Deep Learning Based Recommender Models Comparison

Yani Mo
Department of Decision Science
HEC Montréal
Montréal, QC H3T 2A7
yani.mo@hec.ca

Mahdi Javadi
Department of Decision Science
HEC Montréal
Montréal, QC H3T 2A7
mahdi.javadi@hec.ca

Yuan Wang
Department of Decision Science
HEC Montréal
Montréal, QC H3T 2A7
yuan.wang@hec.ca

Abstract

From previous papers, we notice that the effectiveness of the recommendation system varies greatly from model to model. In this project, we have investigated how deep learning methods and classic recommender models perform in datasets of various sizes and with a "time stamps" variable. More specifically, we have dived into four deep neural network models, NeuMF, Wide and Deep, DeepFM, and improved DIN to analyze their performances in different scenarios.

1 Introduction

Recommender systems use multiple data related to users' interests to predict and recommend items to customers. The more efficient recommendation algorithm can leverage the previous interaction between customers and items to make more accurate recommendations. E-commerce companies leverage recommender systems to help users discover new and relevant items (products, videos, jobs, music, movie), creating a delightful user experience while driving incremental revenue. This project includes five parts. The first section is the introduction of the background on this topic. The second section is the literature research about recommender systems. Section three is the demonstration of our methodologies. In the fourth section, we offered descriptions about our data, the procedure of tuning hyper parameters, and comparisons among models. In the last section, the conclusion and future work are given.

2 Literature Review

There have been numerous researches in this area and especially the recommendations for movies. MOVREC [8], a collaborative filtering-based movie recommendation system, was proposed in 2015. Collaborative filtering gathers data from all users and produces suggestions based on it. A hybrid system that combines both collaborative and content-based method has been presented by Virk in 2015[14]. In another research, Campos offered a combination of Bayesian network and collaborative technique [2]. A clustering approach was adopted by Kuzelewska in 2014 [9]. He proposed two methods for clustering: Centroid-based solution and memory-based methods, and proved that accurate recommendations were generated. A recommendation system based on the user's history was also offered in order to generate recommendations by Chiru[3]. In this paper, different

techniques, including collaborative filtering, content-based and hybrid methods, were analyzed. An inductive learning algorithm built on tree structure was proposed by Li in 2004[10].

3 Methodologies

In this section, we briefly review the models that will be applied in our experiments.

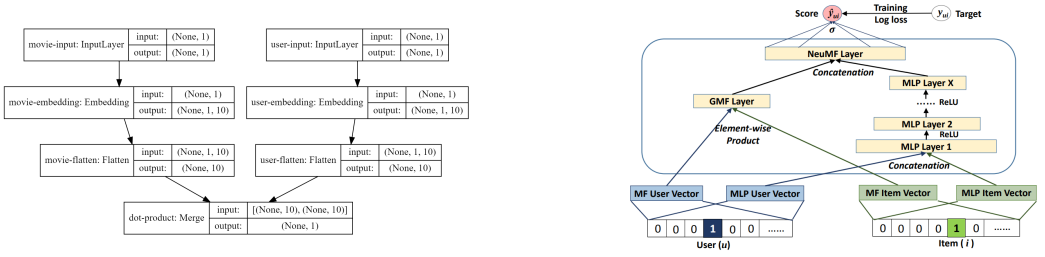
Recommendation Systems try to personalize our web experiences. There are two main Recommendation System methods: Collaborative Filtering and Content-based filtering. Collaborative Filtering filters out items that a user might like based on similar users' reactions. The content-based method employs user preferences for product features to make recommendations. Each of these methods has advantages and disadvantages. For example, Content-Based Filtering needs more data regarding user preferences to find the best match, but it can address the Cold-start problem. Collaborative Filtering needs an extensive data set as well, within which users are active in rating a product beforehand to make accurate predictions. However, it does not require content analysis extraction. There are Hybrid methods as well that combine both methods. Collaborative filtering has two sub-categories that are generally called memory-based and model-based approaches. One advantage of memory-based methods is that they can recommend a larger number of items to a larger number of users than other methods. They have extensive coverage, even when working with large sparse matrices. We will see the K- Nearest Neighbor (KNN) method from the first group and then the second one, the Matrix Factorization (MF).

