Modeling Prediction in Recommender Systems Using Restricted Boltzmann Machine

Hanene Ben Yedder and Umme Zakia
Simon Fraser University
Vancouver, British Columbia, Canada
Email: {hbenyedd, uzakia}@sfu.ca

Aly Ahmed
University of Victoria
Victoria, British Columbia, Canada
Email: alyahmed@uvic.ca

Ljiljana Trajković
Simon Fraser University
Vancouver, British Columbia, Canada
Email: ljilja@sfu.ca

***Abstract*—Collaborative filtering is a well-known technique used for designing recommender systems when advertising services and products offered to the Internet users. In this paper, we employ the Restricted Boltzmann Machine (RBM) for collaborative filtering and propose the neighborhood-conditional RBM (N-CRBM) model based on joint distributions of similarity and popularity scores. The model is trained and evaluated based on the number of hidden units, learning rates, and activation functions. Simulation results using a dataset consisting of 22 million records show that the proposed N-CRBM model achieves 0.46 average root mean square error (RMSE) and 78.5% accuracy in predicting users' selections of recommended advertisements.**

Keywords—Restricted Boltzmann machine; collaborative filtering; recommender systems; similarity score; popularity score.

I. INTRODUCTION

Recommender systems (RSs) significantly enhance the users' experience when accessing online services. Amazon, Facebook, and Netflix extensively use RSs to attract users by advising their products and services. RSs are used to generate lists of suggestions using approaches such as collaborative filtering (CF), content-based filtering, or hybrid methods [1]. The CF methods utilize past activities and preferences of like-minded users to predict a user's interest and generate recommendations. They use the nearest-neighbor approach based on activities and ratings of previous users. Content-based methods rely on users' profiles and descriptions of products for advertising [2] while hybrid methods combine both techniques [3]. It is usually difficult to collect users' information and create profiles from their past activities due to privacy concerns and, therefore, content-based and hybrid methods are less popular. Early CF approaches used singular value decomposition (SVD), which relies on matrix factorization. Few attempts have been made to develop RS learning models using collaborative filtering.

In this paper, we employ the CF method to predict a user's selection of a new advertisement based on past viewing history of users. The employed dataset [4] contains limited information for approximately 2 million users because more than 80% visited the sites only once. Furthermore, each user rated a very small number of advertisements, which limits building the user's profile and making predictions based on the navigation history. While CF is a suitable approach, its prediction accuracy significantly decreases when ratings are very sparse thus limiting the extraction of useful features. The Restricted Boltzmann Machine (RBM) models are useful when designing deep learning models and offer more accurate results using automatically learned features while hiding the details [5]. We select the RBM model for CF to recommend advertisements to a user based on previous ratings and extend it by incorporating neighborhood content related to advertisements. We then identify the optimal model parameters to maximize its performance in terms of RMSE, prediction accuracy, and sensitivity.

The remainder of this paper is organized as follows. In Section II, we describe related work. In Section III, we provide details of the RBM model followed by proposed approach and the N-CRBM model formulation in Section IV. Model training and testing are described in Section V while their performance is given in Section VI. We conclude with Section VII.

II. RELATED WORK

Various RS learning models have been developed for collaborative filtering. An example is the wide and deep learning model [6] that combines the strength of feature memorization of the linear model and feature generalization of the deep learning model. It is proposed for recommending applications for the Google Play store. The deep model is trained over $5x10^{11}$ examples of Google applications and achieves client-side latency of 14 ms during acquisitions by users. The deep neural network used for advertising YouTube videos performs a candidate generation followed by ranking [7]. Authors use a dataset consisting of 10^6 videos and 10^6 search requests (tokens). They consider both positive (clicked) and negative (unclicked) selections of movies shown to a user on a single web page. Selections are weighted by the duration of watch time. The model estimates that the error in predicting the watch time per user is 34.6%.

