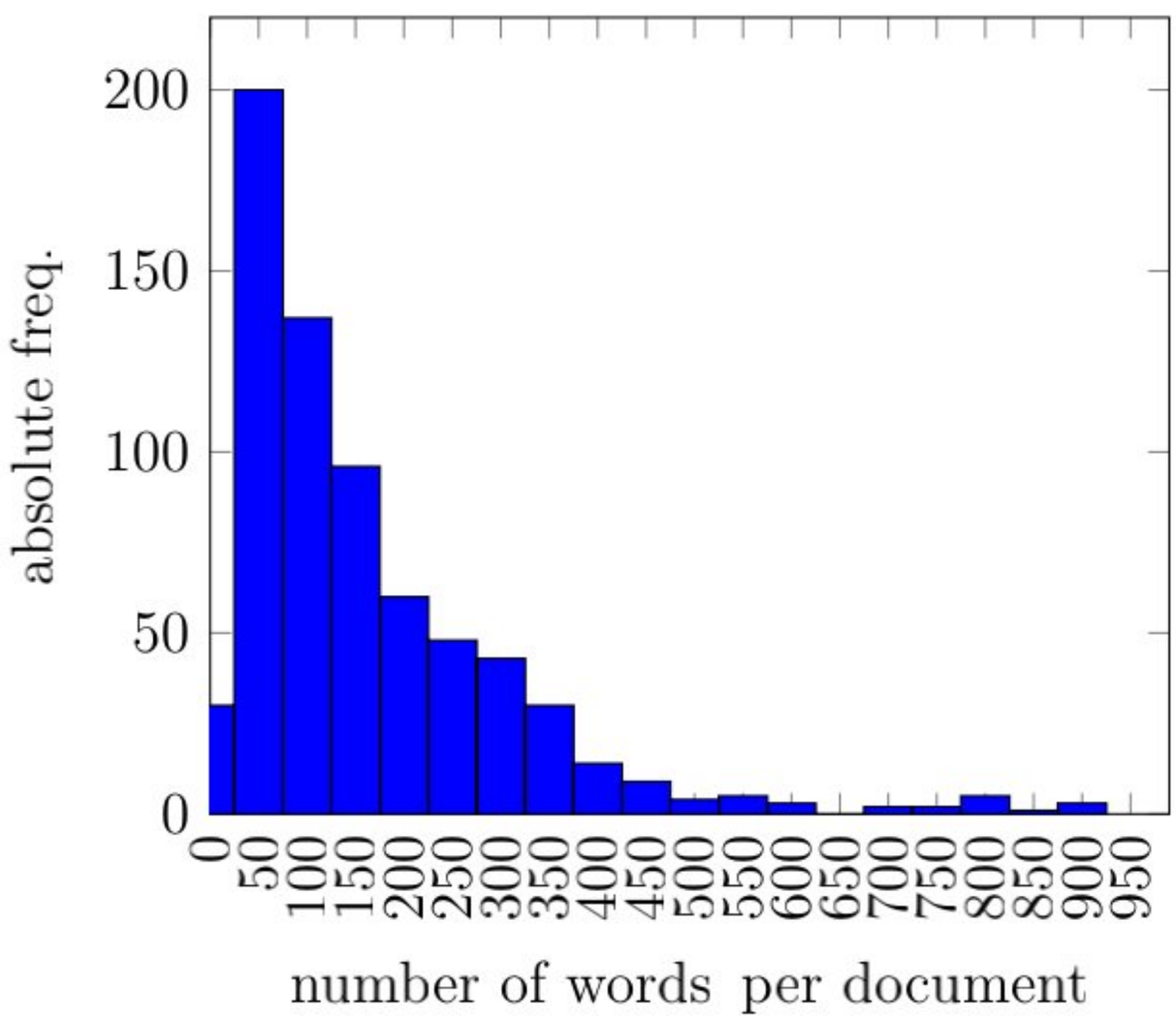


Figure 7.1: Histogram of number of words, per document.

In Table 7.3 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 an empty $TF - IDF$ vector, which is crucial to identify the entities.

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

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

Table 7.3: Number of words in a bag of words.

SOM we used the Orange python library[1]

Figure 7.2 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.