3.1 K- Nearest Neighbour

KNN tries to form groups or clusters of similar users based on their common ratings and then make predictions by the average rating of top k nearest neighbors[11]. There are different kinds of KNN, for example, KNNBasic model,KNNWithMeans, KNNWithZScore etc. KNNWithMeans takes the mean ratings of each user into account. KNNWithZScore takes z-score normalization of each user into account. In our experiments, KNN with Means was chosen as representative due to its better performance, which means the mean rating of each user was considered. We used the Surprise package in python for this.

3.2 Matrix Factorization

Matrix Factorization is the most popular collaborative filtering method, and it tries to factorize the user-item matrix or the utility matrix into vectors of latent features by projecting these vectors into a shared feature space of N dimensions (latent dimension). Then by doing an inner product of these user-item latent vectors (or cosine similarity) in this feature space, it is possible to model the user-item interactions.[1].



(a) Classic Matrix Factorization

(b) Neural Matrix Factorization

Figure 1: Figures for MF part

Let p_u and q_i denote the latent vector for user u and item i, respectively; MF estimates an interaction y_{ui} as the inner product of p_u and q_i (K is the number of latent dimension):

$$\hat{y}_{ui} = f(u, i|p_u, q_i) = p_u^T q_i = \sum_{k=1}^{K} p_{uk} q_{ik}$$

This vanilla implementation of Matrix Factorization could be improved as well. Although Matrix Factorization has shown effectiveness, its performance can be limited because of the simple inner product in modeling the interaction. The inner product that linearly does the multiplication of features may not be enough to comprehend the complex structure of the user interaction[6].

A new method called Neural Collaborative Filtering was offered to address this issue, and it was widely used, but there was a shortcoming with this model. It concatenates interaction vectors of the user and item latent features, which is not adequate for modeling the collaborative filtering. Finally, a Multilayer Perceptron approach was offered by adding hidden layers on the concatenated vector to comprehend the interaction between latent feature vectors. It bestows the model a great amount of flexibility and makes the model able to learn the interactions between p_u and q_i compared to the methods like Generalized Matrix Factorization, where a fixed element-wise multiplication is implemented on them.

$$z_1 = \phi_1(p_u, q_i) = \begin{bmatrix} p_u \\ q_i \end{bmatrix}$$

$$\phi_2(z_1) = a_2(W_2^T z_1 + b_2)$$

$$...$$

$$\phi_L(z_{L-1}) = a_L(W_L^T z_{L-1} + b_L)$$

$$\hat{y}_{ui} = \sigma(h^T \phi_L(z_{L-1}))$$

In the formulas above, the W_x , b_x, and a_x are weights, biases and activation functions, respectively. L layers are stacked on top of each other to capture the interactions better.As seen in the Neural Matrix Factorization figure, the MLP and GMF models are allowed to learn embeddings separately and their outputs are combined by concatenating their last hidden layer:

$$\phi^{GMF} = P_u^G \odot Q_i^G$$

$$\phi^{MLP} = a_L(W_L^T(a_{L-1}(...a_2(W_2^T \begin{bmatrix} p_u \\ q_i \end{bmatrix} + b_2)...)) + b_L)$$

$$\hat{y}_{ui} = \sigma(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}$$

By substituting the inner product with a set of neural networks, Neural Collaborative Filtering (NCF) was offered at first. The model was able to learn arbitrary functions on the data set, and it could include Generalized Matrix Factorization (GMF) method in it. A multi-layer perceptron was introduced to comprehend user-item interactions (embeddings) to boost the model's ability to handle non-linearity.

Multiple experiments showed its superior performance compared to other models as well as the classical matrix factorization.

3.3 Wide and Deep Modeling

The Wide and Deep model[5] has been proposed to address large-scale regression and classification. Compared to traditional Collaborative Filtering methods, Shaoyun Shi (2017)[12] used Wide and Deep model to address cold-start problem in job recommendation task, since Wide and Deep method can efficiently leverage users' and items', and other external information. Yan Liu et al.(2019)[15] compared the traditional models such as LR and GBDT, with deep models such as Wide and Deep and DeepFM in store site recommendation project. According to Google's definition for Wide and Deep model[13], Wide and Deep model combines linear model (wide part) and deep neural network into one model. The details about its architecture can be shown as the following:

The Wide part[13]

The wide linear model is used to memorize sparse feature interactions from input raw features, and it has the form $y = w^T x + b$, where y is the prediction, $x = \{x_1, x_2, ..., x_n\}$ is a vector of n features, $w = \{w_1, w_2, ..., w_n\}$ is the vector of model parameters and b is the bias. The n features include raw

entity features and transformed features. The transformed features are cross-feature transformation defined as the following:

$$\phi_k(x) = \prod_{i=1}^{d} x_i^{c_{ki}}, c_{ik} \in \{0, 1\}$$

The Deep part[13]

