Improving Collaborative Filtering with Long-Short Interest Model

Chao Lv, Lili Yao, Yansong Feng* and Dongyan Zhao

Institute of Computer Science and Technology
Peking University, Beijing 100871, China
{lvchao,yaolili,fengyansong,zhaodongyan}@pku.edu.cn

Abstract. Collaborative filtering (CF) has been widely employed within recommender systems in many real-world situations. The basic assumption of CF is that items liked by the same user would be similar and users like the same items would share a similar interest. But it is not always true since the user's interest changes over time. It should be more reasonable to assume that if these items are liked by the same user in the same time period, there is a strong possibility that they are similar, but the possibility will shrink if the user likes them in a different time period. In this paper, we propose a long-short interest model (LSIM) based on the new assumption to improve collaborative filtering. In special, we introduce a neural network based language model to extract the sequential features on user's preference over time. Then, we integrate the sequential features to solve the rating prediction task in a feature based collaborative filtering framework. Experimental results on three MovieLens datasets demonstrate that our approach can achieve the state-of-the-art performance.

Keywords: Recommender System, Collaborative Filtering, Long-Short Interest Model

1 Introduction

In the modern era of information overload, recommender system (RS) has become more and more popular in many real-world situations. Lots of websites (e.g. Amazon, Netflix, Alibaba and Hulu) use recommender system to target customers and provide them with useful information. An excellent recommendation system can effectively increase the amount of sales. For instance, 80% of movies watched on Netflix come from their recommender system [3].

Lots of classical recommendation methods have been proposed during the last decade, and they can be categorized into two classes: content based methods and collaborative filtering based methods. Content based methods [11] take advantage of user profiles and item properties for recommendation. While collaborative filtering based approaches [15] utilize the past interactions or preferences, such as users' ratings on items, without using user or product content information for

*Corresponding author.

recommendation. Collaborative filtering based approaches have attracted more attention due to their impressive performance, hence they develop for many years and keep to be a hot area in both academia and industry.

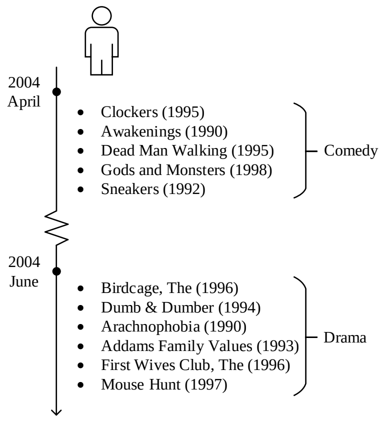

Fig. 1. The preference records of user whose id is 5988 in MovieLens-1M dataset, which are sorted by their rated time.

Collaborative filtering assumes that items liked by the same user would be similar and users like the same items would share a similar interest. However, it is not always true because the user's interest changes over time. For example, given a user in MovieLens-1M dataset whose id is 5988, Figure 1 shows the movies he watched sorted by the rating time. We can find that this user liked watching comedy movies in April 2004 and changed to love watching drama movies in June 2004. These movies are going to be treated similarly in conventional collaborative filtering, but they are not in fact. A more reasonable assumption, aka long-short interest assumption, should be that items liked by the same user in the same time period have a higher possibility to be similar than items liked by the same user in different time period.

Inspired by paragraph2vec algorithm [6] for learning vector representations of words which take advantage of a word order observed in a sentence, we introduce a long-short interest model (LSIM) to extract sequential features of users and items based on the new assumption. As illustrated in Figure 2, user is similar with the sentence, both of them contains a sequence following some order, and items are similar with words because both of them follow the law that the closer they are, the more similar they are. To verify the effectiveness of the learned sequential features of users and items, we integrate them as side information to solve the rating prediction task in a feature based collaborative filtering framework.