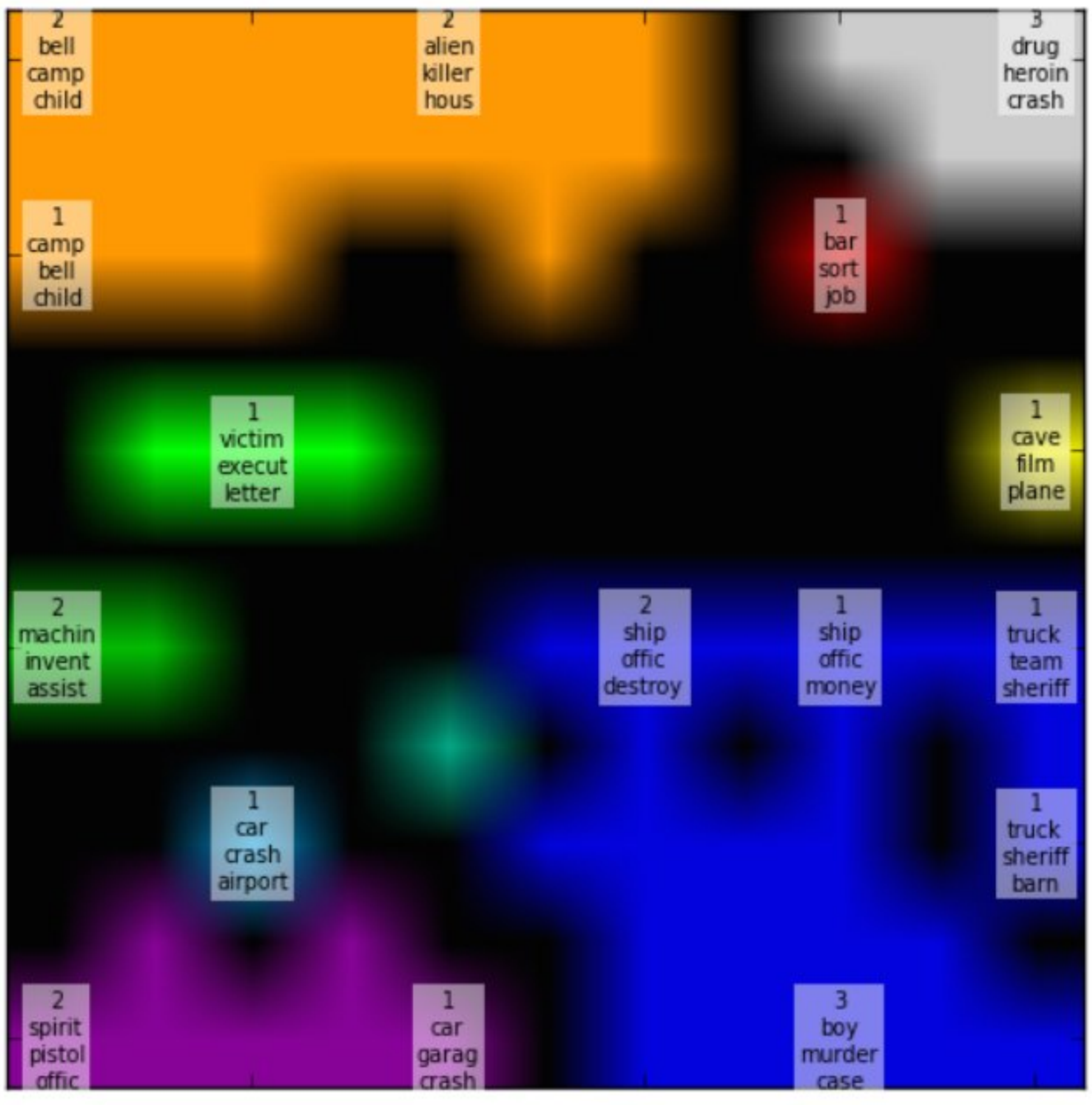


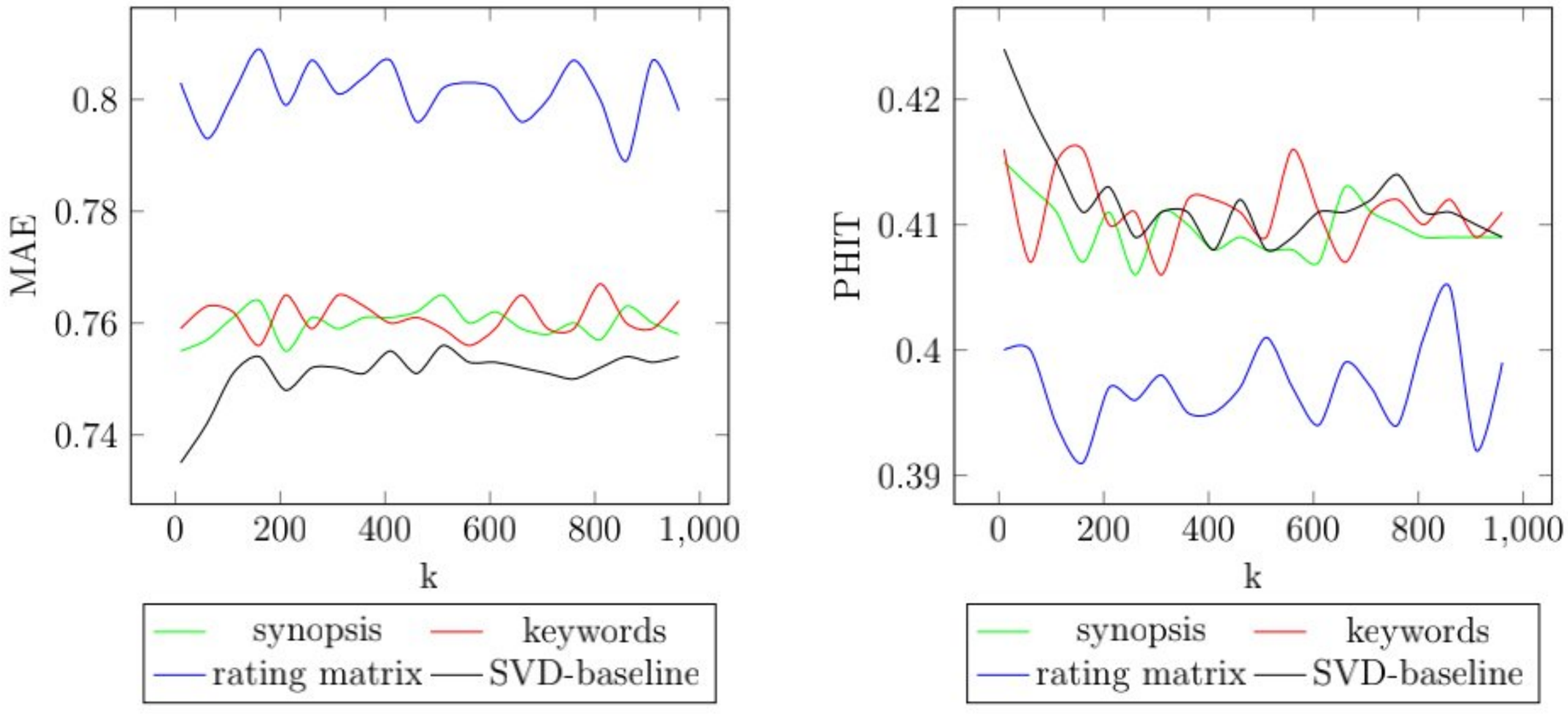
Figure 7.2: Entity visual map.

7.3.2 Recommendation system using SVD

Regarding the RS using SVD, Fig. 7.3 shows the impact of the number of features (k) used. In this figure, we did not perfom the clustering of nodes using a threshold.

[1]Orange python library : `https://pypi.python.org/pypi/Orange`.

One unexpected result concerns the entity model using keywords. In fact, the entity model using keywords, outperforms the one using synopsis. We also tried to cluster the movies using the rating matrix instead of using the synopsis and keywords, yet it did not yield good accuracy results. The performance was, in matter of fact, a bit worst when compared to the other two, probably due to the absence of the information provided by the movie description, via the formed entities. Finally, the PHIT of both synopsis and keywords slightly outperforms the baseline.



(a) MAE

(b) PHIT

Figure 7.3: SVD performance, when using the nodes.

