# Automatic movie ratings prediction using machine learning

Mladen Marović, Marko Mihoković, Mladen Mikša, Siniša Pribil, and Alan Tus
University of Zagreb, Faculty of Electrical Engineering and Computing
Unska 3, Zagreb, Croatia
Email: {mladen.marovic, marko.mihokovic, mladen.miksa, sinisa.pribil, alan.tus}@fer.hr

***Abstract*—Recommendation systems that model users and their interests are often used to improve various user services. Such systems are usually based on automatic prediction of user ratings of the items provided by the service. This paper presents an overview of some of the methods for automatic ratings prediction in the domain of movie ratings. The chosen methods are based on various approaches described in related papers. During the prediction process both the user and item features can be used. For the purpose of this paper, data was gathered from the publicly available movie database IMDb. The paper encompasses the implementation of the chosen methods and their evaluation using the gathered data. The results show an improvement in comparison to the chosen baseline methods.**

## I. Introduction

There are many services today which provide users with information about a large number of different items. Good examples of this are systems used in electronic commerce, various public databases, and many others. Information about items that might appear more interesting to users is used to improve the quality of such services. The user ratings given to some items are used to determine the ratings of other items. Those systems which perform such tasks are called automatic ratings prediction systems.

Automatic ratings prediction systems determine a possible user rating for a specific item using a set of chosen features. The information about a possible user rating can be used for different purposes. For example, bookselling services can identify books more interesting to users and use this information to increase sales. Similarly, while using public databases, a user can be presented with data that might appear more interesting to him, thus improving service quality.

Good ratings prediction requires the correct interpretation of the available data about both the user and the item. Therefore, the design of such systems often uses information retrieval and machine learning techniques to find regularities in the available data. The techniques used can differ in their complexity, the amount of data required, as well as other features.

This paper presents a short overview of some automatic ratings prediction methods. The chosen methods are used in automatic ratings prediction systems of many well-known services and are applied to various items. The paper is based on movie ratings prediction, so an adequate dataset was collected, the chosen methods were implemented and evaluated, and the acquired results were analysed.

The paper is organised as follows. The next section presents a short overview of related papers and research on the subject. Then, the descriptions of the used automatic ratings prediction methods are given, divided by the features they use. This is followed by the description of the evaluation process, as well as the presentation and analysis of the results. The last section concludes the paper.

## II. Predicting movie ratings

Ratings prediction is closely related to the problem of recommending items to users. A lot of research into this problem has already been done. However, most of this research reduces the problem of recommendation to the problem of predicting a suitably defined rating for the user and then recommends the item with the highest rating. Three different approaches have emerged in the research of ratings prediction: content-based methods, collaborative methods, and hybrid methods [1]. Content-based methods predict user ratings using the item's features and the user's affinities, while collaborative methods predict ratings using the ratings of other, similar users. Hybrid methods combine the two aforementioned approaches. These approaches can additionally be subdivided into heuristic and model based approaches.

Li and Yamada created a system to predict movie ratings based on decision trees [5]. Decision trees use selected movie features when predicting a movie rating. User preferences for the above mentioned movie features were modeled by building a separate decision tree for each user.

Neural networks can be used to predict a user's movie preferences by applying both the the content based and collaborative approach. Neural networks using a content based approach were built in [6] to profile users' web page ratings. A comparison of the performance of a naive Bayes to the performances of other machine learning methods, including neural networks, was given to verify the feasibility of using computationally demanding methods. Compared to the naive Bayes, neural networks have achieved slightly worse results.

Resnick et al. built a distributed system for news filtering based on predicted ratings [8]. The method used is a collaborative heuristic method. User ratings are predicted using the ratings of similar users via the *k nearest neighbors algorithm* ($k$-NN). User similarity is based on ratings given to the same articles.

Pennock et al. suggested a method for ratings prediction based on personality diagnosis that combines model based and heuristic approaches [7]. The method uses the probability that a user belongs to a certain personality type and the probability of the predicted rating for that personality type. Hoffman, in his work [3], [4], describes a collaborative method based on a probabilistic model with latent variables. Latent variables are used to describe groups of similar users. The expectation maximization algorithm is used to estimate the model's parameters.