The deep part is deep neural network which is typical feed forward network. Each hidden layer performs the following computation[13]:

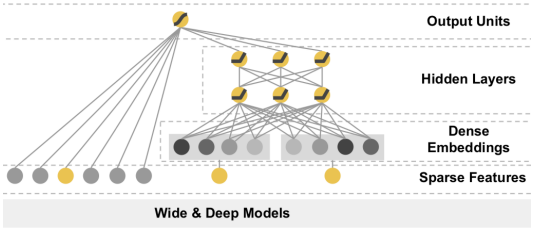
$$a^{l+1} = f\left(W^l a^l + b^l\right)$$

where l is the layer number and f is the activation function, $a^{(l)}, b^{(l)}$, and $W^{(l)}$ are the activations, bias, and model weights at l-th layer.

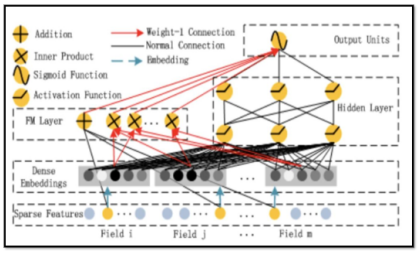
In the deep part, DNN is used to generalize previously unseen feature interactions through low dimensional embeddings by matching items to queries, which are close to each other in the embedding space.

3.4 DeepFM

To effectively learn feature interactions and avoid computation cost, Huifeng Guo(2017)[7] proposed the DeepFM model which is an FM based Neural Network for CTR prediction. Besides, it is an end-to-end deep learning method, which emphasizes both low and high order feature interactions behind users' behaviors. As the following DeepFM figure shows[7], the DeepFM model has a structure similar to the Wide and Deep model. These two models both concatenate the DNN model with another model. DeepFM combines the DNN model with the FM model instead of a linear model to avoid manual feature engineering, and this FM part can learn low and high order feature interactions automatically. The second difference is that both the FM part and the Deep part share raw features as inputs, which guarantees the consistency of features in DeepFM.



(a) Structure of WideDeep



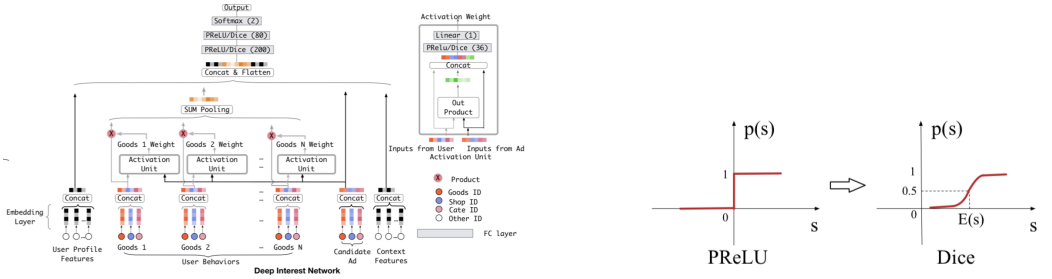
(b) Structure of DeepFM

Figure 2: Figures for WideDeep and DeepFM

3.5 DIN

Deep Interest Network, widely used nowadays to predict CTR, is developed and deployed in online display advertising system by Alibaba. In formal presented models, the basic idea is mapping large scale sparse input features into low dimensional embedding vectors first[4]. Then transform them into fixed-length vectors. In the end, the vectors are concatenated together before being fed into a multi-layer perceptron to learn the nonlinear relations among features[4]. Fixed-length presentation vector of user features brings difficulty for EmbeddingMLP methods to capture user's diverse interests effectively from historical behaviors[4] since the length of vector is fixed towards any candidate product. One solution to this problem is expanding the dimension of the fixed-length vector. However, it will increase the number of learning parameters dramatically[4], which is computationally expensive. In addition, it might encounter the problem of overfitting.

