論文 / 著書情報
Article / Book Information

Title	PNCF: Neural collaborative filtering based on pre-trained embedding
Authors	Jianqi Pan, Masayuki Yamamura, Atsushi Yoshikawa
Citation	PROCEEDINGS OF SPIE, Volume 12258, ,
Pub. date	2022, 7
DOI	https://doi.org/10.1117/12.2639163
Rights	(Copyright) Copyright 2022 Society of Photo-Optical Instrumentation Engineers (SPIE). One print or electronic copy may be made for personal use only. Systematic reproduction and distribution, duplication of any material in this publication for a fee or for commercial purposes, and modification of the contents of the publication are prohibited. (Citation) Jianqi Pan, Masayuki Yamamura, Atsushi Yoshikawa "PNCF: Neural collaborative filtering based on pre-trained embedding", Proc. SPIE 12258, International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022), 122580B (2022); https://doi.org/10.1117/12.2639163

PNCF: neural collaborative filtering based on pre-trained embedding

Jianqi Pan, Masayuki Yamamura, Atsushi Yoshikawa

Jianqi Pan, Masayuki Yamamura, Atsushi Yoshikawa, "PNCF: neural collaborative filtering based on pre-trained embedding," Proc. SPIE 12258, International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022), 122580B (15 July 2022); doi: 10.1117/12.2639163

SPIE.

Event: International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022), 2022, Qingdao, China

PNCF: Neural collaborative filtering based on pre-trained embedding

Jianqi Pan*[a], Masayuki Yamamura[a], Atsushi Yoshikawa[a]
[a]School of Computing, Tokyo Institute of Technology, Japan
*Corresponding author: pan.j.ab@m.titech.ac.jp

ABSTRACT

In this paper, we propose the recommendation algorithm PNCF for neural networks. We designed a pre-training task for a distributed representation of embeddings based on many-to-many information. We used the word2vec technique in natural language processing to implement the embedding of users and items. We also constructed a brand-new video website tag-author pre-training dataset. The code in this paper was implemented in PyTorch and is publicly available on GitHub (github.com/jannchie/ PNCF).

Keywords: Recommendation, Collaborative Filtering, Embedding, Implicit Feedback, Deep Learning

1. INTRODUCTION

In recent years, recommendation algorithms have become increasingly important as information overload occurs. Recommendation algorithms are an important tool for information filtering in the age of information explosion. Traditional collaborative filtering often uses matrix factorization (MF)[1]. To introduce collaborative filtering[2] into the field of deep learning, NCF models have been proposed by He et al. (2017)[3]. However, the simple multi-layer perceptron (MLP) structure, although able to fit any function[4], does not work well for direct application. Therefore, NeuMF has been proposed as an improved solution, fusing both MF and MLP models, with big success. Word embedding is an important approach for natural language processing. Word2Vec[5] is a well-known word embedding model. It approaches assumes that the meanings of words in proximity are always related. Therefore, it needs to use consecutive words from the corpus as input. It constructs a CBOW task where the parameters of the trained model can be treated as a distributed vector representation of words[6]. The model generates word vectors that capture the semantic information of the words. Nowadays, the technique is not limited to the NLP domain, but the algorithm is also used in the recommender system domain[7]. We propose a model based on pre-trained user and item embeddings, called PNCF, to solve the item recommendation problem. We designed a pre-training task to learn the potential semantic relationships of users and items, respectively, from their interaction information in a self-supervised manner. The user and item are then mapped separately to a low-dimensional space, and then a deep neural network is used to find their association. This pre-training greatly improves the prediction. We conducted extensive experiments on the MovieLens dataset to verify the effectiveness of the model. This model also considers information about multiple interactions between items and users. We also validated it on our own collection of video-author-tag datasets containing such multiple interactions and obtained good results.

2. PRE-TRAIN NEURAL COLLABORATIVE FILTERING

2.1. Formal definition of the problem

The recommendation problem is defined in the following form, given an item set I and a user set U, take an item i in I and an item u in U. Construct a model that, for a specified interaction of (u, i), can give the probability of their interaction occurring.