The main contributions of this paper include: (1) We introduce a long-short interest model (LSIM) to extract sequential features of users and items based on

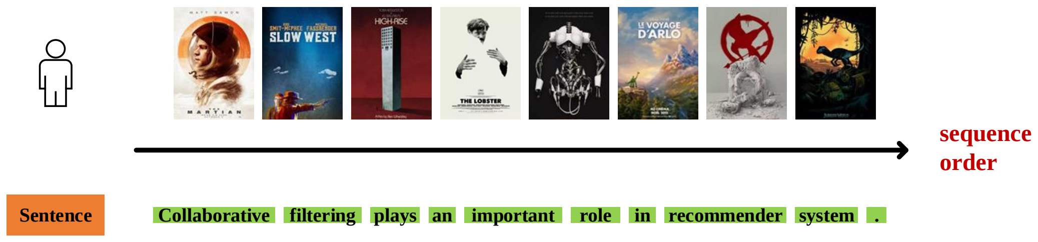
sequence order
Sentence
Collaborative filtering plays an important role in recommender system .

Fig. 2. Paragraph2vec learns vector representations of sentences and words based on the word order while LSIM extracts sequential features of users and items based on the rating order.

the long-short interest assumption. (2) We demonstrate the effectiveness of the sequential features via integrating them as side information to solve the rating prediction task. (3) Experiments on three public MovieLens show that LSIM can achieve the state-of-the-art performance.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. Then, we describe our long-short interest model and the feature based collaborative filtering framework in Section 3. The experimental results, as well as the comparisons with baseline system, are shown in Section 4. Finally, we conclude the paper and outline our future work in Section 5.

sectionRelated Work Our work is closely related to collaborative filtering and neural network language model. We will discuss them in the following subsections.

1.1 Collaborative Filtering

Matrix factorization (MF) is the most popular collaborative filtering methods, their success at the Netflix competition [4] have demonstrated their amazing strength, and lots of variants of it have been proposed in the following works. Basically, the given ratings matrix $\mathbf{R} \in \mathbb{R}^{N*M}$ consisting of the item preferences of the users can be decomposed as a product of two low dimensional matrices $\mathbf{U} \in \mathbb{R}^{N*K}$ and $\mathbf{V} \in \mathbb{R}^{K*M}$. $\mathbf{U}$ could be treated as a user-interest matrix while $\mathbf{V}$ could be treated as an item-interest matrix. K is the amount of interest. The decomposition can be carried out by a variety of methods such as singular value decomposition (SVD) based approaches [9], non-negative matrix factorization approach [7] and regularized alternative least square (ALS) algorithm [16]. Meanwhile, non-linear algorithms are proposed to catch subtle factors, such as Non Linear Probabilistic Matrix Factorization [5], Factorization Machines [12] and Local Low Rank Matrix Approximation [8]. However, these methods group users and treat items they rated equally, which will lose the sequential features to describe the long-short interest.

1.2 Neural Network Language Model

Traditional language model uses a one-hot representation to represent each word as a feature vector, where these feature vectors have the same length as the size of vocabulary, and the position that corresponds to the observed word is equal to 1, and 0 otherwise. However, this approach often exhibits significant limitations in practical tasks, suffering from high dimensionality and severe data sparsity. Mikolov *et al.* [10] proposed the word2vec algorithm to address these issues. They take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent, and have generalized the classic n-gram language models by using continuous variables to represent words in a vector space. The continuous bag-of-words (CBOW) and skip-gram (SG) language models are highly scalable for learning word representations from large-scale corpora. The word2vec algorithm breaks the semantic gap between words. For example, “trade” and “deal” are totally different words in the one-hot representation, but they are similar in word2vec distribution representation.

2 Our Approach

2.1 Problem Definition

Given N users and M items, the rating r_{ij} is the rating given by the i^{th} user for the j^{th} item. In the common real-world situations, users usually rate on a fraction of items, not on the whole items. Therefore, those ratings entail a big and sparse matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$. The goal of recommender system is to make a prediction on the missing ratings. Based on that, we will know the preference of a user on the items he never rates, and recommend high score items to him. Table 1 summarizes the symbols used in our approach.

Table 1. Summary of notations.