In reality, we assumed that only part of user's interests will influence his/her action[4]. For example, a customer has more chance to click a recommended golf ball due to the purchase of golf pole instead of the purchase of high-heels in the last week's shopping history . Xiaoqing[4] introduces attention mechanism by activation unit to adapt the representation vector of user interests towards a candidate product. In other words, DIN will search for historical behaviors and pays attention to the ones most relevant to the candidate product. Then it obtains the representation vector by taking a weighted sum pooling of all values. In conclusion, the most dominant part of the user interests' representation is the behaviors most relevant to the candidate product. The following figure is the structure of DIN:



(a) Structure of DIN

(b) PRelu vs Dice

Figure 3: Figures for DIN

As we can see activation units are applied on the user behavior features. The following statement explains how it works.

$$v_U(A) = f(v_a, e_1, e_2, ..., e_H) = \sum_{j=1}^{H} \alpha(e_j, v_A) e_j = \sum_{j=1}^{H} \omega_j e_j \tag{1}$$

where $e_1, e_2, ..., e_H$ is the list of embedding vectors of behaviors of user U with length of H,v_a is the embedding vector of candidate product A. In this way,$v_U(A)$ varies over different ads. $\alpha()$ is a feed-forward network with output as the activation weight[4]. In other words, query is the item embedding and key is the historical behavior Embedding. The calculated similarity is the weight. However,the constraint of $\sum_i^n w_i = 1$ isn't applied here since we aim to keep the intensity of user interests[4], which is different from traditional attention.

Another innovation about DIN is the Dice activation function, which can be regarded as a general form of PRelu. The difference is shown in figure3. We can conclude that it's more suitable when the inputs of each layer follow different distributions since it doesn't have a strict rectified point with value of 0[4].

4 Experiments

4.1 Introduction of the data

In our project, we used the Douban Movie Dataset from https://github.com/csuldw. Douban Movie is a Chinese platform where netizens can share their comments and ratings on various movies. Besides, this dataset was crawled on September 2019,including userID, movieID, movie name, genre and ratings given by users etc. The other two real-world e-commerce datasets from https://github.com/MengtingWan/marketBiasused in our project are Amazon Electronics which is the Electronics category on Amazon, and ModCloth which is a women's clothing website.

These three datasets have various sizes, and these datasets contain attributes about items and users as shown in figure 4.

After cleaning the datasets, the detailed descriptions of these three datasets can be presented as figure 5.

Users' ratings distribution from 1-5 scale can be shown as figure 6.

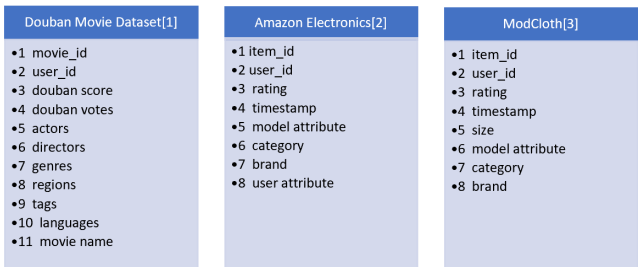


Figure 4: Attributes about items and users

Datasetas	Nomber of users	Nomber of items	Nomber of ratings	Users/Rating ratio
Douban Movie	*115087*	35654	220411	1.9153
Amazon Electronics	40401	1892	45166	1.1229
ModCloth	12657	516	18557	1.4660

Figure 5: Descriptions of three datasets

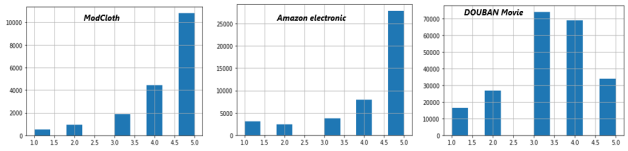


Figure 6: Users' ratings distribution

Then, we divided each of these datasets into two general parts: 80%for training, 20%for testing. For more details, we used 20% of the training set to tune hyper-parameters as the validation set in our experiment.

4.2 Data preprocessing

Resources in python have been used, as the following:

- *thulac* package(THU Lexical Analyzer for Chinese) was applied to separate entity in Chinese vocabulary-by-vocabulary into sequence feature.
- *re* package was used to detect a set of strings that matches.
- *sklearn.preprocessing* for label encoding.
- *DeepCTR* library: SparseFeat, VarLenSparseFeat were applied for spare features and sequence features embedding.

By employing the above resources, we preprocessed raw input from our datasets as following:

1. Because of low portion of missing data, to simplify the process, we removed rows with missing data.

2. In Douban Movie Dataset, most of the entities are in Chinese without a space between two words. Thus, we have divided these entities word-by word with " | " to make them easier for encoding, for instance:

```
Movie Name = ″ 王室的婚礼″ in English is ″Royal Wedding″
Before Splitting:
Movie Name =″ 王室的婚礼 ″
After Splitting:
Movie Name = ″ 王室|的|婚礼″
```

3. To obtain relationships between two features in the datasets, we encoded discrete features, such as UserID, ItemID, ItemSize, etc, and embedded them into vectors of continuous real numbers.