For their hybrid system, [9] are proposing a model which contains user and movie features, as well as all available ratings. Rating prediction is then done by using the *singular value decomposition method* (*SVD*) and the *k nearest neighbors algorithm* (*kNN*). The authors call this method the *SVD-kNN* method.

This paper presents the comparison of different methods for ratings prediction. The aforementioned methods were implemented and their results on acquired datasets were compared in order to find the optimal approach to ratings prediction.

## III. Methods for movie ratings prediction

The problem of predicting a user's ratings can be formally defined as follows [3]. Let $\mathcal{U}$ be the set of all users, $\mathcal{I}$ be the set of all movies, and $\mathcal{R}$ be the set of all possible ratings. The goal in ratings prediction is to learn the function $g : \mathcal{U} \times \mathcal{I} \to \mathcal{R}$ for all values from known values of some subset of $\mathcal{U} \times \mathcal{I}$. The description of the chosen methods is given in the rest of this section. The following subsection describes methods based on movies features. Methods that predict user ratings based on the ratings of other similar users are described in subsection III-B. The last subsection presents a hybrid method that combines movie features and user similarity.

### A. Content based methods

One of the basic content based methods used in this paper is the regression tree. Each tree represents a user profile created using the user's ratings. The features used in this paper are: genres, actors, directors and screenwriters. All the possible features are placed in a binary vector, where each row corresponds to either a genre, an actor, a director or a screenwriter. The rows of each vector that correspond to a specific feature of a movie are marked with a one (1), while the rest is marked with a zero (0). When building a regression tree, the minimum square error criterion was used. Classification trees were taken into consideration as well, but preliminary testing showed that they produced inferior results.

Another content based method for user profiling is the *multilayer feedforward neural network*. The hidden layers consist of sigmoidal units, while the output layer consists of linear units. A separate neural network is trained for each user using only the ratings of that user. The movie features used in the training process are genres, actors and directors. A neural network predicts a specific user's movie rating using those features. The features are represented using a binary vector, similar to the regression tree described in the last paragraph. The training is done using the *Levenberg-Maquardt* modification of the backpropagation learning algorithm [2].

### B. Collaborative methods

Following [8], a simple method was implemented which uses the *k nearest neighbors algorithm* to predict user ratings using ratings of similar users. The similarity of users $u_x$ and $u_y$, $\mathrm{sim}(u_x, u_y)$, can be calculated in two ways. The first way is by calculating the Pearson correlation coefficient, which is the correlation between the ratings users $u_x$ and $u_y$ have given to the same movies:

$$\mathrm{sim}(u_x, u_y) = \frac{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum\limits_{i \in I_{xy}} (r_{x,i} - \bar{r}_x)^2 \sum\limits_{i \in I_{xy}} (r_{y,i} - \bar{r}_y)^2}}, \quad (1)$$

where $I_{xy}$ is the set of all the movies both users have rated, $r_{x,i}$ is the rating of user $u_x$ for movie $i$, and $\bar{r}_x$ is the average rating of user $u_x$. User similarity can also be calculated if two users are treated as vectors in a vector space, where vector components are the ratings of movies both users have rated. The similarity between two users can then be calculated as the cosine of the angle between their vectors:

$$\mathrm{sim}(u_x, u_y) = \frac{\sum\limits_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum\limits_{i \in I_{xy}} r_{x,i}^2} \sqrt{\sum\limits_{i \in I_{xy}} r_{y,i}^2}}. \quad (2)$$