Notation	**Description**
N	Number of users
M	Number of items
K	Dimension of latent factors
D	Dimension of sequential features
$\mathbf{R} \in \mathbb{R}^{N \times M}$	Rating matrix
$\mathbf{U} \in \mathbb{R}^{N \times K}$	Latent factors of users
$\mathbf{V} \in \mathbb{R}^{M \times K}$	Latent factors of items
$\mathbf{X} \in \mathbb{R}^{N \times D}$	sequential features of users
$\mathbf{Y} \in \mathbb{R}^{M \times D}$	sequential features of items

2.2 Long-short Interest Model

The basic assumption of collaborative filtering is that items liked by the same user would be similar or users like the same items would share a similar interest. However, in real-world situations, it is not always true because users' interest may change over a long time period. Meanwhile, the interest distribution of a user in a fixed time period are stable and don't change too much. To describe this phenomenon, we propose the definition of **long interest** and **short interest**.

– **long interest** reflects the interest distribution of a user in a long time period, and it is reflected in the whole items list of the user's preference.
– **short interest** reflects the interest distribution of a user in a short time period, and it is reflected in a fraction of the whole items list of the user's preference in a fixed length sliding window.

Under this definition, two assumptions are proposed as below.

1. items liked in the same short interest of the same long interest have a higher possibility to be similar than ones liked in different short interest of the same long interest.
2. the more times items show in the same short interest of different long interest, the higher possibility they are similar.

For example, as illustrated in the top of Figure 3, **item2** should be more similar with **item1** than **item5** in the low dimensional embedding space.

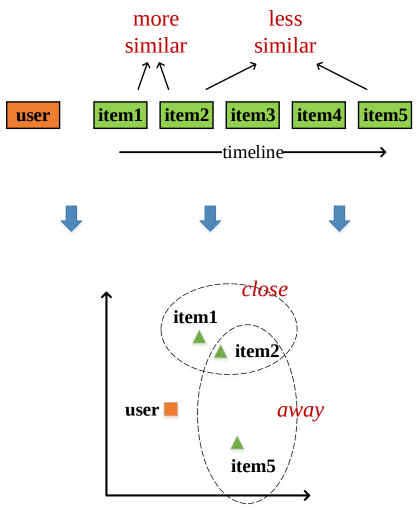

Fig. 3. A Example for Short Interest and Long Interest

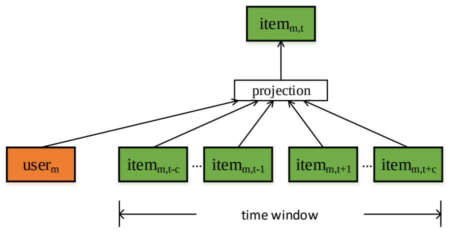

Fig. 4. Embedding Model for Extracting Interest Similarity from Users and Items

Inspired by paragraph2vec algorithm [6] for learning vector representations of words which take advantage of a word order observed in a sentence, we introduce a neural network based language model to carry out the embedding of sequential features. The embedding model simultaneously learns vector representations of users and items by considering the user as a global context, and the architecture of the embedding model is illustrated in Figure 4.

The training data set was derived from users' interaction timeline T, which comprises users $x_i(i = 1, 2, ..., N)$ and their interacted items ordered by the interacted time, y_{i_1}, y_{i_2}, ..., $y_{i_{L_i}}$ [1], where L_i denotes number of items interacted by user x_i, which is much less than the amount of items M. To characterize the *long interest*, we consider the whole items list as the context and generate the long interest of the current user. To characterize the *short interest*, we consider items in the same local interest as the context and generate items in it one by one with the help of long interest. More formally, objective of the embedding model is to maximize the log-likelihood over the set of T of all the interaction timeline,

$$\sum_{i=1}^{N}\left(p(x_i|y_{i_1}, y_{i_2}, ..., y_{i_{L_i}}) + \sum_{j=1}^{L_i} p(y_{i_j}|y_{i_{j-c}} : y_{i_{j+c}}, x_i)\right) \tag{1}$$

where c is the time window size, $y_{i_{j-c}} : y_{i_{j+c}}$ denotes the sequence $y_{i_{j-c}}, y_{i_{j-c+1}}, ..., y_{i_{j+c}}$ excluding y_{i_j}.