The collaborative deep learning model framework [8] jointly performs deep representation learning for the content information and collaborative filtering for the ratings. Authors use three datasets for training the model: CiteULike3-a with 5,551 users and 16,980 items; CiteULike3-t with 7,947 users and 25,975 items; and Netflix dataset with 407,261 users, 9,228 movies, and 15,348,808 ratings. The authors define a performance metric to recommend the top M items. The relational collaborative topic regression develops a hierarchical Bayesian model that integrates the user-item feedback

information, item content information, and their relations using CiteULike3-a and CiteULike3-t datasets [9].

The RBM model was introduced [10] to perform CF in recommending movies for the Netflix competition [11]. Used dataset contains 1,480,000 ratings of 17,700 movies released by Netflix. The reported RMSE of the RBM model used for prediction is 0.91. An RBM model that relies on similar choices of neighboring users is also used for CF recommendations [12]. The MovieLens rating data that consists of 10^6 ratings using a scale of 1 to 5, for 1,700 movies and 10^3 users is used to obtain RMSE of 0.40. A content-boosted RBM [13] for recommendations extends the RBM model by incorporating content-based features. The model is trained with the MovieLens dataset.

III. Restricted Boltzmann Machine

A. *RBM Model*

The RBM is a generative stochastic artificial neural network that learns a probability distribution over a set of inputs [14]. It employs gradient descent approximation algorithms such as contrastive divergence (CD) [15]. The RBM has been used for feature extractions in supervised learning algorithms [16]. It belongs to energy-based models and consists of Bernoulli-valued (binary) hidden and visible units, as shown in Fig. 1. The matrix of weights W represents the strength of connections between visible and hidden units V_i, and H_j, respectively. It includes bias weights (offsets) for these units. Increasing the number of hidden units may improve performance of the RBM model [10].

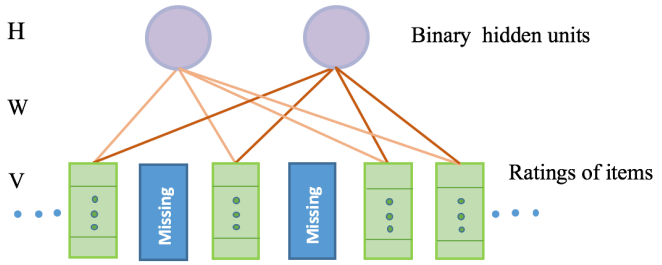

Fig. 1. The RBM model: Each visible unit corresponds to an item that is rated. Items that are not rated are considered missing.

B. *Conditional RBM*

Many users view items without providing feedback and, hence, their ratings are unknown. Furthermore, the test dataset may contain several ratings from a user while the training dataset may have fewer ratings for the same item. If a user views and rates an item, this may lead to his/her choice of similar items. This additional source of information about users and items is considered in the conditional restricted Boltzmann machine (CRBM) model. The CRBM is a probabilistic model [10] that considers in visible layer both rated and unrated items along with additional information. It has been successfully used for modeling temporal data such as motion capture in video sequences, collaborative filtering, and classification. Using a conditional probability of rated and unrated ads/movies/songs improves performance of CFs.

IV. Proposed N-CRBM Model

A. *Model Description*

The major CF challenge is sparsity of data. Therefore, we first design a clusters-based RBM model by clustering users who viewed the same advertisements. We define an RBM for each cluster of users. These RBMs have the same number of hidden units and differ by the number of visible units that depends on the number of common advertisements viewed or selected by users within a cluster. Hence, popularity of the advertisement affects the number of visible units as well as the cluster size. The RBMs calculate the probability distribution for each advertisement and send this information to hidden units that represent features. The weights learned by the RBMs are shared. We consider a sub-weight matrix for each RBM. The weight matrix shown in Fig. 2 is updated after training each cluster of users. If an advertisement is not viewed or selected, it does not contribute to the update of the weight matrix.