The rating $r_{u,i}$ of user $u$ for movie $i$ is predicted using the expression:

$$r_{u,i} = \bar{r}_u + a \sum_{u' \in \hat{U}} \mathrm{sim}(u, u')(r_{u',i} - \bar{r}_{u'}),$$

where $\hat{U}$ is the set of $k$ most similar users who rated the given movie $i$, while $a$ is the normalization factor:

$$a = \frac{1}{\sum\limits_{u' \in \hat{U}} |\mathrm{sim}(u, u')|}.$$

The *personality diagnosis* method, taken from [7], models a user's personality and uses it in the prediction of ratings. If there are $m$ movies in total, the personality type of user $i$ can be presented as a vector $\mathbf{R}_i^{\mathrm{true}} = \langle R_{i1}^{\mathrm{true}}, R_{i2}^{\mathrm{true}}, \ldots, R_{im}^{\mathrm{true}} \rangle$, where ratings $R_{ij}^{\mathrm{true}}$ represent "true" ratings of movie $j$ for user $i$. Additionally, it is assumed that the actual user ratings have Gaussian noise added to the "true" ratings. For user $i$ and movie $j$, the recorded rating presents a realization of a random variable, governed by a normal distribution, whose mean value is equal to the "true" rating $R_{ij}^{\mathrm{true}}$:

$$P(R_{ij} = r | R_{ij}^{\mathrm{true}} = r_t) \propto e^{\frac{-(r - r_t)^2}{2\sigma^2}}.$$

It is assumed that the acquired rating vectors form a representative distribution of personality types present in the population. The user $u$ for which the prediction is being made can belong to any of the observed personalities $\mathbf{R}_1, \ldots, \mathbf{R}_n$ with a probability of $1/n$. Knowing some of the ratings of

user $u$, the probability that he belongs to the personality type $\mathbf{R}_i$ can be calculated by:

$$P(\mathbf{R}_u^{\text{true}} = \mathbf{R}_i | R_{u1} = r_1, \ldots, R_{um} = r_m)$$
$$\propto P(R_{u1} = r_1 | R_{u1}^{\text{true}} = R_{i1})$$
$$\cdots P(R_{um} = r_m | R_{um}^{\text{true}} = R_{im}) P(\mathbf{R}_u^{\text{true}} = \mathbf{R}_i).$$

Finally, the probability distribution of the rating for movie $i$ and the current user $u$ can be estimated using previous probabilities for personality types as follows:

$$P(R_{ui} = r_i | R_{u1} = r_1, \ldots, R_{um} = r_m)$$
$$= \sum_{j=1}^{n} P(R_{ui} = r_i | \mathbf{R}_u^{\text{true}} = \mathbf{R}_j)$$
$$\cdot P(\mathbf{R}_u^{\text{true}} = \mathbf{R}_j | R_{u1} = r_1, \ldots, R_{um} = r_m). \quad (3)$$

The rating with the highest probability is then the predicted rating of user $u$ for movie $i$. The method is defined only with the deviation $\sigma$ of the normal distribution.

The *probabilistic latent semantic analysis* was implemented according to [3]. Intuitively, latent variables represent hidden causes for ratings that user gave to each movie. Users are divided into groups according to similar rating causes using latent variables. Formally, the probability that a user $u$ rates movie $i$ with rating $r$ can be written as:

$$p(r|u,i) = \sum_z p(r|i,z) P(z|u),$$

where $z$ is the latent variable which represents hidden causes. The probability that the cause $z$ is responsible for the ratings of user $u$ is presented by the term $P(z|u)$, while $p(r|i,z)$ is a probability distribution for the ratings of a known movie $i$ according to the cause $z$. It is assumed that this probability distribution is a normal distribution. The expectation maximization algorithm is used to estimate the distribution's parameters $\mu_{i,z}$, $\sigma_{i,z}$ and the probability $P(z|u)$. The $E$-step of the algorithm is represented by the equation:

$$P(z|u,r,i;\hat{\theta}) = \frac{\hat{p}(r|i,z)\hat{P}(z|u)}{\sum_{z'} \hat{p}(r|i,z')\hat{P}(z'|u)},$$

