An Empirical Comparison of Collaborative Filtering Approaches on Netflix Data

Nicola Barbieri, Massimo Guarascio, Ettore Ritacco
ICAR-CNR
Via Pietro Bucci 41/c, Rende, Italy
{barbieri,guarascio,ritacco}@icar.cnr.it

ABSTRACT

Recommender systems are widely used in E-Commerce for making automatic suggestions of new items that could meet the interest of a given user. Collaborative Filtering approaches compute recommendations by assuming that users, who have shown similar behavior in the past, will share a common behavior in the future. According to this assumption, the most effective collaborative filtering techniques try to discover groups of similar users in order to infer the preferences of the group members. The purpose of this work is to show an empirical comparison of the main collaborative filtering approaches, namely *Baseline*, *Nearest Neighbors*, *Latent Factor* and *Probabilistic* models, focusing on their strengths and weaknesses. Data used for the analysis are a sample of the well-known Netflix Prize database.

Categories and Subject Descriptors

H.2.8 [**Database Application**]: Data Mining

Keywords

Recommender Systems, Collaborative Filtering, Netflix

1. INTRODUCTION

The exponential growth of products, services and information makes fundamental the adoption of intelligent systems to guide the navigation of the users on the Web. The goal of *Recommender Systems* is to profile a user to suggest him contents and products of interest. Such systems are adopted by the major E-commerce companies, for example Amazon.com [1], to provide a customized view of the systems to each user. Usually, a recommendation is a list of items, that the system considers the most attractive to customers. User profiling is performed through the analysis of a set of users' evaluations of purchased/viewed items, typically a numerical score called *rating*. Most recommender systems are based on *Collaborative Filtering (CF)* techniques [6], which analyze the past behavior of the users, in terms of previously given ratings, in order to foresee their future choices and discover their preferences. The main advantage in using CF techniques relies on their simplicity: only users' past ratings are used in the learning process, no further informations, like demographic data or item descriptions, are needed (techniques that use this knowledge are called *Content Based* [10, 14]). Four different families of techniques have been studied: *Baseline*, *Neighborhood based*, *Latent Factor* analysis and *Probabilistic* models. This work aims to show an empirical comparison of a set of well-known approaches for CF, in terms of quality prediction, over a real (non synthetic) dataset. Several works have focused on the analysis and performance evaluation of single techniques (i.e. excluding ensemble approaches), but at the best of our knowledge there is no previous work that performed such a deep analysis comparing different approaches.

[1]http://amazon.com/

Appears in the Proceedings of the 1st Italian Information Retrieval Workshop (IIR'10), January 27–28, 2010, Padova, Italy.
http://ims.dei.unipd.it/websites/iir10/index.html

2. BACKGROUND

The following notation is used: u is a user, m is a movie, $\hat{r}_m^u$ is the rating (stored into the data set) expressed by the user u with respect to the movie m (zero if missing), and given a CF model, r_m^u is the predicted rating of the user u for the movie m. On October 2006, Netflix[2], leader in the movie-rental American market, released a dataset containing more of 100 million of ratings and promoted a competition, the Netflix Prize [3], whose goal was to produce a 10% improvement on the prediction quality achieved by its own recommender system, *Cinematch*. The competition lasted three years and was attended by several research groups from all over the world. The dataset is a set of tuple $(u, m, \hat{r}_m^u)$ and the model comparison is performed over a portion of the entire Netflix data [4]. This portion is a random sample of the data, and is divided into two sets: a training set $\mathcal{D}$ and a test set $\mathcal{T}$. $\mathcal{D}$ contains $5,714,427$ ratings of $435,659$ users on $2,961$ movies, $\mathcal{T}$ consists of $3,773,781$ ratings (independent from the training set) of a subset of training users $(389,305)$ on the same set of movies. The evaluation criterion chosen is the **Root Mean Squared Error (RMSE)**:

$$RMSE = \sqrt{\frac{\sum_{(u,m) \in \mathcal{T}} \left(r_m^u - \hat{r}_m^u\right)^2}{|\mathcal{T}|}} \qquad (1)$$

Cinematch achieves (over the entire Netflix test set) an RMSE value equals to 0.9525, while the team BellKor's Pragmatic Chaos, that won the prize, achieved a RMSE of 0.8567. This score was produced using an ensemble of severeal predictors.

