Case study of Deep Learning Hybrid Recommender System for Ranking

Maksimilian Pavlov [1] **Hai Le** [1] **Egor Malkershin** [1]

Abstract

Learning sophisticated and complex interactions behind user behaviors is critical for any recommender systems. This is especially true for top-n/reranking problems; further emphasized by the existence of the cold-start problems. Prior research introduced DeepFM - a combination of factorization machines (FM) for recommendation and deep learning (DL) for feature learning (5). Experiments showcased the effectiveness and efficiency of DeepFM on CTR prediction. In this paper, we implemented and experiment the effectiveness of DeepFM on re-ranking/top-n problem. Furthermore, we compare the performance of this architecture with several baselines including SVD, LightFM, and DSSM. Our experimental results shows that DeepFM performed better in both user cold-start and warm-start scenrios in COV@20 and MRR@20.

Github repository: link

1. Introduction

With the third industrial revolution comes the boom in Web-based services. The growth of accessibility to the internet leads to a substantial number of users consuming online content daily. With the introduction of mobile devices, accessing web pages, purchasing items in e-commerce sites, watching online content via streaming services or interacting with loved ones via social media has never been easier. Understanding the underlining factors that characterize user preferences and their future behaviors is crucial toward improving user experience. This comes in the form of recommendations of new content that aligns with the user's taste. In short, recommendation is the problem of predicting the ratio of interaction between users and items such that a user would prefer an item to another one. In this paper, we focused our research on TV shows/movie/film recommender system (RS).

In general, recommendation lists are created based on user-item past interactions, item properties, user preferences, and some other additional data. There are three main techniques used in traditional RSs: Content Based Filtering (CBF), Collaborative Filtering (CF) and Hybrid RS.

Content-Based Filtering : This technique creates user profiles taking which users previously interact with items into account and simply recommends items with similar contents to user profiles. The recommendation process uses properties of items as contents of users.

Collaborative Filtering: The CF based recommendation systems represent users' preferences as n-dimensional rating vectors where n is the number of items in the system. The key idea of CF is that similar users/items share similar interests. CF recommends items to users based on their liked items by computing similarities between users and items. There are two categories of CF:

- **Item-based**: Calculates the similarity between the items that user rates previously and other items.
- **User-based**: Calculates the similarity between users

Hybrid Systems: These systems combine two or more types of recommendation techniques to produce better recommendations. DeepFM combines both factorization machine (FM) and deep neural network (DNN) to extract both low and high-order feature interaction. For instance, people tends to download food delivery application at meal time. This is a lower order (order-2) interaction between app category and time-stamp. Another example is the male teenagers tends to like RPG and shooting games. This is a higher order (order-3) interaction of app category, user gender and age.

We propose that by utilizing DeepFM's strength of extracting both low and high order feature interaction, this model will significantly improve the quality of top-n accuracy. We hypothesized that the improvement in recall will be emphasized in cold start scenarios. These scenarios are defined as when a only an insignificant amount of training data is available. For instance, we aim to recommend an item to a brand-new user who has only interacted with 1 item

[1]Skolkovo Institute of Science and Technology, Moscow, Russia. Correspondence to: Hai Le <@skoltech.ru>.

Final Projects of the Introduction to Recommender System, Skoltech, Moscow, Russian Federation, 2023.

previously.

Throughout our research, we compared and contrast the re-ranking quality of DeepFM against SVD, LightFM, and DSSM. Our contributions can be summarized as follows:

- We implemented DeepFM as discussed by (1). Thus, we trained and tuned the parameters of DeepFM on the KION dataset.
- To compare the performances of DeepFM, we implemented 3 baselines: SVD, LightFM and DSSM. With the 2 baselines being standard matrix factorization techniques and the latter being a DL technique. Similarly, we also finetuned these baselines on the KION dataset (2)
- Lastly, we evaluate the results of our tuned model and discuss the performance of all 4 models. To compare and contrast the models, we devise two scenerios: 1) warm start/standard scenario and 2) cold start scenario.