$p(x_i|y_{i_1}, y_{i_2}, ..., y_{i_{L_i}})$ is the probability to generate the long interest of u_i based on all items he interacted. The prediction task is typically done via a multi-class classifier, such as softmax. There, we have

$$p(x_i|y_{i_1}, y_{i_2}, ..., y_{i_{L_i}}) = \frac{exp(\overline{\mathbf{v}}_1^{\mathrm{T}} \mathbf{v}'_{x_i})}{\sum_{x'} exp(\overline{\mathbf{v}}_1^{\mathrm{T}} \mathbf{v}'_{x'})} \tag{2}$$

where $\mathbf{v}'_{x_i}$ is the output vector representation of x_i, and $\overline{\mathbf{v}}_1$ is averaged input vector representation of all the items interacted by user x_i, i.e.

[1] we use symbol x and y instead of classic u and v to avoid confusion between v and vector symbol $\mathbf{v}$ in neural network language model.

$$\overline{\mathbf{v}}_1 = \frac{\sum_{j=1}^{T_i} \mathbf{v}_{y_{i_j}}}{T_i} \tag{3}$$

$p(y_{i_j}|y_{i_{j-c}} : y_{i_{j+c}}, x_i)$ is the probability to generate y_{i_j} based on items in the same short interest and the user's long interest. Similarly, using softmax multi-class classifier we have

$$p(y_{i_j}|y_{i_{j-c}} : y_{i_{j+c}}, x_i) = \frac{exp(\overline{\mathbf{v}}_2^{\mathrm{T}} \mathbf{v}'_{y_{i_j}})}{\sum_{y'} exp(\overline{\mathbf{v}}_2^{\mathrm{T}} \mathbf{v}'_{y'})} \tag{4}$$

where $\mathbf{v}'_{y_{i_j}}$ is the output vector representation of y_{i_j}, and $\overline{\mathbf{v}}_2$ is averaged input vector representation of items int the same short interest and corresponding long interest x_i.

$$\overline{\mathbf{v}}_2 = \frac{\mathbf{v}_{x_i} + \sum_{-c \le k \le c, k \ne 0} \mathbf{v}_{y_{i_{j+k}}}}{2c+1} \tag{5}$$

Stochastic Gradient Descent (SGD) are used as the training method, hierarchical softmax and negative sampling are two main approaches to accelerate the computation, and we use negative sampling approach in this paper.

2.3 Feature based Collaborative Filtering

Feature based collaborative filtering [1] is a variety of collaborative filtering, it allows us to build factorization models incorporating side information such as temporal dynamics, neighborhood relationship, and hierarchical information compared with conventional collaborative filtering.

There are two kinds of side information in collaborative filtering: user side information and item side information. Feature based collaborative filtering summarizes the two factors as feature vectors (denoted by $\mathbf{u}_i \in \mathbb{R}^n$ and $\mathbf{v}_j \in \mathbb{R}^m$) and predicts the preference score $\hat{r}$ as

$$\hat{r}_{ij} = \sum_{k=1}^{n} \alpha_k \mathbf{u}_{ik} + \sum_{k=1}^{m} \beta_k \mathbf{v}_{jk} + \left(\sum_{k=1}^{n} \mathbf{u}_{ik} \mathbf{p}_k \right)^{\mathrm{T}} \left(\sum_{k=1}^{m} \mathbf{v}_{jk} \mathbf{q}_k \right) \tag{6}$$

where α and β controls the influence of each feature, $\mathbf{p}_k \in \mathbb{R}^K$ and $\mathbf{q}_k \in \mathbb{R}^K$ are K dimensional latent factors associated with each feature. If we represent one-hot representation of users and items like

$$\mathbf{u}_{ik} = \begin{cases} 1, k = i \\ 0, k \ne i \end{cases}, \mathbf{v}_{jk} = \begin{cases} 1, k = j \\ 0, k \ne j \end{cases} \tag{7}$$

the equation for rating prediction will reduce to the basic matrix factorization

$$\hat{r}_{ij} = \alpha_i + \beta_j + \mathbf{p}_i^{\mathrm{T}} \mathbf{q}_j \tag{8}$$

where α, β are the biases and $\mathbf{p}_i$, $\mathbf{q}_j$ are the latent factors for user $\mathbf{u}_i$ and item $\mathbf{v}_j$.

