2018 17th IEEE International Conference on Machine Learning and Applications

DeepHCF: A Deep Learning Based Hybrid Collaborative Filtering Approach for Recommendation Systems

Meshal Alfarhood, Jianlin Cheng
Department of Electrical Engineering and Computer Science
University of Missouri-Columbia
{may82, chengji}@missouri.edu

***Abstract*—The data sparsity is a significant challenge for collaborative filtering methods in recommendation systems for making accurate recommendations. Several approaches have been published to address this issue. Most of them usually use only one source of data to train their model, and other approaches still have lower performance especially when the sparsity of data is very high. In this paper, we use a deep learning based model, DeepHCF, to tackle this problem. DeepHCF uses two sources of data, ratings matrix and item reviews, to train two deep models via joint training. The user-item ratings matrix data is trained using a Multi-Layer Perceptron (MLP), while other side information is trained using a Convolutional Neural Network (CNN). The users and items' latent features learned by each model are utilized by factorization machines for our model prediction. Extensive experimental results on four different real-world datasets show that DeepHCF achieves on average a 7.42% improvement over the second most accurate method, when the dataset has over 99.9% of sparsity.**

***Index Terms*—Deep Learning, Collaborative Filtering, Neural Networks, Recommendation Systems**

I. Introduction

Recommender systems (RSs) nowadays are everywhere. Most of the commercial and social websites use recommendation engines to show relevant items to their users. This process usually increases user activities and interaction with their websites. According to [23], 60% of the video watches on YouTube come from its homepage recommendation, while 80% of what of the Netflix users watch comes from its recommender system as well. Moreover, companies like Amazon use its recommender system to increase its profit by 35% [13]. Therefore, the ongoing research in this field is important and holds the key for some businesses to effectively engage with and serve their users.

Recommender systems (RSs) generally have two types: personalized and non-personalized RSs. The personalize RSs are based on the preferences of users, while the non-personalized RSs involve no personal preferences in their recommendation. Recommending the most common or highly rated items in the training dataset is an example of a non-personalized such that all users get identical recommendations [17]. Personalized RSs, on the other hand, have three known approaches: collaborative filtering (CF), content-based filtering (CBF), and hybrid approach. The CF approach is based on the past behavior of the users. More precisely, this type of recommendation relies on the similarities between users. For instance, if $User_a$ and $User_b$ are similar to each other, they are more likely to agree on an item in the future. However, CF approach suffers when there are new users or items arrive to the system. This approach is neither able to give recommendation to new users because there is no enough information about those users, nor able to recommend new items to existing users because those items have not seen before by other users [2]. This particular issue is known as the cold-start problem.

On the other hand, content-based filtering approach (CBF) creates profiles for each user and item and offers recommendations based on the similarities of items. For example, if $User_a$ likes $Item_b$, then CBF approach is looking for other items that have same characteristics of $Item_b$ for recommendation. Unlike CF approach that needs other users' ratings, CBF is user independent [12]. Furthermore, new items can be recommended to existing users. However, CBF has two major challenges: limited content analysis and overspecialization. The first one refers to the difficulty of extracting information from items, while the second one appears when the RS only gives item recommendations similar to items they like before [3]. Finally, the hybrid approach is a combination between the CF and the CBF approaches to take the advantage of both approaches and overcome some issues related to each one of them. For example, the sparsity problem and the cold start problem are addressed by this approach. However, this type of approach increases the complexity of the model [2].

Recently, deep learning has become a promising tool for many data domains due to the increase in computational resources like GPUs, the exponential growth of available data, and the improved algorithms. For instance, deep learning models are successfully used in computer vision and natural language processing (NLP). However, in contrast to the huge amount of research on traditional algorithms for recommender systems like matrix factorization, there is much less work of using deep learning in recommendations. In this paper, we propose, DeepHCF, a deep learning model that jointly trains a Multi-Layer Perceptron (MLP) with a Convolution Neural Network (CNN) for collaborative filtering applications. Some of the current work using deep learning model neglect the user-item ratings matrix and focus only on auxiliary information like item reviews [9], [24], while other works show the benefit

978-1-5386-6805-4/18/$31.00 ©2018 IEEE
DOI 10.1109/ICMLA.2018.00021

of using the ratings matrix to train a MLP [7], [8]. However, using only one source of data usually does not scale well when the sparsity of the data is very high.