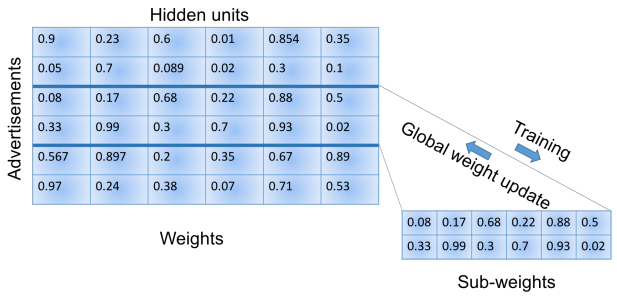

Fig. 2. Sub-weights learned by an RBM that is designed for each cluster of users viewing the same advertisements. The weight matrix is shared among all RBMs.

The clusters-based RBM model is then extended to incorporate advertisements based on the neighborhood content, as shown in Fig. 3. Two additional visible layers S and P are introduced to define similarity and popularity of advertisements, respectively. They reflect the popularity of advertisements in past activities of like-minded users and in documents where the advertisements appear. In this neighborhood-CRBM (N-CRBM) model, the hidden layer (H) and visible layers (V, S, P) are fully connected while there are no connections among the visible layers.

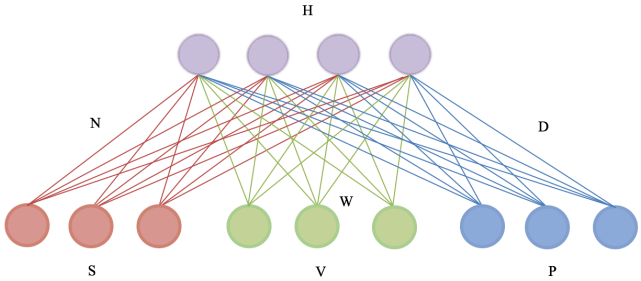

Fig. 3. The proposed neighborhood conditional RBM (N-CRBM) model with similarity (S) and popularity (P) layers. Visible layer V and hidden layer H form a clusters-based RBM for a group of users viewing or selecting the same advertisements.

B. *Model Formulation*

The CF approach is unable to provide recommendations to a new user without navigation history, which is known as the cold-start problem [1]. The N-CRBM model includes similarity and popularity scores of the neighboring users and documents to overcome cold-start and provide the most preferable choices to a user. Similarity and popularity scores $S_i, P_i \in [0,1]$ contribute to the training when their values are close to one.

Similarity is the distribution of ratings provided by a user's neighbors. The neighbor x and user u share the cosine similarity that measures their mutual orientation [17]. The cosine similarity calculates the angle formed by the two rating vectors where only positive ratings are considered while negative ratings are discarded. We consider as neighbors those users who already viewed and/or selected the same advertisement. The similarity score of an advertisement i for the user u is calculated as:

$$S_i\,(u,i) = \frac{\sum_{x \in N_u} Rating_{x,i}}{N(u)}\,, \tag{1}$$

where N_u is the set and $N(u)$ is the number of neighbors of user u. $Rating_{x,i}$ is the viewed/selected (0/1) value of advertisement i by neighbor x.

Popularity is the distribution of ratings of an advertisement in documents that is viewed and/or selected. The popularity score of advertisement i in document d is calculated as:

$$P_i\,(d,i) = \frac{\sum_{y \in D_u} Rating_{y,i}}{D(u)\,R_{max}}\,, \tag{2}$$

where D_u is the set and $D(u)$ is the number of documents that contain advertisement i. $Rating_{y,i}$ is the rating of advertisement i in document y while R_{max} is the maximum rating value of advertisement i in D_u.