where the hat symbol presents the probabilities according to parameters $\hat{\theta}$, that is, previous estimations for the probability's parameters. The probabilities of latent variables $z$ for all recorded ratings are calculated using the stated equation. These probabilities are then used in the $M$-step for the estimation of probabilities $P(z|u)$:

$$P(z|u) = \frac{\sum_{\langle u',r,i\rangle : u'=u} P(z|u,r,i;\hat{\theta})}{\sum_{z'} \sum_{\langle u',r,i\rangle : u'=u} P(z'|u,r,i;\hat{\theta})}$$

and distribution parameters $\mu_{i,z}$ and $\sigma_{i,z}$:

$$\mu_{i,z} = \frac{\sum_{\langle u,r,i'\rangle : i'=i} r\ P(z|u,r,i;\hat{\theta})}{\sum_{\langle u,r,i'\rangle : i'=i} P(z|u,r,i;\hat{\theta})},$$
$$\sigma_{i,z} = \frac{\sum_{\langle u,r,i'\rangle : i'=i} (r-\mu_{i,z})^2\ P(z|u,r,i;\hat{\theta})}{\sum_{\langle u,r,i'\rangle : i'=i} P(z|u,r,i;\hat{\theta})},$$

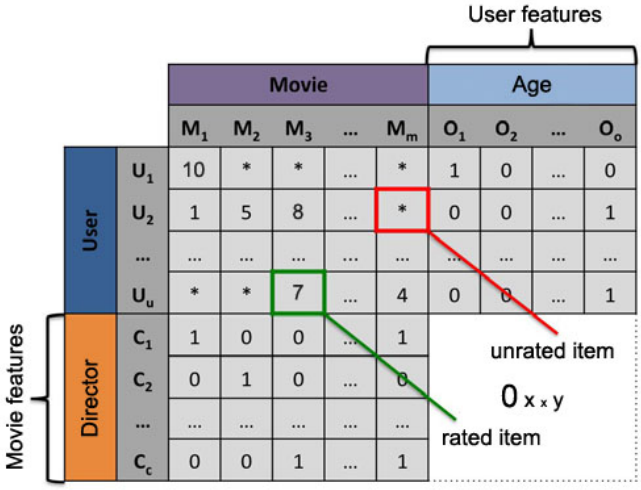


Fig. 1. Example of the matrix on which *SVD-kNN method* is based

where $\langle u,r,i\rangle$ represents the set of all recorded ratings. All ratings are normalized according to the mean rating $\mu_u$ and deviation $\sigma_u$ of the user according to equation:

$$(u,r,i) \mapsto (u,r',i), \quad \text{using} \quad r' = \frac{r-\mu_u}{\sigma_u}. \quad (4)$$

Using the calculated parameters, the predicted rating can be obtained as a mathematical expectation for $p(r|u,i)$:

$$E[r|u,i] = \mu_u + \sigma_u \sum_z P(z|u)\mu_{i,z}.$$

### C. Hybrid methods

The hybrid method *SVD-kNN*, described in [9], was implemented for the purposes of this paper. All available user and movie features are placed in the matrix:

$$H_{m\times n} = \begin{bmatrix} R_{u\times i} & U_{u\times y}\cdot w_2 \\ I_{x\times i}\cdot w_1 & 0_{x\times y} \end{bmatrix}, \begin{pmatrix} m = u+x \\ n = i+y \end{pmatrix}$$

where $R_{u\times i}$ is a matrix of ratings which users gave to movies they watched, $U_{u\times x}$ is a user feature matrix, $I_{x\times i}$ is a movie feature matrix and $w_1$ and $w_2$ are weighted factors that are used to define the ratio at which movie and user features affect the prediction result. Note that filling the matrix $H$ with zeros $0_{x\times y}$ is necessary to preserve its rectangular shape. Figure 1 shows an example of the matrix $H$.