Consequently, DeepHCF is trained on two different sources of data using two sub-networks. The two sub-networks are trained jointly. The joint training is different than ensembles, where models in ensembles are trained separately at the training phase [4]. However, models in joint training are trained together at the training phase under the same objective function. The MLP network takes the user-item ratings matrix as an input, while the CNN network takes the other side information as an input. Lastly, prediction layer with factorization machines are introduced on top of the previous sub-networks as the prediction estimator of our model. Figure 1 shows the abstract architecture of our model.

DeepHCF is evaluated on four distinct datasets with various sparsity. MovieLens-1M is a relatively denser dataset, with 95% sparsity, compared to Amazon datasets that have over 99.9% sparsity. DeepHCF outperforms all other methods on Amazon datasets in terms of prediction error. More precisely, DeepHCF achieves a comparable result with the best method on MovieLens-1M dataset, while it achieves a substantial improvement, 7.42% on average, on Amazon datasets.

The main contributions of this work are summarized as follows:

- We introduce, DeepHCF, a deep learning-based model as a hybrid collaborative filtering approach consisting of a multi-layer perceptron and a convolutional neural network, which takes two different sources of data as input.
- We implement a prediction layer on top of the two deep models such that user-item and review representations can interact with each other to predict final ratings using factorization machines.
- We evaluate our model on four different real-world datasets with different percents of sparsity. We compare the performance of our proposed model with five state-of-the-art approaches. DeepHCF achieves superior performance when the data sparsity is extremely high.

The rest of the paper is organized as follows. In Section 2, we describe the background and related work. Our model, DeepHCF, is described in depth in Section 3. Section 4 presents the experiments that we have done to evaluate our model against the state-of-the-art approaches. Finally, the conclusions are presented in Section 5.

II. Background and Related Work

In this section, we start by defining the problem discussed in this paper, and then have a brief overview on some deep learning models, factorization machines, and related work.

A. Problem Definition

Most of the recommendation problems are represented by a matrix $R^{u.i}$. This matrix, R, has a size of $(u.i)$, for u being the number of users in R, and i the number of items in R. $R_{u,i}$ represents the rating value of User u for Item i. The ratings usually go from 1 to 5. Therefore, the idea of the RSs is to train predictive models with those available ratings in order to predict unknown ratings and recommend an item if the predicted rating is high. However, matrix R is usually very sparse, meaning that users only rate a very small number of items compared to the total number of items in matrix R. This sparsity problem is one frequent problem in collaborative filtering in RSs, which is a real challenge to increase the recommendation performance.

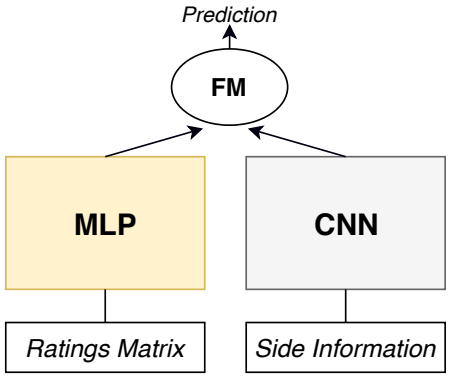


Fig. 1: DeepHCF Overview

B. Deep Learning

The application of deep learning to recommender systems is recent. The first work that applies deep learning concept for collaborative filtering was [16]. It performs inference by using Restricted Boltzmann Machine (RBM). Recently, collaborative deep learning (CDL) [21] has become the state-of-the-art method of deep learning techniques in recommendation systems due to its promising performance. It is composed of a stacked denoising autoencoder (SDAE) and probabilistic matrix factorization (PMF).

Many other types of deep learning models are integrated nowadays into complex structures to solve several types of problems. A multi-layer perceptron (MLP), a feedforward neural network with multiple hidden layers between input layer and output layer [23], is an example. The structure of MLP can have various shapes like the tower shape used in [4], [5], [8], the constant shape used in [18], and the diamond shape used in [22]. YouTube recommender system uses a MLP in its model [5]. The architecture of YouTube recommender system has two deep MLPs. The first deep MLP model is used for the candidate generation, while the second one is used for ranking. The evaluation shows that this new deep collaborative filtering technique outperforms the old recommender system of YouTube that are implemented using Matrix Factorization approaches. Furthermore, Google presented a combination of wide and deep models for recommender systems [4]. It is shown that the wide model is good for memorization, while the deep model is good for generalization. The deep model uses a MLP with embeddings for sparse features. It is evaluated on Google Play and the results show that both wide and deep

model have an improvement in the app acquisition over using wide-only or deep-only models.