A conditional Bernoulli distribution is used to model each column of the observed "visible" binary unit V_i and "hidden" unit H_j, where V_i =1 if the user selects the advertisement and V_i =0, otherwise. The layer S of neighbors of user u with their ratings of advertisement i provides the similarity score S_i. The layer P of documents where advertisement i appears provides the popularity score P_i. Matrices of weights N and D represent the strength of the connections between hidden unit H_j and visible units S_i and P_i, respectively. The joint distribution of scores (V, H), conditional on the similarity score S_i and popularity score P_i, is defined as:

$$P\big(H_j = 1|V,S,P\big) = \sigma\left(a_j + \sum_{i \in V} V_i\,W_{ij} + \sum_{i \in S} S_i\,N_{ij} + \sum_{i \in P} P_i\,D_{ij}\right) \tag{3}$$

$$P\left(V_i = 1|H\right) = \sigma\left(b_i + \sum_{j \in H} H_j\,W_{ij}\right) \tag{4}$$

$$P\left(S_i = 1|H\right) = \sigma\left(c_i + \sum_{j \in H} H_j\,N_{ij}\right) \tag{5}$$

$$P\left(P_i = 1|H\right) = \sigma\left(d_i + \sum_{j \in H} H_j\,D_{ij}\right), \tag{6}$$

where a_j, b_i, c_i, and d_i are biases added to the weight matrices in the N-CRBM model; W, N, and D are the weights; and σ is the activation function. The weights are initialized to small random values chosen from zero-mean Gaussian distribution with standard deviation 0.01 [14].

The parameters required to calculate the gradient descent in the log-likelihood for training are calculated as:

$$\Delta W_{ij} = \epsilon\left(< V_i\,H_j >_{data} - < V_i\;H_j >_T\right) \tag{7}$$

$$\Delta N_{ij} = \epsilon\left(< S_i\,H_j >_{data} - < S_i\,H_j >_T\right) \tag{8}$$

$$\Delta D_{ij} = \epsilon\left(< P_i\,H_j >_{data} - < P_i\,H_j >_T\right), \tag{9}$$

where ϵ is the learning rate. Expectations $< . >_{data}$ define the frequency that advertisement i and feature j are joined in the user-rating data calculated based on joint distribution (3). Expectations $< . >_T$ represent distributions (3)–(6) of samples for T full steps We have used contrastive divergence CD-1 [14], being the most popular gradient approximation.

Given the observed ratings V, the similarity score S, and the popularity score P, the rating for a new advertisement is predicted in a linear time using the mean field updates [10] to obtain the probability distribution for advertisements. The predicted ratings of an advertisement q are used for model testing:

$$\hat{P} = P\big(H_j = 1\big|V,S,P\big) = \sigma\left(a_j + \sum_{i \in \mathrm{V}} V_i\,W_{ij} + \sum_{i \in \mathrm{S}} S_i\,N_{ij} + \sum_{i \in \mathrm{P}} P_i\,D_{ij}\right) \tag{10}$$

$$P\left(V_q = 1\middle|\hat{P}\right) = \sigma\left(a_q + \sum_{j=1}^{F} \hat{P}_j W_{qj}\right). \tag{11}$$

The models are trained and tested using three activation functions [20]:

Logistic sigmoid: $\sigma(a) = \frac{1}{1+e^{-a}}$ (12)

Tanh: $tanh(a) = 2\sigma(a) - 1$ (13)

ReLU: $F(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$. (14)

C. *Performance Metrics*

Performance is measured based on various metrics such as root mean square error (RMSE), accuracy, sensitivity (recall), and precision [19]. The RMSE measures the distance between predicted and true preferences over samples:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (Y_i - Y'_i)^2}{N}}\,, \tag{15}$$

where $(Y_i - Y'_i)$ denotes the residual difference between the

actual values and the predicted values and N is the size of the test set. RMSE is useful for predicting tasks because it computes errors for either negative or positive ratings. The performance of the clusters-based RBM and the N-CRBM is evaluated based on RMSE [10], [12].