Let m and n denote the number of users and items, respectively. We define the user-item interaction matrix $Y \in R^{m \times n}$ from implicit interaction as $y_{u,i} = k$, where k represents the number of interactions between the user and the project, or 0 if there are no interactions.

2.2. Pre-train Task

Pre-training allows using as much training data as possible to extract as many common features as possible from it, thus enabling the model to have a lighter learning burden for a specific task[8].

In this work, a pre-training task is constructed based on word2vec. For the recommendation task, we need to construct the relevant datasets for users and items separately. We extract the relevance of users and items from their implicit feedback[9].

International Conference on Neural Networks, Information, and Communication Engineering (NNICE 2022),
edited by Rajeev Tiwari, Proc. of SPIE Vol. 12258, 122580B · © 2022 SPIE
0277-786X · doi: 10.1117/12.2639163

Consider a user may interact with multiple items. For each user, we take out the items with which he interacts, from which we take items as a sample. Repeat this operation to obtain a sample of items associated with.

Most recommendation tasks do not consider multiple interactions. In order to model the relationship of potential multiple interactions, the probability of sampling objects is not equal. The higher the number of interactions, the higher the probability of being taken out, which is calculated as:

$$probability(u,i) = \frac{count(u,i)^{0.75}}{\sum_{j=1}^{|I_u|} count(u,j)^{0.75}} \quad (1)$$

where $probability(u,i)$ denotes the probability that the i-th item associated with user u is sampled and $count(u,i)$ denotes the number of the i-th item associated with user u. The 0.75 in the formula is a constant to reduce the probability of high-frequency items being sampled. Such a set of samples is called a bag, and a distributed representation of the samples is obtained by taking one sample as the center and a certain number of other samples as the context.

There is no explicit one-way constraint between users and items, meaning that performing the same operation, for each item, we can then map the items to a vector space and obtain a distributed representation of the items. After pre-training, items that are often used together by users are clustered together, and similarly, users who often use the same items are clustered together.

2.3. Feed-forward Neural Network

The embeddings of users and items exist in two different vector spaces. We need a way to link the two vector spaces. This high-dimensional mapping relationship is very complex, and it is a natural idea to model it with a feed-forward neural network. It is important to note that for training, we use the following training techniques. The specific structure is shown in Figure 1.

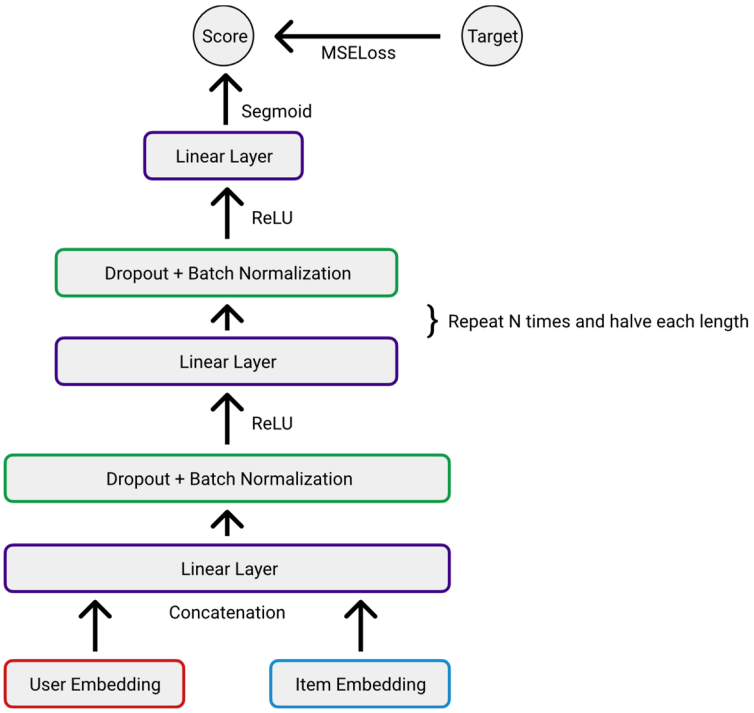

Figure 1. Neural network structure of NCF part

Negative sampling. When training this model, all weights in the network are not updated, but only some of the negative samples sampled are updated. There is experimental evidence that negative sampling not only speeds up the training but also helps the model to improve its effectiveness[5].

