Distributed Representation for Neighborhood-based Collaborative Filtering

Yi Yang
Institute of Automation
Chinese Academy of Sciences
Beijing, China
yangyi@ia.ac.cn

Guigang Zhang
Institute of Automation
Chinese Academy of Sciences
Beijing, China
guigang.zhang@ia.ac.cn

Jian Wang*
Institute of Automation
Chinese Academy of Sciences
Beijing, China
jian.wang@ia.ac.cn

Weixing Huang
Institute of Automation
Chinese Academy of Sciences
Beijing, China
hwx0904@vip.sina.com

*Corresponding author: Jian Wang

ABSTRACT

Collaborative filtering is widely used in recommender systems. When training data are extremely sparse, neighbor selection methods work ineffectively. To address this issue, this paper proposes a distributed representation model that represents users as low-dimensional vectors for neighbor selection by considering the chronological order of users' ratings. Experiments show that the proposed method outperforms the state-of-the-art methods solving the sparsity problem with regard to precision and ranking quality.

CCS CONCEPTS

• **Information systems → Recommender systems**

KEYWORDS

Recommender system, collaborative filtering, neighbor selection, word embedding, word2vec, NLP, deep learning

1 INTRODUCTION

In the last decade, Collaborative Filtering (CF) [1] is widely used in recommender systems (RS). The neighborhood-based CF is one of the famous CF methods. It predicts the items that users may like by analyzing the co-occurrence of users' ratings. An open issue of CF is the sparsity problem. When the rating matrix is sparse, CF will have poor results. Researchers proposed solutions, for example, clustering-based CF [2] and matrix factorization [3] [5] [4]. However, the rating matrix of a large RS or of a newly built RS for large-scale existing users is extremely sparse and high dimensional. In this case, the solutions suffer from the high-dimensional extremely sparse data [6]. The sparsity problem of CF is still critical when meeting the high-dimensional extremely sparse data, for example, the density is below the range of 10^{-3}. In this case, the effectiveness of the existing approaches will rapidly reduce. Therefore, they can hardly deal with the situation well. Our contribution in this paper is to propose a neighbor selection method for the user-based CF by considering the chronological order of the ratings in order to address the high-dimensional sparse issue. Our method presents users with low-dimensional vectors by applying the distributed representation. Experiments show that our method can provide better precision and ranking quality for the high-dimensional extremely sparse data.

We organize the rest of the paper as follows. Section 2 presents the related work and background. Section 3 introduces our method and Section 4 shows experiments to evaluate our method. Finally, conclusion and future work are given in Section 5.

2 BACKGROUND AND RELATED WORK

2.1 Neighborhood-based Collaborative Filtering

The neighborhood-based CF [7] includes user-based CF [8] and item-based CF [9]. The user-based CF has the following main steps: 1) compute user similarity based on the rating matrix; 2) find K nearest neighbors; 3) predict ratings of the unrated items according to the neighbors' ratings; 4) find the first N items for recommendation. The item-based CF is similar to the user-based CF while identifying item-neighbors instead of user-neighbors.

Neighbor selection has been studied in previous work. Verstrepen and Goethals [10] unified the neighbor selection algorithms for the user-based CF and the item-based CF. The study [11] proposed a probabilistic neighbor selection method by considering similarity levels and weighted sampling of neighbors. Bellogin and Parapar [12] introduced a normalized cut clustering-based neighbor selection method for the user-based CF. The user interests drift over time. Koren [13] discussed the modeling methods for the time-aware interests for CF. The study [14] proposed a probabilistic generative

978-1-5386-2937-6/17 $31.00 © 2017 IEEE
DOI 10.1109/ISM.2017.104

model to represent the interest change. Koenigstein, Dror, and Koren [15] presented a music recommendation method with consideration of the different temporal dynamics of music ratings. The research [16] proposed an algorithm to compute the changing time weights of items.