4. Label encoding all sparse features with multiple values into sequence features, such as 'Item category','REGIONS', 'TAGS', 'LANGUAGES', etc.

5. Then embedding and padding sequence features to make them into the dense features with the same size as the other features.

4.3 Model training and hyper-parameters tuning

In the tuning step, we take the Amazon Electronic dataset(https://github.com/MengtingWan/marketBiasused) as an example to show the tuning results.

4.3.1 WideDeep model and DeepFM model

We can construct the Wide and Deep model by using the above information. As we mentioned in the previous dataset description part, the size of the Amazon Electronic dataset is 45166 with 8 features. According to Deepctr API[13], firstly, we can construct, and train the Wide and Deep model on the 80% of the training set and then tune hyper-parameters with the rest 20% of the training dataset to avoid over-fitting. Early stopping is also implemented if validation accuracy keeps decreasing. We applied the same logic when building DeepFM model.

Keeping the other hyper-parameters setting fixed, when we are tuning one hyper-parameter, We focus on: 1.Optimizer; 2. Learning-rate; 3. DNN-dropout; 4. DNN-hidden-unit; 5. Embedding-dimension; 6. Epochs; 7. Batch-size to detect the impact of different hyper-parameters to the model performance. Finally, we can identify the best hyper-parameters setting for model comparison. The tuning results for two models can be shown as follow:

Hyper-parameters\Test Set	Optimal setting
Optimizer	SGD
Learning_rate	0.01
DNN_dropout	0.2
DNN_hidden_units	(256,128)
Embedding_dimension	32
Epochs	5
Batch_size	32
MSE	**1.4896**

(a) hyper-parameters for WideDeep

Hyper-parameters\Test Set	Optimal setting
Optimizer	Adam
Learning_rate	0.0001
DNN_dropout	0.5
DNN_hidden_units	(128,128)
Embedding_dimension	32
Epochs	3
Batch_size	32
MSE	**1.5261**

(b) hyper-parameters for DeepFM

4.3.2 NeuMF and DIN

In Neural Matrix Factorization, the hyper parameters we focused on are 1.Optimizer; 2. Learning-rate; 3. DNN-dropout; 4. Latent-dimension; 5. Activation-function; 6. Epochs; 7. Batch-size. we also had two fully connected layers (instead of inner product in the classical matrix factorization).

In terms of DIN, we regarded the sequence of ratings as historical behaviors, which are the keys. We first sort the behavior according to the "timestamp" variable. Since some of the users have few ratings, we constrained to at least ten ratings for each user to better train our model. For those users who have less than ten ratings, We randomly chose items to recommend and the missing ratings were replaced by zero. Then, we applied the same logic to train and tune hyper-parameters. We focused on the hyper-parameters mentioned before and struck a balance between time consumption and accuracy. The best combinations are:

Hyper-parameters\Test Set	Optimal setting
Optimizer	Adam
Learning_rate	0.01
DNN_dropout	0.2
Latent_dimension	10
Activation_function	Relu
Epochs	50
Batch_size	64
MSE	0.904

(a) Hyper-Parameters for NeuMF

Hyper-parameters\Test Set	Optimal setting
Optimizer	Adam
Learning_rate	0.0001
DNN_dropout	0.2
DNN_hidden_units	(128,64)
Activation_function	Dice
Epochs	5
Batch_size	32
MSE	1.471

(b) Hyper-Parameters for DIN

4.4 Models' Comparison

Since we are predicting ratings and our output is continuous, we used mean squared error to measure the performance of our models. We applied KNN with means and Classic MF as our bench marks. As we can see from the table, Wide and Deep performes best in Movie dataset. NeuMF performs best in both Electronics and ModCloth datasets.

MSE	KNN with Means	Classic MF	NeuMF	Wide&Deep	DeepFM	DIN
Movies	1.077	3.037	0.976	0.964	1.087	1.105
Electronics	1.566	3.791	0.904	1.489	1.526	1.471
ModCloth	1.177	3.759	0.754	1.109	1.575	1.315

Figure 9: MSE for Models

5 Conclusions and Discussion

Our goal of this project is to implement several recommender models based on neural networks and to discuss the conditions, in which each model will be suitable. In our experiment, we took three different size of datasets into consideration. Douban Movie has 220411 ratings in total without "timestamp". Amazon Electronics and ModCloth have 45166, 18557 ratings respectively with "timestamp".