Accuracy reflects the true prediction over the entire dataset:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (16)$$

Sensitivity (recall) computes the portion of favored items that were suggested:

$$Sensitivity = \frac{TP}{TP + FN}. \quad (17)$$

Precision refers to the closeness of recommended advertisements to user preferences where false positive rate defines the proportion of unsuitable advertisements recommended. Since a rating "0" of an advertisement implies that the user is either uninterested in the advertisement or not aware of its existence [3], precision cannot capture the feedback of not-selected advertisement properly. Presenting unsuitable advertisement is undesirable for the recommender system and, therefore, precision is not an appropriate performance metric. Hence, we only consider accuracy and sensitivity of the models. They are computed based on the confusion matrix shown in Table I. The true positive (TP) and the true negative (TN) are advertisements (whether or not selected by a user) that are correctly classified (predicted) while the false negative (FN) and false positive (FP) are selections that are misclassified (wrongly predicted).

TABLE I. CONFUSION MATRIX

Actual class	**Predicted class**	
	Selected (positive)	Not-selected (negative)
Selected (positive)	TP	FN
Not-selected (negative)	FP	TN

V. MODEL TRAINING AND TESTING

We use the *Kaggle* dataset provided by the Outbrain click prediction contest [4]. It contains navigation histories of advertisements viewed or selected by users. Each viewed or selected advertisement is accompanied by semantic attributes of the visited documents. The dataset consists of 2 million users, 22 million advertisements, 330,000 unique advertisements, and 87 million users' navigation histories. Users who had visited zero or one advertisement are excluded from the dataset [8].

The SQL server was used to extract model features from the *Kaggle* dataset. A Java script is used to cluster users who have visited the same advertisements and to prepare input data. The clusters-based RBM and the N-CRBM models are implemented using Python and developed using the open source code [18]. The code is modified to share the same matrix of weights, include clusters for training, and enable training large datasets. The code was further extended to include the neighborhood layers and additional functionalities for the N-CRBM model. The overview of the proposed recommender system is shown in Fig. 4.

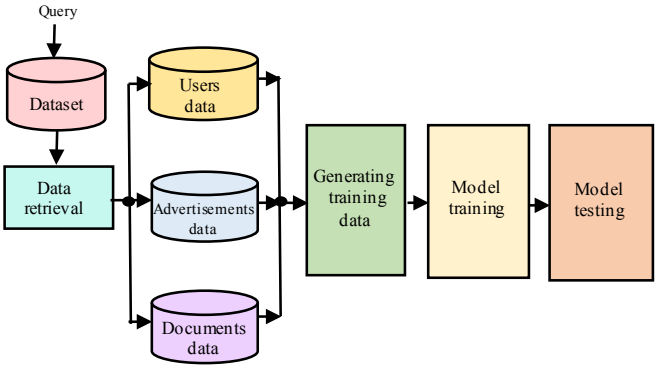

Fig. 4. Overview of the proposed recommender system: training and testing processes.

A. *Datasets for Training the Models*

We randomly choose a *sample* subset of the *Kaggle* dataset to train and test the clusters-based RBM model with CD-1 learning procedure to select various model parameters such as hidden units, learning rates, and activation functions. We then apply these learned parameter values for training and testing the N-CRBM model with *Kaggle* dataset. The 10% of both datasets is selected for testing and the remaining 90% is used for training. The last rating selected by each user is used for predicting the selection of the advertisement. Details of the two datasets are shown in Table II.

TABLE II. DATASETS

Number	***Sample* subset**	***Kaggle* dataset**
Unique advertisements	3×10^3	330×10^3
Unique users	9×10^3	2×10^6
Records	27×10^3	22×10^6

B. *Selection of Model Parameters*

We use the *sample* subset to determine the optimal number of hidden units, learning rates, random vs. learned weights, and activation functions for the clusters-based RBM model. The size of cluster may affect the learning process. Since the created clusters are small and vary in size depending on the popularity of the advertisements, the learning is more efficient than in cases when clusters are larger in size [14].