Convolutional Neural Network (CNN), in addition, is popular in computer vision and natural language processing (NLP). It usually consists of convolutional layers followed by pooling and fully connected layers. CNN has less parameters than MLP with the same number of units, which makes it easy to train [11]. The convolutional layers extract features from the input, and generates k feature maps, where k is the number of filters. The pooling layers are then responsible to reduce the dimensionality of feature maps. ConvMF [9], the state-of-the-art approach in this paper, combines CNN with Probabilistic Matrix Factorization (PMF). It effectively utilizes the contextual information of documents in order to solve the data sparsity problem. The bag of word model doesn't work well since it ignores the word order. Therefore, they use CNN to generate the document latent vector, then integrate it with the epsilon variable in the PMF model to generate the final prediction.

C. Factorization Machines

Factorization Machines (FM) were proposed by [19] where it models all interaction between feature variables. A 2-way FM with degree, d=2, is sufficient (Equation 1) in most of the cases. It captures all single (first order) and pairwise (second order) interactions of the input features.

$$FM(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} w_{ij} x_i x_j, \qquad (1)$$

where w_0 is the global bias, w_i is the weight assigned to each feature, and w_{ij} is the weight assigned for each feature pair $x_i x_j$. w_{ij} can be calculated as the dot product of the embedding vector of feature i (V_i) and the embedding vector of feature j (V_j) as $w_{ij} = V_i^T V_j$.

FM has been successfully integrated in recommendation systems [4], [6], [24]. CoNN [24], for example, jointly trains two CNNs that are coupled in the last layer. One CNN is used to train only the user reviews, while the other one is used to train the item reviews. MF is used to predict the corresponding rating at the final layer. One major limitation of this approach is that CoNN can't deal with users and item with no ratings.

III. Methodology

Our proposed model, DeepHCF, is described thoroughly in this section. DeepHCF is a hybrid model where it takes advantage of both ratings matrix and other side information to address the sparsity problem. It models users behavior and items characteristics using two sources of data: user-item ratings matrix and item reviews. It learns hidden latent features from both data, such that the learned latent features can be employed to approximate ratings for each user. We first describe the general architecture of our model in Section A. Then, MLP layers, CNN layers, and prediction layer are presented in Sections B, C, and D respectively.

A. Architecture

The architecture of our model for ratings prediction is shown in Figure 2. DeepHCF network consists of two parallel sub-networks, connected with a prediction layer at the top. The two parallel sub-networks are a multi-layer perceptron (MLP) and a convolutional neural network (CNN) trained in a joint manner to predict final ratings in order to minimize prediction error. MLP is a widely used model with excellent scalability, generalization, performance, and ability to learn complex relationships. CNN, similarly, is an impressive approach to extract high-level representations from raw data. The learned latent features of each model are multiplied by each other, and then factorized by factorization machines to estimate ratings values. The difference between our predicted value and the actual value is then computed to estimate the loss function of our model. Further specifics about all layers in the architecture are presented in the following three sub-sections.

B. MLP Layers

The MLP structure in our model has three main components: input layer, embedding layer, and hidden layers. User ID and item ID are the inputs of our MLP model and they are separately embedded into two different embedding layers, which are then concatenated and fed into three hidden layers. Embedding is a function, $f\colon X \rightarrow \mathbb{R}^n$, mapping individual data values from a large sparse categorical domain to dense vectors. It is particularly useful when some of the values may be similar in some meaningful way to other values.

The three hidden layers afterward are in a tower shape, means that each hidden layer has less neurons than the previous layer. The activation of each neuron $a_j^{(\ell)}$ is computed as:

$$a_j^{(\ell)} = \sigma(\sum_{i=1} a_i^{(\ell-1)} w_{ij}^{(\ell)} + b_j^{(\ell)}), \qquad (2)$$

where (ℓ) is the layer number, and σ is a non-linear activation function. The non-linear activation function used in the first two hidden layers is the Rectified Linear Unit, $ReLU$, (Equation 3), while the third hidden layer has a $\tanh$ activation function (Equation 4) to be consistent with the CNN latent feature layer. The learned latent features in MLP model represents the user behavior in collaborative filtering approach.

$$f(x) = max\{0, x\} \qquad (3)$$

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \qquad (4)$$

C. CNN Layers

The CNN model in this paper is inspired by [10]. The CNN structure has five main components: input layer, embedding layers, convolution layers, pooling layers, and a fully connected layer at the end. First, distinct words of the item review are embedded into different embedding layers. Word embedding is very influential in natural language processing techniques. It maps each word into a dense vector. This process captures the semantic information in the review text such that