2. Data Pre-processing & Pipeline

Before we discuss the implemented DeepFM and the baselines, we would like to introduce the experimented dataset Moreover, in this section, we will discuss the data preprocessing stage as well as provide a general overview of our pipeline.

2.1. Dataset Information

For our experiments, we chose the MTS Kion Implicit Contextualised Sequential Dataset for Movie Recommendation (2). As an alternative to the popular MovieLens dataset (4), the MTS Kion dataset is based on the implicit interactions registered at the watching time, unlike Movielens's explicit ratings. This dataset also provide rich contextual and side information including interactions characteristics (such as temporal information, watch duration and watch percentage), user demographics and rich movies meta-information. This is ideal for the proposed DeepFM model to extract both low and high order feature interaction.

Kion dataset provides interactions over a 5 months period starting on 13.03.2021 and ending on 22.08.2021. The dataset contains 5,476,251 interactions of 962,179 users with 15,706 items.

2.2. Data Pre-processing

As mentioned in section 1, we experimented our models on both cold and warm start scenarios. To prepare for both scenarios, our pre-processing stage can be summarized in the following steps:

- Filter out all users who consumed less than 2 items.
- Prepare our train data by filtering out users who consumed more than 5 items.
- Prepare our **cold start** test data by filtering out users who consumed more than 2 items and less than 5 items. The rest of the unfiltered items to be the test data for our **warm start** scenario.
- Prepare the holdout data for evaluation metrics computation. We apply the leave last out strategy for the holdout data.
- For holdout and training, we stratify by movies. Lastly, we apply one-hot-encoding for items and users features.

We utilized the following features to train our model:
Interaction features: date, duration, watched percent.
User features: age, income, kids, sex.
Item features: content type, title, release year, genres, age rating, studio, director, actors, keyword.

2.3. Pipeline Overview

2.3.1. SVD & LightFM

For our pure matrix factorization baseline collaborative filtering fine-tuned scaled SVD was implemented as a standard approach. LightFM (3) was implemented as hybrid model baseline. In (SVD & LightFM), we simply feed our train data into the model. We then evaluate both models for warm and cold start data on the metrics mentioned below.

Mostly LightFM utilize Weighted Approximate-Rank Pairwise (WARP) (7) loss function in order to optimize the parameters of the model. WARP loss is defined as:

$$\sum_{i=1}^{|\mathcal{D}|} \sum_{j=1}^{m_i} \max_{k=1}^{|I|-m_i+1} [\Delta(i,j,k) - \Delta(i,j,i_k)] \tag{1}$$

where:

- $\mathcal{D}$ is the set of training examples.
- m_i is the number of items in the candidate set I for example i.
- i_k is the index of the k-th ranked item for example i.
- $\Delta(i,j,k)$ is the difference between the score of the k-th ranked
- and the score of the j-th ranked item, for example i and position j.
- $\Delta(i,j,i_k)$ is the difference between the score of the k-th ranked item and the score of the ideal item (i.e. the item that maximizes the score) for example i and position j.

2.3.2. DSSM

In case of DSSM, for warm start scenario, we excluded items that has been consumed by less than 10 users; and items that has been consumed at least 35% of item. For cold start scenario, data pre-processing is similar to other models. For the loss function of DSSM, We used triplet loss. It refers to the difference in distance from user $f(A)$ to good item $f(P)$ and user to bad item $f(N)$:

$$\mathrm{L} = \max(||f(A)-f(P)||_2-||f(A)-f(N)||_2+\alpha, 0) \quad (2)$$

2.3.3. DeepFM

For the DeepFM model, we used watch percentage to be our target feature. This feature refers to how much of the content the user has consumed. We then binarize this feature into 1's and 0's based on a proposed cutoff (for instance, if the user consumed $\leq 50\%$, replace with 0, else replace with 1). This cutoff were tuned during our experiments. In addition to the binarization, the original train data were split further into train and validation (95/5 ratio). Moreover, we utilized Binary Cross Entropy (BCE) as our loss function for DeepFM.

$$\mathrm{L} = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \cdot \log(\hat{y_i}) + (1-y_i) \cdot \log(1-\hat{y_i}) \quad (3)$$

Similar to our pure matrix factorization baseline, we then evaluate this model for both warm and cold start with the metrics below.