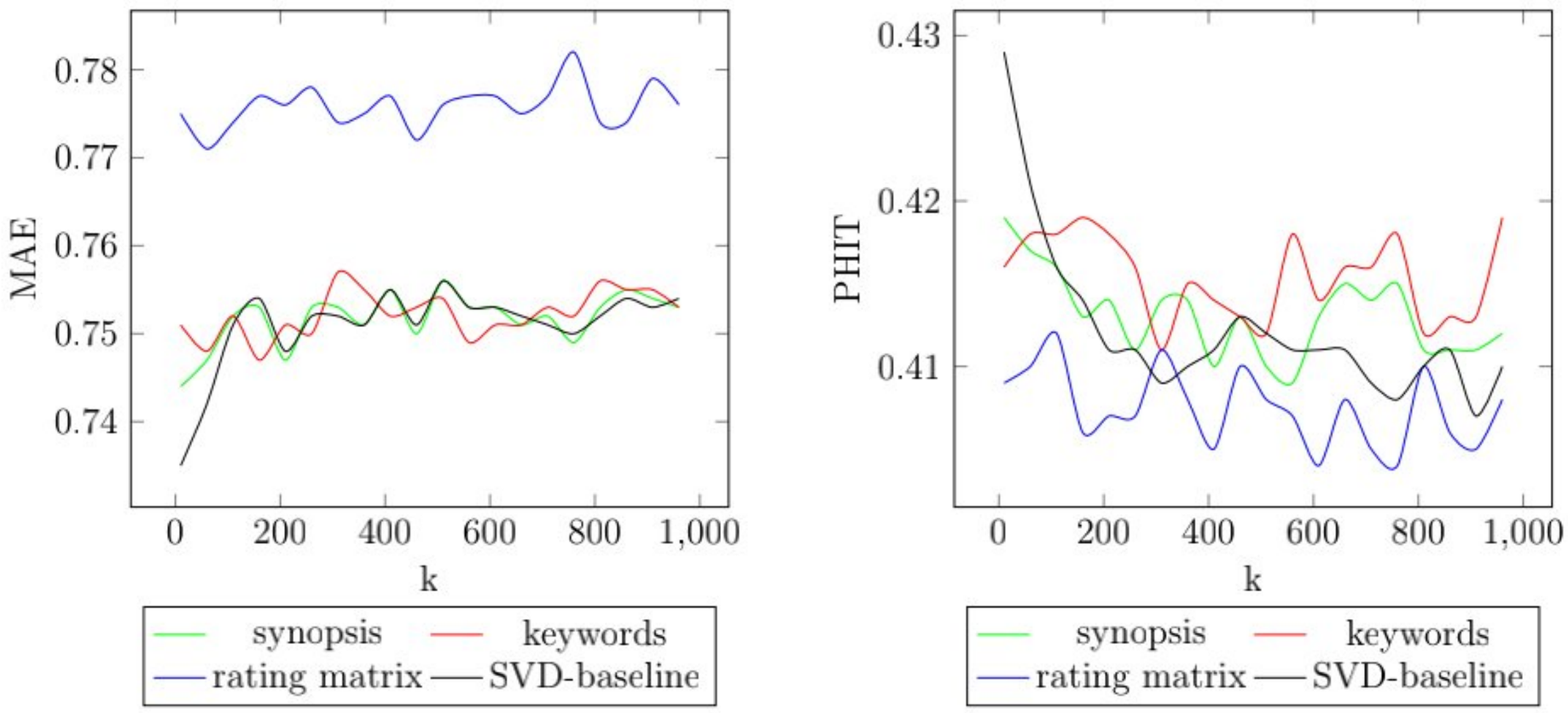
In Fig. 7.4, the RS uses the entities, which have been clustered using the Node clustering module. In this case, the performance in MAE and PHIT increases, in contrast to Fig. 7.3, this occurs because the entities are less specific than the nodes.

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 and PHIT metrics.

7.3.3 Recommendation system using KNN

To evaluate the RS using KNN, we began by comparing the performance of the developed metrics: modified Pearson correlation (Eq. 6.3) and Weighted similarity between users and entities (Eq. 6.4). 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. 7.5a) and PHIT (Fig. 7.5b). Note that, for $\alpha = 0$ only the users are going