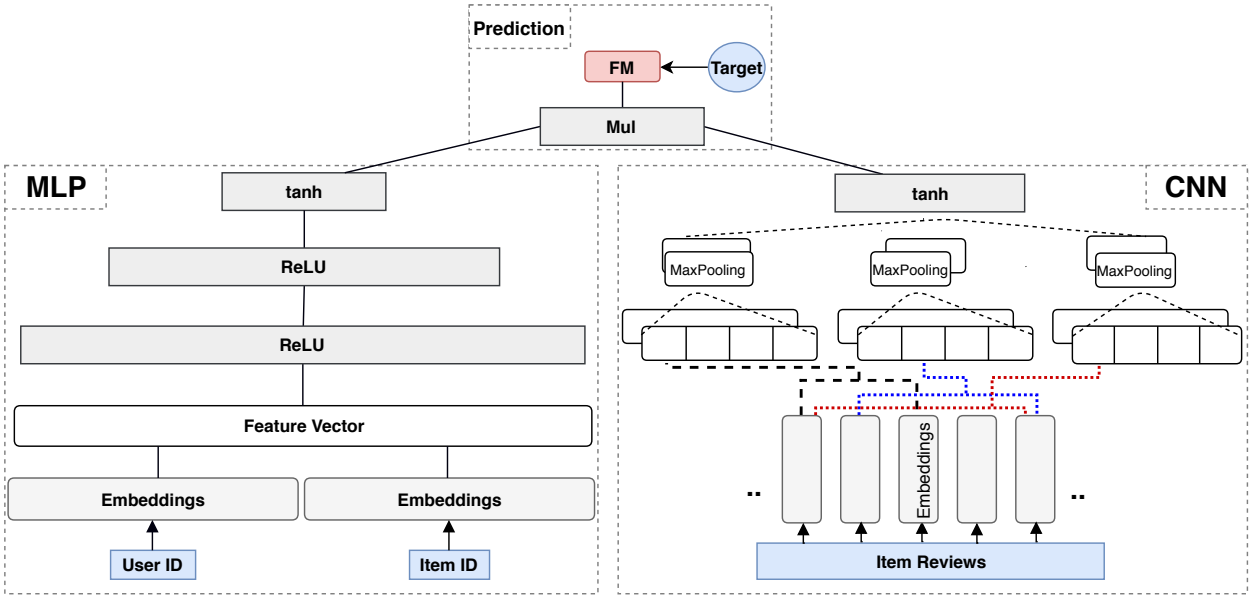


Fig. 2: DeepHCF Architecture

similar words have nearly similar vectors. Plotting all the words in embedding space would result into words with the same meaning or semantically related to each other are placed close to each other. Word embeddings can be initialized either with pre-trained word embeddings, or randomly such that it learns through training. Pre-trained word embeddings are usually preferred and enabled the training to converge faster.

Second, the convolutional layers extract features from its inputs. It uses multiple filters along with three different word window sizes, w, to capture the surrounding words interactions. Let $x_i \in \mathbb{R}^k$ represent the k-dimensional word embeddings for the i-th word in the sentence. The whole sentence can be represented as the concatenation of all words embedding as $x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n$. Each feature, c_i, extracted from convolution layers is computed as:

$$c_i = \sigma(W \cdot x_{i:i+w-1} + b), \tag{5}$$

where b is the bias, $W \in \mathbb{R}^{wk}$ is the filter, and σ is a non-linear activation function. We use ReLU as activation functions for the convolutional layers for faster convergence and to reduce the vanishing gradient problem. The feature map, j, generated from convolutions (Equation 6) are then fed into the max-pooling layer to capture the most important information.

$$c^j = (c_1^j, c_2^j, c_3^j, c_4^j, ..., c_{n-w+1}^j) \tag{6}$$

Max-pooling layer reduces the dimensionality of each feature map by taking the maximum value from values in Equation 6. Lastly, max-pooling layer feds its output into a fully connected layer with a tanh activation function. The size of this layer is equivalent to the size of the last hidden layer of the MLP model.