2.3.4. Evaluation Metrics

The metrics used throughout our experiments includes COV@N - catalog coverage and MRR@N - mean reciprocal rank (for our research, we focused on the top-20 recommendations). Catalog coverage refers to how many different items ever appear in the top-k recommendations. On the other hand, mean reciprocal rank refers to the calculation of the reciprocal of the rank at which the first relevant document was retrieved. Both evaluation metrics can be shown as the following equations:

$$MRR@20 = \frac{1}{\text{test users}} \sum_{\text{test users}} \text{hit} \cdot \frac{1}{\text{hit rank}} \quad (4)$$

$$COV@20 = \frac{|U_{u \in U_{\text{test}}} R_u|}{|I|} \quad (5)$$

3. Model Description

With our data preparation and pipeline in mind, we will discuss the models implemented and its architecture in this section.

3.1. Singular value decomposition (SVD)

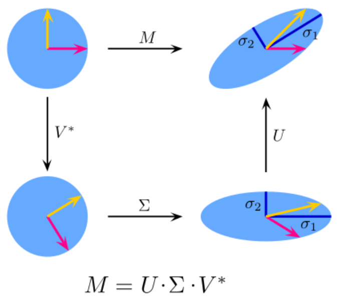


Figure 1. Singular Value Decomposition

SVD is a popular baseline for recommender systems. It can handle sparse data, and it can capture complex relationships between users and items. SVD better performance can be achieved by low-rank approximation:

$$A = U \Sigma V^T \quad (6)$$

$$||A - R||_F^2 \Rightarrow min \quad (7)$$

$$R = U_d \Sigma_d V_d^T \quad (8)$$

where A - initial user-item interaction matrix, R - truncated matrix by rank d The U matrix represents user preferences in terms of latent factors, while the V matrix represents item attributes in terms of latent factors. The Σ matrix contains the singular values, which represent the strength of each latent factor. To predict a user's rating for an item, SVD takes scalar product of the user's latent feature space and the item's columns in item's latent feature space:

$$r_{ui} \approx p_i^T q_j = \sum_{k=1}^{d} p_{ik} q_{jk} \quad (9)$$

where p_i - latent factors vector for user i, q_j - latent factors for item j

This gives an estimate of the user's rating for the item based on their preferences and the item's attributes. Additionally, it is useful to apply data normalization to SVD data input, because SVD is sensitive to the scale of the input data. If the data is not normalized, the results of SVD may be biased towards the variables that have larger scales. Normalizing the data ensures that all variables have equal importance in the analysis and prevents the dominance of certain variables over others.

$$\hat{A} = D^{f-1} A \quad (10)$$

where f is a scaling parameter.

3.2. LightFM

LightFM is a hybrid recommender system model that uses both collaborative filtering and content-based filtering techniques to recommend items to users. It is a matrix factorization model that learns the latent features of users and items by factorizing the user-item interaction matrix. The model uses a combination of matrix factorization and neural networks to predict the likelihood of a user interacting with an item. It also incorporates item metadata such as genre, author, and other item features to improve the recommendations. LightFM utilizes parametrization to obtain user and items feature embeddings.
User latent representation:

$$q_u = \sum_{j \in f_u} e_j^U + \sum_{j \in f_u} b_j^U \tag{11}$$

Item latent representation:

$$p_i = \sum_{j \in f_i} e_j^U + \sum_{j \in f_i} b_j^U \tag{12}$$

The model's prediction for user and item i defines as a dot product of user and item representations.

$$\hat{r_{ui}} = f(q_u \cdot p_i + b_u + b_i)$$

As an activation function sigmoid and other can be used. Algorithm is optimized by using the gradient descent algorithm.