[2]http://www.netflix.com/
[3]http://www.netflixprize.com/
[4]http://repository.icar.cnr.it/sample_netflix/

3. COLLABORATIVE FILTERING MODELS

Studied models belong to four algorithm families: Baseline, Nearest Neighbor, Latent Factor and Probabilistic models. A detailed description of all the analyzed techniques follows.

3.1 Baseline Models

Baseline algorithms are the simplest approaches for rating prediction. This section will focus on the analysis of the following algorithms: *OverallMean*, *MovieAvg*, *UserAvg*, *DoubleCentering*. *OverallMean* computes the mean of all ratings in the training set, this value is returned as prediction for each pair (u, m). *MovieAvg* predicts the rating of a pair (u, m) as the mean of all ratings received by m in the training set. Similarly, *UserAvg* predicts the rating of a pair (u, m) as the mean of all ratings given by u. Given a pair (u, m), *DoubleCentering* compute separately the mean of the ratings of the movie r_m, and the mean of all the ratings given by the user r_u. The value of the prediction is a linear combination of these means:

$$r_m^u = \alpha\, r_m + (1 - \alpha)\, r_u \tag{2}$$

where $0 \leq \alpha \leq 1$. Experiments on $\mathcal{T}$ have shown that the best value for α is 0.6 (see Fig. 1).

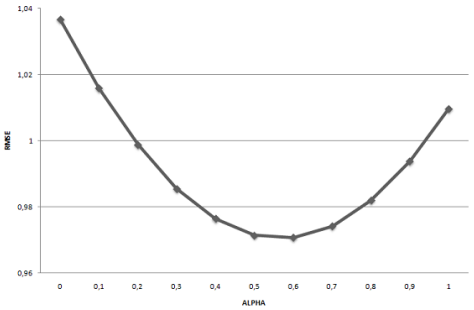

Figure 1: RMSE vs. α

3.2 Nearest Neighbor models

Neighborhood based approaches compute the prediction basing on a chosen portion of the data. The most common formulation of the neighborhood approach is the *K-Nearest-Neighbors (K-NN)*. r_m^u is computed following simple steps. A similarity function associates a numerical coefficient to each pair of user, then K-NN finds the *neighborhood* of u selecting the K most similar users to him, said *neighbors*. The rating prediction is computed as the average of the ratings in the *neighborhood*, weighted by the similarity coefficients. User-based K-NN algorithm is intuitive but doesn't scale because it requires the computation of similarity coefficients for each pair of users. A more scalable formulation can be obtained considering an *item-based* approach [15]: the predicted rating for the pair (u, m) can be computed by aggregating the ratings given by u on the K most similar movies to m: $\{m_1, \ldots, m_K\}$. The underlying assumption is that the user might prefer movies more similar to the ones he liked before, because they share similar features. In this approach the number of similarity coefficients (respectively $\{s_1, \ldots, s_K\}$) depends on the number of movies which is much smaller than the number of users. The prediction is computed as:

$$r_m^u = \frac{\sum_{i=1}^{K} s_i\, r_{m_i}^u}{\sum_{i=1}^{K} s_i} \tag{3}$$

In the rest of the paper, only item-based K-NN algorithms will be considered. The similarity function plays a central role : its coefficients are necessary for the identification of the neighbors and they act as weights in the prediction. Two functions, commonly used for CF, are *Pearson Correlation* and *Adjusted Cosine* [15] coefficients: preliminary studies proved that Pearson Correlation is more effective in detecting similarities than Adjusted Cosine. Moreover as discussed in [9], similarity coefficients based on a larger support are more reliable than the ones computed using few rating values, so it is a common practice to weight the similarity coefficients using the support size, technique often called *shrinkage*. Shrinkage is performed as follows. Let $U(m_i, m_j)$ be the set of users that rated movies m_i and m_j, and let s_{m_i,m_j} be the similarity coefficient between these two movies:

$$s'_{m_i,m_j} = \frac{s_{m_i,m_j}\ |U(m_i, m_j)|}{|U(m_i, m_j)| + \alpha} \tag{4}$$