D. Prediction Layer

The outcome of the MLP model and the CNN model are the learned latent features with same dimensions. The prediction layer concatenates the two sets of latent features and does multiplication (Equation 7). The multiplication function is the element-wise multiplication that takes two vectors of the same size ,n, and return a single vector of size n also. It simply multiplies vectors A and B element by element.

$$C^n = (A^n \times B^n) = A_i \times B_i \tag{7}$$

The result of the multiplication function is then fed to factorization machines (FM) layer that applies a sigmoid function (Equation 8) to its output. The sigmoid neuron outputs a continuous range of values between 0 and 1 as the predicted rating of our model.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

Therefore, it is essential to normalize our ratings before the training phase to be in the same range [0,1], instead of being in the range of [1,5]. In the testing phase, we need to convert the predicted rating back into the original range of [1,5] by de-normalization. The normalization process is shown in Equation 9, while the de-normalization process is shown in Equation 10:

$$N(\hat{x_i}) = \frac{x_i - min(x)}{max(x) - min(x)} \tag{9}$$

$$D(x_i) = \hat{x_i} \times (max(x) - min(x)) + min(x), \tag{10}$$

where $min(x)$ is the minimum rating value, $max(x)$ is the maximum rating value, and i refers to each sample in the dataset.

Since the output of our model is a real-value between 0 and 1, the most straightforward and suitable loss function for

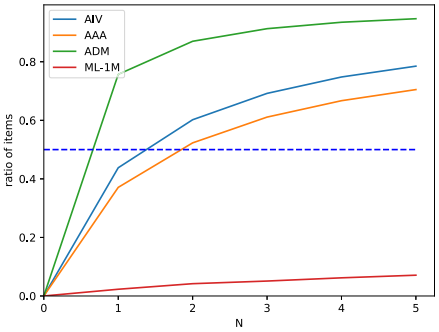


Fig. 3: Ratio of items who have been rated by N users or less.

TABLE I: Datasets

Dataset	#users	#items	#ratings	sparsity
MovieLens-1M	6,040	3,544	993,482	95.35%
Amazon Instant Video	29,757	15,149	135,188	99.97%
Amazon Android Apps	240,931	51,598	1,322,838	99.98%
Amazon Digital Music	56,810	156,493	351,762	99.99%

the training is the binary cross entropy (Equation 11). As it is shown in the equation, the cross entropy loss value increases as the predicted rating value varies from the actual rating value.

$$Entropy(y_i, \hat{y_i}) = -(y_i log(\hat{y_i}) + (1 - y_i) log(1 - \hat{y_i})), \quad (11)$$

where y_i is the actual rating value, $\hat{y_i}$ is the predicted rating value, and i refers to each sample in the dataset.

IV. Experiments

We have performed extensive experiments on different datasets to evaluate our model against the state-of-the-art approaches. Our model is implemented in Python with a Tensorflow [1], a well-known Python library for deep learning implemented by Google. We first describe the datasets and the evaluation metrics that are used to evaluate DeepHCF in Section A. Then, we describe the state-of-the-art approaches in Section B. The experimental settings are then defined in Section C. Section D shows the impact of using pre-training for word embeddings. Lastly, the results are shown in Section E.

A. Datasets

In our experiments, we have used four real-world datasets from two different platforms for our evaluation. Three datasets are collected from Amazon product data[1], while the remaining dataset is taken from MovieLens[2]. The four datasets are categorized into two categories based on the platform they are collected from as the following:

- **MovieLens**: This movie rating dataset has been widely used to evaluate collaborative filtering approaches. This version of dataset has one million ratings with a sparsity of around 95.35%. It is considered a relatively denser dataset compared to Amazon datasets. It has explicit ratings that go from 1 to 5. However, MovieLens dataset does not have movie reviews in its original format. Therefore, reviews for each movie are obtained from IMDB[3].
- **Amazon**: This category has three datasets: Amazon Instant Video, Amazon Android Apps, and Amazon Digital Music. This category is critical for most of existing approaches due to the high data sparsity that exceeds 99.9%. All datasets contain explicit ratings between 1 to 5 and product reviews from Amazon that are collected between May 1996 - July 2014. More details about the datasets are shown in Table 1.

Figure 3 shows the ratio of items who have been rated only by 5 users or less. As the plot shows, more than half of items in Amazon datasets have only 2 ratings or less. Moreover, 95% of items in Amazon Digital Music dataset have 5 ratings or less. On the contrary, MovieLens-1M is much denser where at least 90% of movies have been rated by more than 5 users.