In our work, the representation of users and items learned from long-short interest model can be treated as a kind of side information. Hence, we define $\tilde{\mathbf{u}}_i$ as the learned vector representation of users and $\tilde{\mathbf{v}}_j$ as the learned vector representation of items. Then, we get new features of users and items by add the learned vector representation to original one-hot representation.

$$\mathbf{u}_i = \{\mathbf{u}_i, \tilde{\mathbf{u}}_i\}, \mathbf{v}_j = \{\mathbf{v}_j, \tilde{\mathbf{v}}_j\} \tag{9}$$

There, the equation for rating prediction will change to

$$\hat{r}_{ij} = \sum_{k=1}^{N+D} \alpha_k \{\mathbf{u}_i, \tilde{\mathbf{u}}_i\}_k + \sum_{k=1}^{M+D} \beta_k \{\mathbf{v}_j, \tilde{\mathbf{v}}_j\}_k + \left(\sum_{k=1}^{N+D} \{\mathbf{u}_i, \tilde{\mathbf{u}}_i\}_k \mathbf{p}_k\right)^{\mathrm{T}} \left(\sum_{k=1}^{M+D} \{\mathbf{v}_j, \tilde{\mathbf{v}}_j\}_k \mathbf{q}_k\right) \tag{10}$$

where N is the number of users, M is the number of items, D is the dimension of sequential features of users and items learned from our long-short interest model, $\mathbf{p}_k \in \mathbb{R}^K$ and $\mathbf{q}_k \in \mathbb{R}^K$ are K dimensional latent factors associated with each feature.

3 Experiment

In this section, we conduct several experiments to evaluate the effectiveness of our proposed long-short interest model on three public MovieLens [2] datasets. In these experiments, we also conduct corresponding analysis to investigate: (1) the rating prediction performance of our long-short interest model compare with other benchmark models; (2) the effect of our long-short interest model on users own different interaction number.

3.1 Experimental Setup

Dataset We conduct experiments on three MovieLens datasets, i.e. MovieLens-1M, MovieLens-10M and MovieLens-20M, which are commonly used for evaluating collaborative filtering algorithms. Table 2 summarizes the statistics of three datasets. These datasets also provide some side information about users and items, such as gender, age and occupation of users and movies' genres, but we don't use the side information in all our experiments.

Metrics We employ the Root Mean Square Error (RMSE) to measure the rating prediction quality. The metric RMSE is defined as:

$$RSME = \sqrt{\frac{1}{N} \sum_{i,j} I_{ij} (R_{ij} - \hat{R}_{ij})^2} \tag{11}$$

[2] http://grouplens.org/datasets/movielens/

Table 2. Statistics of three MovieLens datasets and Netflix dataset.

Dataset	**#Users**	**#Items**	**#Ratings**	**Sparsity**
ML-1M	6,040	3,706	1,000,209	95.53%
ML-10M	69,878	10,677	10,000,054	98.66%
ML-20M	138,493	26,744	20,000,263	99.46%

where R_{ij} denotes the ground-truth rating the user i gives to the item j, $\hat{R}_{ij}$ denotes the corresponding predicted rating, I_{ij} is a binary matrix that indicates the ratings in the test set, and N is the total number of ratings in the test set.

3.2 Benchmark Models

we use two popular toolkits, LibMF and LibFM, that are widely used in both academia and industry as our benchmark models.

LibMF is an open source tool for approximating an incomplete matrix using the product of two matrices in a latent space. It provides solvers for real-valued matrix factorization, binary matrix factorization, and one-class matrix factorization, and also supports parallel computation in a multi-core machine using CPU instructions (e.g., SSE) to accelerate vector operations. Its paper [2] won the best paper award in RecSys 2013.