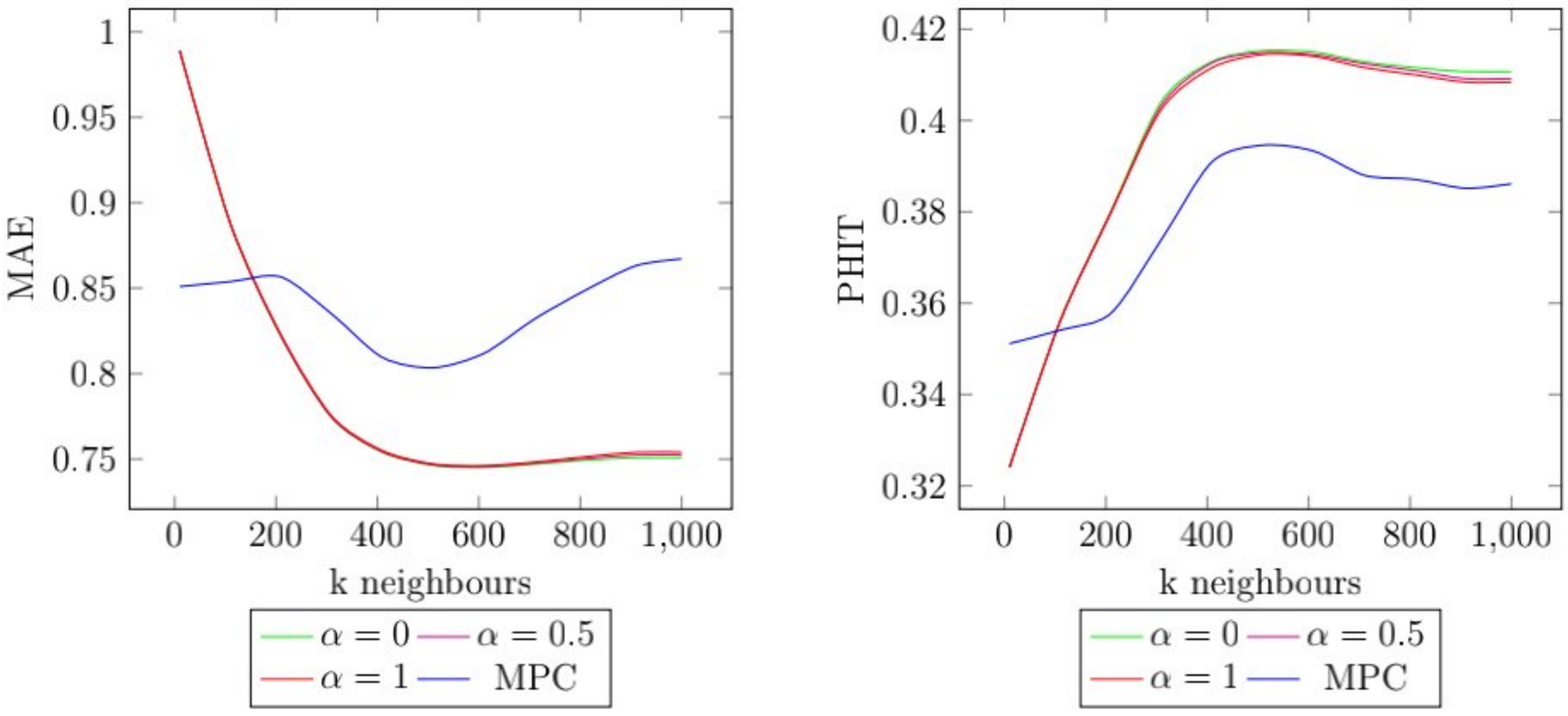


(a) MAE (b) PHIT

Figure 7.4: SVD performance, when using the entities.

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. 7.5.