In order to achieve better rating predictions, it is necessary to remove user preferences, movie popularities and other global effects from matrices $R$, $U$ and $I$. For the rating matrix $R$ this is done by "averaging" rating values in the form of subtracting weighted combination of overall- ($\bar{r}$), user- ($\bar{r}_u$) and movie-average ($\bar{r}_i$) from the original rating:

$$\tilde{r}_{ui} = r_{ui} - \alpha\bar{r} - \beta\bar{r}_u - \gamma\bar{r}_i.$$

The parameters $\alpha$, $\beta$ and $\gamma$ determine the influence of each effect. Feature matrices, $U$ and $I$, consist only of values 0 or

1, so the previously mentioned normalization method is not applicable here. Instead, the following expression is used:

$$\tilde{F} = MFN,$$

where $F$ is a user- or movie-feature matrix and $M$ and $N$ are diagonal matrices with values:

$$M_{xx} = \frac{1}{\sqrt{\sum_n F_{xn}}} \quad \text{and} \quad N_{yy} = \frac{1}{\sqrt{\sum_m F_{my}}}.$$

After removing global effects in the rating and feature data, the *SVD* method is used on the resulting matrix $H$. *SVD* is an eigenvalue generalization for non-square matrices which is often used in signal processing and statistics. The linear decomposition procedure factorizes the starting matrix into three low-dimensional resulting matrices containing left-singular vectors ($V$), singular values ($S$) and right-singular vectors ($W$) respectively:

$$H_k = V_{m\times k} \cdot S_{k\times k} \cdot W^T_{k\times n}.$$

The parameter $k$ defines the number of extracted eigenvectors and is crucial for model accuracy. Matrices $V$, $S$ and $W$ are used to calculate compound matrices $X = [VS^{\frac{1}{2}}]$ and $Y = [S^{\frac{1}{2}}W^T]$, which can be considered as separate user and movie concepts and be used as a basis for collaborative rating prediction methods. Using the k nearest neighbor algorithm, the prediction is done by calculating:

$$\hat{r}_{ui} = \frac{\sum\limits_{j\subset ratedMovies(u)} s_{ij}r_{uj}}{\sum\limits_{j\in ratedMovies(u)} |s_{ij}|},$$

where $s_{ij}$ is a cosine similarity defined analogous to expression (2). Model accuracy is also dependent on the number of neighbor elements ($n$) that are included in the calculation.

## IV. Evaluation

All methods mentioned before are implemented and tested using the *Matlab* software package. This section describes the training and test data, as well as the performance measures that are used in experiments. The last subsection contains experiment results and their analysis.

### A. Training and test data

Movie and user data was collected from the publicly available IMDb[1] database. Movie ratings in the collected data are represented by the ratings that users gave in their reviews. Apart from the ratings, the following movie features were collected: title, genres, year of release, directors, screenwriters and actors. It is possible that some movies do not have all of the listed features. Although a bigger dataset was collected, the set of 1059 users, 9428 movies, 1000 actors, 1066 directors, 1060 screenwriters, 28 genres and 65581 ratings in range from 1 to 10 was used for the purposes of this paper.

The training dataset was generated using the *AllButOne* method described in [3]. One rating from every user in the training set was left out and used to create a test set. Thus the training set contains 64522 ratings, while the test set contains 1059. The average number of ratings per movie is 7.57, per user is 60.93, and the average rating value is 7.22.

[1] http://www.imdb.com/

### B. Performance measures

Two performance measures were used to present the success of the various methods used in the paper: the *average absolute deviation* (AAD) and the *root mean square error* (RMSE). The first measure is calculated using the expression:

$$AAD(D) = \frac{1}{|D|}\sum_{j=1}^{|D|} |r_o(j) - r_t(j)|, \tag{5}$$

where $D$ is the training set which contains records about users, movies and ratings, $r_o(j)$ is the predicted user rating for the movie in record $j$, and $r_t(j)$ is the real user rating. Using similar terms, the RMSE measure can be calculated:

$$RMSE(D) = \sqrt{\frac{1}{|D|}\sum_{j=1}^{|D|} (r_o(j) - r_t(j))^2}. \tag{6}$$

### C. Results

The results for each of the implemented methods are shown in table I. The average absolute deviation (AAD) and rooted mean square error are shown for every method. Additionally, the table shows the number of examples that belong to one of the selected ranges of absolute deviation. Two baseline methods were implemented: for each movie and user pair the first baseline method returns the movie's mean rating, while the second method returns the user's mean rating, both calculated on the train set. The rest of this sections contains comments for the evaluation results of each method, following the order in which they appear in the table.