Each dataset is split randomly into a training set (80%), a validation set (10%), and a test set (10%). The training set is used to train our model. The validation set is used for early stopping the training of our model to avoid overfitting. This technique is a form of regularization. The test set is used to compute the performance of our model against the state-of-the-art methods. We use the same sets with all the state-of-the-art methods for the purpose of fair comparison.

Mean Absolute Error (MAE) and Root-Mean-Square Error (RMSE) are used for evaluation. Lower MAE (Equation 12) and lower RMSE (Equation 13) indicate a better performance.

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \quad (12)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \quad (13)$$

B. State-of-the-art Approaches

Since Convolutional Matrix Factorization - ConvMF+ [9] was extensively evaluated against the previous state-of-the-art models like Probabilistic Matrix Factorization (PMF) [14], Collaborative Topic Regression (CTR) [20], and Collaborative Deep Learning (CDL) [21], and shown to have better performance by a wide margin, we decide not to repeat those same comparisons in this paper and only compare our model with ConvMF+ and the following baseline approaches:

- **User Average**: It calculates the average rating for each user in the training set and assigns this value for future predication. If the user is not found in the training set, the global average will be assigned as a prediction.

[1]http://jmcauley.ucsd.edu/data/amazon
[2]https://grouplens.org/datasets/movielens
[3]http://www.imdb.com

TABLE II: Performance Results (Best values are marked in bold)

Approach	MovieLens-1M		Amazon Instant Video		Amazon Android Apps		Amazon Digital Music	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
User Average	0.830	1.036	0.778	1.168	1.005	1.373	0.569	0.917
Item Average	0.781	0.979	0.788	1.079	0.975	1.250	0.625	0.943
BPMF	0.678	0.869	0.817	1.103	1.024	1.326	0.612	0.910
ALS-WR	0.675	**0.861**	0.856	1.141	1.087	1.423	0.631	0.919
ConvMF+	0.678	**0.861**	0.866	1.149	0.994	1.307	0.661	0.934
DeepHCF	**0.674**	0.867	**0.688**	**0.994**	**0.896**	**1.180**	**0.520**	**0.830**
Improvement%	0.14%	-	11.43%	7.87%	8.10%	5.60%	8.61%	8.97%
Average Improvement for Amazon Datasets%			MAE= 9.38%, RMSE= 7.42%					

- **Item Average**: It calculates the average rating for each item in the training set and assigns this value for future predication. If the item is not found in the training set, the global average will be assigned as a prediction.
- **BPMF**: Bayesian Probabilistic Matrix Factorization [15] has proven to be an improvement over PMF by adding a bayesian treatment into its model. It is trained using Markov Chain Monte Carlo (MCMC).
- **ALS-WR**: Alternating Least Squares with Weighted Regularization [25] is a matrix factorization method that uses alternating least squares (ALS) with Weighted lambda regularization.

ConvMF+ [9] uses both ratings matrix and item reviews as DeepHCF to train a CNN with PMF, while other baselines only use rating matrix data for their predictions. Although User Average and Item Average approaches seem very naive, however, we intentionally add these two basic methods to show that some recent approaches, ConvMF+ for example, fail in some cases as it is described in detailed at the performance comparison Section. For BPMF and ALS-WR, we have used the GitHub[4] implementation for both approaches.

C. Experimental Settings

For each dataset, we run three experiments and the average error is reported in the performance comparison Section. We follow the same experimental settings as the state-of-the-art approaches [9], [20], [21]. Items that don't have reviews are removed from the dataset. Moreover, users who have less than 3 ratings are also removed in Amazon datasets. The reviews in each dataset are processed such that stop words are removed, corpus-specific stop words who have document frequency more than 0.5 are removed, only the top 8000 distinct words are selected, and only 200 words are taken as the maximum length of a review.

For MLP layers, we set the embedding size for both user and item attributes to 64 dimensions. Also, we use the dropout technique with values in [0.1, 0.25] for regularization. In the same way for CNN, we use three different window sizes [1, 2, and 3] for the convolutional layers to represent the surrounding words, and we use 64 filters per window size. In addition, we use a dropout rate equal to 0.25 to avoid overfitting. Lastly, the pre-trained word embeddings[5] of 200 dimensions are used to initialize the word embedding vectors.