Where α is an empirical value. Experiments showed that the best value for α is 100, so in the following K-NN algorithms with Pearson Correlation and shrinkage with $\alpha = 100$ will be considered. This first model will be called *SimpleK-NN*. An improved version can be obtained considering the difference of preference of u with respect to the movies in the neighborhood ($\{m_1, \ldots, m_K\}$) of m. Formally:

$$r_m^u = b_m^u + \frac{\sum_{i=1}^{K} s_i\, (\hat{r}_{m_i}^u - b_{m_i}^u)}{\sum_{i=1}^{K} s_i} \tag{5}$$

Where $\{s_1, \ldots, s_K\}$ are the similarity coefficients between m and its neighbors, b_m^u and $b_{m_i}^u$ are baseline values computed using Eq. 2. In this case the model is named *BaselineK-NN*, otherwise, if the baseline values are computed according to the so called *User Effect Model* [2], the model will be called *K-NN (user effect)*. An alternative way to estimate item-to-item interpolation weights is by solving a least squares problem minimizing the error of the prediction rule. This strategy, proposed in [1, 3], defines the *Neighborhood Relationship Model*, one of the most effective approaches applied during the Netflix prize. r_m^u is computed as:

$$r_m^u = \sum_{i=1}^{K} w_{m_i}^m\, \hat{r}_{m_i}^u \tag{6}$$

Where m_i is a generic movie in the neighborhood of m, and $w_{m_i}^m$ are weights representing the similarity between m and m_i computed as the solution of the following optimization problem:

$$min_w \sum_{v \neq u} \left(r_{m_i}^v - \sum_{j=1}^{K} w_{m_i}^m\, \hat{r}_{m_j}^v \right)^2 \tag{7}$$

Fig. 2 shows the behaviors of K-NN models with different values of K. Best performances are achieved by the *Neighborhood Relationship Model*.

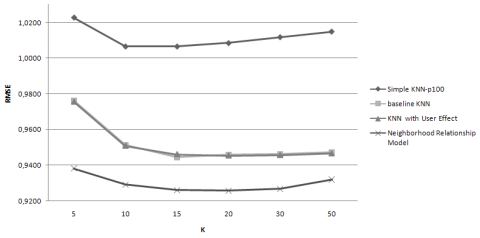

Figure 2: RMSE vs. α

3.3 Latent Factor Models via Singular Value Decomposition (SVD)

The assumption behind Latent Factor models is that the rating value can be expressed considering a set of contributes which represent the interaction between the user and the target item on a set of features. Let A be a matrix $[|users| \times |movies|]$, $A_{u,m}$ is equal to the rank chosen by the user u for the movie m. A can be approximated as the product between two matrices: $A \approx U \times M$, where U is a matrix $[|users| \times K]$ and M is a matrix $[K \times |movies|]$, K is an input parameter of the model and represents the number of features to be considered. Intuitively, A is generated by a combination of users (U) and movies (M) with respect to a certain number of features. Fixed the number of features K, SVD algorithms try to estimate the values within U and M, and give the prediction of r_m^u as:

$$r_m^u = \sum_{i=1}^{K} U_{u,i}\, M_{i,m} \tag{8}$$

where $U_{u,i}$ is the response of the user u to the feature i, and $M_{i,m}$ is the response of the movie m on i. Several approaches have been proposed to overcome the sparsity of the original rating matrix A and to determine a good approximation solving the following optimization problem:

$$(U, M) = \arg\min_{U,M} \left[\sum_{(u,m) in\ \mathcal{D}} \left(\hat{r}_m^u - \sum_{i=1}^{K} U_{u,i}\, M_{i,m} \right) \right] \tag{9}$$