2.2 Distributed Representation

Word embedding is a widely used distributed representation method [17] in NLP. It represents words with fixed-length low-dimensional vectors. Word2Vec [18][19] is a state-of-the-art word embedding technique that builds word vectors by evaluating language models. Word2Vec is a simplified deep learning method that has a neural network with three layers: input layer, projection layer, and output layer. Word2Vec can be implemented by the language model CBOW (Continuous Bag-of-Words) that describes a prediction process of a word according to the contexts of the word. For a corpus C consisting of a sequence of words $w_1, \dots, w_z$, the cost function is defined as

$$J = \frac{1}{Z}\sum_{w \epsilon C} \log p(w|Context(w)) = \frac{1}{Z}\sum_{w \epsilon C} \log \frac{\exp(v_{sum}^{\mathrm{T}}\theta)}{\sum_{i=1}^{V} exp(v_i^{\mathrm{T}}\theta)} \quad (1)$$

where $Context(w_t) = (w_{t-s}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+s})$ and s is the size of sliding window. The objective of the model is to maximize Equation (1). For a given word $w_t \in C$, the input of the model is $Context(w_t)$. Each $w_{t'} \in Context(w_t)$ is randomly initialized with a unique fixed-length vector $v_{t'}$. The projection layer builds a vector $v_{sum} = \sum v_{t'}$ that sums the contextual vectors. The output layer works on predicting w_t that is encoded with Huffman code. The model applies the Softmax function for the prediction (see Equation (1)), where V is the size of the vocabulary of corpus C, and θ is the weight parameters. By using stochastic gradient descent (SGD), the contextual word vectors and the parameters θ can be updated. After rounds of iterations, the word vectors can be obtained.

Paragraph Vector (also called Doc2Vec) [20] extends Word2Vec for learning paragraph vectors that are used to compute paragraph similarity. Doc2Vec provides PV-DM (Distributed Memory Model of Paragraph Vector). The most important difference to CBOW is that in each round of iteration, Doc2Vec samples training words from a paragraph and treats the paragraph id as an additional training word. The paragraph vector has the same dimension size as the word vectors. Afterwards, the input vectors are averaged in the projection layer. Finally, the word vectors can be computed using SGD in the output layer.

3 Distributed Representation for Neighbor Selection

In this paper, we propose a distributed representation model called Rating2Vec for neighbor selection of the user-based CF. As shown in Figure 1, the learning framework of Rating2Vec is similar to that of Doc2Vec. We build Rating2Vec considering user interests that are represented by ratings in the rating matrix. The rating score means how much a user likes an item. The higher the score is, the stronger the user interests are. In Rating2Vec, we treat the rating score as the weight of item vector. For computing the model, we use Hierarchical Softmax based on the Huffman Tree that structures items and users according to their frequency. For items, we treat the rating scores of an item as its frequency. We consider the chronological order of the ratings in our model because user interests change over time. We select training data using a sliding windows like in Word2Vec. The order of ratings in a relative short time interval usually can hardly represent interest drift. The inner order may introduce over-fitting of the training. Therefore, we do not consider the inner order of ratings.

Formally, we define a rating matrix $RM^{m \times n} = \{r_{u,i}\}$ where user $u \in U^{1 \times n}$ and item $i \in I^{m \times 1}$. A user is represented as a continuous sequence of ratings with the chronological order: $u = (r_1, \dots, r_n)$. The user vectors can be obtained by maximizing Equation (2) for each round of iteration.

$$\mathcal{L}_t = log\, p\,(i_t|u, i_{t-s}, \dots, i_{t-1}, i_{t+1}, \dots, i_{t+s}) \quad (2)$$

Comparing with Doc2Vec, we add a weighting layer between the input layer and the projection layer. In this layer, the vector $v_{t'}$ of item $i_{t'} \in Context(i_t)$ is weighted by multiplying $v_{t'}$ and rating score $r_{t'}$. The weight of a user vector is always 1. The projection layer sums the vectors: $v_{sum} = v_u + \sum_{t'} v_{t'} r_{t'}$. The output layer trains the prediction model by updating vectors and parameters.