The *regression tree* was trained using ten-folded cross-validation that showed the best places to trim the tree. The results obtained in this paper are inferior to baseline methods. Such a result could likely be explained with an insufficient number of training samples per user and a very sparse distribution of features per film. A small number of sparse vectors cause problems when discerning a movie.

The complexity of *neural networks*, the number of hidden layers and the number of neurons in the hidden layer was chosen according to the guidelines in [6]. Using two hidden layers, one consisting of ten, and the other of five units, is not optimal, as can be seen from the results. Selecting the optimal number of neurons in the hidden layer by searching the entire parameter space could not be performed due to high computing demands. The experimental results are inferior to baseline methods, which could be attributed to the same reasons as it was with the regression tree. In addition, the results are worse than those acquired using the regression tree, probably because the regression tree selects features that better discriminate movies, while neural networks predict movie ratings based on features more common in the set. The inability of a network's structural adaptation to an individual user could have also

TABLE I
RESULTS

| Methods | Measures | | Deviation | | | |
|---|---|---|---|---|---|---|
| | AAD | RMSE | $[0, 0.5\rangle$ | $[0.5, 1.5\rangle$ | $[1.5, 2.5\rangle$ | $[2.5, \infty\rangle$ |
| Baseline methods | | | | | | |
| Mean movie rating | 1.676 | 2.289 | 199 | 405 | 248 | 207 |
| Mean user rating | 1.457 | 1.921 | 258 | 392 | 227 | 182 |
| Content-based methods | | | | | | |
| Regression tree | 1.593 | 2.293 | 278 | 352 | 216 | 213 |
| Neural network | 1.691 | 2.307 | 242 | 370 | 207 | 240 |
| Collaborative methods | | | | | | |
| $k$-NN algorithm | 1.319 | 1.747 | 263 | 454 | 201 | 141 |
| Personality diagnosis | 1.482 | 2.009 | 238 | 400 | 248 | 173 |
| Latent variables (EM) | **1.254** | **1.739** | **307** | 438 | 185 | **129** |
| Hybrid methods | | | | | | |
| SVD-kNN | 1.379 | 1.897 | 267 | 415 | 212 | 165 |

contributed to poor results. Currently, neural networks built for users with a smaller number of ratings are probably overfitted, while more complex networks would be more suitable for users with a larger number of ratings.

The parameter $k$ of the $k$ *nearest neighbors* method was trained using ten-folded cross-validation on the given training set. The training procedure was performed twice, for different user similarity measures, given by expressions (1) and (2). Both cases resulted with value $k_{opt} = 40$ as the optimal value of the parameter. A new similarity table, which contained similarities between pairs of users, was obtained for each iteration of the cross-folded validation. The uses of both the Pearson coefficient and the cosine similarity resulted in equal error values, given in table I. Both error values are smaller than those acquired through the use of baseline methods. The main advantage of this method is its simplicity and solid results, while the largest flaw is the inability to add new users without recalculating the similarity table.

The *personality diagnosis* method was trained using three-folded cross-validation with the step value of $0.5$ for the parameter $\sigma$. The optimal value for the parameter was determined to be $\sigma_{opt} = 5.0$. The results obtained on the test data show results inferior to one of the baseline methods. A possible reason for this is a small number of ratings per movie on average, which can cause a lower possibility of finding similar users. The advantage of this method is its simplicity, while one of the main disadvantages is that all of the user information needs to be stored in memory.

