Exploiting SDAE Model for Recommendations

Qing Yang, Xianhe Yao
Guangxi Key Laboratory of Automatic Detecting Technology and Instruments
Guilin University of Electronic Technology, Guilin, China
gtyqing@hotmail.com, 407953822@qq.com

Jingwei Zhang
Guangxi Key Laboratory of Trusted Software
Guilin University of Electronic Technology
Guilin, China
gtzjw@hotmail.com

Zhongqin Bi
College of Computer Science and Technology
Shanghai University of Electronic Power
Shanghai, China
zqbi@shiep.edu.cn

***Abstract*—The data for recommendations, usually a matrix composed of users and items, include a large number of missing data, noise data, etc, which have a negative effect on the accuracy of recommendations. In order to improve recommendation performance, this paper put forwards an improved model named Stacked Denoising AutoEncoder (SDAE), which improves the autoencoder by both indicators and denoising parts to construct an effective stacked autoencoding network. The first layer of the encoding network is responsible for dealing with missing data with the help of indicators and to get a new encoding for features, and then stacked denoising is applied to process noise data for a further optimization. SDAE's output can be accepted by collaborative filtering methods to provide a more accurate recommendation. Three data sets are used to verify the proposed model, the experimental results show that the proposed model presents an active ability on improving recommendation performance and mitigates the negative influence caused by missing data, noise data, etc.**

***Index Terms*—recommendation, data preprocessing, stacked encoding**

I. Introduction

Data play a key role for recommendations to match users' interests, such as smart tourism, online shopping. But missing data, noise data etc are often included when collecting data from applications. These inaccurate data bring negative impact on the effectiveness of recommendations. For example, as a popular strategy for recommendation, collaborative filtering will be weakened on computing the similarity between users or items when using data containing both missing values and noise data.

In addition, it is also difficult to capture users' characteristics accurately when facing large amount of missing values and noise data. For example, 0 is usually a default value for those unobserved behaviors, but may also be a score from a real user, the same value meaning different actions for users will yield unsatisfied recommendations. Obviously, it is also possible to output unexpected decision on users' similarity caused by noise data, which can even result in a series of failures on recommendation for an initializer. Aiming at the above challenges, this paper focuses on data preprocessing models to improve data quality for recommendations. Depending on deep learning and encoding technologies, this paper put forwards a stacked denoising autoencoder to preprocess missing data and noise data in recommendation data, which can then cooperate with collaborative filtering to improve recommendation accuracy. The major contributions are as followings,

- introducing indicators to deal with missing data and to compute the hidden features for recommendations;
- constructing a stacked denoising autoencoder to process noise data and to cooperate with collaborative filtering for improving recommendation performance;
- conducting comprehensive experiments to verify the effectiveness of the proposed model.

This paper is organized as following. Section II summarized the related work on recommendation and data preprocessing technologies. Section III presented the primary objective of this paper and the problem statement. Section IV detailed the proposed model, including the network structures for encoding and parameter training. Section V designed experiments to provide proof for verifying and analyzing our model. Section VI concluded the whole paper and discussed the future work.

II. Related Work

Data preprocessing is a key part to ensure the effectiveness of recommendations. Considering the negative effect on recommendations brought by missing values, these current popular strategies are to fill missing values, dimensionality reduction or clustering. [1] proposed a model-based collaborative filtering method that used a specific

Jingwei Zhang and Zhongqin Bi are corresponding authors.
DOI reference number: 10.18293/SEKE2018-094

value to substitute those missing values and to compute the similarity between items. [2] put forwards a two-stage clustering method on sparse data, which combined graph summarization with content-based similarity to provide topic recommendation based on users' interests.

Aiming at suppressing noise data, principal component analysis(PCA) is a classical strategy on both denoising and dimensionality reduction. On denoising applications, PCA belongs to fixed effect model and uses a fixed structure with low noise to generate data. [3] analyzed the difference between PCA and regularized PCA and proved that the regularized PCA performs better even facing high-noise data. In addition, neural networks are becoming active for processing noise data. [4] applied radial basis function neural networks on denoising of ECG signal and improved analysis performance on ECG data.