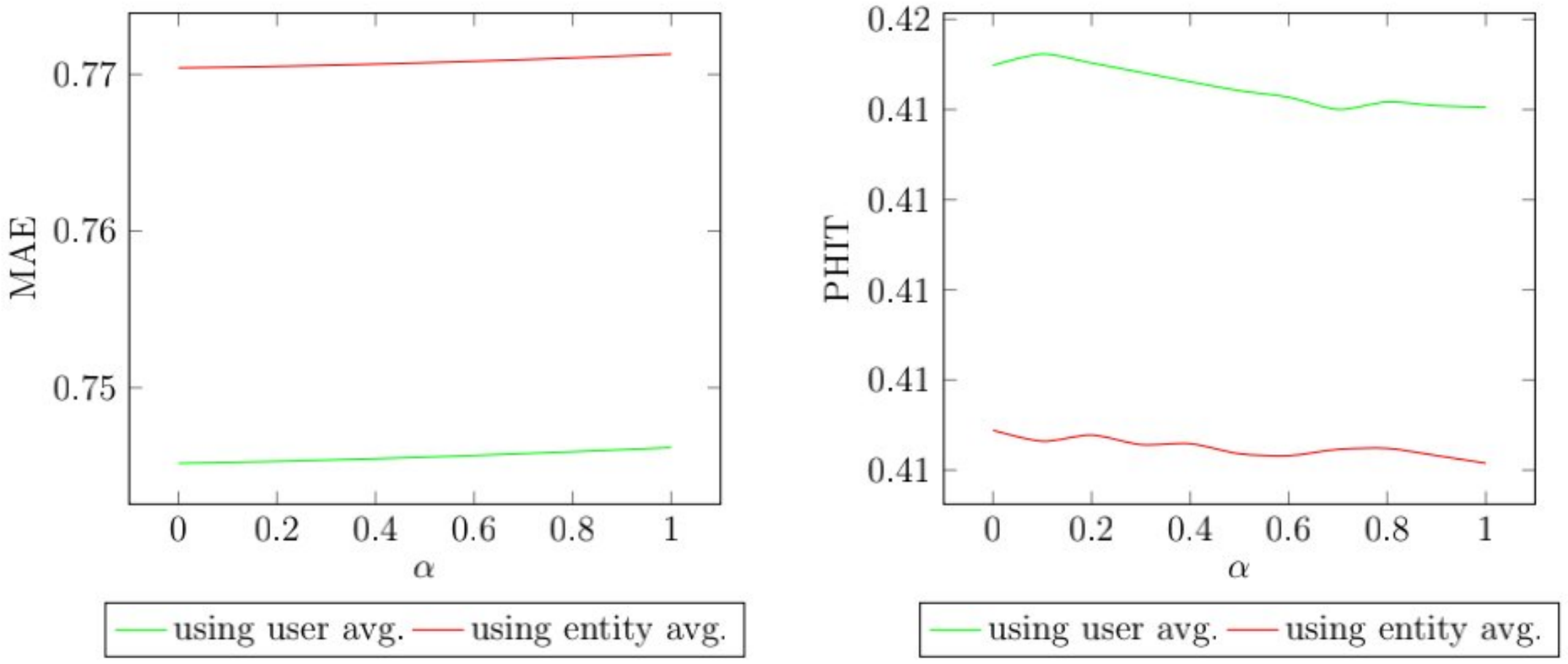
(a) MAE (b) PHIT

Figure 7.5: KNN performance, when tuning the number of neighbours.

Figure 7.6 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 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 the 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.



(a) MAE

(b) PHIT

Figure 7.6: KNN performance, when tuning the coefficient α.

To conclude the evaluation, we compared the impact of using different similarity metrics. To compare the user and entity similarities, we combined the Pearson correlation and the novel metric. The results are shown in Table 7.4. The number of neighbours used was 500 and $\alpha = 0.1$, motived by the results obtained previously. We concluded that the best combination of the similarity metrics used: novel metric as user similarity metric and the Pearson correlation as entity similarity to minimize the MAE and use the Pearson correlation for both user and entity similarity to maximize the PHIT.

user similarity metric	entity similarity metric	MAE	PHIT
Pearson	Pearson	0.7463	0.4152
Pearson	novel	**0.7462**	**0.4154**
novel	Pearson	0.7480	0.4118
novel	novel	0.7464	0.4130

Table 7.4: Impact of using different similarity metrics for users and entities.

7.3.4 Overall RS results

The RS proposed in the work, combined several techniques and to conclude this Chapter. We present two Tables that summarize all the obtained RSs and their best registered results regarding MAE and PHIT.

Table 7.5 presents the performance of the entity models when combined with the SVD to create a RS. The results are compared with a baseline, which is the RS using SVD applied to the original rating matrix, provided by the MovieLens-100k dataset (ML).

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 7.5: 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 7.6. In this evaluation, we configure the RS parameters based on the entity model using synopsis which was the only one evaluated in Section 7.3.3, for entity models based on synopsis and keywords. We did not include the rating matrix because of the obtained results in Table 7.5.

The results presented in Table 7.6, show that both Entity models performed better than the baseline, yet the improvements were marginal. The best accuracy performance, accomplished by the Entity model based on the synopsis was obtained when using WS with α equal to 0.1. The users similarity was computed with pearson correlation and the novel metric (Eq. 3.4) was used to compute entities similarity. As for the prediction the average was obtained using the user average rating.

The best accuracy performance, accomplished by the Entity model based on the keywords, had the best MAE when using WS with α equal to 0.1. The users and

Entity Model	similarity metric			predition avg. based on	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 7.6: Evaluation results for the RS using KNN, using $k = 500$, with different Entity models.

entities similarities both with the novel metric. As for the $PHIT$ the best value was obtained using the WS with α equal to 0.1, with pearson correlation to compute the users similarity and the novel metric to compute the entities similarity.