Funk in [5] proposed an incremental procedure, based on gradient descent, to minimize the error of the model on observed ratings. User and movie feature values are randomly initialized and updated as follows:

$$U'_{u,i} = U_{u,i} + \eta\,(2e_{u,m} \cdot M_{i,m}) \tag{10}$$
$$M'_{i,m} = M_{i,m} + \eta\,(2e_{u,m} \cdot U_{u,i}) \tag{11}$$

where $e_{u,m} = \hat{r}_m^u - r_m^u$ is the prediction error on the pair (u, m) and η is the learning rate. The initial model could be further improved considering regularization coefficients λ. Updating rules become:

$$U'_{u,i} = U_{u,i} + \eta\,(2e_{um} \cdot M_{i,m} - \lambda \cdot U_{u,i}) \tag{12}$$
$$M'_{i,m} = M_{i,m} + \eta\,(2e_{um} \cdot U_{u,i} - \lambda \cdot M_{i,m}) \tag{13}$$

An extension of this model could be obtained considering user and movie bias vectors, which define a parameter for each user and movie:

$$r_m^u = c_u + d_m + \sum_{i=1}^{K} U_{u,i}\, M_{i,m} \tag{14}$$

Where c is the user bias vector and d is the movie bias vector. An interesting version of the SVD model was proposed in [13]. According to this formulation, known as *Asymmetric SVD*, each user is modeled through her the rated items:

$$U_{u,i} = \frac{1}{\sqrt{|M(u)| + 1}} \sum_{m \in\ M(u)} w_{i,m} \tag{15}$$

Where $M(u)$ is the set of all the movies rated by the user u. A slight different version, called *SVD++*, proposed in [9], models each user by using both a user-features vector and the corresponding implicit feedback component (movies rated by each user in the training set and the ones for whom is asked the prediction in the test-set).
Latent factor models based on the SVD decomposition change according to the number of considered features and the structure of model, characterized by presence of bias and baseline contributes. The optimization procedure used in the learning phase plays an important role: learning could be incremental (one feature at the time) or in batch (all features are updated during the same iteration of data). Incremental learning usually achieves better performances at the cost of learning time. Several version of SVD models have been tested, considering the batch learning with learning rate 0.001. Feature values have been initialized with the value $\sqrt{\frac{\mu}{K}} + rand(-0.005, 0.005)$ where μ is the overall rating average and K is the number of the considered features. The regularization coefficient, where needed, has been set to 0.02. To avoid overfitting, the training set has been partitioned into two different parts: the first one is used as actual training set, while the second one, called validation set, is used to evaluate the model. The learning procedure is stopped as soon the error on the validation set increases. Performance of the different SVD models are summarized in Tab.1, while Fig.3 shows the accuracy of the main SVD approaches. An interesting property of the analyzed models is that they reach convergence after almost the same number of iteration, no matter how many features are considered. Better performances are achieved if the model includes bias or baseline components; the regularization factors decrease the overall learning rate but are characterized by an high accuracy. In the worst case, the learning time for the regularized versions is about 60 min. The *SVD++* model with 20 features obtains the best performance with a relative improvement on the Cinematch score of about 5%.

Model	Best RMSE	Avg #Iter.
SVD	0.9441	43
SVD with biases	0.9236	45
SVD with baseline	0.9237	45
Reg. SVD	0.9388	32
Reg. SVD with biases	0.9053	186
Reg. SVD with baseline	0.9062	190
SVD++	0.9039	8

Table 1: Performance of SVD Models

3.4 Probabilistic Approaches

Several probabilistic methods have been proposed for the CF, they try to estimate the relations between users or products through probabilistic clustering techniques. The *Aspect Model* [8, 7], also called *pLSA*, is the main probabilistic model used in the CF, and belongs to the class of *Multinomial Mixture Models*. Such models assume that data were independently generated, and introduce a *latent vari-*