Tower structure. The NCF part of the model is a feedforward neural network, and its input is the embedding of users and items. Experience shows that a tower neural network structure with progressively fewer layers in each layer generally works better[3].

Regularization. The structure also uses some advanced means to prevent overfitting. Dropout[10] and Batch Normalization[11] mechanisms are introduced to improve the generalization ability of the model while accelerating the convergence speed.

The model finally connects a single-node Sigmoid layer as the output to obtain the similarity score. Calculate losses using mean squared error (MSE).

3. EXPERIMENT

3.1. Experimental Settings

Datasets. In this paper, we use two datasets of different sizes, MovieLens 100K, and 1M for testing. In this dataset, the user and the project will interact at most once. We tested on these datasets using PNCF, NeuMF[3] respectively to verify the effectiveness of PNCF for general recommendation tasks.

In addition to the public test set, we also introduced a dataset with a list of 500,000 videos, authors, and tags from Bilibili, which is one of the most popular video sites in China for testing. In this dataset, authors and tags have a many-to-many relationship. Our task is to suggest tags that the author hasn't used before.

We uniformly pre-processed the dataset with Core10. This can reduce the negative impact of data noise on the model effect to some degree. The statistical information of the dataset is shown in Table 1. We can find that the Bilibili dataset is larger and sparser than the MovieLens dataset.

Table 1. Data set statistics

name	user	item	density
Bilibili Video	36,693	17,209	0.0019
MovieLens_100K	943	1,152	0.0901
MovieLens_1M	6,040	3,260	0.0507

A user-based scaling approach was used to determine the training set and test set. Specifically, for each user, records are placed in the training set and the testing set in a 4:1 ratio.

Evaluation Metrics. To facilitate comparison, the Top-K recommendation method, which is the mainstream in this field, is used for evaluation. For each user, the model generates a list of items to be recommended and compares them with the data in the test set for evaluation.

The evaluation metrics include Precision, Recall, Hit Ratio (HR), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). We choose the NeuMF model as the baseline, which is a widely used classical NCF model. We refer to the code implementation of daisyRec[12] for the evaluation.

Parameter Settings. The adjustable hyperparameters are listed in Table 2. We sliced the training set again to obtain the development set for hyperparameters tuning. We use grid Search to find the optimal hyperparameters.

Table 2. Adjustable hyperparameters

Hyperparameter	Description
Batch size	Size of training batches
Bag size	Number of samples per pre-training task
Vector size	Number of dimensions of the embedding vector
Pre-train negative number	Number of negative samples in the pre-training task
Fine-tuning negative number	Number of negative samples during fine-tuning

During the pre-training phase, the crucial metrics are batch size, bag size, and vector size. Bag size has the most significant effect on the results. We were able to get the best results when the scale was moderate. We also find that using larger batch

sizes yields better results but requires more memory for training. Vector size showed a significant performance degradation if it was too small.