In 2006, both deep learning and the improvement on model training broke the bottleneck of traditional BP neural networks, Geoffrey Hinton elaborated systematically the strong learning ability on features contributed by multi-layer neural networks and proved that those features learned by deep learning model can express those initial data better [5]. [6] applied local denoising to learn the effective features in deep networks. Aiming at the local minimum of gradient descent caused by stochastic weight initialization of neural networks, [7] put forwards stacked autoencoder on stacked RBM and trained neural networks layer by layer in greedy mode to capture the deep features, which can learn deep nonlinear network structures from large-scale data. This paper made a comprehensive consideration on both current data preprocessing for recommendation applications and the popularity of deep learning, and constructed an improved stacked denoising autoencoder model to provide more effective and accurate input data for further recommendations.

III. Problem Statement

This paper focuses on more effective data preprocessing and feature capturing method, which can cooperate with those popular recommendation methods, such as collaborative filtering, top-N recommendations, etc, for better recommendations. The specific problem can be stated as,

Given an initial dataset D, we need to design a function df for improving their effectiveness on recommendations, $D^{'} = df(D)$, which should satisfy $ACC(rf(D^{'})) > ACC(rf(D))$ when given a specific recommendation function rf and a concrete evaluation metric ACC.

Our basic strategy for the function df is to introduce autoencoders and neural networks for dealing with two kinds of abnormal data in the initial dataset, namely missing data and noise data. The following parts will detail our solutions.

IV. SDAE Model

A Stacked Denoising AutoEncoder(SDAE) model is proposed to optimize data for recommendations. SDAE model introduces an autoencoder with indicators to deal with the data sparsity and to construct their initial encoding. Then, a Gaussian noise is applied on the output encoding and the stacked denoising technology is combined with autoencoding to suppress the noise. The proposed model reduces the negative effect caused by both the sparsity and noise data to improve recommendation quality.

A. Sparsity Optimization by Introducing Autoencoding with Indicators

Autoencoder adopts unsupervised learning model to train on non-labeled dataset, which constructs a three-layer network model to draw the compression characteristics of the input data. The encoding network structure is illustrated in Fig. 1. For encoding, the input x will be converted into a hidden feature h, which is computed as Equation. 1. W_1 is a weight matrix, b_1 is a bias vector. $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function [8]. For decoding, the weight matrix W_2 and the bias vector b_2 are applied to restore the implied feature y from the compressed hidden layer h, which is showed in Equation. 2. We introduce stochastic gradient descent strategy to optimize the weight matrix W_1, W_2 and the bias vector b_1, b_2, and use minimum average error to measure the encoding performance. The average error between the input x and the encoding output y is defined as their $L2$ norm, which is presented in Equation. 3. The value h outputted by the hidden layer is just the encoding output of the initial data in the first phase. $\| \cdot \|_2$ represents the $L2$ norm of vectors or matrix.

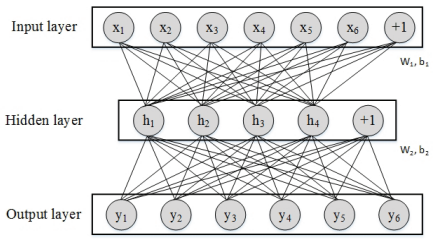

Fig. 1. A three-layer autoencoder

$$h = \sigma(W_1^T x + b_1) \tag{1}$$

$$y = \sigma(W_2^T h + b_2) \tag{2}$$

$$\iota(x, y) = \| x - y \|_2 \tag{3}$$

A large number of missing data are acting as negative samples when recommendation algorithms work on them [9]. It is difficult for a pure autoencoder to capture the

data feature accurately. The following will construct an autoencoding network with indicators to filter out missing data and to reduce their side effect on similarity computation. We define an indicator matrix as Formula. 4 to identify those missing data in the input data, where i, j represents the user ID and the ID of recommended objects. The indicator matrix can help to remove those missing data from the training process of autoencoders, whose structure is illustrated in Figure. 2.

$$indicator_{i,j} = \begin{cases} 1 & if \quad data_{i,j} \neq 0 \\ 0 & otherwise \end{cases} \tag{4}$$