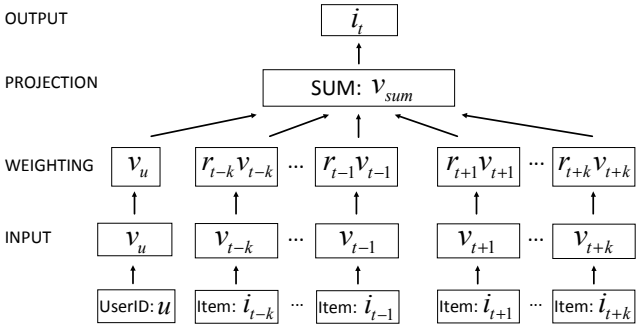


Figure 1: Learning Process of Rating Vector in Rating2Vec.

ALGORITHM 1 shows the algorithm of Rating2Vec. At first, for a given dimension d, it initializes the parameter set θ, users and items of the given rating matrix with random d-dimensional vectors (line 1). Then, it uses SGD to train the vectors. At the beginning, a user u is randomly selected from the user set U and an item $item_t$ that is rated by u is randomly selected from the item set I. t is the index of the item in the sequence in chronological order of the rated items of u (lines 4-5). The context of $item_t$ are sampled with the windows size $2\,s$ (line 6). The contextual item vectors are weighted by the corresponding rating scores. The user vector and the weighted contextual item vectors are added up (lines 7-11). Afterwards, the gradients of θ and v_{sum}, and the accumulative error e are computed. The parameter set θ^{item_t} is updated with the learning rate η and the gradient $\nabla\theta^{item_t}\mathcal{L}_t$ (lines 12-16). The user vector and the contextual item vectors are updated with the accumulative error e (lines 17-20). After rounds of iterations, the results are obtained including the parameter set θ, the contextual item vectors, and the user vectors (line 23).

ALGORITHM 1: Rating2Vec Algorithm

Input: rating matrix $RM^{m \times n}$ having a user set U, and an item set I; vector dimension $d \in \mathbb{N}^+$; radius $s \in \mathbb{N}^+$; total number of iterations $total \in \mathbb{N}^+$ and $total > 1$; learning rate of stochastic gradient descent η.

Output: vector set v, parameter set θ.

$initialization()$
$round \leftarrow 0$
while $round < total$ **do**
  $u \leftarrow RandomUser(U)$
  $item_t \leftarrow RandomItem(u)$
  $context \leftarrow ContextItems(u, t, s)$
  $v_{sum} \leftarrow v_u$
  for $t' \in \{t-s, \dots, t-1, t+1, \dots, t+s\}$ **do**
    $r_{t'} \leftarrow Rating(u, item_{t'})$ /* $item_{t'} \in context$ */
    $v_{sum} \leftarrow v_{sum} + v_{t'} r_{t'}$
  end
  $e \leftarrow 0$
  for $j = 2 : l$ **do** /* l: length of the Huffman code of i_t */
    $e \leftarrow e + \eta \frac{\partial \mathcal{L}_t}{\partial v_{sum}}$
    $\theta_{j-1}^{item_t} \leftarrow \theta_{j-1}^{item_t} + \eta \frac{\partial \mathcal{L}_t}{\partial \theta_{j-1}^{item_t}}$
  end
  for $t' \in \{t-s, \dots, t-1, t+1, \dots, t+s\}$ **do**
    $v_{t'} \leftarrow v_{t'} + e$
  end
  $v_u \leftarrow v_u + e$
  $round \leftarrow round + 1$
end
return v, θ

The training complexity of each round of iteration in Rating2Vec combines the complexities of the weighting layer, the projection layer, and the output layer: $2s \times d + 2s \times d + d \times log_2 V$, where V denotes size of the vocabulary and d denotes dimension size of vector. Rating2Vec builds feature vectors of users and items. After that, the user-based CF computes user similarity according to the user vectors using Pearson correlation coefficient.

4 EXPERIMENTS

4.1 Dataset

We used Netflix Prize dataset [21] for the experiments. The dataset contains 17,770 movies, over 480,000 users, and c.a. 1 billion ratings. As shown in Table 1, we sampled five high-dimensional sparse sub-datasets from the Netflix dataset. The densities are in the range from 1.7×10^{-3} to 6.2×10^{-2}. In each sub-dataset, we sampled 80% data for training and 20% data for test.