3.3. DSSM

DSSM is a DNN used to model semantic similarity between a pair of strings. In other words, semantic similarity of two sentences is the similarity based on their meaning (i.e. semantics), and DSSM helps us capture that.

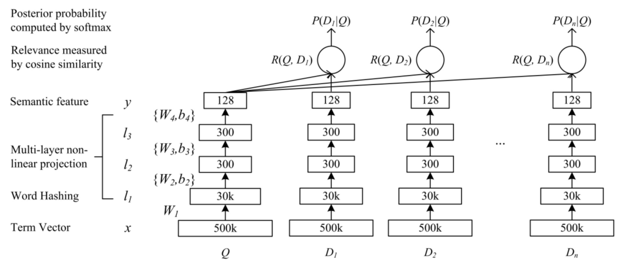


Figure 2. Architecture of original DSSM model

Our DSSM model consists of three dense layers (fully connected NN), each with 128 neurons. The input layer takes 8813 (item df shape) product parameters. Within each layer, DSSM utilized ReLu activation function and L2 regularization to reduce the chance of model over-fitting. Lastly, the last layer of DSSM includes a dense layer with a linear activation function is used, which gives a prediction of the product rating.

3.4. DeepFM

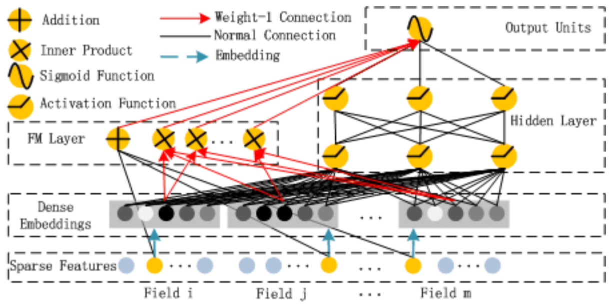


Figure 3. Wide & Deep Architecture of DeepFM

As depicted in the figure above, DeepFM consists of two components - FM component and DNN component; both of which shares the same input. For any given feature i, a scalar w_i is used to measure order-1 importance, and a latent vector V_i is used to measure i's impact of interaction with other features (order-2 importance). Moreover, V_i is also fed into our DNN component to model high-order feature interaction. All parameters are trained jointly, summarized in our equation below:

$$\hat{y} = \text{sigmoid}(y_{FM} + y_{DNN}), \hat{y} \in (0, 1) \tag{14}$$

where $\hat{y}$ is the predicted percentage of whether the user like this item or not, y_{FM} is the output of our FM part, and y_{DNN} is our DNN's output. In other words, DeepFM will predict the likeliness of user u liking a given item i. Thus, as a note,

3.5. Factorization Machine Component

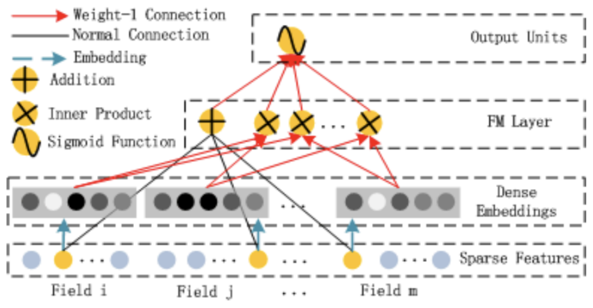


Figure 4. Factorization Machine Architecture

Our FM component, proposed by (6), aims to learn feature interactions for recommendation. FM can capture order 1 and 2 feature interactions effectively, especially for sparse data. FM models pairwise feature interaction (order-2) as as inner product of respective feature latent vector V_i and V_j. With this flexible design, FM can train any given latent vector V_i whenever i appears. With the rarity of feature interaction in train data, FM can learn these features more effectively. As shown in the figure above, FM is the summation of an **addition** unit and a number of **inner product** units:

$$y_{FM}(x) = (w, x) + \sum_{i=1}^{d} \sum_{j=i+1}^{d} (V_i, V_j) x_i \cdot x_j \qquad (15)$$

where $w \in R^d$ and $V_i \in R^k$ (k is given by fast inner/outer matrix multiplication). The **addition** unit reflects the importance of order-1 while the **inner product** reflects order-2.