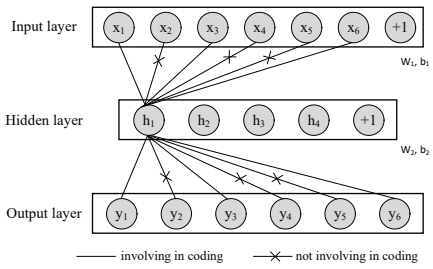

Fig. 2. A partial illustration of autoencoder with indicators

The specific process for training autoencoding network is detailed in Algorithm. 1, which is composed of two stages with the help of indicators, encoding and decoding. Those related symbols are as following. X is the initial input data, n_v is the node number in the input/visual layer, n_h is the node number in the hidden layer. l_r denotes the learning rate, ϵ holds the threshold of the minimum average error, which respectively serve for the network constructing speed and the stopping condition. The weight matrix are denoted as W and $W^{'}$, their corresponding bias vectors are b and $b^{'}$. For functions, the encoding function is denoted by *get-hidden-units*, the decoding function is *get-reconstruction-units*, who correspond to the Formula. 1 and 2. The function for computing reconstruction error is *get-cost* that is expressed in 3. *get-gradient* is used to compute gradients, which cooperates with l_r and applies stochastic gradient descent [10] to update parameters and to minimize the reconstructed errors. By the indicators, the weights corresponding to the missing parts are ignored at both input and output, and an activation function is designed to compute both the hidden and the reconstructed representations for the initial data.

B. Improving Recommendations by Stacked Denoising AutoEncoder(SDAE)

Noise is another factor disturbing the accuracy of recommendations. Based on the autoencoder model with indicators, a denoising autoencoder model is proposed to remove the disturbance caused by noise. In order to improve the local minimizing problem of gradient descent,

Algorithm 1 *Autoencoding with Indicators*

Input: the initial data X,
the indicator matrix $indicator$,
the node number in the input/visible layer n_v,
the node number in the hidden layer n_v,
the learning rate l_r,
the threshold for minimum average error ϵ,
the initial weight matrix W and $W^{'}$,
the initial bias vectors b and $b^{'}$

Output: the encoding output Y_{out}

$X^{'} = X * indicator$;
while $cost > \epsilon$ do
  $Y = get - hidden - units(X^{'}, W, b, n_v, n_h)$;
  $Z = get - reconstruction - units(Y, W^{'}, b^{'}, n_h, n_v)$;
  $Z_{out} = Z * indicator$;
  $cost = get - cost(X^{'}, Z_{out})$;
  $gradient$=$get - gradient(cost, param)$;
  $\backslash\backslash$ $param$ is $(W, W^{'}, b, b^{'})$
  $param$=$param - l_r * gradient$;
end while
return $Y_{out} = Z$;

denoising autoencoders(DAE) are stacked together to form a new stacked denoising autoencoder (SDAE), an iteration computation is applied to capture the features of the initial input. The bottom-up and layered training strategy is used to decide the parameters of the whole network. In the phase of data preprocessing, a group of noise are firstly merged into the initial data to maintain the consistency between the initial data and the features learned from the merged data. This measure can help to improve the ability on resisting noise for recommendations.

Considering that most of noise in real life conform to Gaussian distribution [11], we use Gaussian noise to simulate those possible noise in the initial collected data, which are formalized as $x^{'} \sim q_D(x^{'}|x)$. Here, x is the initial collected data, $x^{'}$ is the data merged with noise and is also the input of recommendation applications, D represents the whole dataset. The structure of the denoising autoencoder is presented in Fig. 3. The Denoising AutoEncoder model (DAE) is responsible for minimizing the reconstructed error between the input merged with noise and its output, which is presented in Formula. 5. Multiple DAE are stacked by submitting the output from the previous DAE to the input of the next DAE to improve recommendation performance.

$$\jmath(x, y) = (\|x^{'} - y\|)_2 \tag{5}$$

Algorithm. 2 presents the detailed process to capture the features of the original data. Compared with Algorithm. 1, a specific value for the layers of neural networks, n_l, is