D. Impact of Pre-Training

Figure 4 shows the state of each epoch of our model, DeepHCF, with and without pre-training CNN embeddings on MovieLens-1M and Amazon Instant Video dataset. We can observe from Figure 4 that Amazon datasets in general converge very fast with only 2 epochs and that's due clearly to the high sparsity of the datasets, while MovieLens-1M converges with 18 epochs of training. On the other hand, with or without using Glove word embeddings to initialize our CNN embeddings vectors as pre-training, DeepHCF displays a fast convergence on both datasets. Thus, pre-training does not improve our model performance, and a random initialization can achieve results that are equivalent to the results with word embeddings pre-training. In contrast to the significant impact of pre-training on other approaches, this result demonstrates that DeepHCF is insensitive to parameter initialization of the embedding layer.

E. Performance Comparison

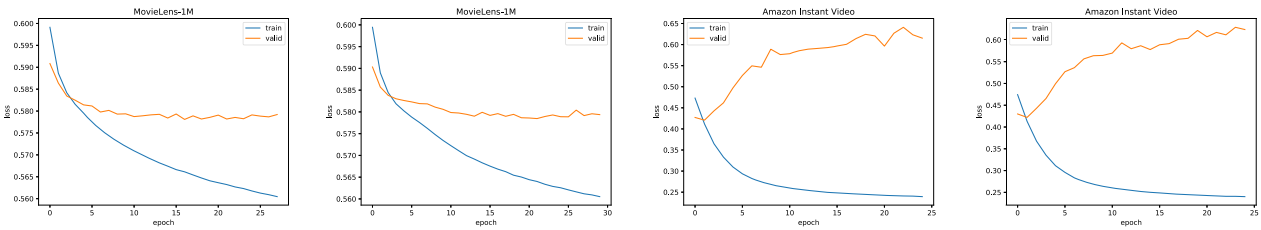
The performance of DeepHCF and the other approaches mentioned in Section 4.2 are reported in the next two subsections. DeepHCF outperforms all baselines on Amazon datasets and shows competitive results on MovieLens-1M dataset according to both metrics: MAE and RMSE. The improvement rate in each table indicates the improvement in error for our model, DeepHCF, over the second most accurate method.

ConvMF+ has two attributes, $\lambda_i \lambda_j$, that should be manually set and they are significant to its performance. Thus, we have used the same values they used in their paper for MovieLens-1M, $\lambda_i = 100$ and $\lambda_j = 10$. However, for all Amazon datasets, we have searched in the grid of [1, 10, 100] and we have found that best performance is given when $\lambda_i = 1$ and $\lambda_j = 100$.

1) MovieLens-1M Dataset: This dataset has a level of sparsity equal to 95.35%. Table 2 summarizes the performance on this dataset. As the table shows, BPMF, ALS-WR, and ConvMF+ perform very well and they have much better performance than the other straightforward methods, User Average and Item Average. DeepHCF as well performs well on

[4]https://github.com/chyikwei/recommend

[5]http://nlp.stanford.edu/projects/glove



(a) ML-1M w/ pre-training (b) ML-1M w/o pre-training (c) AIV w/ pre-training (d) AIV w/o pre-training

Fig. 4: Training and validation loss value of each epoch of MovieLens-1M (ML-1M) and Amazon Instant Video (AIV) datasets with and without pre-training.

this dataset and achieves a comparable performance with other baselines. This result demonstrates that most of the existing collaborative filtering approaches, including DeepHCF, are effective and powerful with relatively denser datasets.

2) Amazon Datasets: This category has three datasets as described in Section 4.A. This category has extremely sparse datasets, which poses a significant challenge to all collaborative filtering approaches to make accurate recommendations. All datasets have a sparsity of more than 99.9%. Table 3 summarizes the performance of all methods on Amazon datasets. we can observe from Table 3 that ConvMF+ performs poorly compared to other approaches in this category due to the high percentage of sparsity. Furthermore, the two naive methods, Item Average and User Average, have a better score in terms of both metrics than ConvMF+. We believe that the user latent vector of ConvMF+, which is learned by PMF, doesn't represent the user features very accurate in Amazon datasets.

On the contrary, our model, DeepHCF, achieves a substantial improvement over the second most accurate method in Amazon Instant Video, Amazon Android Apps, Amazon Digital Music datasets by a marginal improvement in RMSE of 7.87%, 5.60%, and 8.97% respectively. The average improvement in RMSE of the three datasets is 7.42%. This significant gain demonstrates that DeepHCF model works much better than the other methods on very sparse datasets, such as Amazon datasets. We believe that using MLP to learn the user representation, and CNN to analyze the sentiment in text reviews are more powerful and robust with extraordinarily sparse datasets.