Factorization machines (FM) are a generic approach that allows to mimic most factorization models by feature engineering. This way, factorization machines combine the generality of feature engineering with the superiority of factorization models in estimating interactions between categorical variables of a large domain. LibFM [13] is a software implementation for factorization machines that features Stochastic Gradient Descent (SGD) and Alternating Least Squares (ALS) optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC).

3.3 Overall Results

In this section, we report the experimental results to demonstrate the effectiveness of our long-short interest model. The model proposed in LibMF toolkit is denoted as **LibFM**, the dimension of latent factors is set to 100 and the number of iterations is set to 1000. For the model proposed in LibFM, we denote the LibFM model optimized by SGD as **LibFM-SGD**, the learn rate is set to 0.01, the regular parameter is set to $(0, 0, 0.01)$ and the stdev for initialization of 2-way factors is set to 0.1. Then we denote the LibFM model optimized by ALS as **LibFM-ALS**, the regular parameter is set to $(0, 0, 10)$ and the stdev for initialization of 2-way factors is set to 0.1. Lastly, the LibFM model optimized by MCMC is denoted as **LibFM-MCMC**, the stdev for initialization of 2-way factors is set to 0.1. Meanwhile, we denote our long-short interest model as **L-SIM**, the slide window size is set to 10, the dimension of latent factors is set to 256 and the dimension of sequential features is set to 100. For feature based collaborative filtering framework, the learn rate is set to 0.005 and the regular parameter is set to 0.024 for both users and items.

Table 3. RMSE Performance comparison of our proposed model and benchmark models with a training ratio of 90%/10% on three MovieLens datasets.

Algorithms	**MovieLens-1M**	**MovieLens-10M**	**MovieLens-20M**
LibMF	0.8554 ± 0.0013	0.8090 ± 0.0004	0.8023 ± 0.0004
LibFM-SGD	0.8641 ± 0.0015	0.8022 ± 0.0013	0.7945 ± 0.0023
LibFM-ALS	0.8453 ± 0.0015	0.7936 ± 0.0004	0.7860 ± 0.0004
LibFM-MCMC	0.8460 ± 0.0011	0.7866 ± 0.0004	0.7787 ± 0.0005
LSIM	**0.8355 ± 0.0014**	**0.7740 ± 0.0013**	**0.7656 ± 0.0019**

We split the rating data in each dataset into random 90%-10% training-test datasets, the training dataset are used for building our proposed model and benchmark models, the remaining data are used for testing. This process is repeated five times, and we report the mean value and standard deviation of the RMSE scores on Table 3. From this table, we can clearly observe:

1. The rating prediction performance of **LibMF** is worst among all the models, the reason should be that it focuses on accelerating the training speed of matrix factorization by parallelization, and doesn't pay enough attention on optimizing the prediction precision.
2. In three **LibFM** models, MCMC optimization **LibFM-MCMC** shows a better performance in large datasets (MovieLens-10M and MovieLens-20M) while ALS optimization **LibFM-ALS** shows a better performance in small datasets (MovieLens-1M). The performance of SGD optimization **LibFM-SGD** is the worst.
3. Our **LSIM** has the best performance between among all models in RMSE performance on all three MovieLens datasets, which shows the effectiveness of the long-short interest model. To the best of our knowledge, the best results published regarding MovieLens-1M and MovieLens-10M are reported by both [8][14] with a final RMSE of 0.831 ± 0.003 and 0.782 ± 0.003. These scores are obtained with a training ratio of 90%/10% and without side information. That means our proposed model can achieve the start-of-the-art performance.

3.4 Effects on Different Users

In our long-short interest model, the change of users' interest over time has been embedding into the sequential features. For these users with a lot of interactions, their long interest and short interest will be detailedly extracted via LSIM. But for the others with few interactions, LSIM only can extract marginal sequential features because their interest keeps stable. This phenomenon means that our LSIM will show a better performance on users with lots of interactions than ones with few interactions.

In order to exhibit this character of LSIM, we carry on some experiments on the MovieLens-1M dataset. First, we sort the users on MovieLens-1M according to their respective number of ratings, those users are grouped by their ranking order into 5 clusters (i.e. 0%-20%, 20%-40%, 40%-60%, 60%-80% and 80%-100%), and RMSE is computed by cluster respectively with a training ratio of 90%/10%. For instance, the first cluster contains the 20% of users with the least number of ratings and the last cluster contains the 20% of users with the highest number of ratings. The provided results are the mean reported through 5-cross validation.