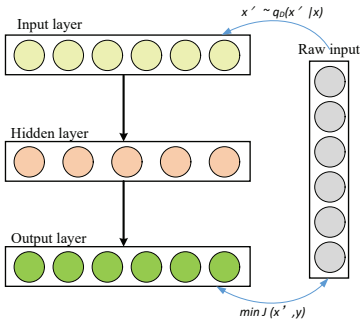

Fig. 3. The model for denoising autoencoder.

introduced to represent how many times for DAE model to be stack. In addition, an array holding the node numbers for each hidden layer, $n_H[n_l]$, and the signal-noise ratio c_r are designated. $get-input$ is a new function for mixing Gaussian noise.

Algorithm 2 *Stacked Denoising Autoencoding for Recommendations*

Input: the initial data X,
the layers of neural networks n_l,
the node number in the input/visible layer n_v,
the node number in the hidden layer $n_h[n_l]$,
the signal-noise ratio c_r,
the learning rate l_r,
the threshold for minimum average error ϵ,
the initial weight matrix W and $W^{'}$,
the initial bias vectors b and $b^{'}$

Output: the encoding output Y_{out}

for k from 1 to n_l do
  $X^{'} = get-input(X, c_r)$
  while $cost > \epsilon$ do
    $Y = get-hidden-units(X^{'}, W[i], b[i], n_v, n_h[k])$;
    $Z = get-reconstruction-units(Y, W^{'}[i], b^{'}[i], n_h[k], n_v)$;
    $cost = get-cost(X^{'}, Z)$;
    $gradient$=$get-gradient(cost, param)$;
    $\backslash\backslash param$ is $(W[i], W^{'}[i], b[i], b^{'}[i])$
    $param$=$param - l_r * gradient$;
  end while
end for
return $Y_{out} = Z$;

V. Experiments and Evaluation

A. Experimental Setup and Datasets

The proposed method is tested and analyzed in this section. The concrete experimental configurations are composed of MacOS Sierra and Matlab2015b run on Intel core i7 with 2.2GHz and 16GB DDR3 memory. Three types of datasets covering different scenarios are applied to verify the effectiveness of the proposed method, which are listed in Table. I. The dataset from *GroupLens* uses 0 to express all missing scorings, in which you can not make a clear distinction between a real score and a default value when meeting 0. Though 0 is still used as default value, the *EachMovie* dataset accompanies all 0s with a weight, which can help you to distinguish that this value is from a real user or is a default one. The dataset from *Jester* is about online joke recommendation, which presents different characteristics from the above two, and consists of less items for recommendations and a larger range for user scoring, and the default values are also not overlapped with the scoring range.

TABLE I
The Details of Data Sets

Name	Num. of users	Num. of items	Scoring range	Default values
MovieLens-100k	943	1682	$[0, 5]$	0
EachMovie	61265	1623	$[0, 1]$	0 (with weight)
Jester Dataset1	24983	100	$[-10, 10]$	99

Since collaborative filtering is the most popular method for recommendations, we will execute collaborative filtering [12] on the preprocessed data contributed by our proposed method to verify its effectiveness. $Top-N$ recommendation is used to evaluate the improvement on recommendations. Three evaluation metrics,*Precision*, *Recall*, and *F1-Score*, are used to evaluate our proposed method, which are computed as Formula. 6- 8. Here, *true positive* is the number of items that should be recommended and have also been in the top-N list, *false positive* represents the number of items that should not be recommended but have been in the top-N list, and *false negative* corresponds to the number of items that should be recommended but have not been in the top-N list.

$$Precision = \frac{truepositive}{truepositive + falsepositive} \tag{6}$$

$$recall = \frac{truepositive}{truepositive + falsenegative} \tag{7}$$

$$F1 - Score = \frac{2 * precision * recall}{precision + recall} \tag{8}$$

In order to verify the improvement on the recommendation accuracy brought by the different compression dimensions in SDAE model, we introduce the user-based collaborative filtering, autoencoding with indicators, as well as a SAE model to compare with the SDAE model on recommendation performance. The specific SAE model

has both the same parameters and the same structure with the SDAE model, whose input dose not mix with noise. The comparison between the SAE model and the SDAE model can show the recommendation effectiveness contributed by noise suppression.