Table 1: Experimental Datasets

Dataset	User	Item	Density
1	22573	16294	0.0017
2	3596	8647	0.0035
3	22342	17381	0.0064
4	1024	9247	0.011
5	166	8132	0.062

4.2 Metrics

The experiments contain the tasks regarding top-N recommendations (N=5, 10, 20). We evaluated the algorithms for precision and nDCG (Normalized Discounted Cumulative Gain) [22]. The precision measures the correctness of recommendations and the nDCG indicates the ranking quality of recommendations.

4.3 Baselines

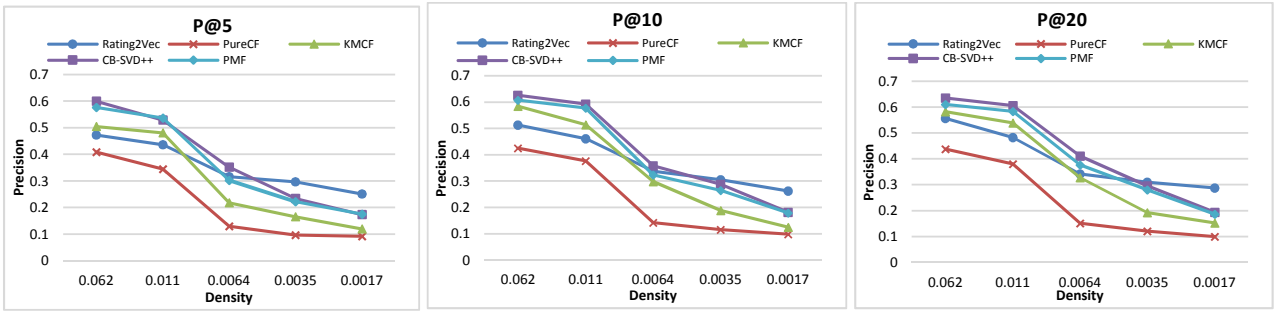
We selected baseline methods from classic CF, clustering-based CF, and matrix factorization methods. We applied the traditional CF (PureCF), the K-means-based CF (KMCF) [2], the clustering-based SVD++ (CB-SVD++) [23], and the probabilistic matrix factorization (PMF) [5] as the baseline methods. PureCF directly uses the original ratings as the features of users and items. Rating2Vec-based CF (Rating2Vec) improves the PureCF with distributed user vectors for neighbor selection. KMCF uses the k-means clustering for users, and then selects neighbors from related clusters. For Rating2Vec, PureCF, and KMCF, after identifying user vectors, these methods can predict users' ratings based on neighbors' ratings. In different ways, CB-SVD++ and PMF predict users' ratings basically by factorizing the rating matrix instead of analyzing neighbors' ratings. All the baseline methods have the same rest steps for obtaining the recommended items after the rating prediction.

4.4 Settings

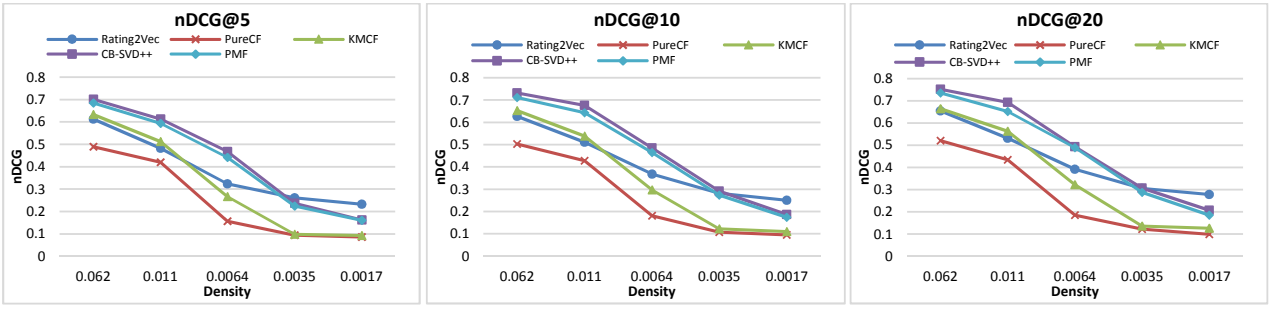
For the experiments, we set the neighbor size: n=50; the dimension size for Rating2Vec: d=100. The computer server has four Intel Xeon 8-core CPUs and 128 GB total memory.