Chapter 8

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.

Bibliography

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowledge-Based Systems*, vol. 46, no. 0, pp. 109 – 132, 2013.

[2] C.-F. Tsai and C. Hung, "Cluster ensembles in collaborative filtering recommendation," *Applied Soft Computing*, vol. 12, no. 4, pp. 1417 – 1425, 2012.

[3] D. Bogdanov, M. Haro, F. Fuhrmann, A. Xambó, E. Gómez, and P. Herrera, "Semantic audio content-based music recommendation and visualization based on user preference examples," *Information Processing and Management*, vol. 49, no. 1, pp. 13 – 33, 2013.

[4] 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*, vol. 17, no. 6, pp. 734–749, 2005.

[5] G. Song, S. Sun, and W. Fan, "Applying user interest on item-based recommender system," in *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*, pp. 635–638, 2012.

[6] S. Gabrielsson and S. Gabrielsson, "The use of Self-Organizing Maps in Recommender Systems." Uppsala University, Department of Information Technology, 2006.

[7] M. Mandl, A. Felfernig, E. Teppan, and M. Schubert, "Consumer decision making in knowledge-based recommendation," *Journal of Intelligent Information Systems*, vol. 37, no. 1, pp. 1–22, 2011.

[8] M. Lee, P. Choi, and Y. Woo, "A hybrid recommender system combining collaborative filtering with neural network," in *Adaptive Hypermedia and Adaptive Web-Based Systems* (P. Bra, P. Brusilovsky, and R. Conejo, eds.), vol. 2347 of *Lecture Notes in Computer Science*, pp. 531–534, Springer Berlin Heidelberg, 2002.

[9] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[10] 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*, pp. 1–7, 2010.

[11] 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*, pp. 3580–3585, Oct 2011.

[12] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, pp. 61–70, Dec. 1992.

[13] J. Bobadilla, F. Serradilla, and A. Hernando, "Collaborative filtering adapted to recommender systems of e-learning," *Knowledge-Based Systems*, vol. 22, no. 4, pp. 261 – 265, 2009. Artificial Intelligence (AI) in Blended Learning (AI) in Blended Learning.

[14] X. Zhou, Y. Xu, Y. Li, A. Josang, and C. Cox, "The state-of-the-art in personalized recommender systems for social networking," *Artificial Intelligence Review*, vol. 37, no. 2, pp. 119–132, 2012.

[15] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. application to the movies domain," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10990 – 11000, 2012.

[16] M. Dell'Amico and L. Capra, "Sofia: Social filtering for robust recommendations," in *Trust Management II* (Y. Karabulut, J. Mitchell, P. Herrmann, and C. Jensen, eds.), vol. 263 of *IFIP – The International Federation for Information Processing*, pp. 135–150, Springer US, 2008.

[17] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowledge-Based Systems*, vol. 23, no. 6, pp. 520 – 528, 2010.

[18] L. Candillier, F. Meyer, and M. Boullé, "Comparing state-of-the-art collaborative filtering systems," in *Machine Learning and Data Mining in Pattern Recognition* (P. Perner, ed.), vol. 4571 of *Lecture Notes in Computer Science*, pp. 548–562, Springer Berlin Heidelberg, 2007.

[19] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Moviexplain: a recommender system with explanations," in *Proceedings of the third ACM conference on Recommender systems*, RecSys '09, (New York, NY, USA), pp. 317–320, ACM, 2009.

[20] U. Shardanand and P. Maes, "Social information filtering: Algorithms for automating ;word of mouth;," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, (New York, NY, USA), pp. 210–217, ACM Press/Addison-Wesley Publishing Co., 1995.

[21] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system – a case study," in *IN ACM WEBKDD WORKSHOP*, 2000.

[22] D. Kalman, "A singularly valuable decomposition: The svd of a matrix," *College Math Journal*, vol. 27, pp. 2–23, 1996.

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

[24] S. Lahabar and P. Narayanan, "Singular value decomposition on gpu using cuda," in *Parallel Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pp. 1–10, May 2009.

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

[26] N. Bhamidipati and S. Pal, "Stemming via distribution-based word segregation for classification and retrieval," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, pp. 350–360, April 2007.