B. Experiments on MovieLens-100k

In this part, we test the effectiveness of our proposed method on a traditional dataset, *MovieLens-100k*, which consists of 100 thousands of scores on 1,682 movies contributed by 943 users. Each score is between 0 and 5, a hidden challenge is that you can not tell whether the value is written by a real user or is just a default value when it is 0. First, we designed a group of experiments to test the recommendation performance under different proportions of noise, the experimental results are presented in Fig. 4. Different amount of noise present different effect on the recommendation accuracy. In the following experiments, 13% noise will be introduced into the datasets since it contributed the best performance on recommendations.

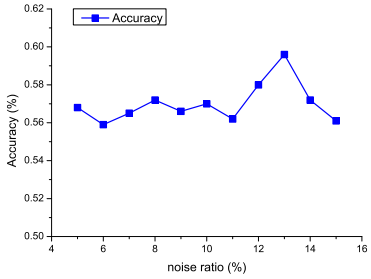

Fig. 4. Recommendation performance under different noise.

Second, we designed a group of experiments to observe the top-N recommendation performance on *MovieLens-100k*, here N is designated to be 10. Fig. 5 presents the recommendation performance contributed by four methods, which are user-based collaborative filtering(User-based), autoencoder with indicators(AE-indicator), stacked autoencoder(SAE) and stacked denoising autoencoder(SDAE). Here, indicators are used to remove all values equal to 0 and do not consider whether this value is from a real user or is just a default value. Compared to the user-based collaborative filtering without data preprocessing, AE-indicator, SAE, SDAE all contributed a better performance on precision, which proved that both the indicators and the stacked encoding technologies have made a difference on the original dataset. Especially, SDAE presented the best precision among the four models. Considering recall, user-based collaborative filtering performed better than AE-indicator, SAE and SDAE. The above situation can be attributed to two factors, one is that the indicators removed some values from real users, and the other is due to the sparse data, the whole data sparsity is $S_{MovieLens-100k} = \frac{100000}{1682*943} = 0.063$, which caused the introduction of noise to have a more obvious influence on a sparse dataset.

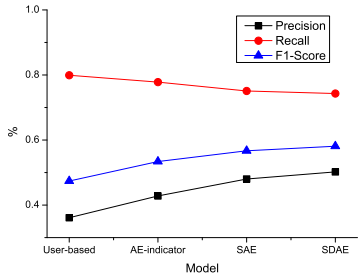

Fig. 5. Top-N recommendation performance on *MovieLens-100k*.

C. Experiments on EachMovie

EachMovie is a dataset collected by HP/Compaq, which consists of 2,811,983 scores on 1,623 movies contributed by 61,265 users. Except for more users than *MovieLens-100k*, the scoring weights are introduced to distinguish those default values from values contributed by real users. Fig. 6 presents the effects of scoring weights on improving recommendation accuracy, both user-based collaborative filtering and AE-indicators have a little improvement on accuracy when those real values with an appointed scoring weight are applied. The following experiments will associate indicators with scoring weights to reserve those values contributed by real users.

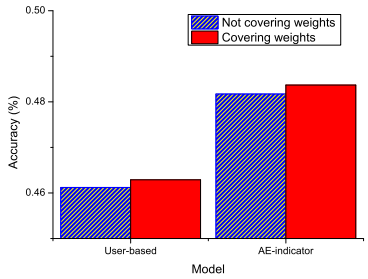

Fig. 6. The recommendation effectiveness test of scoring weight on *EachMovie*.

In this group of experiments, we chose 30,000 users and their scorings randomly to be the training data and then used the remaining data to test their recommendation performance, the experimental results are presented in Fig. 7. SDAE still contributed the best precision, which also declared that indicators, stacked technologies and denoising worked well on preprocessing data. But all the four models presented a lower recall than those on *MovieLens-100k* since the dataset is a more sparse one, whose sparsity is $S_{EachMovie} = \frac{2811983}{1623*61265} = 0.028$.