NeuMF performs best on average since NeuMF is fusing the GMF and MLP. GMF and MLP are the two parts of Neural Collaborative Filtering; the GMF part uses a linear kernel to model user-item interactions, similar to vanilla or classic MF. A MLP component that models nonlinear interactions using multiple neural layers. NCF blends these models in order to superimpose their desired properties. Then the output of NCF model, which is the concatenation of GMF and MLP, is fed to the NeuMF layer. It bestows the model a great amount of flexibility and makes the model able to learn the interactions between user and item latent vectors.

Since the datasets ModCloth and Electronics contain timestamp as one of the explanatory variables, we did expect DIN could beat the others at least in these two datasets. The reason explaining the actual outcome of our experiment is the datasets. In DIN, we forced the ratings as historical sequence of behavior regardless of lacking of ratings for bunch of users. In other words, rating itself can't present users' real time reactions towards a candidate product accurately. Maybe, the customers consumed products without any ratings. Maybe, they changed their interest without consumed certain

type of products before. In the original report, DIN beats other models when predicting advertisement based on click through rate.

Our next step to elaborate our experiment is to get real-time data and make dynamic predictions.

References

[1] Josef Bauer and A. Nanopoulos. Recommender systems based on quantitative implicit customer feedback. *Decis. Support Syst.*, 68:77–88, 2014.

[2] L. M. D. Campos, J. M. Fernández-Luna, J. Huete, and M. A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *Int. J. Approx. Reason.*, 51:785–799, 2010.

[3] Costin-Gabriel Chiru, Catalina Preda, V. Dinu, and M. Macri. Movie recommender system using the user's psychological profile. *2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, pages 93–99, 2015.

[4] Xiaoqiang Zhu Guorui Zhou, Chengru Song. Deep interest network for click-through rate prediction. *arXiv:1706.06978v4*, 2018.

[5] J.Harmsen T.Shaked T. Chandra H. Aradhye G. Anderson G. Corrado W.Chai M. Ispir R. Anil Z. Haque L. Hong V. Jain X. Liu H.Cheng, L.Koc and H. Shah. Wide deep learning for recommender systems. *CoRR*, abs/1606.07792, 2016.

[6] X. He, Lizi Liao, Hanwang Zhang, L. Nie, X. Hu, and Tat-Seng Chua. Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[7] Yunming Ye Zhenguo Li Xiuqiang He Huifeng Guo, Ruiming Tang. Deepfm: A factorization machine based neural network for ctr prediction. *In Proceedings of the IJCAI*, pages 2782–2788, 2017.

[8] M. Kumar, D. Yadav, Ashutosh Kumar Singh, and V. K. Gupta. A movie recommender system: Movrec. *International Journal of Computer Applications*, 124:7–11, 2015.

[9] U. Kuzelewska. Clustering algorithms in hybrid recommender system on movielens data. *Studies in Logic, Grammar and Rhetoric*, 37:125 – 139, 2014.

[10] P. Li and S. Yamada. A movie recommender system based on inductive learning. *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, 1:318–323 vol.1, 2004.

[11] Wenzhong Liang, G. Lu, Xiaoyu Ji, J. Li, and Dingrong Yuan. Difference factor' knn collaborative filtering recommendation algorithm. In *ADMA*, 2014.

[12] Hongyu Lu Yiqun Liu Shaopin Ma Shaoyun Shi, Min Zhang. Wide deep learning in job recommendation: An empirical study. *13th Asia Information Retrieval Societies Conference, AIRS 2017*, pages 112–124, 2017.

[13] Weichen Shen. Deepctr: Easy-to-use,modular and extendible package of deep-learning based ctr models. *https://github.com/shenweichen/deepctr*, 2017.

[14] H. Virk, Er. Maninder Singh, and Er. Amritpal Singh. Analysis and design of hybrid online movie recommender system. 2015.

[15] Nuo Li Jing Zhang Jingmin Chen Daqing Zhang Yinxiao Liu Zhiwen Yu Sizhe Zhang Yan Liu, Bin Guo and Lina Yao. Deepstore: An interaction-aware wide-and-deep model for store site recommendation with attentional spatial embeddings. *IEEE Internet of things journal 6, no. 4*, pages 7319–7333, 2019.