The number of variables in the *probabilistic latent semantic analysis* was determined using five-folded cross-validation. On each iteration of cross-validation $30$ random restarts of the algorithm were carried out, in order to minimize the influence of randomly chosen initial conditions. The error for that iteration was then set to be the mean of the errors from the random restarts. The optimal number of latent variables obtained by cross-validation was equal to $1$ and the error grew with the number of variables. The results show the best ability to predict ratings compared to the rest of the tested methods. The primary advantage of the method is the ability to semantically interpret its results as groups of similar users, while the main disadvantage is its inability to cope with new users or movies.

All parameters of the SVD-kNN hybrid method ($\alpha$, $\beta$, $\gamma$, $k$ and $n$) were trained using three-folded cross-validation, but due to extreme computational complexity not all combinations of values were tested. Instead, the parameter space was examined more thoroughly as parameter values were closer to optimal. A new singular value decomposition of the matrix $H$ and a new similarity table for the *kNN* algorithm were calculated in every iteration of the cross-folded validation. Optimal values are as follows: $\alpha_{opt} = -0.069$, $\beta_{opt} = -0.795$, $\gamma_{opt} = -0.789$, $k_{opt} = 112$ and $n_{opt} = 3308$. It is clear that the results of the hybrid method are better than those of the baseline methods, but not the best overall. The performance would probably benefit from additional user information and more ratings. The main advantage of this method is that the prediction is based on all available information (user and movie features as well as ratings), which might lead to better results. The main disadvantage is its extreme computational complexity because new users cannot be added to the model without recalculating the singular value decomposition of the matrix $H$ and creating a new similarity matrix.

Of all the tested methods, the probabilistic latent semantic analysis using one latent variable proved to be the best. In this case, the method can be simply interpreted by the equation:

$$r = \mu_u + \sigma_u \bar{\mu}_i,$$

where $r$ is the predicted rating, $\mu_u$ is the user's mean rating, $\sigma_u$ is the user's deviation, and $\mu_i$ is the mean value of normalized movie ratings, where normalization was performed according to (4). This result demonstrates relative similarity in ratings between users, that is, most users give movie the same rating modified by their own personal scale. It is possible that these results are consequence of the small number of ratings per movie and users tendency to rate movies that they prefer, which results in small deviation between ratings. Using a more representative dataset could possibly lead to better results.

## V. Conclusion

Predicting user ratings for some items presents an interesting and well formed problem. Many different approaches to solving this problem were used with different methods and performance. This paper gives a comparison of a number of methods used in the automatic prediction of movie ratings. The used dataset was collected from the publicly available IMDb database. The results show that the probabilistic latent semantic analysis achieved the best predictions.

Future plans include more detailed testing with different datasets because the currently used dataset does not contain the required diversity of user ratings. Some of the implemented methods were not tested thoroughly enough because of their computational complexity and should be tested in detail. Other methods suggested in related work should be taken into consideration as well. A combination of classifiers and the use of weighted voting could be tested for prediction purposes. Finally, developing a new algorithm for movie ratings prediction is a possible option for the future.

## Acknowledgments

The authors thank the anonymous reviewer for his or her useful comments.

## 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," *IEEE Trans. Knowl. Data Eng.*, pp. 734–749, 2005.

[2] M. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, 2002.

[3] T. Hofmann, "Collaborative filtering via gaussian probabilistic latent semantic analysis," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 259–266.

[4] ——, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, pp. 89–115, 2004.

[5] P. Li and S. Yamada, "A movie recommender system based on inductive learning," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, vol. 1. IEEE, 2005, pp. 318–323.

[6] M. Pazzani and D. Billsus, "Learning and revising user profiles: The identification of interesting web sites," *Machine learning*, vol. 27, no. 3, pp. 313–331, 1997.

[7] D. Pennock, E. Horvitz, S. Lawrence, and C. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 473–480.

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*. ACM, 1994, pp. 175–186.

[9] S. Spiegel, J. Kunegis, and F. Li, "Hydra: a hybrid recommender system [cross-linked rating and content information]," in *Proceeding of the 1st ACM international workshop on Complex networks meet information & knowledge management*. ACM, 2009, pp. 75–80.