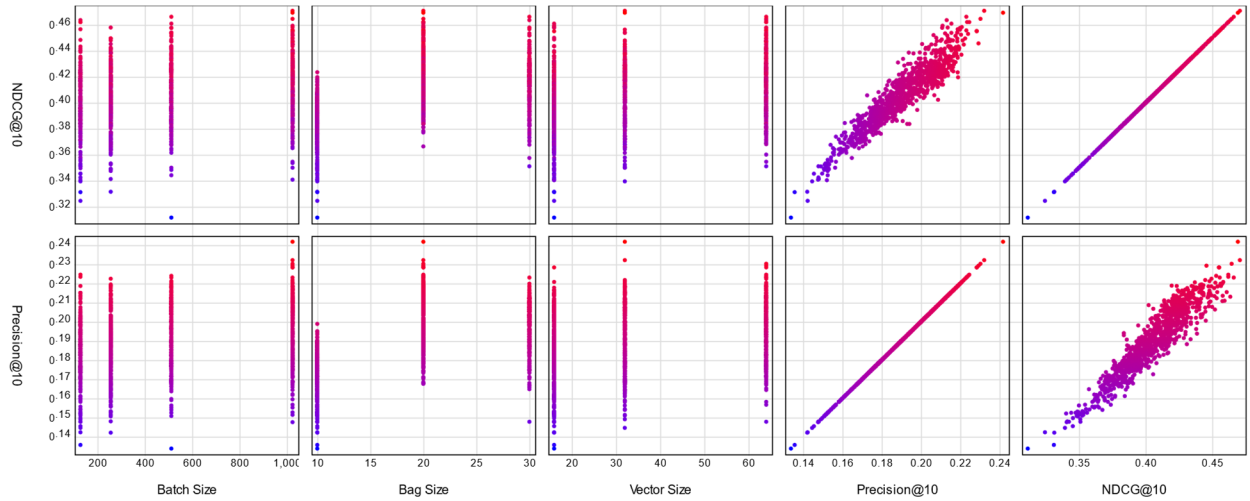

Figure 2. Evaluation of the model under different hyperparameters

In the fine-tuning phase, we find that the critical hyperparameter is the negative sample number. Selecting a larger number of negative samples can greatly improve the final performance of the model.

3.2. Performance experiment

We compare the NeuMF model as a baseline. To ensure the rigor of the experiment, we refer to the PyTorch implementation of Sun, Zhu, et al[12]. we also used the Grid Search technique to find the hyperparameters of the baseline model and obtained better results than the original paper.

We conducted multiple tests on MovieLens 100K and MovieLens 1M datasets, the top-10 data are taken for observation the results are shown in the Table 3.

Table 3. In the MovieLens data set, the evaluation results of top-K recommendation are conducted when K=10

	MovieLens 100K					MovieLens 1M				
Model	**Precision**	**Recall**	**HR**	**NDCG**	**MRR**	**Precision**	**Recall**	**HR**	**NDCG**	**MRR**
PNCF	0.257	0.331	0.969	0.641	1.140	0.274	0.231	0.947	0.647	1.271
NeuMF	0.243	0.309	0.957	0.626	1.112	0.235	0.193	0.910	0.587	1.058
Improve	5.4%	6.6%	1.3%	2.4%	2.4%	14.0%	16.4%	3.8%	9.3%	16.8%

The results show that PNCF's performance is significantly better than NeuMF's on the larger dataset except HR, which is already very high. PNCF also improved in the small dataset, but it was not obvious.

Of more interest are the results on the Bilibili dataset. On this dataset, we expect to predict the tags that users are likely to use in the future by using information about their interactions with the tags. PNCF leverages information from multiple users and multiple interactions of tags.

Observing the experimental results, PNCF achieves better performance in each of the metrics. When $k = 10$, PNCF gained 39.9% over NeuMF on the recall metric, while 37.7% was gained on the NDCG metric.

The experimental results prove that after PNCF can effectively utilize the many-to-many information in the data sources and substantially improve the effectiveness of recommendations. The experimental results in Table 4 show the specific magnitude of the improvement.

Table 4. Bilibili author's tag recommendation results

Model	K	Precision	Recall	HR	NDCG	MRR
NeuMF	1	0.1306	0.022	0.1306	0.1306	0.1306
	2	0.103	0.0865	0.4007	0.2651	0.2575
	5	0.0862	0.1453	0.5645	0.315	0.3031
	10	0.0691	0.2317	0.7242	0.35	0.3385
	20	0.0583	0.2917	0.798	0.3625	0.3532
	50	0.0455	0.375	0.8693	0.3712	0.3666
PNCF	1	**0.2705**	0.0498	0.2705	0.2705	0.2705
	2	0.1737	0.1546	0.5927	0.4332	0.4753
	5	0.1307	0.2296	0.7301	0.4679	0.5332
	10	0.0936	0.3241	0.8388	0.4819	0.572
	20	0.0748	0.3853	0.8846	**0.4828**	0.587
	50	0.055	**0.4666**	**0.9287**	0.4798	**0.6**

3.3. Ablation experiment

We also performed ablation experiments. to validate the effectiveness of pre-trained embeddings.