1) Hidden Units

The number of hidden units affects the RBM learning capacity because they represent the automatically learned features. The clusters-based RBM model is trained and tested with various number of hidden units using logistic sigmoid function. The optimal overall result is achieved using 20 hidden units, as shown in Fig. 5. Experiments with large number of hidden units generate larger RMSE.

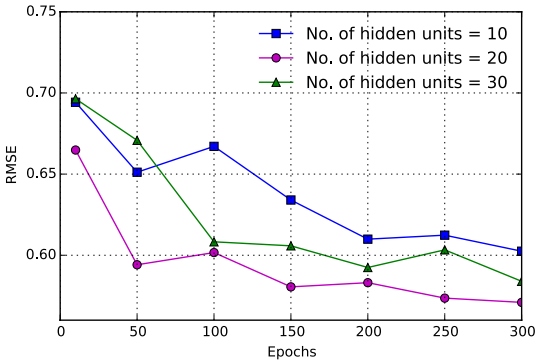

Fig. 5. Effect of the number of hidden units on the model's performance.

2) *Learning Rates*

Before updating the sub-weight matrices, we divide the learned gradient (7)–(9) by the size of the cluster to avoid variation of the learning rate depending to the cluster size, [14]. The optimal results are achieved with learning rate 10^{-3}. However, using pre-trained models (learned weights), the rate 10^{-4} leads to better performance as shown in Fig. 6. The activation function of the visible unit is always set to logistic sigmoid function because it gives the optimal RMSE value compared to tanh and ReLU functions.

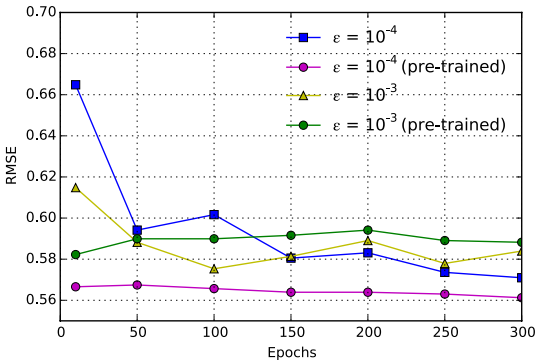

Fig. 6. RMSE values for various learning rates (ϵ).

3) *Activation Functions*

The clusters-based RBM model is trained and tested using various activation functions [20] in the hidden layer, as shown in Fig. 7.

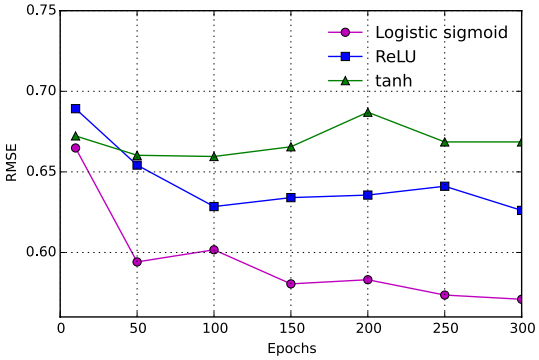

Fig. 7. Performance of clusters-based RBM for various activation functions used in the hidden layer.

VI. Performance of RBMs

Performance of the clusters-based RBM (with *V* and *H* layers only) and N-CRBM (with *V, S, P,* and *H* layers) models is evaluated using the *sample* subset and the *Kaggle* dataset. The learning rate of 10^{-4} with 20 hidden units and logistic sigmoid function are used in training and testing both models. The models are developed using core i7-7700K CPU at 4.4 GHz with 32 GB RAM and pre-trained for better performance. The execution time required for training and testing the models using the *sample* subset is 10-15 min. It required 4 days for the *Kaggle* dataset.

1) *Variation of RMSE Using the Sample Subset*

RMSE of the two models is shown in Fig. 8. The N-CRBM model shows better results because the neighborhood layers (*S* and *P*) supply additional features about the advertisements to the hidden layer through the similarity and popularity scores.