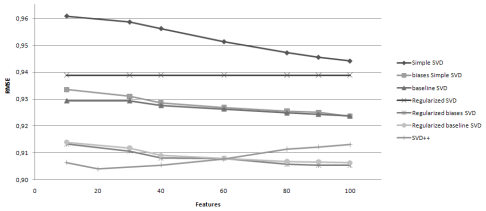

Figure 3: SVD Models Performance

able (also called hidden), namely Z, that can take K values. Fixed a value of Z, u and m are conditionally independent. The hidden variable is able to detect the hidden structure within data in terms of user communities, assuming that Z, associated to observation $(u, m, \hat{r}_m^u)$, models the reason why the user u voted for the movie m with rating $\hat{r}_m^u$. Formally, assuming the user community version, the posterior probability of $\hat{r}_m^u = v$ is:

$$P(\hat{r}_m^u = v|u, m) = \sum_{z=1}^{K} P(\hat{r}_m^u = v|m, z)P(Z = z|u) \quad (16)$$

Where $P(Z = z|u)$ represents the participation in a pattern of interest by u, and $P(\hat{r}_m^u = v|m, z)$ is the probability that a user belonging to pattern z gives rating v on the movie m. A simplified version of the Aspect Model is the *Multinomial Mixture Model* that assumes there is only one type of user [11]:

$$P(\hat{r}_m^u = v|u, m) = \sum_{z=1}^{K} P(\hat{r}_m^u = v|m, z)P(Z = z) \quad (17)$$

The standard learning procedure, for the Multinomial Mixture Model, is the Expectation Maximization algorithm [12]. Fig. 4 shows the RMSE achieved by the Multinomial Mixture Model with different number of latent class. The model

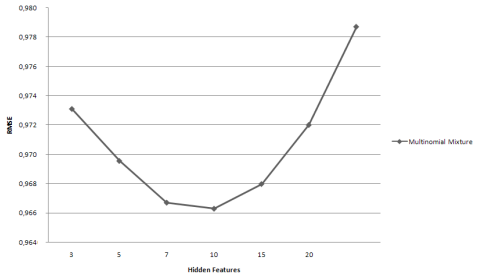

Figure 4: RMSE - Multinomial Mixture

has been initialized randomly and the learning phase required about 40 iterations of the training set but since the first 10 iterations the model reaches the 90% of its potentiality. The best result (0.9662) is obtained considering 10 latent settings for Z. The pLSA model was tested assuming a *Gaussian distribution* for the rating probability given the state of the hidden variable and the considered movie m, in the user-community version. The model was tested for different values of user-communities, as in Fig. 5. To avoid overfitting was implemented the early stopping strategy, described in the previous section. The best pLSA model produces an improvement of around 1% on Cinematch. The

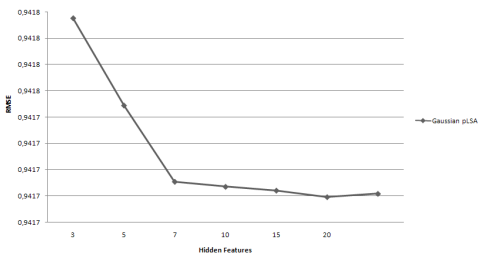

Figure 5: RMSE - pLSA

drawback of the model is the process of learning: a few iterations (3 to 5) of the data are sufficient to overfit the model.

4. MODEL COMPARISON

In this section it is performed a comparative analysis of the above described models. Each model is tuned with it best parameters settings. As said before *Cinematch*, the Netflix's Recommender System, achieves an RMSE equals to 0.9525. Figure 6 shows the RMSE of all Baseline models mentioned. The best model is the *doubleCentering*, but

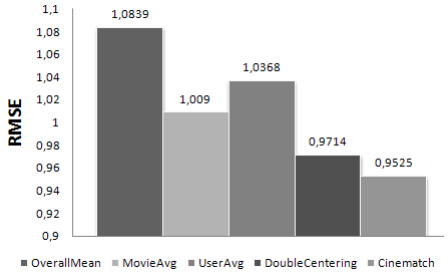

Figure 6: Baseline models

no one of them outcomes the accuracy of Cinematch. Figure 7 shows the mentioned K-NN models performances. Performances are really better than baseline ones. Except