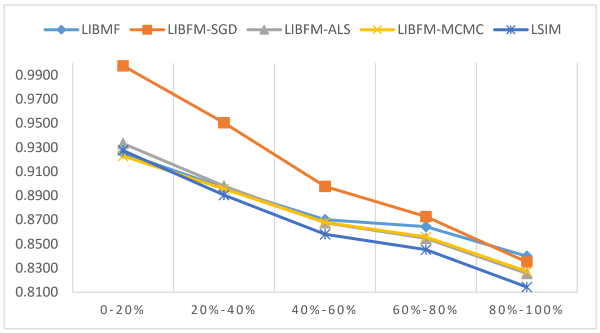

Fig. 5. RMSE computed by cluster of users sorted by their respective number of ratings on MovieLens-10M with a training ratio of 90%/10%.

It can be clearly observed from Figure 5 that all these recommendation models benefit from the increase of the count of items the user interact with on RSME metric. Our proposed **LSIM** don't perform well at the first "0%-20%" user cluster, its RMSE score is 0.9275 which is less than **LibFM-MCMC**. But when the interactions become more, its performance rapidly increases, and beats the baseline methods obviously, which shows the effectiveness of our long-short interest model.

4 Conclusions

In this study, we propose to use long-short interest model to utilize the rich sequential information on users' interaction history to solve the rating prediction task in recommender system. By incorporating the sequential features into feature based collaborative filtering framework, better prediction performance can be obtained. Our thorough evaluation, using three standard MovieLens datasets, demonstrates the effectiveness of the proposed method.

Acknowledgement

This work was supported by National High Technology R&D Program of China (Grant No. 2015AA015403, 2014AA015102), Natural Science Foundation of China (Grant No. 61202233, 61272344, 61370055) and the joint project with IBM Research. Any correspondence please refer to Yansong Feng.

References

1. Chen, T., Zheng, Z., Lu, Q., Zhang, W., Yu, Y.: Feature-based matrix factorization. arXiv preprint arXiv:1109.2271 (2011)
2. Chin, W.S., Zhuang, Y., Juan, Y.C., Lin, C.J.: A fast parallel stochastic gradient method for matrix factorization in shared memory systems. ACM Transactions on Intelligent Systems and Technology (TIST) 6(1), 2 (2015)
3. Gomez-Uribe, C.A., Hunt, N.: The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS) 6(4), 13 (2015)
4. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer (8), 30–37 (2009)
5. Lawrence, N.D., Urtasun, R.: Non-linear matrix factorization with gaussian processes. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 601–608. ACM (2009)
6. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. arXiv preprint arXiv:1405.4053 (2014)
7. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in neural information processing systems. pp. 556–562 (2001)
8. Lee, J., Kim, S., Lebanon, G., Singer, Y.: Local low-rank matrix approximation. In: Proceedings of The 30th International Conference on Machine Learning. pp. 82–90 (2013)
9. Mazumder, R., Hastie, T., Tibshirani, R.: Spectral regularization algorithms for learning large incomplete matrices. The Journal of Machine Learning Research 11, 2287–2322 (2010)
10. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
11. Pazzani, M.J., Billsus, D.: Content-based recommendation systems. In: The adaptive web, pp. 325–341. Springer (2007)
12. Rendle, S.: Factorization machines. In: Data Mining (ICDM), 2010 IEEE 10th International Conference on. pp. 995–1000. IEEE (2010)
13. Rendle, S.: Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST) 3(3), 57 (2012)
14. Sedhain, S., Menon, A.K., Sanner, S., Xie, L.: Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web Companion. pp. 111–112. International World Wide Web Conferences Steering Committee (2015)
15. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence 2009, 4 (2009)
16. Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-scale parallel collaborative filtering for the netflix prize. In: Algorithmic Aspects in Information and Management, pp. 337–348. Springer (2008)