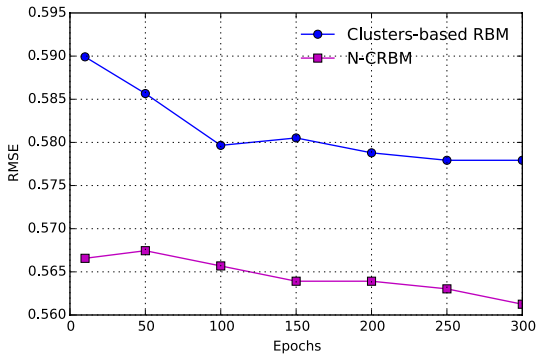

Fig. 8. Variations of RMSE for the clusters-based RBM and N-CRBM models using the *sample* subset.

2) *Variation of RMSE Using the Kaggle Dataset*

For the *Kaggle* dataset, performances of the clusters-based RBM and N-CRBM models improves as the number of iterations (epochs) increases. The RMSE of clusters-based RBM model decreases to 0.49 with 300 epochs while the RMSE of the N-CRBM model decreases to 0.46, as shown in Fig. 9.

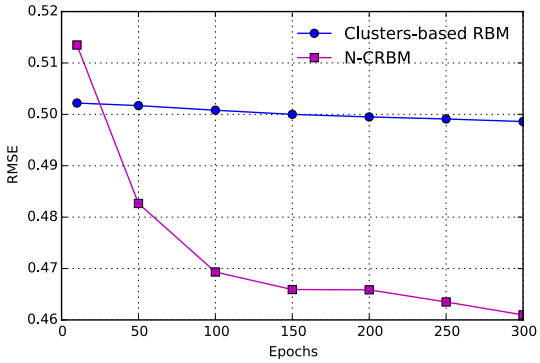

Fig. 9. Improved RMSE of the models using the *Kaggle* dataset.

Both models exhibit reduced RMSE when trained with additional data. Models based on the *Kaggle* dataset predict user preferences better than using the *sample* subset.

3) Variation of Accuracy

Accuracy in predicting selection of advertisements is 76% for the clusters-based RBM models using the *Kaggle* dataset. The N-CRBM models achieve better accuracy in predicting user selection of an advertisement with both the *sample* subset and the *Kaggle* dataset, as shown in Table III. The proposed N-CRBM model is useful to generate a list of recommended advertisements for a user based on the prediction accuracy.

TABLE III. ACCURACY OF THE MODELS

Dataset	Clusters-based RBM (%)	N-CRBM (%)
Sample subset	66.6	68.9
Kaggle dataset	76.0	78.5

4) Variation of Sensitivity

Sensitivity in predicting selection of advertisements for a user is higher in case of the *sample* subset than the *Kaggle* dataset for both the clusters-based RBM and the N-CRBM models. It is noted that 76% of the advertisements in the *Kaggle* dataset are viewed either because users are not interested or not aware of the advertisements. Hence, the training dataset contains very sparse ratings (low ratio of selected advertisements vs. not-selected advertisements) and many neighboring scores are zero. Since the *sample* subset consists of fewer records and may have less zeros in user ratings and neighboring scores, the models may better memorize and learn, as shown in Table IV.

TABLE IV. SENSITIVITY OF THE MODELS

Dataset	Clusters-based RBM (%)	N-CRBM (%)
Sample subset	65.1	69.5
Kaggle dataset	18.7	29.4

VII. CONCLUSION

The recommender systems based on predictions currently play an important role in advertisements of Internet services and products. Due to the sparsity of navigation histories and privacy concerns, it is quite a challenge to predict a user's behavior. In this paper, we propose the N-CRBM model with joint distribution conditional on similarity and popularity scores that are used to predict and recommend advertisements to a user. Before training the model using *Kaggle* dataset, the optimal model parameters were identified by pre-training using the *sample* subset.