V. Conclusions

In this paper, we develop a deep learning method to address the data sparsity problem of collaborative filtering approaches in recommendation systems. We introduce DeepHCF, a deep learning network for hybrid collaborative filtering, that is composed of two sub-networks trained jointly with two different sources of data. A prediction layer with factorization machines are built on top of the two models to utilize their outcome for our final prediction. We have evaluated our model on four different real-world datasets against five methods. The evaluation results show that DeepHCF achieves superior performance in both metrics, MAE and RMSE, in case of three datasets with extreme data sparsity, while it achieves a competitive performance when the data is relatively denser. In conclusion, our model is effective on both dense and sparse data.

Acknowledgment

The authors would like to thank King Saud University for supporting this project. Also, we would like to thank the authors of ConvMF paper for making their data public.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. Corrado, A. Davis, J. Dean, M. Devin, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." arXiv preprint arXiv:1603.04467, 2016.

[2] B. Betru, C. Onana, and B. Batchakui. "Deep learning methods on recommender system: A survey of state-of-the-art." volume 162. International Journal of Computer Applications, 2017.

[3] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez. "Recommender systems survey." Knowledge-based systems, 46:109132, 2013.

[4] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, and R. Anil. "Wide and deep learning for recommender systems." In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. ACM, 2016.

[5] P. Covington, J. Adams, and E. Sargin. "Deep neural networks for youtube recommendations." In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016. ACM, 2016.

[6] H. Guo, R. Tang, Y. Ye, Z. Li and X. He. "Deepfm: A factorization-machine based neural network for CTR prediction." IJCAI, 2017.

[7] X. He and T. Chua. "Neural factorization machines for sparse predictive analytics." In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 17. ACM, 2017.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. "Neural collaborative filtering." In Proceedings of the 26th International Conference on World Wide Web. WWW, 2017.

[9] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu. "Convolutional matrix factorization for document contextaware recommendation." In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016.

[10] Y. Kim. "Convolutional neural networks for sentence classification." In Proceedings of the 2014 Empirical Methods in Natural Language Processing, page 17461751. EMNLP, 2014.

[11] J. Liu and C. Wu. "Deep Learning Based Recommendation: A Survey," pages 451458. Springer Singapore, Singapore, 2017.

[12] P. Lops, M. Gemmis, and G. Semeraro. "Content-based recommender systems: State of the art and trends." Springer US, 2011.

[13] I. MacKenzie, C. Meyer, and S. Noble. "How retailers can keep up with consumers," 2013. https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers. [Accessed Online Apr 18, 2018].

[14] A. Mnih and R. Salakhutdinov. “Probabilistic matrix factorization.” In Advances in neural information processing systems, pages 12571264, 2008.
[15] R. Salakhutdinov, and A. Mnih. “Bayesian probabilistic matrix factorization using Markov chain Monte Carlo.” Proceedings of the 25th international conference on Machine learning. ACM, 2008.
[16] R. Salakhutdinov, A. Mnih, and G. Hinton. “Restricted Boltzmann machines for collaborative filtering.” In Proceedings of the 24th International Conference on Machine Learning, pages 791798. ACM, 2007.
[17] J. Schafer, J. Konstan, and J. Riedl. “Recommender systems in e-commerce.” In Proceedings of the 1st ACM conference on Electronic commerce, pages 158166. ACM, 1999.
[18] Y. Shan, T. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. “Deep crossing: Webscale modeling without manually crafted combinatorial features.” In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 255262. ACM, 2016.
[19] R. Steffen. “Factorization machines.” Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 2010.
[20] C. Wang and D. Blei. “Collaborative topic modeling for recommending scientific articles.” In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 448456. ACM, 2011.
[21] H. Wang, N. Wang, and D. Yeung. “Collaborative deep learning for recommender systems.” In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 12351244. ACM, 2015.
[22] W. Zhang, T. Du, and J. Wang. “Deep learning over multi-field categorical data.” In European conference on information retrieval, pages 4557. Springer, 2016.
[23] S. Zhang, L. Yao, and A. Sun. “Deep learning based recommender system: A survey and new perspectives.” In arXiv preprint arXiv:1707.07435., 2017.
[24] L. Zheng, V. Noroozi, and P. Yu. “Joint deep modeling of users and items using reviews for recommendation.” In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. ACM, 2017.
[25] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. “Large-scale parallel collaborative filtering for the netflix prize.” International Conference on Algorithmic Applications in Management. Springer, Berlin, Heidelberg, 2008.