4.5 Results

Figure 2 illustrates the diagrams of the experimental results for precision and ranking quality.



(a) Precision: P@5, P@10, and P@20



(b) nDCG: nDCG@5, nDCG@10, nDCG@20

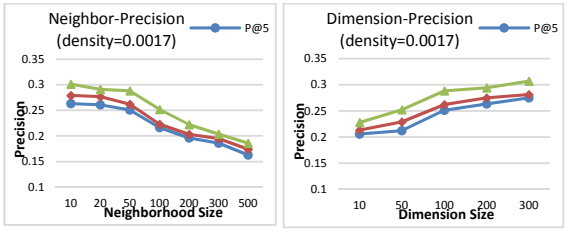
Figure 2: Experiments for precision and nDCG.

Figure 2 (a) compares the methods regarding precision. For the datasets with densities of 0.062 and 0.011, CB-SVD++ and PMF have higher precision than that of other experimental methods. KMCF is better than Rating2Vec in this case. PureCF has the worst results among the CF methods in the experiments. When the dataset becomes more sparse (density=0.0064), the precision of CB-SVD++ and PMF reduce to the similar level of Rating2Vec. In this case, Rating2Vec outperforms KMCF. PureCF still has the worst result. When the density drops to 0.0035, precision of CB-SVD++ and PMF reduce below that of Rating2Vec, particularly for P@5. PureCF and KMCF are overtaken by Rating2Vec in this case. When using the extremely sparse dataset (density=0.0017), Rating2Vec outperforms the baseline methods. The precision experiments show that our method definitely has better precision than that of the commonly used solutions of sparsity problem when data are extremely sparse.

Figure 2 (b) presents the experimental results regarding nDCG. Similar to the precision experiments, for the not extremely sparse datasets (density>0.011), CB-SVD++, PMF, and KMCF are able to outperform Rating2Vec. When the density reduces to 0.0064, Rating2Vec becomes better than KMCF, but dose not as good as the matrix factorization-based baseline methods. When the dataset is very sparse (density<0.0035), the advantage of Rating2Vec becomes obvious.

In sum, the experiments show that our method can provide better precision and ranking quality of recommendations for the high-dimensional sparse datasets. The benefits become stronger when the density is below the range of 10^{-3}. Therefore, we conclude that the Rating2Vec-based CF outperforms the state-of-the-art CF methods for the extremely sparse datasets.

We also performed experiments for the variable settings of Rating2Vec. Figure 3 (a) presents the experimental results for neighbor size. The results show that the precision reduces rapidly when Rating2Vec has more than 50 neighbors. Figure 3 (b) presents the experimental results for dimension size of vectors. We evaluated the dimension size between 10 and 300. The results show that the large dimension size gives better precision. However, because the large dimension size will increase the complexity, usually we need a trade-off size. The trends of the precision show that the growth of precision becomes slow when the dimension size is over 100. Therefore, we conclude that the dimension size of 100 is suitable for the extremely sparse datasets.



(a) Neighbor size (b) Dimension size

Figure 3: Experiments for Variables of Rating2Vec.

Moreover, the variable experiments also show that P@20 is higher than P@5 and P@10. That means, the larger the number of recommendation items is, the better the precision is. Because we only used the number between 5 and 20 in our study, it still needs more experiments to validate the conclusion.