In addition to the PNCF model, NCF models called NPNCF with identical structure were constructed, but the embeddings of users and items were randomly generated and needed to be learned. Top K is taken for each user and evaluated against the test set, and the results are as Figure 3.

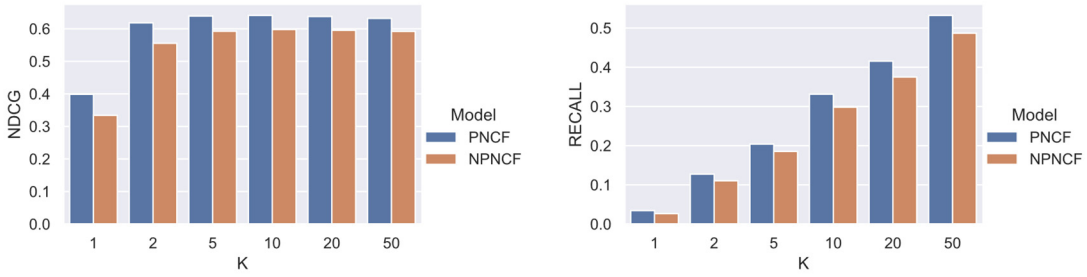

Figure 3. Ablation experiment results, using MovieLens 100K dataset

It can be found that the PNCF with pre-training is significantly better than the NCF model alone. The results of the experiments show that the recall is improved by up to 29% and the NDCG index is improved by up to 19% when using pre-trained embeddings.

4. CONCLUSION AND FUTURE WORK

In this work, we elaborate a pre-training task that makes it possible to complete user and item embeddings from the user and item interaction information separately, and then learn the relevance for the Top-K recommendation task through a feedforward neural network. We call this model PNCF, and our results show that this model can be very effective and can improve the effectiveness of recommendations. But there is still much work to be done. For simplicity, we have used a simple feed-forward neural network architecture. There is still a lot of room for optimization of this architecture. We can optimize the dropout rate, add regularization terms, or add residual networks or self-attention layers[13] to improve performance. We propose that it is possible to perform semantic embedding of users and items through interaction information, and we have experimentally demonstrated its effectiveness. However, although there is some research[14] on why embedding is so effective, there is no definitive academic conclusion. This is a very interesting question to explore. Many studies[15][16] have found that user interest drift and topic heat evolution information can improve recommendation performance. Integrating temporal information on top the PNCF may further improve the recommendation performance.

REFERENCES

[1] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model," Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (2008).

[2] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms," Proceedings of the 10th international conference on World Wide Web (2001)

[3] He, Xiangnan, et al. "Neural collaborative filtering," Proceedings of the 26th international conference on world wide web (2017).

[4] Hornik, Kurt, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators." Neural networks 2.5 (1989).

[5] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space," arXiv preprint arXiv:1301.3781 (2013).

[6] Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality," Advances in neural information processing systems (2013).

[7] Wang, Penghua, et al. "A Personalized Recommendation System based on Knowledge Graph Embedding and Neural Network," 2019 3rd International Conference on Data Science and Business Analytics (ICDSBA). IEEE, (2019).

[8] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805 (2018).

[9] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets," 2008 Eighth IEEE International Conference on Data Mining. Ieee (2008).

[10] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting," The journal of machine learning research 15.1, 1929-1958 (2014).

[11] Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift," International conference on machine learning. PMLR (2015).

[12] Sun, Zhu, et al. "Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison," Fourteenth ACM Conference on Recommender Systems (2020).

[13] Vaswani, Ashish, et al. "Attention is all you need," Advances in neural information processing systems (2017).

[14] Goldberg, Yoav, and Omer Levy. "word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method," arXiv preprint arXiv:1402.3722 (2014).

[15] Yin, Hongzhi, et al. "Dynamic user modeling in social media systems," ACM Transactions on Information Systems (TOIS) 33.3, 1-44 (2015).

[16] Wangwatcharakul, Charinya, and Sartra Wongthanavasu. "A novel temporal recommender system based on multiple transitions in user preference drift and topic review evolution," Expert Systems with Applications 185 (2021).