[27] R. Li and X. Guo, "An improved algorithm to term weighting in text classification," in *Multimedia Technology (ICMT), 2010 International Conference on*, pp. 1–3, Oct 2010.

[28] W. Carrer-Neto, M. L. Hernández-Alcaraz, R. Valencia-García, and F. García-Sánchez, "Social knowledge-based recommender system. application to the movies domain," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10990 – 11000, 2012.

[29] P. C. Vaz, D. M. de Matos, B. Martins, and P. Calado, "Improving an hybrid literary book recommendation system through author ranking," *CoRR*, vol. abs/1203.5324, 2012.

[30] N. Manouselis, H. Drachsler, K. Verbert, and E. Duval, "Recommender systems for learning," 2012.

[31] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[32] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 5–53, Jan. 2004.

[33] A. A. Kardan and M. Ebrahimi, "A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups," *Information Sciences*, vol. 219, no. 0, pp. 93 – 110, 2013.

[34] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Eighteenth National Conference on Artificial Intelligence*, (Menlo Park, CA, USA), pp. 187–192, American Association for Artificial Intelligence, 2002.

[35] B. M. Kim, Q. Li, C. S. Park, S. G. Kim, and J. Y. Kim, "A new approach for combining content-based and collaborative filters," *J. Intell. Inf. Syst.*, vol. 27, pp. 79–91, July 2006.

[36] A. Eckhardt, "Similarity of users' (content-based) preference models for collaborative filtering in few ratings scenario," *Expert Systems with Applications*, vol. 39, no. 14, pp. 11511 – 11516, 2012.

[37] J. Bobadilla, F. Ortega, A. Hernando, and G. G. de Rivera, "A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors algorithm," *Knowledge-Based Systems*, vol. 51, no. 0, pp. 27 – 34, 2013.

[38] Netflix, "Netflix prize," Jan. 2009.

[39] T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," in *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, (Washington, DC, USA), pp. 625–628, IEEE Computer Society, 2005.

[40] J. J. Jung, "Attribute selection-based recommendation framework for short-head user group: An empirical study by movielens and {IMDB}," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4049 – 4054, 2012.

[41] S.-M. Choi, S.-K. Ko, and Y.-S. Han, "A movie recommendation algorithm based on genre correlations," *Expert Systems with Applications*, vol. 39, no. 9, pp. 8079 – 8085, 2012.

[42] G. Lekakos and P. Caravelas, "A hybrid approach for movie recommendation," *Multimedia Tools and Applications*, vol. 36, no. 1-2, pp. 55–70, 2008.

[43] M. Kim and S. Park, "Group affinity based social trust model for an intelligent movie recommender system," *Multimedia Tools and Applications*, vol. 64, no. 2, pp. 505–516, 2013.

[44] F. Mourão, L. Rocha, J. A. Konstan, and W. Meira, Jr., "Exploiting non-content preference attributes through hybrid recommendation method," in *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, (New York, NY, USA), pp. 177–184, ACM, 2013.

[45] O. Osmanli and I. Toroslu, "Using tag similarity in svd-based recommendation systems," in *Application of Information and Communication Technologies (AICT), 2011 5th International Conference on*, pp. 1–4, Oct 2011.

[46] Y.-L. Chen and Y.-T. Chiu, "An ipc-based vector space model for patent retrieval," *Information Processing and Management*, vol. 47, no. 3, pp. 309 – 322, 2011.

[47] G. Stockman and L. G. Shapiro, *Computer Vision*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1st ed., 2001.

[48] W.-Y. Deng, Q.-H. Zheng, S. Lian, and L. Chen, "Adaptive personalized recommendation based on adaptive learning," *Neurocomputing*, vol. 74, no. 11, pp. 1848 – 1858, 2011. Adaptive Incremental Learning in Neural Networks Learning Algorithm and Mathematic Modelling Selected papers from the International Conference on Neural Information Processing 2009 (ICONIP 2009) {ICONIP} 2009.

[49] R. Ronen, N. Koenigstein, E. Ziklik, and N. Nice, "Selecting content-based features for collaborative filtering recommenders," in *Proceedings of the 7th ACM conference on Recommender systems*, RecSys '13, (New York, NY, USA), pp. 407–410, ACM, 2013.

[50] R. 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, (San Francisco, CA, USA), pp. 1137–1143, Morgan Kaufmann Publishers Inc., 1995.