5 CONCLUSIONS

The neighbor selection method strongly influences collaborative filtering. The state-of-the-art neighbor selection methods do not work well for the high-dimensional sparse datasets. In this paper, we propose a feature modeling method Rating2Vec for neighbor selection by considering the time factor of ratings. We use the distributed representation in our method to represent users with the fixed-length low-dimensional vectors. The experiments show that the Rating2Vec-based CF method can provide better top-N recommendations for the extremely sparse data than those of the commonly used CF methods with regard to precision and ranking quality. In the future, we will extend our method to item-based CF. Furthermore, we will focus on different distributed representation models, e.g., GloVe, for collaborative filtering.

ACKNOWLEDGMENTS

We thank the National Key Technology Support Program of the National '12th Five-Year-Plan of China' under Grant No. 2015BAK25B04 for partially supporting our work. We also thank

all colleagues and graduate students who helped us for our visualization system and experiments.

REFERENCES

[1] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. Commun. ACM 35, 12 (December 1992), 61-70.

[2] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. 2005. Scalable collaborative filtering using cluster-based smoothing. In Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '05). ACM, New York, NY, USA, 114-121.

[3] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '08). ACM, New York, NY, USA, 426-434.

[4] Yehuda Koren. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Trans. Knowl. Discov. Data 4, 1, Article 1 (January 2010), 24 pages.

[5] Salakhutdinov and A. Mnih. Probabilistic matrix factorization. Advances in neural information processing systems, pages 1257–1264, 2008.

[6] Miha Grčar, Blaž Fortuna, Dunja Mladenič, & Marko Grobelnik. 2006. Knn versus svm in the collaborative filtering framework. Workshop on Knowledge Discovery on the Web, 251-260.

[7] X. Ning, C. Desrosiers, and G. Karypis. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. Recommender Systems Handbook. Springer US, (2015), 107-144.

[8] Jon Herlocker, Joseph A. Konstan, and John Riedl. 2002. An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. Inf. Retr. 5, 4 (October 2002), 287-310.

[9] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (WWW '01). ACM, New York, NY, USA, 285-295.

[10] Koen Verstrepen and Bart Goethals. 2014. Unifying nearest neighbors collaborative filtering. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, 177-184.

[11] Panagiotis Adamopoulos and Alexander Tuzhilin. 2014. On over-specialization and concentration bias of recommendations: probabilistic neighborhood selection in collaborative filtering systems. In Proceedings of the 8th ACM Conference on Recommender systems (RecSys '14). ACM, New York, NY, USA, 153-160.

[12] A. Bellogin and J. Parapar. 2012. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In Proceedings of the sixth ACM conference on Recommender systems (RecSys '12). ACM, New York, NY, USA, 213-216.

[13] Y. Koren. 2010. Collaborative filtering with temporal dynamics. Commun. ACM 53, 4 (April 2010), 89-97.

[14] Wei Chen, Wynne Hsu, and Mong Li Lee. 2013. Modeling user's receptiveness over time for recommendation. In Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval (SIGIR '13). ACM, New York, NY, USA, 373-382.

[15] Noam Koenigstein, Gideon Dror, and Yehuda Koren. 2011. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In Proceedings of the fifth ACM conference on Recommender systems (RecSys '11). ACM, New York, NY, USA, 165-172.

[16] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In Proceedings of the 14th ACM international conference on Information and knowledge management (CIKM '05). ACM, New York, NY, USA, 485-492.

[17] G. E. Hinton.1986. Learning Distributed Representations of Concepts. Proceedings of the Eighth Annual Conference of the Cognitive Science Society. Hillsdale, NJ: Erlbaum. (1986), 1-12.

[18] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. Advances in Neural Information Processing Systems, 26(2013), 3111-3119.

[19] T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. Computer Science (2013).

[20] Quoc V. Le and T. Mikolov. 2014. Distributed Representations of Sentences and Documents. Computer Science 4(2014): 1188-1196.

[21] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. 2007. KDD Cup and workshop 2007. SIGKDD Explor. Newsl. 9, 2 (December 2007), 51-52.

[22] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '00). ACM, New York, NY, USA, 41-48.

[23] Nima Mirbakhsh and Charles X. Ling. 2013. Clustering-based factorized collaborative filtering. In Proceedings of the 7th ACM conference on Recommender systems (RecSys '13). ACM, New York, NY, USA, 315-318.