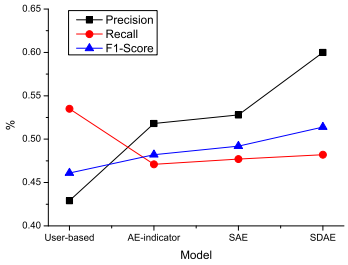

Fig. 7. Top-N recommendation performance on *EachMovie*.

D. Experiments on Jester

In this part, we introduced *Jester* datset to test the effectiveness of the proposed method when facing a small number of features since the node number of the hidden layer are often less than those of the input layer or the output layer in an autoencoder. *Jester* dataset includes scores on 100 jokes contributed by 24,983 users, each score is from -10 to 10 and all default values are expressed as 99. We only consider those values greater than 0 as the positive examples. The experimental results are presented in Fig. 8. Compared with the results on *MovieLens-100k* and *EachMovie*, all the four models presented a good accuracy though SDAE has a little advantage, which show that the data sparsity is a key factor for recommendation performance. One obvious change is that the recall contributed by SDAE covered the other three models, in fact, the effects contributed by indicators can be ignored since a default value is expressed as 99, the denoising and stacked technologies played a major role on improving recommendation performance for SDAE.

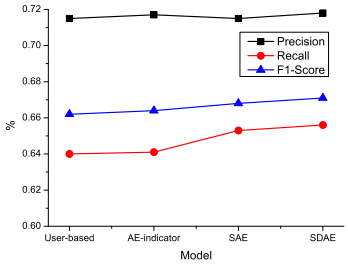

Fig. 8. Top-N recommendation performance on *Jester*.

VI. Conclusion

This paper provides an optimized data preprocessing model, SDAE, to provide more robust data for improving recommendation performance. SDAE synthesized the indicators, stacked encoding and denoising technologies to improve data quality. An autoencoder with indicators are firstly introduced to remove default values and to reduce the loss on recommendation accuracy brought by sparse data. Then a stacked denoising autoencoder is designed to relieve recommendation disturbance caused by noise data, which provides an iterative computation on a three-layer neural network. We verified the effectiveness of the proposed model on different scenarios, including sparse datasets and small number of recommendation objectives. In future, we will apply neural networks to capture the user interest migration and to provide a dynamic recommendation model, the time cost on data preprocessing will also be covered.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (No.61462017, 61363005, 61662015, U1501252, U1711263), Guangxi Natural Science Foundation of China(No.2017GXNSFAA198035), and Guangxi Cooperative Innovation Center of Cloud Computing and Big Data.

References

[1] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295.

[2] K. Chen, P. Han, and J. Wu, "User clustering based soical network recommendation," *Chineser Journal of Computers*, vol. 36, no. 2, pp. 349–359, 2013.

[3] M. Verbanck, J. Josse, and F. Husson, "Regularised pca to denoise and visualise data," *Statistics and Computing*, vol. 25, no. 2, pp. 471–486, Mar 2015.

[4] K. A., "Rbf neural networks for ecg beat classification and arrhythmia detection," *International Journal of Artificial Intelligence and Knowledge Discovery*, vol. 3, no. 3, pp. 9–15, 2013.

[5] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[6] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010.

[7] Y. Bengio, E. r. Thibodeau-Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, pp. II–226–II–234.

[8] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, June 2005.

[9] F. Zhang and H. Chang, "Employing bp neural networks to alleviate the sparsity issue in collaborative filtering recommendation algorithms," *Journal of Computer Research and Development*, vol. 43, no. 4, pp. 667–672, 2006.

[10] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, and J. M. de Sá, "Using different cost functions to train stacked auto-encoders," in *Proceedings of the 2013 12th Mexican International Conference on Artificial Intelligence*, ser. MICAI '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 114–120.

[11] S. Gong, H. Ye, and H. Tan, "Combining memory-based and model-based collaborative filtering in recommender system," in *2009 Pacific-Asia Conference on Circuits, Communications and Systems*, May 2009, pp. 690–693.

[12] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "The adaptive web," P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Collaborative Filtering Recommender Systems, pp. 291–324.