Experiments show that learning rate 10^{-4} and 20 hidden units produce relatively good results. The sigmoid function used in hidden layer provides the optimal accuracy. Despite the sparsity of data in the *Kaggle* dataset, the N-CRBM model outperforms the clusters-based RBM model in terms of RMSE, accuracy and sensitivity due to the similarity and popularity scores. The additional neighborhood features help to overcome the CF cold-start problem and enhance the ability of N-CRBM model in recommending advertisements to a user.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez, "Recommender systems survey," *J. Knowledge Based Systems*, vol. 46, pp. 109–132, July 2013.

[2] K. Lang, "Newsweeder: learning to filter netnews." in *Proc. Int. Conf. Mach. Learning*, Tahoe City, CA, USA, July 1995, pp. 331–339.

[3] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proc. ACM Int. Conf. Multimedia*, Orlando, FL, USA, Nov. 2014, pp. 627–636.

[4] (Apr. 04, 2017) Outbrain click prediction contest. [Online]. Available: https://www.kaggle.com/c/outbrain-click-prediction.

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

[6] H. T. Cheng, L. Koc, J. Harmsen, et al., "Wide and deep learning for recommender systems*,"* in *Proc. ACM Workshop on Deep Learning for Recommender Syst.*, Boston, MA, USA, Sept. 2016, pp. 7–10.

[7] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for YouTube recommendations," in *Proc. ACM Conf. Recommender Syst.*, Boston, MA, USA, Sept. 2016, pp. 191–198.

[8] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. ACM Int. Conf. Knowledge Discovery and Data Mining*, Sydney, Australia, Aug. 2015, pp. 1235–1244.

[9] H. Wang and W. Li, "Relational collaborative topic regression for recommender systems," *IEEE Trans. Knowledge and Data Eng.*, vol. 27, no. 5, pp. 1343–1355, May 2015.

[10] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. Int. Conf. Mach. Learning*, Corvallis, OR, USA, June 2007, pp. 791–798.

[11] (Apr. 04, 2017) Netflixprize movie rating contest. [Online]. Available: http://www.netflixprize.com/index.html.

[12] B. Abdollahi and O. Nasraoui, "Explainable restricted Boltzmann machines for collaborative filtering," in *Proc. ICML Workshop Human Interpretability in Mach. Learning*, New York, NY, USA, June 2016, pp. 31–35.

[13] Y. Liu, Q. Tong, Z. Du, and L. Hu, "Content-boosted restricted Boltzmann machine for recommendation," in *Lecture Notes in Computer Science,* D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, C. Pandu Rangan, B. Steffen, D. Terzopoulos, D. Tygar, G. Weikum, Eds. Springer, 2014, vol. 8681, pp. 773–780.

[14] G. E. Hinton, "A practical guide to training restricted Boltzmann machines," in *Lecture Notes in Computer Science, Neural Networks: Tricks of the Trade*, G. Montavon, G. B. Orr, K. R. Müller, Eds. Springer, 2012, vol. 7700, pp. 599–619.

[15] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," in *Proc. Int. Conf. Mach. Learning*, San Diego, CA, USA, July 2008, pp. 1064–1071.

[16] M. Eastwood and J. Chrisina, "Restricted Boltzmann machines for pre-training deep Gaussian networks." in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Dallas, TX, USA, Aug. 2013, pp. 1–8.

[17] H. V. Nguyen and L. Bai, "Cosine similarity metric learning for face verification," in *Proc. Asian Conf. Comput. Vision,* Queenstown, New Zealand, Nov. 2010, pp. 709–720.

[18] (Apr. 04, 2017) Restricted Boltzmann machines. [Online]. Available: https://github.com/echen/restricted-boltzmann-machines.

[19] A. Gunawardana and G. Shani, "A survey of accuracy evaluation metrics of recommendation tasks," *J. Mach. Learning Res.*, vol. 10, pp. 2935–2962, Dec. 2009.

[20] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006, Ch. 2, p. 133.