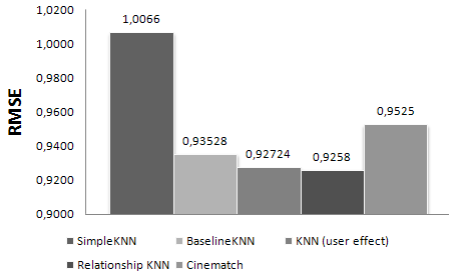

Figure 7: K-NN models

the *SimpleK-NN*, all approaches improve Cinematch's precision, especially the *Neighborhood Relationship Model*. Quality of SVD models is shown in figure 8. SVD models show the best performances, note *SVD++*. Figure 9 shows the behavior of the two proposed probabilistic models. Only *pLSA* outcomes Cinematch. Finally, figure 10 compare the best models for each algorithm family. In this experimen-

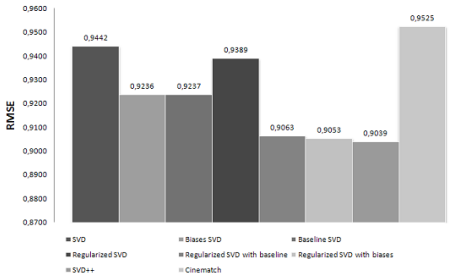

Figure 8: SVD models

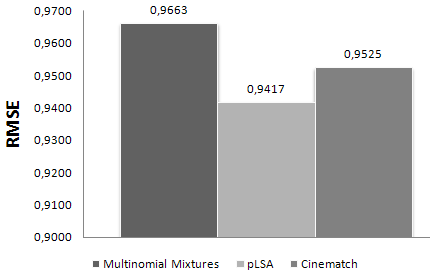

Figure 9: Probabilistic models

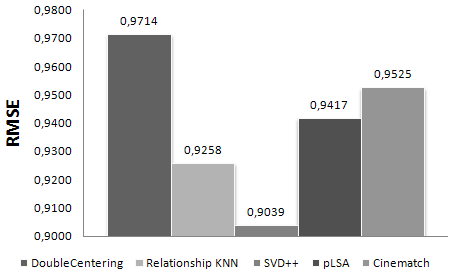

Figure 10: Best models

tation *SVD++* results to be the best model among all proposed ones.

5. CONCLUSIONS AND FUTURE WORK

This work has presented an empirical comparison of some of the most effective individual CF approaches applied to the Netflix dataset, with their best settings. Best performances are achieved by the *Neighborhood Relationship* and the *SVD++* models. Moreover, the symbiosis of standard approaches with simple baseline or biases models improved the performances, obtaining a considerable gain with respect to Cinematch. From a theoretical point of view, probabilistic models should be the most promising, since the underlying generative process should in principle summarize the benefits of latent modeling and neighborhood influence. However, these approaches seem to suffer from overfitting issues: experiments showed that their RMSE value is not comparable to the one achieved by SVD or K-NN models. Future works will focus on the study of the *Latent Dirichlet Allocation (LDA)* [4] that extends the *pLSA* model reducing the risk of over fitting, and on the integration of baseline/bias contributes in probabilistic approaches.

6. REFERENCES

[1] R. Bell, Y. Koren, and C. Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 95–104, New York, NY, USA, 2007. ACM.

[2] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering. In *In Proc. of KDD-Cup and Workshop at the 13th ACM SIGKDD International Conference of Knowledge Discovery and Data Mining*, pages 7–14, 2007.

[3] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, 2007. IEEE Computer Society.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[5] S. Funk. Netflix update: Try this at home. URL: http://sifter.org/ simon/Journal/20061211.html.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:61–70, 1992.

[7] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266, New York, NY, USA, 2003. ACM.

[8] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, January 2004.

[9] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[10] H. Lieberman. Letizia: An Agent that Assists Web Browsing. In *Proc. of Int. Joint Conf. on Artificial Intelligence*, pages 924 – 929, 1995.

[11] B. Marlin. Modeling user rating profiles for collaborative filtering. In *In NIPS*17*, 2003.

[12] T. K. Moon. The expectation-maximization algorithm. *Signal Processing Magazine, IEEE*, 13(6):47–60, 1996.

[13] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. *Proceedings of KDD Cup and Workshop*, pages 39–42, 2007.

[14] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2007.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.