3.6. Deep Neural Network Component

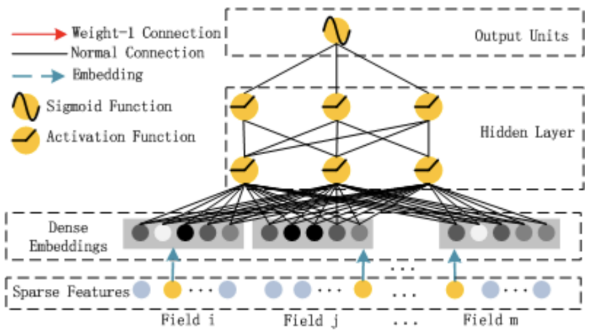


Figure 5. Deep Neural Network Architecture

The deep component is a feed-forward neural network, which is used to learn high-order feature interactions. The sparse features can be of different length but their embeddings remain the same size. The latent feature vector (V) in FM are now the DNN's learning weights. In this DeepFM, FM model was not pre-trained but included in the combined model as the overall learning architecture. Given the output of embedding layers:

$$a^{(0)} = [e_1, e_2, ...e_m] \qquad (16)$$

where e_i is i-th the embedding, m is the number of embeddings, and $a^{(0)}$ is the input into our DNN, thus our forwarding process is defined as:

$$a^{(l+1)} = \sigma(W^{(l)} a^{(l)} + b^{(l)}) \qquad (17)$$

where l is layer depth, σ is activation function, $a^{(l)}$, $W^{(l)}$, $b^{(l)}$ are the output, model weight, l-th bias respectively. After that, a dense real-value feature vector is generated, which is finally fed into the sigmoid function for our prediction of preferences (thus used for re-ranking).

$$y_{DNN} = \sigma(W^{|H|+1} \cdot a^H + b^{|H|+1}) \qquad (18)$$

where $|H|$ is the number of hidden layers.

4. Experiments & Results

In this section, we will discuss the experiment descriptions, methodologies and results.

4.1. Experiments Methodology

As mentioned in the section 2.1, we utilized the Kion dataset. Moreover, the experiment methodology consists of two main parts: warm-start scenario and cold-start scenario. As discussed section 2.2, each scenario is split to train and test by time-point splitting. Holdout data was defined as the last movie a given user watched.

4.1.1. SVD

We fine-tuned following hyper-parameters for SVD:

Table 1. Rank hyper-parameter grid

Ranks	16	24	32	48	64	128
	192	256	384	512	768	

Table 2. Scaling hyper-parameter grid

Scalings	0.2	0.4	0.6	0.8

4.1.2. LightFM

LightFM was trained with the following parameters: number of components - 50, WARP loss (as mentioned in section 2.3.1), 100 number of epochs, user regularization - 10^{-3}, item regularization - 10^{-3}

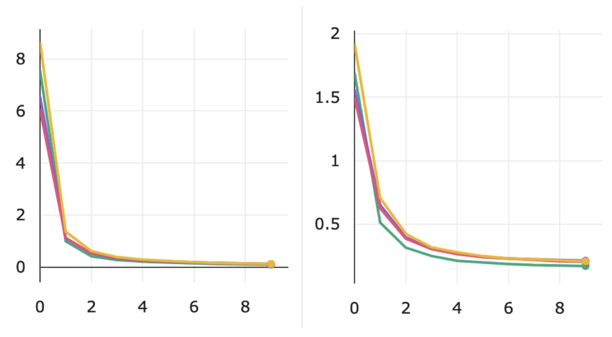
4.1.3. DSSM

As mentioned in section 2.3.3, DSSM is trained with triplet loss as the loss function. The following are the parameters that gave us the best result:

- epoch/steps-per-epoch: 30/100
- optimizer: Adam
- learning rate: 0.001
- batch size: 64

4.1.4. DeepFM

As discussed in section 2.3.3, we utilized BCE loss as DeepFM's loss function. The loss plots for both training and validation are included below:



((a)) Train Loss ((b)) Val Loss

Figure 6. Loss Plots

Both plots showed similar behavior. We experimented with several hyper-parameters. Mainly, we tuned the cutoff for binarization (mentioned in 2.3) and learning rate. The following are the parameters that gave us the best result:

- epoch/steps per epoch: 10
- optimizer: Adam
- learning rate: 0.001
- batch size: 256
- binarization cutoff: 75

4.2. Results

Using the ranks and scalings mentioned in 4.1.1, the MRR@20 and COV@20 (2.3.4) scores is presented in the following grids in figure **??** and **??**.

To summarize our experiments result, we've compiled all MRR@20 and COV@20 scores of all 4 implemented models below:

Table 3. Final results table

	Warm-start		Cold-start	
	MRR@20	COV@20, 10^{-2}	MRR@20	COV@20, 10^{-2}
Scaled SVD	0.064	0.450	0.002	0.456
LightFM	0.043	0.176	0.002	0.166
DSSM	0.022	0.240	0.003	0.244
DeepFM	0.086	0.185	0.01966	0.185

Aligning with our hypothesis, DeepFM performed better than our baselines in both warm-start and cold-start scenarios in terms of MRR metric. However, coverage metric demonstrate that DeepFM is popularity biased, which means that it recommends mostly popular items. Collaborative fine-tuned SVD demonstrated close results to the DeepFM for warm scenario, but much worst results in cold start scenario.

5. Conclusion

In this paper, we examined the performance of a hybrid deep learning model - DeepFM. Thus, we compared DeepFM against several implemented baselines - SVD, LightFM and DSSM. We hypothesized that due to the effectiveness of DeepFM in extracting low and high order feature interaction, this hybrid model will outperformed our baselines in both cold and warm start scenarios. We performed our experiments on the Kion dataset - a real-world dataset that offers implicit interactions of user and items. Our results demonstrated higher accuracy in both MRR@20 and COV@20 in favor of DeepFM.

There are several limitations with our work. Although some fine-tuning occurred, we did not extensively tune all four models implemented. Furthermore, our data pre-processing stage differs slightly between DSSM and the other 3 models. Future work is aimed to fine-tune the proposed algorithm and make it more robust and regularized to the input data.

References

[1] Guo H., Tang R., Ye Y., Li Z., He X. (2015) DeepFM: An End-to-End Wide Deep Learning Framework for CTR Prediction.

[2] Petrov A., Safilo I., Tikhonovich D., Ignatov D. MTS Kion Implicit Contextualised Sequential Dataset for Movie Recommendation. arXiv:2209.00325

[3] Maciej Kula (2015). Metadata Embeddings for User and Item Cold-start Recommendations. arXiv:1507.08439v1

[4] Movielens dataset. link

[5] Yu D., Seltzer M., Li J., Hyang J., Seide F. (2013) Feature Learning in Deep Neural Networks - Studies on Speech Recognition Tasks. arXiv:1301.3605

[6] S. Rendle and L. Schmidt-Thieme, "Pairwise interaction tensor factorization for personalized tag recommendation," Proceedings of the third ACM international conference on Web search and data mining, 2010.

[7] Weston J., Bengio S., Usunier N. WSABIE: Scaling Up To Large Vocabulary Image Annotation